ERGM–Project Report

Ninad Khargonkar

Contents

1 Introduction

Networks form an integral part of our daily life. The world is becoming highly connected due to increasing use of internet and social media websites with many of the popular websites having millions of active users. As a consequence of the communication networks, information can travel at blazing fast speed throughout the world. This leads to human interactions and collective behavior becoming more and more complex as they are no longer restricted to some local boundaries. Due to such high level of connection between people, it has become difficult to predict and model reactions to particular situations.

2 Social Networks

A social network is a set of people or groups of people with some pattern of contacts or interactions between them. Social networks represent the broad class of networks which express some kind of social relationship between their nodes and these relationships could be related to family, friendship and communities. The nature of the relationship can be of different types like positive and negative, ascribed and achieved.

The underlying structure of networks are graphs with the social units acting as nodes of the graph. In all the cases there is an assumption of binary nature of ties between the nodes relating to the presence or absence of the relationship between the nodes.
The graph could be directed, undirected and the directed graph could also additionally signify a symmetric/asymmetric relationship between the nodes.

The presence of an edge between two nodes indicates the presence of a relationship. In some cases the edges of the graphs also have weights associated with them to indicate strength of the tie. Nodes(vertices) and edges are the primary constituent data set in a network. Even if these are fixed, the distribution of the ties among the nodes and the structure of the network can vary significantly in different networks.

There are many techniques used to measure the properties of a network like density, centrality and clustering. In addition to these, there exist statistical models which are estimated from available data and then are used for simulation and studying hypotheses for social processes. They are useful to observe and understand the uncertainty associated with some outcomes in complex social networks and answer some questions relating to presence of social groups.

3 Sampling from graphs

The massive size of underlying graphs of such networks makes many analysis tasks prohibitively expensive and most of the time it is not possible to get a complete picture of the graph and its characteristics. Therefore efficient sampling of such networks plays an important role in scaling down and reducing the size of graph and accurately representing its properties. There are many known algorithms to compute interesting measures (shortest paths, centrality, between- ness, etc.), but several of them become impractical for large graphs. Thus graph sampling is essential.

The problem we consider, is how to efficiently estimate global graph properties such as number of edges, triangles and vertices using random walks. We present an analysis of such walks to determine the expected value of the property, and the accuracy of our estimate.

3.1 Basics of graphs

Graph $G = (V, E)$ is viewed as consisting a set of nodes V and a set of edges between the nodes E. It follows that $E \subseteq VxV$ as each edge e is a pair of nodes (a, b) which are its ends. There may also be weights attached to each edge and the pair of nodes for each edge can be ordered or unordered depending on whether the graph is directed or undirected.

Adjacency matrix : Another standard way of representing a graph G is through its *adjacency matrix* A_G which is square matrix which may additionally be symmetric if the G is undirected. Each entry in the matrix indicates a possible connection between the nodes and may be also be given different weights.

$$a_{c,d} = \begin{cases} 1 & \text{if } (c,d) \in E \\ 0 & \text{if } otherwise \end{cases}$$

As the graph is undirected, $a_{c,d} = ad, c$, i.e the matrix is symmetric.

A **walk** is a sequence of edges $(c_1, c_2), (c_2, c_3), ..., (c_{k-1}, c_k)$.

A **path** between two nodes i and j is a walk with no repeated nodes.

Cycle is defined as the path where the initial node is equal to final node.

A **geodesic** between two nodes i and j is the shortest path between them. Shortest path has the minimum number of edges in it.

Diameter is defined as the length (number of edges) of the longest geodesic between any two nodes in the graph.

Degree of node is the number of edges connected to it. A directed graph will have both *in-degree* and *out-degree* for a node.

Some more basic theory can be found in [12].

The standard ways to traverse the entire graph are the popular methods, BFS (Breadth First Search) and DFS (Depth First Search). BFS tries to visit all neighbors first and then moves on to subsequent neighbors whereas DFS tries to go far (depth) from the starting vertex v.

3.2 Random Walks and Markov Chains

One use of random walks and Markov chains is to sample from a distribution over a large universe. Informally, we set up a graph over the universe such that if we perform a long random walk over the graph, the distribution of our position approaches the distribution we want to sample from.

A random walk is a process for traversing a graph where at every step we follow an outgoing edge chosen uniformly at random. We choose a starting vertex u and length of the walk t which is defined as the number of steps with each step being the process of randomly choosing a neighbor v of u. If the graph is directed then we only move to those neighbors of u to which it is pointing.

The process can be thought of as a Markov chain if the outgoing neighbor edge is chosen according to a fixed distribution. Markov chains are special because they have the property that the probability of an event is only dependent on its previous state.
Given a finite state space, a Markov chain is a random process in which the probability of the state X_t occupied at step $t > 0$ depends only on the previous state X_{t-1}, i.e:

$$P(X_t = x_t | X_{t-1} = x_{t-1}, ..., X_0 = x_0) = P(X_t = x_t | X_{t-1} = x_{t-1})$$

Transition matrix :

A Markov chain is usually represented by a transition matrix P. It is a square matrix in which each entry denotes the probability of transition from one state to another. In a graph the states are the nodes and the entries are the probability of transitioning from a node to one of its neighbors. Therefore the transition probability is as follows :

$$p_{ij} = P(X_t = j | X_{t-1} = i)$$

Stationary distribution :

One of the properties of interest is the distribution in position after running the random walk for a large number of steps. If we start with a π distribution on nodes where π is a $nx1$ vector, after the first step in the walk the distribution will be given by $P^T\pi$.

Under certain special conditions the limiting distribution given below will exist and will also be independent of the initial one.

$\lim_{t\to\infty}(P^T)^t\pi$

With such conditions, the limiting distribution will be identical to the *stationary distribution* which follows the equation:

$\pi^* = P^T\pi^*$

It can be seen any further step in the walk produces no change in the distribution and the chain has essentially converged. Convergence to the stationary distribution means the state of the chain no longer depends on the initial state.

3.3 Scale-down sampling

The goal in scale down sampling is to create a sample graph S on graph G with v nodes and sample u nodes such that $u << v$. S should have similar graph properties as G like similar degree distribution and/or diameter. Some sampling methods by exploration of a graph are presented:

Random Node/Edge selection :

This is a straightforward way to create a sample graph when the entire node and edge list is available. The method is to uniformly at random select a set of nodes and then see the induced graph by the sampled nodes. It is called Random-Node sampling. One can also do the same with edges by selecting a uniform sample randomly and then observing induced sub-graph.

Classic Random Walk :

In the classic random walk, from the current node the next node to visit is chosen uniformly at random from its set of neighbors. Thus the probability of transition from v to u is:

$$P_{v,u} = \begin{cases} \frac{1}{d(v)} & \text{if } u \in neighbors(v), \\ 0 & \text{if } otherwise. \end{cases}$$

Here $d(v)$ represents the degree of node v.

The probability of being at a particular node v, π_v will converge to the stationary distribution :

$\pi_v = \frac{d(v)}{2m}$ where m is equal to number of edges in the graph or $|E|$.

As $\pi_v \sim d(v)$, this method is biased toward nodes with higher degree as such will be visited more often by the random walk.

Random Walk with jumps:

In Random Walk with jump sampling, a starting node is selected uniformly at random for a random walk. But the difference is at every step with some

probability p (usually $p = 0.15$), there is a possibility of going to any random node in the graph and continue the walk.

This avoids the problem of getting stuck in a node with no neighbors or all edges directed inwards (in case of a directed graph) or in small and isolated region of the graph.

The jump to random node with fixed probability is implemented with generating a random number and based on its value selecting either the jump or going with the standard random walk.

Metropolis Hastings Random Walk :

It works like classic random walk but has a correction factor so that the bias towards high-degree nodes is eliminated. The transition probabilities are modified so that the markov chain converged to the desired uniform distribution, that is each node has the same probability.

$$P_{v,u} = \begin{cases} min(\frac{1}{d(v)}, \frac{1}{d(u)}) & \text{if } u \in neighbors(v), \\ 1 - \sum_{w \neq v} P_{v,w} & \text{if } u = v, \\ 0 & \text{if } otherwise. \end{cases}$$

The stationary distribution in this case comes out to be $\pi_v = \frac{1}{|V|}$ which is a uniform distribution and not dependent on the degree of the node.

This algorithm removes the bias towards the higher degree nodes by always accepting the jump towards a smaller degree node and rejecting some jumps towards higher degree nodes.

4 R package–igraph

The *igraph* library ([6],[5]) provides options and tools for network analysis and visualization. Its core functionality is implemented as a *C* library but its also available as a *python* and *R* package and is a open-source and free software. It can be installed and loaded in *R* by:

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

The package provides functions for creating certain kinds of graphs, reading and loading graphs and plotting them, setting node and edge attributes and also getting various kinds of properties of graph like edge density, clustering, triangles, degree distribution etc. USED intergraph package [3]

5 Network decriptives

Some of the summary statistics of the graph help in answering questions who cannot be answered by manual analysis due to large amount of data. Sometimes one cannot even visualize the graph due to it sheer size and these properties help in giving insights into the graph structure.
Models of the networks help in understanding the meaning of properties and also predicting behavior based on the observed properties. Some graph properties are given below:

5.1 Edge density

Edge density refers to the ratio of actual number of edges to the theoretically maximum number of edges in the graph. If graph G has n nodes then the maximum number of edges will be when each node is connected to every other $n-1$ nodes, i.e $n(n-1)/2$ edges. It gives an idea about how *dense* a graph may be.

$edgedensity = \frac{m}{n(n-1)/2}$, where m is actual number of edges.

5.2 Mean–geodesic distance

It is simply the mean of the lengths of all geodiscs between the nodes of graph. It is by definition bounded above by the diameter. If the graph is not connected (making some geodisc lengths undefined), one can look at the largest connected part pf graph.

$l = \frac{\sum_{i \geq j} l_i j}{n(n-1)/2}$

Intuitively paths refer to connectivity and are related to centrality measures. It can also be used to study the small-world effect i.e most pairs of vertices are connected by a short path through the network.

5.3 Degree distribution

Degree is the simplest centrality measure in a network. The average degree of the graph is simply: $1/n \sum_v degree(v)$

Degree distribution represents the relative frequency of different values for the degree . If θ_d represents the fraction of nodes having degree equal to d, then the degree distribution can be viewed as a histogram for different values of θ_d. Usually the degree distribution follows a power-law and some kind of heavy tailed distribution with a very long right tail and then its known as scale-free.

5.4 Clustering

A clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together and transitivity in graph. In most social networks, nodes tend to create tightly knit groups characterized by a relatively high density of ties. The global clustering coefficient C_g is defined as :

$C_g = \frac{count_trg}{count_trp}$, where $count_trg$ is 3x number of triangles and $count_trp$ is the number of connected triples.

The local clustering coefficient C_loc measures how influential a node and clustering can be regarded as a type of centrality that takes small values for powerful individuals rather than large ones. The local clustering coefficient for node v is:

$C_loc(v) = \frac{trg(v)}{trip(v)}$, where $trg(v)$ is number of triangles connected to v and $trip(v)$ is number of triples who have v as their center.

6 Analysis of the Network

The dataset used for graphical demonstration of the network properties is the *ERDOS* dataset which contains the direct joint-authors of Paul Erdős. It was taken from the pajek datasets [1] and it is a file of people who have the Erdős number one from 1999.

In the *R* environment we load the *igraph* package and read the data file . One can get the basic details of the network graph by the inbuilt functions of igraph.

```
> library(igraph)
> eg1<-read_graph("./ERDOS991.NET", format = "pajek")
> nnodes <- vcount(eg1)
> nedges <- ecount(eg1)
> nnodes
[1] 492
> nedges
[1] 1417
> summary(eg1)
IGRAPH U--- 492 1417 --
+ attr: id (v/c)
```

The summary of the network graph shows that there are 492 vertices and 1417 edges between them. There is also a vertex attribute called 'id' which contains the names of the 492 direct collaborators of Erdős.

Then the disconnectedness and directness of the graph is checked. The graph is undirected with no weights for edges and also not connected which means that there must be some isolated nodes.

```
> is.connected(eg1)
[1] FALSE
> is.directed(eg1)
[1] FALSE
> is.weighted(eg1)
[1] FALSE
```

Next the graph is plotted so as to gain some insight into its properties like density and centrality. One can set more vertex attributes like color and size (relative to degree) and edge thickness (related to weight).

For large graphs, one can even choose the graph layout algorithm which optimize the spread so as to avoid crossing edges. The minimal spanning tree is also plotted to get a clearer picture of the graph.

```
> set.seed(42)
> plot(eg1, layout=layout_nicely, vertex.label = NA, vertex.size=4,
    edge.color="gray20", edge.width = 1.5)
> mstp <- minimum.spanning.tree(graph = eg1, weights = NULL)
> plot(mstp, vertex.label=NA, vertex.color="lightblue", vertex.size
    = 3)
```

See the Figure 1 for the plot. It has the isolated nodes in red color and the nods in diameter(longest path) in golden color. The figure 2 shows the minimal spanning tree of the network for clear look.

Figure 1: Network plot

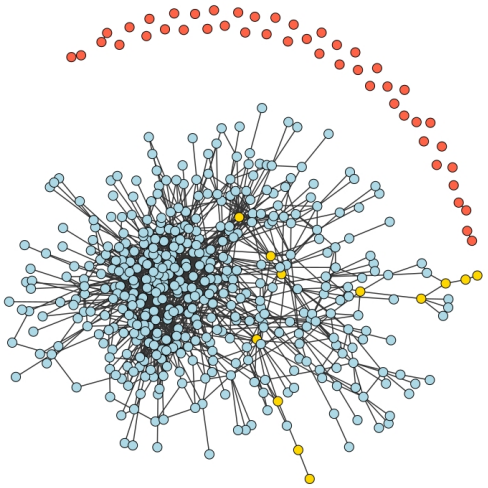
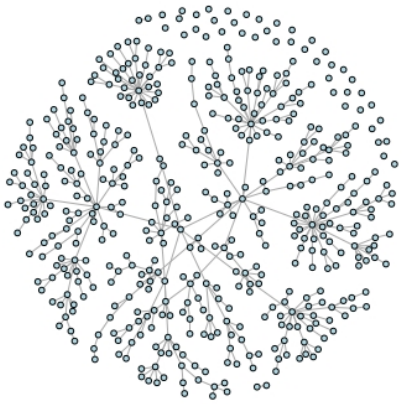


Figure 2: Minimal spanning tree



The basic node and network descriptives are available through in-built commands.

```
> edge_density(eg1)
[1] 0.01173149
> transitivity(eg1, type = "global")
[1] 0.2128726
> sum(count_triangles(eg1))/3 #nof triangles
[1] 1302
```

The degree and its distribution with histogram is very helpful in analysis of the graph. The range indicates there are some isolated nodes along with highly connected node having degree 42.

From the histogram in Fig. 3 there is a substantial fraction of nodes of lower degree but there are also some number of nodes with higher order degrees. A log–log scale used in Fig. 4 is more useful as the distribution decays and follows a power law tail.

```
> deg<-degree(eg1, mode="all")
> deg.dist <- degree_distribution(eg1, cumulative=T, mode="all")
> range(deg)
[1]  0 42
> hist(deg, breaks = max(deg) ,main="Histogram of node degree",
    xlab="Degree")
```

Figure 3:

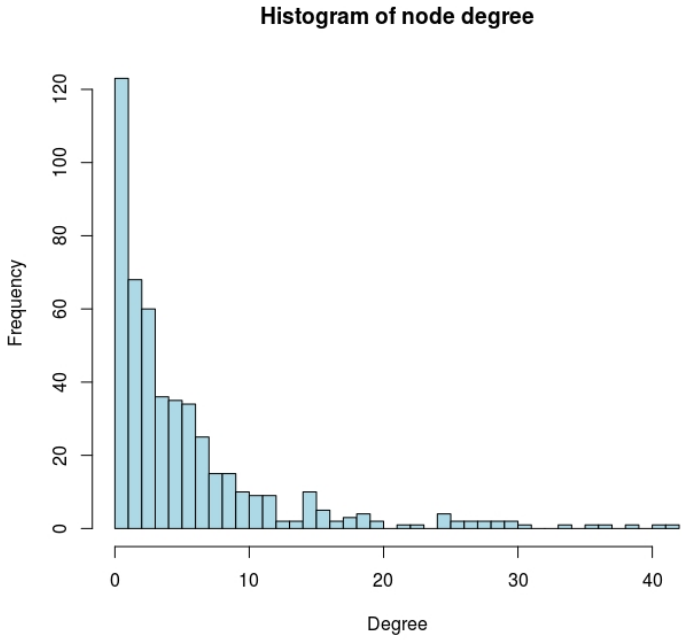
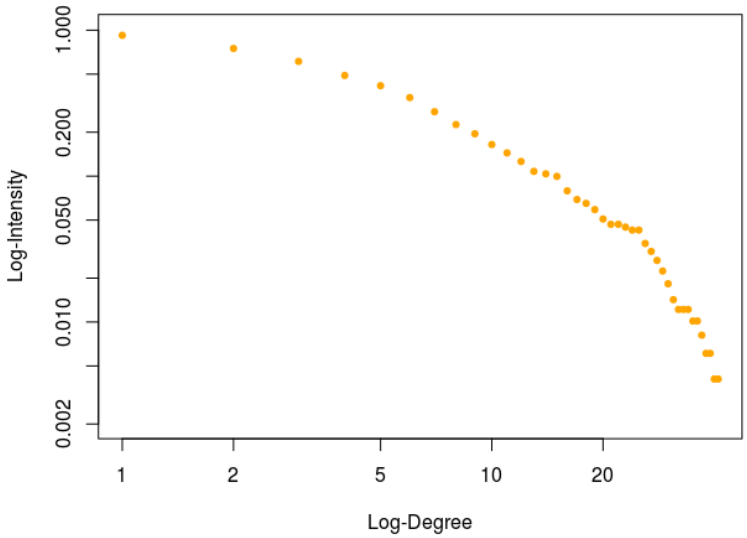
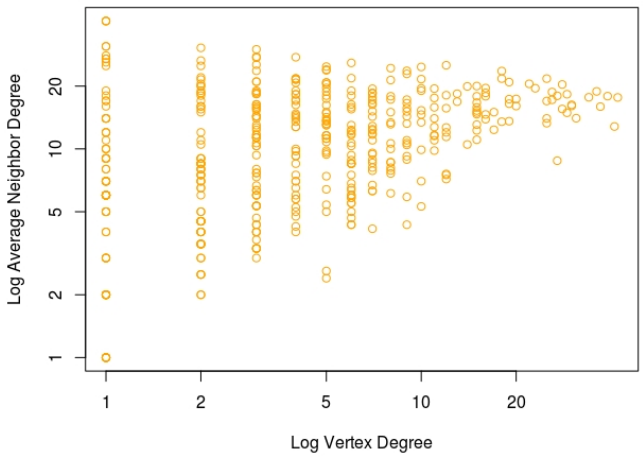


Figure 4: Log Degree plot



One can also see how nodes of different degree are connected to one another. The average neighbor degree can used for this analysis. In a plot of average neighbor degree versus node degree in Fig. 5 it is seen that high degree nodes tend to connect with similar higher degree nodes while low degree connect with nodes across the spectrum.

Figure 5: Average neighbor degree



The mean path length and diameter of the graph is also available.

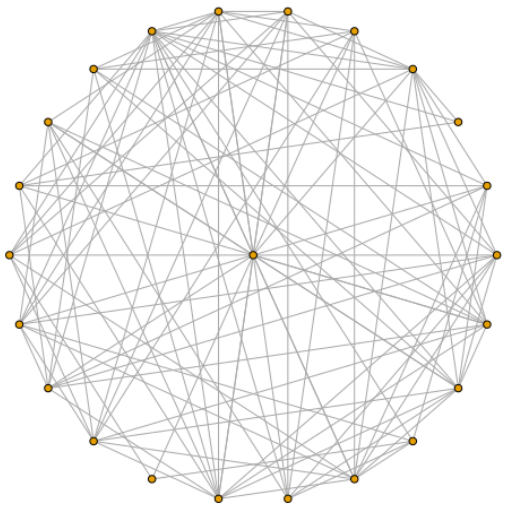
```
> diam <- diameter(graph = eg1, directed = F, weights = NA)
> v_diam <- get_diameter(eg1, directed = F)
> mean_distance(eg1, directed = FALSE)
[1] 3.951914
```

In addition to the basic properties, one can use the functions for community detection and finding cliques. A cohesive subset of node is by definition that is strongly connected in itself and is separated from other groups.

```
clq<-cliques(eg1)
largest_clq <- largest_cliques(eg1)
maximal_clq <- max_cliques(eg1)

ceb <- cluster_edge_betweenness(eg1)
plot(ceb, eg1, vertex.label=NA) #Rplot for groups..
```

Also, the plot in Fig. 6 shows the network's induced subgraph with nodes having a degree greater than 20. This shows how highly inter-connected they are to other nodes of similar degree.

Figure 6: high degree nodes



7 Performance of sampling methods

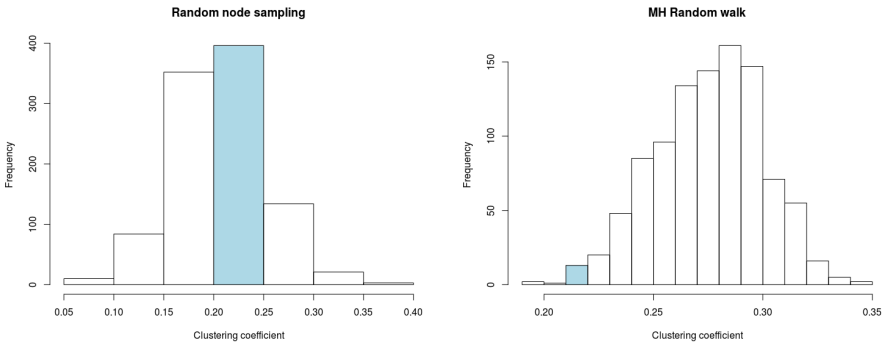
The four different sampling algorithms: Random node selection(rns), Simple random walk(rw-simple), Random with jump (rw-jump) and Metropolis-Hastings random walk(mhrw) are tested using the same dataset of Erdos number one persons. One thousand samples are taken for every algorithm with each sample containing 150 nodes ($\sim 30\%$ of size).
During run of each procedure four graph properties are taken into account: edge density, global clustering coefficient, average path length and the mean degree of the sampled graph. The values for the original graph are as follows:

```
transitivity(graph = eg1, type = "global") #clustering coef
[1] 0.2128726

edge_density(graph = eg1)
[1] 0.01173149

mean_distance(graph=eg1, directed = FALSE, unconnected = FALSE)
[1] 91.01475

mean(degree(graph = eg1))
[1] 5.760163
```

Now we look at the histograms of the observations to get some idea about how the procedures compare. In the following figures, the bin in which the actual value lies is marked in blue.

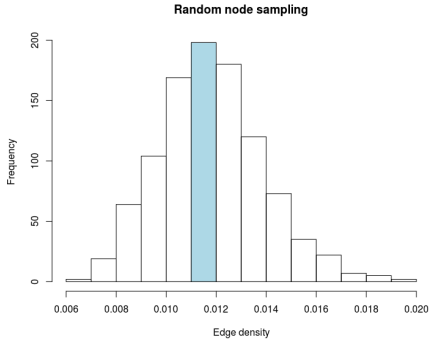
For the **clustering coefficient** in Fig. 7 , the rw-jump and rw-simple fail to get the actual value in any of their 1000 samples. But the mhrw and rns methods are successful in it although the mhrw method is only able to capture the desired value for a small fraction of the total samples.

Figure 7: Clustering coefficient



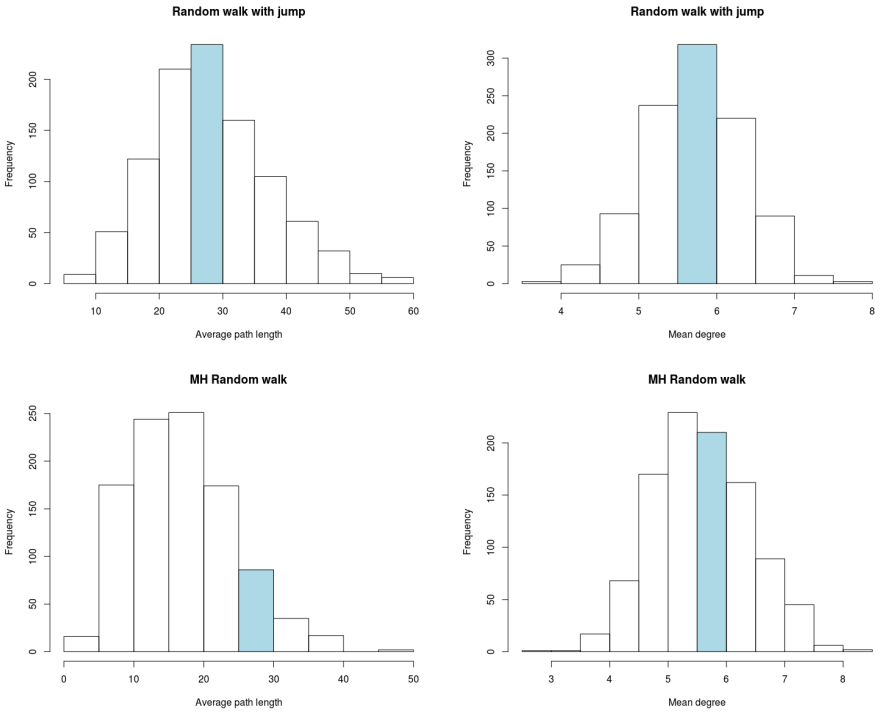
For the **edge density** in Fig. 8 only the rns method is able to observe the actual value while all other methods tend to over-estimate the value in their sample. This is likely because of their preference towards higher degree nodes.

Figure 8: Edge density



In the case of **average path length** and **mean degree** in Fig. 9 , the rns method fails badly while all the other methods are able to take the actual values into their samples.

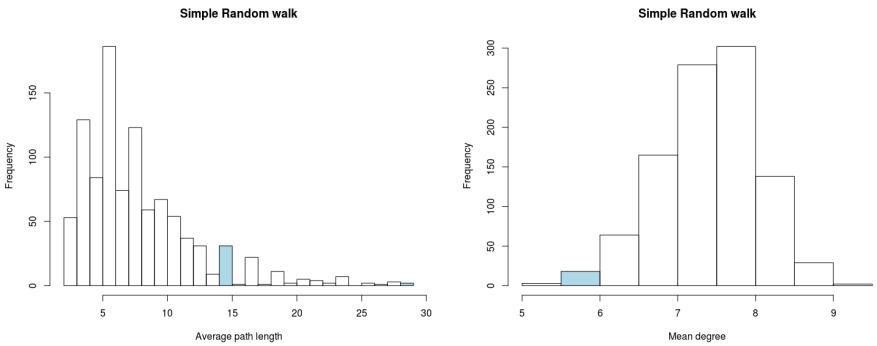
Figure 9:



Both the mhrw and rw-jump method have a significant proportion of their samples including the actual values. Note that the value of average path length is scaled down from the original graph according to the size of the sample.

Also, the rw-simple procedure performs worse than both the above methods as it has a much lower fraction of 'good' samples as it is evident from the Fig. 10

Figure 10:



The errors in the observations are quantified by using the metrics like relative error and root mean square deviation.

Figure 11: Relative error

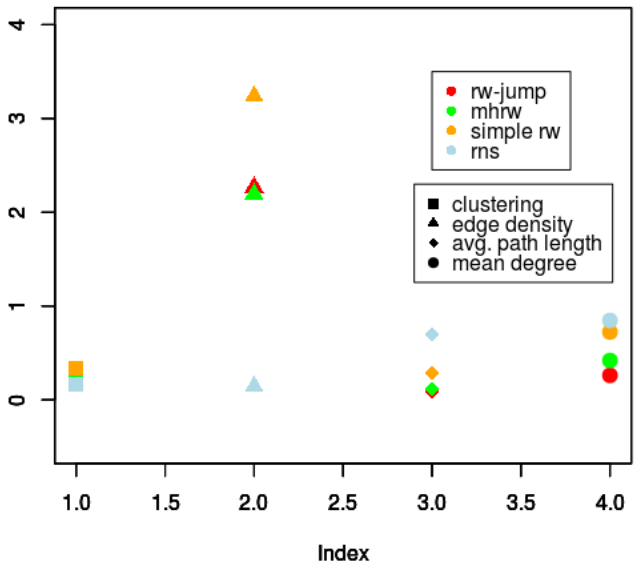
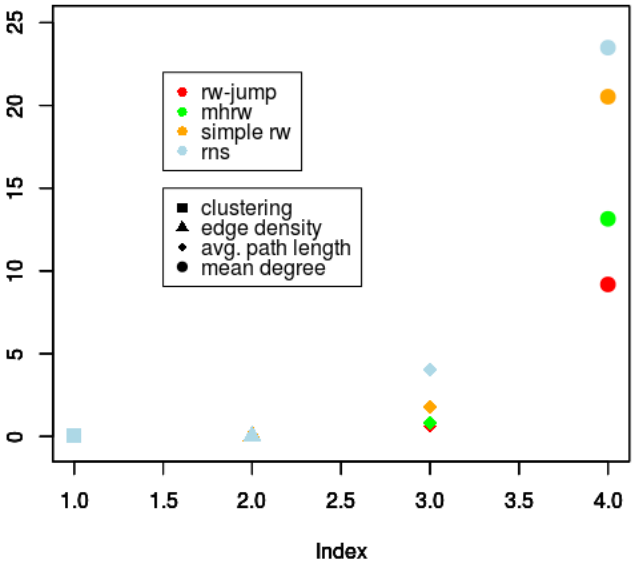


Figure 12: Root mean square deviation



As we can see from the above error plots, random node selection (rns) usually performs the best in the case of clustering coefficient and edge density but falls behind the other methods like in the case of average path length where it tends to highly under estimate the value and likewise in the case of mean degree.

Metropolis-Hastings random walk and Random walk with jump both prove to be better than the generic random walk and especially mhrw as it was found to include the true value of parameter in three out of four cases.

As for comparison between mhrw and rw-jump, there are very close to each other but mhrw (after a sufficient burn-in period) can eliminate the bias towards the higher degree nodes that is present in rw-jump though it is still better than the simple random walk as rw-jump incorporates the random jump to any node with some non-zero probability.

8 Statistical Models

Models are used to study mechanisms for producing observed characteristics and test significance. Statistical models also allow inferences about whether certain network substructures – usually represented in the model by one or a small number of parameters. Models are also useful in analysis of complex networks.

Each network tie is regarded as a random variable i.e for i and j who are distinct Y_{ij} represents the random variable and y_{ij} it's value which is equal to one if there is a tie between i and j and zero otherwise in the case of binary ties.

Exponential random graph models have the form given below. The probability of observing a graph $\mathbf{y}$ in the distribution is given by the equation :

$P(\mathbf{Y} = \mathbf{y}) = (\frac{1}{\kappa})\ exp(\sum_A \theta_A g_A(y))$

Variable A represents a configuration which might include edges, reciprocated ties, transitive triads etc. and the sum is over all such configurations.
θ_A is the parameter corresponding to the configuration. It is non-zero only if all pairs of variables are assumed to be conditionally dependent on the rest of the graph.

$g_A(y) = \prod_{y_{ij} \in A} y_{ij}$ It is the network statistic corresponding to the configuration A. The constant κ is the normalizing constant for the probability distribution.

In the case of simplest assumption, we only assume that the edges are independent and occur randomly with a fixed probability i.e all possible distinct ties are independent of each other.

$P(\mathbf{Y} = \mathbf{y}) = (\frac{1}{\kappa})\ exp(\sum_{i,j} \theta_{ij} * y_{ij})$

Every edge Y_{ij} is a possible configuration and $g_A(y) = y_{ij}$. We may also assume that for each tie $\theta_{ij} = \theta$ and denote $L(\mathbf{y}) = \sum_{ij} y_{ij}$. Then the formula becomes much simpler: $P(\mathbf{Y} = \mathbf{y}) = (\frac{1}{\kappa})\ exp(\theta L(\mathbf{y})$

In the case of Markov random graphs in addition to edges, other configurations like 2-stars, 3-stars and triangles are also taken into account. If $S_2(\mathbf{y})$ represents the count of two-stars, $S_3(\mathbf{y})$ represents the count of three-stars and $T(\mathbf{y})$ means the count of triangles then the formula becomes:

$P(\mathbf{Y} = \mathbf{y}) = (\frac{1}{\kappa})\ exp(\theta L(\mathbf{y} + \sigma_2 S_2(\mathbf{y}) + \sigma_3 S_3(\mathbf{y}) + \tau T(\mathbf{y}))$

One can also include higher order star configurations or other complex cases with node level variables. Further information can be found in [10].

8.1 Model fitting

We use the same Erdos dataset that was used in sampling analysis for ergm modeling and testing the goodness of fit of different models. First we take a look at the network object by using the package **statnet** ([8]) and its associated packages **sna, ergm, network**. Also, the package **intergraph**([3]) is useful for coercing network object to graph objects and vice versa.

```
> install.packages("statnet")
> library(intergraph)
> # 'eg1' is the loaded igraph object (erdos graph)
> enet <- asNetwork(eg1)

> enet
 Network attributes:
  vertices = 492
  directed = FALSE
  hyper = FALSE
  loops = FALSE
  multiple = FALSE
  bipartite = FALSE
  total edges= 1417
    missing edges= 0
    non-missing edges= 1417

 Vertex attribute names:
    id vertex.names

 Edge attribute names not shown
```

This already displays some of the basic information related to the network. Some other features can be accessed through the summary function which accepts many terms for display like density, triangle count, two-stars, three-stars etc. and also for finding isolated nodes.

```
# summary function

> summary(enet ~ edges + meandeg + density)
       edges      meandeg      density
1.417000e+03 5.760163e+00 1.173149e-02

> summary(enet ~ triangle + kstar(2) + kstar(3))
triangle   kstar2   kstar3
    1302    18349   129697

> length(isolates(enet))
[1] 38
```

As seen in the output the triangle, 2-star and 3-star counts are obtained.

Exponential random graph models (ERGM) can be fit into a network dataset using the **ergm** package. Many different parameters can be used as the model terms and they are know as ergm-terms (found in the help section for the package). They are functions of network statistics that we hypothesize may be more or less common than what would be expected in a simple random graph (where all ties have the same probability). For example, specific degree distributions, or triad configurations.

One key difference in model terms is that terms are either dyad independent or dyad dependent. Dyad independent terms (like nodal homophily terms) imply no dependence between dyads—the presence or absence of a tie may depend on nodal attributes, but not on the state of other ties. Dyad dependent terms (like degree terms, or triad terms), on the other hand imply dependence between dyads. A model with dyad dependent terms may also requires a different estimation algorithm.

First we begin by taking the simplest model of interest, the one parameter model which gives an equal probability to all the edges in the network. It is called a Bernoulli or Erdos-Renyi model. The vector of statistics thus only contains the number of edges in the network.

```
> model1 <- ergm(enet ~ edges) # model estimation

==========================
Summary of model fit
==========================

Formula:   enet ~ edges

Iterations:  7 out of 20

Monte Carlo MLE Results:
      Estimate Std. Error MCMC % p-value
edges -4.43368    0.02672      0  <1e-04 ***
---
#Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 167445  on 120786  degrees of freedom
 Residual Deviance:  15416  on 120785  degrees of freedom

AIC: 15418    BIC: 15427    (Smaller is better.)
```

The interpretation for the results is as follows: the log-odds of a tie will be equal to the estimate multiplied by the change statistic i.e change in the number of ties, which is equal to one since adding or removing a tie to the network changes the statistic (number of edges) by one.

Therefore the log-odds for a tie to exist is equal to -4.43368. Then for $a = -4.43368$, $p = \exp(a)/1+\exp(a)$ will give us the probability corresponding to the log-odds score. p comes out to be 0.0117314 which is equal to the density of the actual network found be the summary command for it.

8.2 Simulation

To see how this model captures the structure in our original data. To do this, we use the model fit that we have just generated to simulate new networks at random, and consider how these are similar to or different from our data.

```
> sim1 <- simulate(model1, burnin = 1e+4, verbose = TRUE, seed =
    10)
Starting MCMC iterations to generate 1 network
Sampler accepted  97.870% of 10000 proposed steps.

> summary(sim1 ~ edges)
edges
 1396
> summary(sim1 ~ density)
   density
0.01155763
> summary(sim1 ~ triangle)
triangle
 27
```

One can change the burn-in from its default value of 1000 and a higher value for it gives a chance that the output is approximately independent of initial conditions. The object then returned from the simulate command is of class network.

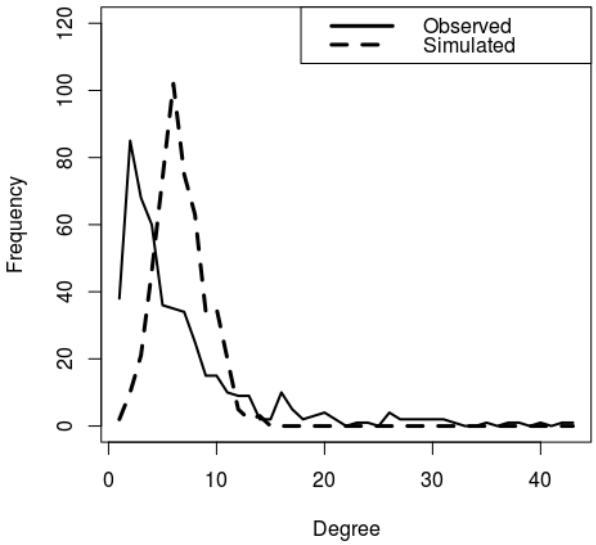
The model fit can then be compared to actual network for example by using the degree distribution. As seen in Fig.13 the simulated network somewhat matches the actual network.

In the sense that it still has some bias giving a higher count to some degrees and looks like shifted to the right but in fact the mean degree remains close to the actual value probably due to under estimation of higher degree nodes in the simulated network.

While the simple model is able to properly estimate the mean degree and number of edges, it fails in estimation of other cases like triangles and stars. One can try introducing such terms in model fitting command like (edges + triangle) or (edges + triangle + kstar(2) + kstar(3)).

If we try with edges and triangle in the ergm command, a warning about degeneracy is shown in the console. Therefore it may be the case that these models involving dyadic dependence are maybe degenerate for this network. They are poor descriptions of the actual network and the proper convergence is tough and the model cannot be properly estimated.

Figure 13: Degree distribution



To resolve the case of degeneracy, we try a new statistic called 'geometrically weighted edgewise shared partner' or gwesp. From the paper[11], it is gives a definition as:

The geometrically weighted degree parameter explicitly models the degree distribution but puts more weight on the numbers of nodes with lower degrees, with weights decreasing geometrically as the degrees increase.

The edge-wise shared partner (or ESP) parameter models the distribution of shared partners of tied actors, but with weights decreasing geometrically as the number of shared partners increase.

The gwesp statistic has a scaling parameter called (α). When it is equal to zero, the statistic is equivalent to the number of edges that are in at least one triangle and as it approaches infinity, the statistic approaches three times the number of triangles. The model is less likely to be degenerate with a lower value of α.

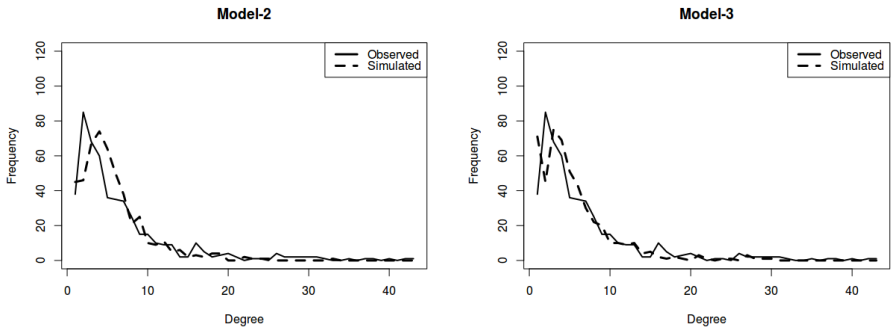
In beginning we take a $\alpha = 0.1$ and compare the count of triangles with the the first model which could estimate only 27 triangles in the network. Clearly the second model works much better as it is able to estimate 397 triangles.

Later on, we can keep increasing the value of α to give better estimates of triangle count and also tweak some parameters for the ergm function. In the third model we take $\alpha = 0.2$ and and also increase the interval size.

```
# Taking alpha = 0.1 and 0.2

> model2 <- ergm(enet ~ edges + gwesp(0, fixed=TRUE))

> sim2 <- simulate(model2, burnin = 1e+4, verbose = TRUE, seed=100)
Starting MCMC iterations to generate 1 network
Sampler accepted  32.640% of 10000 proposed steps.

> summary(sim2 ~edges)
edges
 1155
> summary(sim2 ~ triangle)
triangle
 397
> summary(sim2 ~ density)
    density
0.009562366

> model3 <- ergm(enet ~ edges + gwesp(0.2, fixed=TRUE),
    MCMCsamplesize = 1e+3, interval = 1000)

> sim3 <- simulate(model3, burnin = 1e+4, verbose = TRUE, seed=100)
Starting MCMC iterations to generate 1 network
Sampler accepted  24.480% of 10000 proposed steps.

> summary(sim3 ~ edges)
edges
 1117
> summary(sim3 ~ density)
    density
0.009247761
> summary(sim3 ~ triangle)
triangle
 506
```

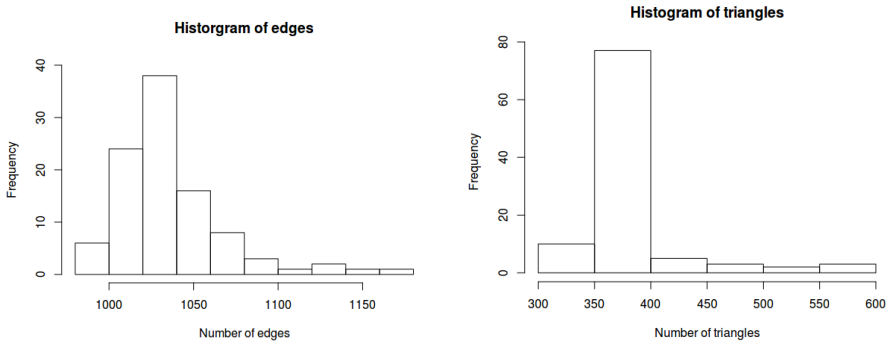
It can be seen that increasing the value of α and interval size has clearly helped in the estimating triangle count but not without taking a minor hit in the count of edges and density. But on the other hand these later models fit the degree distribution much better than in earlier case as seen in Fig.14. Some other higher values of α can be tested along with tweaking the MCMC parameters.

Figure 14: Degree distribution



Instead of comparing a single simulation, we simulate many networks and compare the full distribution of statistics to that from the observed data in the original network. We take 100 simulations from model-3 and the distributions for triangles and edges can be seen in Fig.15

Figure 15: Edges and Triangles



We can also observe goodness of fit tests by using **gof** command so that we observe the estimation for properties that were not included in the model description.

```
# Simulation of 100 networks

> sim31 <- simulate(model3, nsim = 100, burnin = 1e+4, verbose =
    TRUE)
> sim31
Number of Networks: 100
Model: enet ~ edges + gwesp(0.2, fixed = TRUE)
Reference: ~Bernoulli
Constraints: ~.
Parameters:
          edges gwesp.fixed.0.2
      -6.340758        2.292049

> sedge <- summary(sim31 ~ edges)[,1] # Number of edges
> strig <- summary(sim31 ~ triangle)[,1] # Number of triangles

> mod3.gof <- gof(model3 ~ degree + esp + distance)
> plot(mod3.gof)
```

References

[1] V. Batagelj and A. Mrvar. *Pajek datasets*, 2006. `http://vlado.fmf.uni-lj.si/pub/networks/data/`.

[2] S. Bhattacherjee. The one-hop neighborhood of paul erdos. `https://wiki.cs.umd.edu/cmsc734_f13/images/4/45/One-hop-neighborhood.pdf`.

[3] M. Bojanowski. *intergraph: Coercion Routines for Network Data Objects*, 2015. `http://mbojan.github.io/intergraph`.

[4] C. Cooper, T. Radzik, and Y. Siantos. Estimating network parameters using random walks. *Social Network Analysis and Mining*, 4(1):1–19, 2014.

[5] G. Csardi. et nepusz, t.(2006). the igraph software package for complex network research. *InterJournal, Complex Systems*, 1695.

[6] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. `http://igraph.org`.

[7] M. Denny. Introduction to social network theory. `http://www.mjdenny.com/workshops/SN_Theory_I.pdf`.

[8] M. S. Handcock, D. R. Hunter, C. T. Butts, S. M. Goodreau, and M. Morris. *statnet: Software tools for the Statistical Modeling of Network Data*. Seattle, WA, 2003. `http://statnetproject.org`.

[9] M. S. Handcock, D. R. Hunter, C. T. Butts, S. M. Goodreau, and M. Morris. A statnet tutorial. *Journal of statistical software*, pages 1–27, 2008.

[10] G. Robins, P. Pattison, Y. Kalish, and D. Lusher. An introduction to exponential random graph (p*) models for social networks. *Social networks*, 29(2):173–191, 2007.

[11] G. Robins, T. Snijders, P. Wang, M. Handcock, and P. Pattison. Recent developments in exponential random graph (p*) models for social networks. *Social networks*, 29(2):192–215, 2007.

[12] Satish. Introduction to graph theory. *International Journal of Research in Engineering and Applied Sciences*, 3(6), 2013.

[13] T. Wang, Y. Chen, Z. Zhang, T. Xu, L. Jin, P. Hui, B. Deng, and X. Li. Understanding graph sampling algorithms for social network analysis. In *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*, pages 123–128. IEEE, 2011.