Page Rank

Ryan Greenup

October 15, 2020

Contents

1 Introduction

Any collection of interconnected information can form a network structure, consider for example citations, webpages, wikis, power grids, wiring diagrams, encyclopedias and interpersonal relationships. The analysis of these networks can be used to draw insights about the behaviour of such networks.

One important form of analysis is *netowork centrality*, a concept concerned with the measure of the importance, popularity and relevance of a node. In a relatively small graph, visualised in such a way so as to minimise the overlapping of edges, a general expectation would be that the centrality score would be correlated with geometric-centrality, this is demonstrated in figure 5 where the 2nd vertex has the highest *PageRank* score and is geometrically very central.

1.1 The PageRank Method

There are multiple ways to measure network centrality but this report is concerned with the *PageRank* method, this method asserts that the centrality of a vertex can be measured by the frequency of incidence with that vertex during a random walk.

This approach only makes sense if the random walk can:

1. Traverse the entire network
2. Escape dead ends on a directed graph

and so the *PageRank* method involves:

- Altering a corresponding transition probability matrix such that it corresponds to a stochastic primitive *Markov Chain*.
- Considering the stationary distribution of this new graph.

1.2 Power Walk and the Random Surfer

The typical method to adjust the transition probability matrix is the *Random Surfer*, introduced by Page and Brin in 1998 [21] as a distinguishing feature of the *Google* search engine, this aproach essentially introduces some probability of teleporting to other nodes during a random walk, this is illustrated in figure 4.

A shortcoming of this approach is that it assumes all edges are positively weighted. This menas that the model treats any link as an endorsement of the destination node, this may not necessarily always be true (consider for exmple burned-in advertisements or negative reviews). In the past attributing weights to links was not particularly feasible, recent developments in sentiment analysis has however made this possible meaning that this limitation is more significant.

The *Power Walk* approach, introduced by Park and Simoff in 2013 [25] is an alternative way to create a transition probability matrix that is defined for real weighted edges and could be used with sentiment analysis to more effectively measure network centrality.

These individual apprahces are discussed in more detail at 2 .

1.3 Stability and Convergence

The rate at which the algorithm for *PageRank* converges to a solution and the stability of that solution can both be measured by the second eigenvalue of the corresponding transition probability matrix (The details of this are discussed at 6).

It is not clear how the second eigenvalue is related to the method parameters of the *Power Walk* algorithm [25, §3.4] and this report aims to:

1. Implement methods to perform *PageRank* analysis using:
   (a) The *Random Surfer* model
   (b) The *Power Walk* model
2. Investigate the Relationship between the parameters of the *Power Walk* transition probability matrix and the second eigenvalue

Part I
Implementing PageRank

2 Mathematics of Page Rank

2.1 The Stationary Distribution of a Probability Transition Matrix

A graph can be expressed as an adjacency matrix $\mathbf{A}$:

$$A_{i,j} \in \{0, 1\}$$

Where each element of the matrix indicates whether or not travel from vertex j to vertex i is possible with a value of 1. [1]

During a random walk the probability of arriving at vertex j from vertex i can similarly be described as an element of a transition probability matrix $\mathbf{T}_{i,j}$, this matrix can be described by the following relationship [2]:

$$\mathbf{T} = \mathbf{A}\mathbf{D}_{\mathbf{A}}^{-1} : \tag{1}$$

$$\mathbf{D}_{\mathbf{A}} = \operatorname{diag}\left(\vec{1}\mathbf{A}\right) \tag{2}$$

The value of $\mathbf{D}$ is such that under matrix multiplication $\mathbf{A}$ will have columns that sum to 1 (i.e. a *column stochastic matrix*, see § 9.2), for a reducible or non-stochastic graph the definition of $\mathbf{D}$ would need to be adjusted to acheive this, this is discussed below

During the random walk, the running tally of frequencies, at the i^{th} step of the walk, can be described by a vector $\vec{p}$, this vector can be determined for each step by matrix multiplication:

$$\vec{p_{i+1}} = \mathbf{T}\vec{p_i} \tag{3}$$

This relationship is a linear recurrence relation, more importantly however it is a *Markov Chain* [20, §4.4].

Finding the Stationary point for this relationship will give a frequency distribution for the nodes and a metric to measure the centrality of vertices.

2.2 Random Surfer Model

2.2.1 Problems with the Stationary Distribution

The approach in 2 has the following issues

1. Convergence of (3)

    (a) Will this relationship converge or diverge?

[1] Some authors define an adjacency matrix transposed (see e.g. [**AdjacencyMatrix2020a**, 1, 23]) this unfourtunately includes the `igraph` library [13] but that convention will not be followed in this paper

[2] In this paper $\vec{1}$ refers to a vector containing only values of 1, the size of which should be clear from the context

(b) How quickly will it converge?

(c) Will it converge uniquely?

2. Reducible graphs

   (a) If it is not possible to perform a random walk across an entire graph for all initial conditions, this approach doesn't have a clear analogue.

3. Cycles

   (a) A graph that is cyclical may not converge uniquely

      i. Consider for example the graph $A \to B$.

2.2.2 Markov Chains

The relationship in (3) is a *Markov Chain* and it is known that the power method will converge: [3]

- for a stochastic irreducible markov chain [11, §1.5.5],
- regardless of the initial condition of the process for an *aperiodic* Markov chain [20, §4.4]

Stochastic If a vertex had a 0 outdegree the corresponding column sum for the adjacency matrix describing that graph would also be zero and the matrix non-stochastic, this could occur in the context of a random walk where a link to a page with no outgoing links was followed (e.g. an image), this would be the end of the walk.

So to ensure that (3) will converge, the probability transition matrix must be made stochastic, to acheive this a uniform probability of teleporing from a dead end to any other vertex can be introduced:

$$\mathrm{S} = \mathrm{T} + \frac{\vec{a} \cdot \vec{1}^{\mathrm{T}}}{n} \tag{4}$$

This however would not be sufficient to ensure that (3) would converge, in addition the transition probability matrix must be made irreducible and aperiodic (i.e. primitive). [20]

Irreducible A graph that allows travel from any given vertex to any other vertex is said to be irreducible [20], see for example figure 2, this is important in the context of a random walk because only in an irreducible graph can all vertexes be reached from any initial condition.

Aperiodic An a periodic graph has only one eigenvalue that lies on the unit circle, this is important because $\lim_{k\to\infty} \left(\frac{\mathbf{A}}{r}^k\right)$ exists for a non-negative irreducible matrix $\mathbf{A}$ if and only if A is aperiodic. A graph that is a periodic can be made aperiodic by interlinking nodes [4]

[3] A *Markov Chain* is simply any process that evolves depending on it's current condition, it's interesting to note however that the theory of *Markov Chains* is not mentioned in any of the original papers by page and brin [20, §4.4]

[4] Actually it would be sufficient to merely link one vertex to itself [20, §15.2] but this isn't very illustrative or helpful in this context

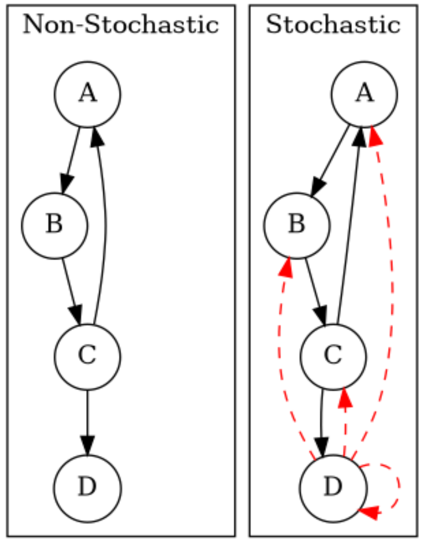


Figure 1: D is a *dangling node*, a dead end during a random walk, the corresponding probability transition matrix ($\mathbf{T}$) is hence non-stochastic (and also reducible), Introducing some probability of teleporting from a dead end to any other vertex as per (4) (denoted in red) will cause $\mathbf{T}$ to be stochastic.

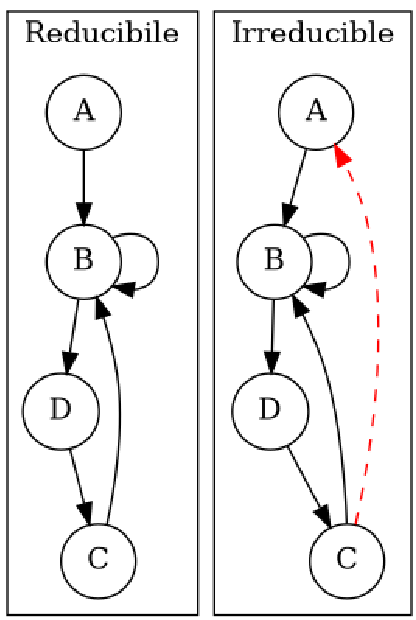


Figure 2: Example of a reducible graph, observe that although C is not a dead end as discussed in 2.2.2 , there is no way to travel from C to A, by adding an edge such an edge in the resulting graph is irreducible. The resulting graph is also aperiodic (due to the loop on B) and stochastic, so there will be a stationary distribution corresponding to (3).

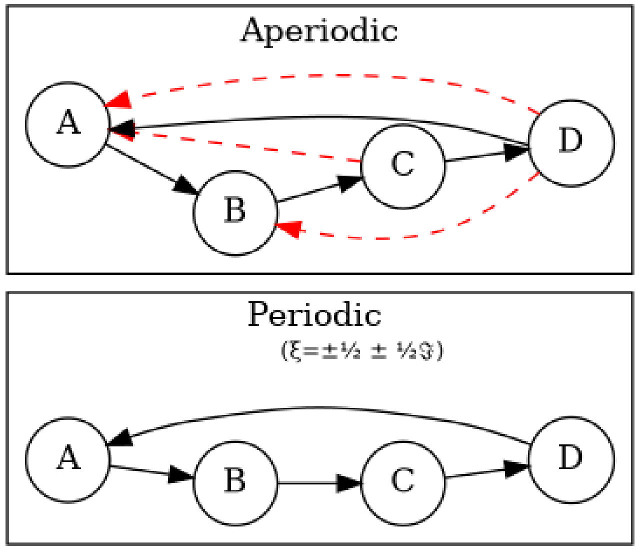


Figure 3: A periodic graph with all eigenvalues on the unit circle $\xi = \frac{\sqrt{2}}{2} e^{\frac{\pi i}{4} k}$, by adding in extra edges the graph is now aperiodic, this does not represent the random surfer model, which would in theory connect every vertex but with some probability.

The Fix To ensure that the transition probability matrix is primitive (i.e. irreducible and aperiodic) as well as stochastic, instead of introducing the possible to teleport out of dead ends, introduce a probability of teleporting to any node at any time (α), this approach is known as the *Random Surfer* model and the transition probability matrix is given by [21] :

$$\mathbf{S} = \alpha \mathbf{T} + \frac{(1-\alpha)}{n} \mathbf{J} \tag{5}$$

This matrix is primitive and stochastic and so will converge (it is also unfourtunately completely dense, see 3.1 [20, §4.5].

The relation ship in (3) can now be re expressed as:

$$\vec{p_{i+1}} \rightarrow \mathbf{T} \vec{p_i} \tag{6}$$

2.2.3 Limitations

The *Random Surfer* Model can only consider positively weighted edges, it cannot take into account negatively weighted edges. This limitation is increasingly important as techniques of sentiment analysis are developed which could indicate that links promote aversion rather than endorsement (e.g. a negative review or an innapropriate advertisement).

2.3 Power walk

The *Power Walk* method is an alternative approach to develop a probability transition matrix to use in place of (3).

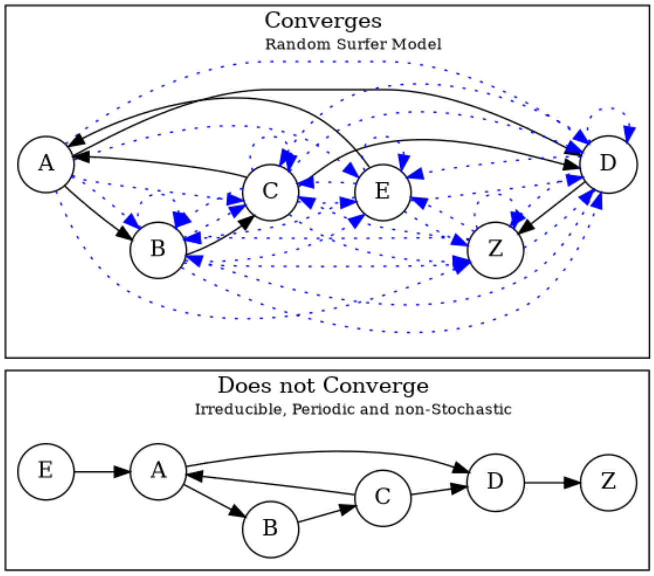


Figure 4: A graph that is aperiodic, reducible and non-stochastic, by applying the random surfer model (5) blue *teleportation* edges are introduced, these may be followed with a probability of $1 - \alpha$

Let the probability of travelling to a non-adjacent vertex be some value x and β be the ratio of probability between following an edge or teleporting to another vertex.

This transition probability matrix would be such that the probability of travelling some vertex $j \to i$ would be :

$$\mathbf{W}_{i,j} = x\beta^{\mathbf{A_{i,j}}} \tag{7}$$

Where $\mathbf{W}$ denotes the power walk probability transition matrix.

Whe probability of travelling to any given vertex must be 1 and so:

$$1 = \sum_{j=1}^{n} \left[x\beta^{\mathbf{A_{i,j}}} \right] \tag{8}$$

$$\implies x = \left(\sum_{j=1}^{n} \beta^{\mathbf{A_{i,j}}} \right)^{-1} \tag{9}$$

Substituting the value of x from (9) into (**??**) gives the probability as:

$$\mathbf{W}_{i,j} = \frac{\beta^{\mathbf{A}_{i,j}}}{\sum_{i=j}^{n} \left[\beta^{\mathbf{A}_{i,j}} \right]} \tag{10}$$

In this model all vertices are interconnected by some probability of jumping to another vertex, so much like the random surfer model (5) discussed at 2.2.2 $\mathbf{W}$ will be a primitive stochastic matrix and so if $\mathbf{W}$ was used in place of $\mathbf{T}$ in (3) a solution would exist.

3 Sparse Matrices

Most Adjacency matrices resulting from webpages and analagous networks result in sparse adjacency matrices (see figure 11), this is a good thing because it requires far less computational resources to work with a sparse matrix than a dense matrix [20, §4.2] .

Sparse matrices can be expressed in alternetive forms so as to reduce the memory footprint associated with that matrix, one such method is the *Compressed Row Storage* method, this involves listing the elements as a table as in (11) and (12).

This is implemented in ***R*** with the `Matrix` package [**batesMatrixSparseDense2019a**] .

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \phi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \pi \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{11}$$

$$\begin{array}{ccc} \text{Row Index} & \text{Col Index} & \text{Value} \\ 1 & 1 & 1 \\ 3 & 2 & \phi \\ 4 & 5 & \pi \end{array} \tag{12}$$

3.1 Solving the Stationary Distribution

The relationship in (3) [5] is equivelant to the eigenvalue value problem, where $\vec{p} = \lim_{i\to\infty} (\vec{p_i})$ is the eigenvector [6] $\vec{x}$ that corresponds to the eigenvalue $\xi = 1$:

$$\vec{p}(1) = \mathbf{S}\vec{p} \tag{13}$$

Solving eigenvectors for large matrices can be very resource intensive and so this approach isn't suitable for analysing large networks.

Upon iteration (3) will converge to stable stationary point, as discussed in 2.2.2 , this approach is known as the power method [22] and is what in practice must be implemented to solve the stationary distribution of (6) and (3).

As mentioned in 2.2.2 and 2.3 , the *Random Surfer* and *Power Walk* transtition probability matrices are completely dense, that means applying the power method will not be able to take advantage of using sparse matrix algorithms.

With some effort however it is possible to express the algorithms in such a way that only involves sparse matrices.

[5] This assumes that the transition probability matrix is stochastic and primitive as it would be for $\mathbf{S}$ and $\mathbf{W}$

[6] More accurately the eigenvector specifically scaled specifically to 1, so it would be more correct to say the eigenvector $\frac{\vec{x}}{\sum \vec{x}}$

4 Implementing the Models

To Implement the models, first they'll be implemented using an ordinary matrix and then improved to work with sparse matrices and algorithms, the implementation has been performed with ***R*** and the preamble is provided in listings 1

```r
  if (require("pacman")) {
      library(pacman)
    }else{
      install.packages("pacman")
      library(pacman)
    }

    pacman::p_load(tidyverse, Matrix, igraph, plotly,
     ↪  mise, docstring, mise, corrplot, latex2exp)
#   options(scipen=20) # Resist Scientific Notation
```

Listing 1: Implemented Packages used in this report

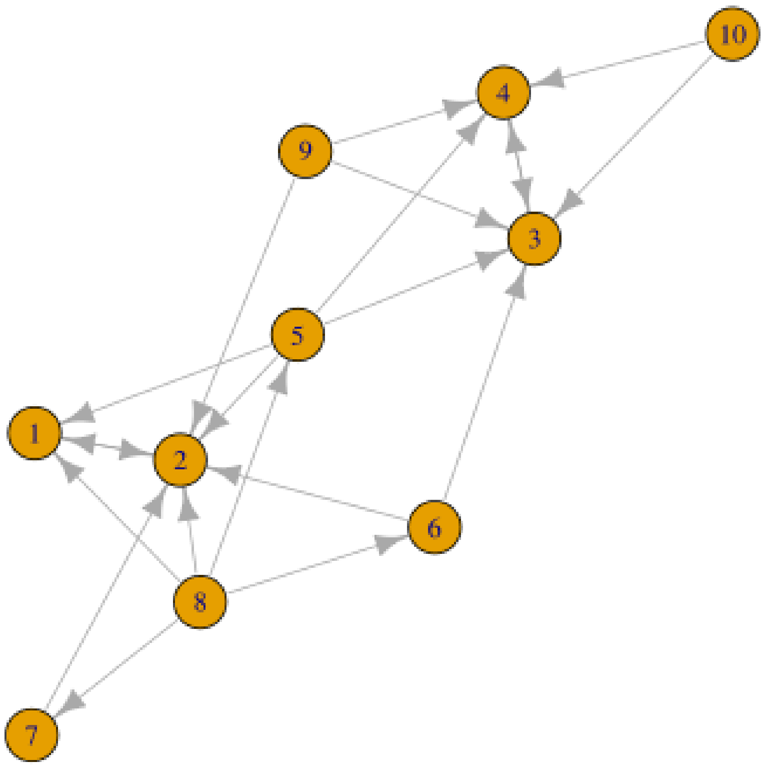
. .

4.0.1 Example Graph

Consider the following graph:

```r
g1 <- igraph::graph.formula(
                1++2, 1+-8, 1+-5,
                2+-5, 2+-7, 2+-8, 2+-6, 2+-9,
                3++4, 3+-5, 3+-6, 3+-9, 3+-10,
                4+-9, 4+-10, 4+-5,
                5+-8, 6+-8, 7+-8)
plot(g1)
```

Listing 2: Produce exemplar graph in figure 5



4.1 Implementing the Random Surfer

4.1.1 Ordinary Matrices

Adjacency Matrix The adjacency Matrix is given by:

```r
A <- igraph::get.adjacency(g1, names = TRUE, sparse = FALSE)

## igraph gives back the transpose
(A <- t(A))
```

Listing 3: Return the Adjacency Matrix corresponding to figure 5

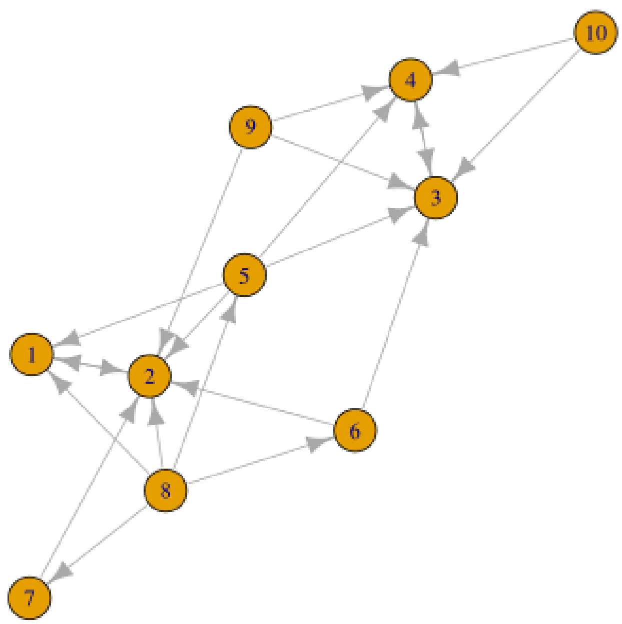


Figure 5: Exemplar graph for *PageRank* examples, produced in listing 2

```
   1 2 8 5 7 6 9 3 4 10
1  0 1 1 1 0 0 0 0 0  0
2  1 0 1 1 1 1 1 0 0  0
8  0 0 0 0 0 0 0 0 0  0
5  0 0 1 0 0 0 0 0 0  0
7  0 0 1 0 0 0 0 0 0  0
6  0 0 1 0 0 0 0 0 0  0
9  0 0 0 0 0 0 0 0 0  0
3  0 0 0 1 0 1 1 0 1  1
4  0 0 0 1 0 0 1 1 0  1
10 0 0 0 0 0 0 0 0 0  0
```

```
   1 2 8 5 7 6 9 3 4 10
1  0 1 1 1 0 0 0 0 0  0
2  1 0 1 1 1 1 1 0 0  0
8  0 0 0 0 0 0 0 0 0  0
5  0 0 1 0 0 0 0 0 0  0
7  0 0 1 0 0 0 0 0 0  0
6  0 0 1 0 0 0 0 0 0  0
9  0 0 0 0 0 0 0 0 0  0
3  0 0 0 1 0 1 1 0 1  1
4  0 0 0 1 0 0 1 1 0  1
10 0 0 0 0 0 0 0 0 0  0
```

Probability Transition Matrix The probability transition matrix is such that each column of the initial state distribution (i.e. the transposed adjacency matrix) is scaled to 1.

if $\mathbf{A}$ had vertices with a 0 out-degree, the relationship in (1) would not work, instead columns that sum to 0 would need to be left while all other columns be divided by the column sum to get $\mathbf{T}$. An alternative approach using sparse matrices will be presented below and in this case there exists corresponding $\mathbf{T}$ that is stochastic and so it is sufficient to use the relationship at (1), this is shown in listing 4.

```r
(T <- A %*% diag(1/colSums(A)))
```

Listing 4: Solve the Transition Probability Matrix by scaling each column to 1 using matrix multiplication.

```
   [,1] [,2] [,3] [,4] [,5] [,6]      [,7] [,8] [,9] [,10]
1     0    1  0.2 0.25    0  0.0 0.0000000    0    0   0.0
2     1    0  0.2 0.25    1  0.5 0.3333333    0    0   0.0
8     0    0  0.0 0.00    0  0.0 0.0000000    0    0   0.0
5     0    0  0.2 0.00    0  0.0 0.0000000    0    0   0.0
7     0    0  0.2 0.00    0  0.0 0.0000000    0    0   0.0
6     0    0  0.2 0.00    0  0.0 0.0000000    0    0   0.0
9     0    0  0.0 0.00    0  0.0 0.0000000    0    0   0.0
3     0    0  0.0 0.25    0  0.5 0.3333333    0    1   0.5
4     0    0  0.0 0.25    0  0.0 0.3333333    1    0   0.5
10    0    0  0.0 0.00    0  0.0 0.0000000    0    0   0.0
```

Create a Function

```r
adj_to_probTrans <- function(A) {
  A %*% diag(1/colSums(A))
}

(T <- adj_to_probTrans(A)) %>% round(2)
```

```
   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
1     0    1  0.2 0.25    0  0.0 0.00    0    0   0.0
2     1    0  0.2 0.25    1  0.5 0.33    0    0   0.0
8     0    0  0.0 0.00    0  0.0 0.00    0    0   0.0
5     0    0  0.2 0.00    0  0.0 0.00    0    0   0.0
7     0    0  0.2 0.00    0  0.0 0.00    0    0   0.0
6     0    0  0.2 0.00    0  0.0 0.00    0    0   0.0
9     0    0  0.0 0.00    0  0.0 0.00    0    0   0.0
3     0    0  0.0 0.25    0  0.5 0.33    0    1   0.5
4     0    0  0.0 0.25    0  0.0 0.33    1    0   0.5
10    0    0  0.0 0.00    0  0.0 0.00    0    0   0.0
```

```
##    [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## 1     0    1    0    0 0.25  0.0    0  0.2 0.00   0.0
## 2     1    0    0    0 0.25  0.5    1  0.2 0.33   0.0
## 3     0    0    0    1 0.25  0.5    0  0.0 0.33   0.5
## 4     0    0    1    0 0.25  0.0    0  0.0 0.33   0.5
## 5     0    0    0    0 0.00  0.0    0  0.2 0.00   0.0
## 6     0    0    0    0 0.00  0.0    0  0.2 0.00   0.0
## 7     0    0    0    0 0.00  0.0    0  0.2 0.00   0.0
## 8     0    0    0    0 0.00  0.0    0  0.0 0.00   0.0
## 9     0    0    0    0 0.00  0.0    0  0.0 0.00   0.0
## 10    0    0    0    0 0.00  0.0    0  0.0 0.00   0.0
```

Page Rank Random Surfer Recall from 2.2.2 the following variables of the *Random Surfer* model:

$$\mathbf{B} = \alpha T + (1-\alpha) B : \tag{14}$$

$$\tag{15}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \\ \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n} & \cdots & \frac{1}{n} \end{bmatrix} \tag{16}$$

$$n = ||V|| \tag{17}$$

$$\alpha \in [0,1] \tag{18}$$

These are assigned to ***R*** variables in listing 5.

```r
B <- matrix(rep(1/nrow(T), length.out =
l <- 0.8123456789


(S <- l*T+(1-l)*B) %>% round(2)
```

Listing 5: Assign Random Surfer Variables, observe the unique value given to `l`, this will be relevant later.

```
   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
1  0.02 0.83 0.18 0.22 0.02 0.02 0.02 0.02 0.02  0.02
2  0.83 0.02 0.18 0.22 0.83 0.42 0.29 0.02 0.02  0.02
8  0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02  0.02
5  0.02 0.02 0.18 0.02 0.02 0.02 0.02 0.02 0.02  0.02
7  0.02 0.02 0.18 0.02 0.02 0.02 0.02 0.02 0.02  0.02
6  0.02 0.02 0.18 0.02 0.02 0.02 0.02 0.02 0.02  0.02
9  0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02  0.02
3  0.02 0.02 0.02 0.22 0.02 0.42 0.29 0.02 0.83  0.42
4  0.02 0.02 0.02 0.22 0.02 0.02 0.29 0.83 0.02  0.42
10 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02  0.02
```

Eigen Value Method The eigenvector corresponding to the the eigenvalue of 1 will be the stationary point, this is shown in listing 6

```r
print(eigen(S, symmetric = FALSE, only.values =
 ↪  TRUE)$values, 9)
print(eigen(S, symmetric = FALSE)$vectors, 3)
```

Listing 6: Solve the Eigen vectors and Eigen values of the transition probability matrix corresponding to the graph.

```
 [1]  1.00000000e+00+0.0000000e+00i -8.12345679e-01+0.0000000e+00i
 [3]  8.12345679e-01+0.0000000e+00i -8.12345679e-01+0.0000000e+00i
 [5]  5.81488197e-10+0.0000000e+00i -5.81487610e-10+0.0000000e+00i
 [7] -6.74980227e-16+0.0000000e+00i  3.21036747e-17+0.0000000e+00i
 [9]  1.34928172e-18+1.1137323e-17i  1.34928172e-18-1.1137323e-17i
            [,1]         [,2]         [,3]         [,4]         [,5]
 [1,] 0.4873+0i -7.07e-01+0i  5.00e-01+0i -2.07e-03+0i -6.74e-01+0i
 [2,] 0.5268+0i  7.07e-01+0i  5.00e-01+0i  2.07e-03+0i -9.62e-02+0i
 [3,] 0.0424+0i  9.09e-18+0i -3.50e-17+0i -5.05e-17+0i  1.38e-09+0i
 [4,] 0.0493+0i -1.25e-18+0i -1.65e-16+0i  4.25e-17+0i  3.85e-01+0i
 [5,] 0.0493+0i -8.30e-18+0i -3.75e-17+0i  3.71e-17+0i  3.85e-01+0i
 [6,] 0.0493+0i -8.30e-18+0i -3.75e-17+0i  9.76e-18+0i  3.85e-01+0i
```

```
 [7,] 0.0424+0i -1.32e-18+0i -3.50e-17+0i  1.60e-17+0i -3.01e-08+0i
 [8,] 0.4915+0i -2.98e-03+0i -5.00e-01+0i -7.07e-01+0i -9.62e-02+0i
 [9,] 0.4804+0i  2.98e-03+0i -5.00e-01+0i  7.07e-01+0i -2.89e-01+0i
[10,] 0.0424+0i  5.57e-18+0i -3.77e-17+0i  3.14e-18+0i -3.24e-08+0i
              [,6]         [,7]         [,8]                   [,9]
 [1,]  6.74e-01+0i  6.53e-01+0i -2.15e-01+0i -2.00e-01+1.53e-01i
 [2,]  9.62e-02+0i  1.09e-01+0i -1.96e-01+0i -1.59e-01+0.00e+00i
 [3,]  1.38e-09+0i  1.42e-15+0i -2.84e-16+0i -6.73e-17+1.32e-16i
 [4,] -3.85e-01+0i -4.37e-01+0i  7.85e-01+0i  6.37e-01+0.00e+00i
 [5,] -3.85e-01+0i -3.56e-01+0i  2.81e-01+0i  2.84e-02-1.63e-01i
 [6,] -3.85e-01+0i -3.58e-01+0i -3.68e-01+0i  4.84e-02-2.68e-01i
 [7,] -3.01e-08+0i -2.63e-02+0i -2.34e-01+0i -3.47e-02+4.29e-01i
 [8,]  9.62e-02+0i  1.32e-01+0i -6.40e-02+0i -1.09e-01-2.84e-01i
 [9,]  2.89e-01+0i  3.11e-01+0i  1.20e-01+0i -1.34e-01-1.50e-01i
[10,] -3.24e-08+0i -2.82e-02+0i -1.08e-01+0i -7.64e-02+2.83e-01i
                    [,10]
 [1,] -2.00e-01-1.53e-01i
 [2,] -1.59e-01-0.00e+00i
 [3,] -6.73e-17-1.32e-16i
 [4,]  6.37e-01+0.00e+00i
 [5,]  2.84e-02+1.63e-01i
 [6,]  4.84e-02+2.68e-01i
 [7,] -3.47e-02-4.29e-01i
 [8,] -1.09e-01+2.84e-01i
 [9,] -1.34e-01+1.50e-01i
[10,] -7.64e-02-2.83e-01i
```

So in this case the stationary point corresponds to the eigenvector given by:

$$\langle -0.49, -0.53, -0.49, -0.48, -0.05, -0.05, -0.05, -0.04, -0.04, -0.04 \rangle$$

this can be verified by using identity (13):

$$1\vec{p} = S\vec{p}$$

```
(p      <- eigen(S)$values[1] *
  eigen(S)$vectors[,1]) %>% Re() %>%  round(2)
```

```
[1] 0.49 0.53 0.04 0.05 0.05 0.05 0.04 0.49 0.48 0.04
```

```
(p_new <- S %*% p) %>% Re()  %>% as.vector() %>%
  round(2)
```

```
[1] 0.49 0.53 0.04 0.05 0.05 0.05 0.04 0.49 0.48 0.04
```

However this vector does not sum to 1 so the scale should be adjusted (for probabilities the vector should sum to 1):

```r
(p_new <- p_new/sum(p_new)) %>% Re() %>%
  as.vector() %>% round(2)
```

```
[1] 0.22 0.23 0.02 0.02 0.02 0.02 0.02 0.22 0.21 0.02
```

Power Value Method Using the power method should give the same result as the eigenvalue method, again but for scale:

```r
p_new <- p_new *123456789

while (sum(round(p, 9) != round(p_new, 9))) {
    (p     <- p_new)
    (p_new <- S %*% p)
}

round(Re(p_new), 2) %>% as.vector()
```

```
[1] 26602900 28759738  2316720  2693115  2693115  2693115  2316720 26834105
[9] 26230539  2316720
```

If scaled to 1 the same value will be returned:

```r
(p_new <- p_new/sum(p_new)) %>% Re %>% as.vector()
  %>% round(2)
```

```
[1] 0.22 0.23 0.02 0.02 0.02 0.02 0.02 0.22 0.21 0.02
```

Scaling If the initial state sums to 1, then the scale of the stationary vector will also sum to 1, so this isn't in practice an issue for the power method:

```r
p     <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
p_new <- S %*% p

while (sum(round(p, 9) != round(p_new, 9))) {
    (p     <- p_new)
    (p_new <- S %*% p)
}

cbind(p_new, p)
```

```
        [,1]       [,2]
1  0.21548349 0.21548349
2  0.23295388 0.23295388
```

```
8  0.01876543 0.01876543
5  0.02181424 0.02181424
7  0.02181424 0.02181424
6  0.02181424 0.02181424
9  0.01876543 0.01876543
3  0.21735625 0.21735625
4  0.21246737 0.21246737
10 0.01876543 0.01876543
```

4.1.2 Sparse Matrices

Creating the Probability Transition Matrix Implementing the page rank method on a larger graph requires the use of more efficient form of matrix storage as discussed at 3

A sparse matrix can be created using the following syntax, which will return a matrix of the class `dgCMatrix`:

```r
library(Matrix)
## Create Example Matrix
n <- 20
m <- 10^6
i <- sample(1:m, size = n); j <- sample(1:m, size =
↪  n); x <- rpois(n, lambda = 90)
A <- sparseMatrix(i, j, x = x, dims = c(m, m))

summary(A)
```

```
1000000 x 1000000 sparse Matrix of class "dgCMatrix", with 20 entries
        i      j   x
1  832961  14530  77
2  410264  57606  97
3  782033 111998  86
4   82383 176945  93
5  110039 239517 103
6  713327 249015  98
7    3377 387382  87
8  183673 466594  90
9  459326 509037  98
10 360156 554024  91
11 697837 573216 106
12 460554 582729  80
13 353957 654474  87
14 941579 683010 108
15 955791 763690 104
16 726278 790608  85
17 317527 867693  90
18  71267 949427  81
19 126551 992218  96
```

```
20 723320 992960   84
```

As before in section 4.1.1 , the probability transition matrix can be found by:

1. Creating adjacency matrix
   (a) Transposing as necessary such that $\mathbf{A}_{i,j} \neq 0$ indicates that j is connected to i by a directed edge.
2. Scaling the columns to one

To implement this for a sparseMatrix of the class `dgCMatrix`, the same technique of multiplying by a diagonalised matrix as in (2) may be implemented, using sparse matrices has the advantage however that only non-zero elements will be operated on, meaning that columns that some to zero can still be used to create a probability transition matrix [7] pracice an error however to create this new matrix, a new `sparseMatrix` will need to be created using the properties of the original matrix, this can be done like so:

```r
sparse_diag <- function(mat) {

 ## Get the Dimensions
 n <- nrow(mat)

 ## Make a Diagonal Matrix of Column Sums
 D <- sparseMatrix(i = 1:n, j = 1:n, x =
  ↪  colSums(mat), dims = c(n,n))

 ## Throw away explicit Zeroes
 D <- drop0(D)

 ## Inverse the Values
 D@x <- 1/D@x

 ## Return the Diagonal Matrix
 return(D)
}
```

Applying this to the previously created sparse matrix:

```r
D <- sparse_diag(t(A))
summary(D)
```

```
1000000 x 1000000 sparse Matrix of class "dgCMatrix", with 20 entries
```

[7] Although this matrix may still have columns that sum to zero and will hence be non-stochastic

```
        i      j           x
1    3377   3377 0.011494253
2   71267  71267 0.012345679
3   82383  82383 0.010752688
4  110039 110039 0.009708738
5  126551 126551 0.010416667
6  183673 183673 0.011111111
7  317527 317527 0.011111111
8  353957 353957 0.011494253
9  360156 360156 0.010989011
10 410264 410264 0.010309278
11 459326 459326 0.010204082
12 460554 460554 0.012500000
13 697837 697837 0.009433962
14 713327 713327 0.010204082
15 723320 723320 0.011904762
16 726278 726278 0.011764706
17 782033 782033 0.011627907
18 832961 832961 0.012987013
19 941579 941579 0.009259259
20 955791 955791 0.009615385
```

and hence the probability transition matrix may be implemented by performing matrix multiplication accordingly:

```r
summary((T <- t(A) %*% D))
```

```
1000000 x 1000000 sparse Matrix of class "dgCMatrix", with 20 entries
        i      j x
1  387382   3377 1
2  949427  71267 1
3  176945  82383 1
4  239517 110039 1
5  992218 126551 1
6  466594 183673 1
7  867693 317527 1
8  654474 353957 1
9  554024 360156 1
10  57606 410264 1
11 509037 459326 1
12 582729 460554 1
13 573216 697837 1
14 249015 713327 1
15 992960 723320 1
16 790608 726278 1
17 111998 782033 1
18  14530 832961 1
```

```
19 683010 941579 1
20 763690 955791 1
```

Solving the Random Surfer via the Power Method Solving the eigenvalues for such a large matrix will not feasible, instead the power method will need to be used to find the stationary point.

However, creating a matrix of background probabilites (denoted by `B` in section 4.1.1) will not be feasible, it would simply be too large, instead some algebra can be used to reduce B from a matrix into a vector containing only $\frac{1-\alpha}{N}$.

The power method is given by:

$$\vec{p} = \mathbf{S}\vec{p} \tag{19}$$

where:

$$\begin{aligned} S &= \alpha \mathbf{T} + (1-\alpha)\,\mathbf{B} && (20) \\ \vec{p} &= (\alpha \mathbf{T} + (1-\alpha)\,\mathbf{B})\,\vec{p} && (21) \\ &= \alpha \mathbf{T}\vec{p} + (1-\alpha)\,\mathbf{B}\vec{p} && (22) \end{aligned}$$

Let $\mathbf{F} = \mathbf{B}\vec{p}$, consider the value of $\mathbf{F}$:

$$\begin{aligned} \mathbf{F} &= \begin{bmatrix} \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \end{bmatrix} \begin{bmatrix} \vec{p_1} \\ \vec{p_2} \\ \vdots \\ \vec{p_m} \end{bmatrix} && (23) \\ &= \begin{bmatrix} \left(\sum_{i=0}^{m} [p_i]\right) \times \frac{1}{N} \\ \left(\sum_{i=0}^{m} [p_i]\right) \times \frac{1}{N} \\ \vdots \\ \left(\sum_{i=0}^{m} [p_i]\right) \times \frac{1}{N} \end{bmatrix} && (24) \\ & \text{Probabilities sum to 1 and hence:} && (25) \\ &= \begin{bmatrix} \frac{1}{N} \\ \frac{1}{N} \\ \frac{1}{N} \\ \vdots \\ \frac{1}{N} \end{bmatrix} && (26) \end{aligned}$$

So instead the power method can be implemented by performing an algorithm that involves only sparse matrices:

```r
## Find Stationary point of random surfer
N     <- nrow(A)
alpha <- 0.85
F     <- rep((1-alpha)/N, nrow(A))  ## A nx1 vector
 ↪  of (1-alpha)/N

```

```r
## Solve using the power method
p      <- rep(0, length.out = ncol(T)); p[1] <- 1
p_new <- alpha*T %*% p + F

## use a Counter to debug
i <- 0
while (sum(round(p, 9) != round(p_new, 9))) {
    p      <- p_new
    p_new <- alpha*T %*% p + F
    (i <- i+1) %>% print()
}

p %>% head() %>% print()
```

```
[1] 1
[1] 2
6 x 1 Matrix of class "dgeMatrix"
        [,1]
[1,] 1.5e-07
[2,] 1.5e-07
[3,] 1.5e-07
[4,] 1.5e-07
[5,] 1.5e-07
[6,] 1.5e-07
```

4.2 Power Walk Method

Recall from 2.3 that the power walk is given by:

$$\mathbf{T} = \mathbf{B}\mathbf{D}_B^{-1}$$

4.2.1 Ordinary Matrices

Implementing the Power walk using ordinary matrices is very similar to the *Random Surfer* model be done pretty much the same as it is with the random surfer, but doing it with Sparse Matrices is a bit trickier.

Create the Adjacency Matrix

```r
  A <- igraph::get.adjacency(g1, names = TRUE, sparse
  ↪  = FALSE)

## * Function to create Prob Trans Mat
adj_to_probTrans <- function(A, beta) {
    B      <- A
    B      <- beta^A            # Element Wise
     ↪  exponentiation
```

```r
    D      <- diag(colSums(B)) # B is completely dense
    ↪  so D   0
    D_in   <- solve(D)         # Solve returns inverse
    ↪  of matrix
    W      <- B %*% D_in

    return(as.matrix(W))
}

beta <-   <- 0.867
(W <- adj_to_probTrans(A, beta =  )) %>% round(2)
```

```
   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
1  0.10 0.09  0.1 0.10 0.10 0.10  0.1 0.11 0.11   0.1
2  0.09 0.11  0.1 0.10 0.10 0.10  0.1 0.11 0.11   0.1
8  0.09 0.09  0.1 0.09 0.09 0.09  0.1 0.11 0.11   0.1
5  0.09 0.09  0.1 0.10 0.10 0.10  0.1 0.09 0.09   0.1
7  0.10 0.09  0.1 0.10 0.10 0.10  0.1 0.11 0.11   0.1
6  0.10 0.09  0.1 0.10 0.10 0.10  0.1 0.09 0.11   0.1
9  0.10 0.09  0.1 0.10 0.10 0.10  0.1 0.09 0.09   0.1
3  0.10 0.11  0.1 0.10 0.10 0.10  0.1 0.11 0.09   0.1
4  0.10 0.11  0.1 0.10 0.10 0.10  0.1 0.09 0.11   0.1
10 0.10 0.11  0.1 0.10 0.10 0.10  0.1 0.09 0.09   0.1
```

Look at the Eigenvalues:

```r
eigen(W, only.values = TRUE)$values %>% round(9)
eigen(W)$vectors/sum(eigen(W)$vectors)
```

```
 [1]  1.000000000+0.000000000i  0.014269902+0.000000000i
 [3] -0.014148391+0.000000000i  0.014147087+0.000000000i
 [5]  0.007672842+0.004095136i  0.007672842-0.004095136i
 [7]  0.000000000+0.000000000i  0.000000000+0.000000000i
 [9]  0.000000000+0.000000000i  0.000000000+0.000000000i
             [,1]             [,2]            [,3]            [,4]
 [1,] 0.10153165+0i  5.107247e-02+0i  0.073531664+0i  0.009918277+0i
 [2,] 0.10159353+0i -1.161249e-01+0i  0.071987451+0i -0.009531974+0i
 [3,] 0.09609664+0i -2.162636e-01+0i  0.198568750+0i  0.141245296+0i
 [4,] 0.09725145+0i  6.794340e-02+0i -0.012230606+0i -0.001148014+0i
 [5,] 0.10153165+0i  5.107247e-02+0i  0.073531664+0i  0.009918277+0i
 [6,] 0.10008449+0i  1.115133e-01+0i -0.005625969+0i -0.156796770+0i
 [7,] 0.09865794+0i  1.175228e-01+0i -0.084225633+0i  0.008563891+0i
 [8,] 0.10157348+0i -6.053608e-02+0i -0.078607240+0i  0.165540590+0i
 [9,] 0.10155286+0i -6.104664e-03+0i -0.079165209+0i -0.166535117+0i
[10,] 0.10012631+0i -9.522175e-05+0i -0.157764873+0i -0.001174456+0i
```

```
                        [,5]                     [,6]              [,7]
 [1,]  0.00633946+0.04208220i  0.00633946-0.04208220i  3.014602e-16+0i
 [2,]  0.00757768+0.03910216i  0.00757768-0.03910216i  1.909248e-16+0i
 [3,]  0.22697603+0.00000000i  0.22697603+0.00000000i  3.985744e-02+0i
 [4,] -0.11628681-0.11808928i -0.11628681+0.11808928i -2.471407e-01+0i
 [5,]  0.00633946+0.04208220i  0.00633946-0.04208220i  7.520823e-02+0i
 [6,] -0.03494625-0.01031801i -0.03494625+0.01031801i  1.719325e-01+0i
 [7,] -0.07581902-0.06371153i -0.07581902+0.06371153i  6.131013e-03+0i
 [8,]  0.00717270+0.04008639i  0.00717270-0.04008639i  5.526770e-17+0i
 [9,]  0.00675977+0.04107970i  0.00675977-0.04107970i  1.105354e-16+0i
[10,] -0.03411300-0.01231382i -0.03411300+0.01231382i -4.598845e-02+0i
                  [,8]             [,9]            [,10]
 [1,] -1.791605e-17+0i -4.365749e-17+0i  1.179767e-17+0i
 [2,] -7.334385e-17+0i -8.731498e-17+0i -5.190977e-17+0i
 [3,] -1.241234e-01+0i -1.401965e-01+0i -8.894098e-02+0i
 [4,]  1.691000e-01+0i  1.687523e-01+0i  1.041947e-01+0i
 [5,] -2.144546e-01+0i  2.715852e-02+0i  3.085359e-02+0i
 [6,]  4.535455e-02+0i -1.959109e-01+0i -1.350483e-01+0i
 [7,]  7.398187e-02+0i  3.163948e-02+0i -1.260060e-01+0i
 [8,]  8.062225e-17+0i  3.638124e-17+0i  5.898837e-18+0i
 [9,]  2.687408e-17+0i  3.638124e-17+0i  5.662884e-17+0i
[10,]  5.014155e-02+0i  1.085570e-01+0i  2.149470e-01+0i
```

Unlike the *Random Surfer* Model in listing 6 at 4.1.1 the relationship between the second eigenvalue and the model parameters is not as clear, this provides that the

Use the power method

```r
## * Power Method
p     <- rep(0, nrow(W))
p[1] <- 1
p_new     <- rep(0, nrow(W))
p_new[2]     <- 1

while (sum(round(p, 9) != round(p_new, 9))) {
    (p      <- p_new)
    (p_new <- W %*% p)
}


p %>% as.vector()
```

```
[1] 0.10153165 0.10159353 0.09609664 0.09725145 0.10153165 0.10008449
[7] 0.09865794 0.10157348 0.10155286 0.10012631
```

4.2.2 Sparse Matrices

Theory; Simplifying Power Walk to be solved with Sparse Matrices The Random Surfer model is:

$$\mathbf{S} = \alpha \mathbf{T} + \mathbf{F}$$

where:

- $\mathbf{T}$
  - is an $i \times j$ matrix that describes the probability of travelling from vertex j to i
    - This is transpose from the way that `igraph` produces an adjacency matrix.
- $\mathbf{F} = \begin{bmatrix} \frac{1}{n} \\ \frac{1}{n} \\ \frac{1}{n} \vdots \end{bmatrix}$

Interpreting the transition probability matrix in this way is such that $\mathbf{T} = \mathbf{A}\mathbf{D}_A^{-1}$ under the following conditions:

- No column of $\mathbf{A}$ sums to zero
  - If this does happen the question arises how to deal with $\mathbf{D_A^{-1}}$
    - I've been doing $\mathbf{D}^{\mathrm{T}}_{\mathbf{A},i,j} := \mathtt{diag}\left(\frac{1}{\mathtt{colsums}(\mathbf{A})}\right)$ and then replacing any 0 on the diagonal with 1.
  - What is done in the paper is to make another matrix $\mathbf{Z}$ that is filled with 0, if a column sum of $\mathbf{A}$ adds to zero then that column in $\mathbf{Z}$ becomes $\frac{1}{n}$
    - This has the effect of making each row identical
    - The probability of going from an orphaned vertex to any other vertex would hence be $\frac{1}{n}$
    - The idea with this method is then to use $D^{-1}_{(\mathbf{A}+\mathbf{Z})}$ this will be consistent with the *Random Surfer* the method using $\mathbf{F}$ in [[#eq:sparse-RS][]] (4.2.2)

  where each row is identical that is a 0

The way to deal with the *Power Walk* is more or less the same.
observe that:

$$\left(\mathbf{B} = \beta^{\mathbf{A}}\right) \land (\mathbf{A}_{i,j}) \in \mathbb{R} \implies |\mathbf{B}_{i,j}| > 0 \quad \forall i,j > n \in \mathbb{Z}^+ \tag{27}$$

Be mindful that the use of exponentiation in (27) is not an element wise exponentiation and not an actual matrix exponential.
So if I have:

- $\mathbf{O}_{i,j} := 0, \quad \forall i,j \leq n \in \mathbb{Z}^+$
- $\vec{p_i}$ as the state distribution, being a vector of length n

Then It can be shown (see (4.2.2) at 4.2.2):

$$
\begin{aligned}
\mathbf{O}\mathbf{D}_{\mathbf{B}}^{-1}\vec{p_i} &= (\overrightarrow{\delta^T}\overrightarrow{p_i})\vec{1} && (28)\\
&= \texttt{repeat}\left(\vec{p} \bullet \vec{\delta}^{\mathrm{T}}, \texttt{n}\right) && (29)\\
& && (30)
\end{aligned}
$$

where:

- $\vec{\delta_i} = \frac{1}{\texttt{colsums}(\mathbf{B})}$
  - A vector...($n \times 1$ matrix)

$\vec{1}$ is a vector containing all 1's

  - A vector...($n \times 1$ matrix)

$\vec{\delta}^{\mathrm{T}}$ refers to the transpoxe of $\vec{}$($1 \times n$ matrix)

$\vec{\delta}^{\mathrm{T}}\vec{p_i}$ is some number (because it's a dot product)

This means we can do:

$$
\begin{aligned}
\overrightarrow{p_{i+1}} &= \mathbf{T}_{\mathrm{pw}}\overrightarrow{p_i} && (31)\\
&= \mathbf{B}\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} && (32)\\
&= (\mathbf{B} - \mathbf{O} + \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} && (33)\\
&= \left((\mathbf{B} - \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1} + \mathbf{O}\mathbf{D}_{\mathbf{B}}^{-1}\right)\overrightarrow{p_i} && (34)\\
&= (\mathbf{B} - \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} + \mathbf{O}\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} && (35)\\
&= (\mathbf{B} - \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} + \vec{1}(\overrightarrow{\delta^{\mathrm{T}}}\overrightarrow{p_i}) && (36)\\
&= (\mathbf{B} - \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} + \texttt{rep}(\overrightarrow{\delta^{\mathrm{T}}}\overrightarrow{p_i}) && (37)
\end{aligned}
$$

where:
Let $(\mathbf{B} - \mathbf{O}) = \mathbf{B}_{\mathbf{O}}$:

$$
\overrightarrow{p_{i+1}} \quad = \mathbf{B}_{\mathbf{O}}\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} + \texttt{rep}(\overrightarrow{\delta^{\mathrm{T}}}\overrightarrow{p_i})
$$

Now solve D_B^{-1} in terms of $\mathbf{B}_{\mathbf{O}}$:

$$
\begin{aligned}
\mathbf{B}_{\mathbf{O}} =& (\mathbf{B} - \mathbf{O}) && (38)\\
\mathbf{B} =& \mathbf{B}_{\mathbf{O}} + \mathbf{O} && (39)
\end{aligned}
$$

If we have $\delta_{\mathbf{B}}$ as the column sums of $\mathbf{B}$:

$$
\begin{aligned}
\delta_{\mathbf{B}}^{-1} &= \vec{1}\mathbf{B} && (40)\\
&= \vec{1}\left(\mathbf{B_O} + \mathbf{O}\right) && (41)\\
&= \vec{1}\mathbf{B_O} + \vec{1}\mathbf{O} && (42)\\
&= \vec{1}\mathbf{B_O} + \langle n, n, n, ...n\rangle && (43)\\
&= \vec{1}\mathbf{B_O} + \vec{1}n && (44)\\
\delta_{\mathbf{B}} &= \mathtt{1}/(\mathtt{colSums}(\mathbf{B_O}) + \mathtt{n}) && (45)
\end{aligned}
$$

Then if we have $D_B = \mathtt{diag}(\delta_B)$:

$$
\begin{aligned}
D_B^{-1} &= \operatorname{diag}\left(\delta_{\mathbf{B}}^{-1}\right)\\
&= \mathtt{diag}\left(\mathtt{ColSums}(\mathtt{B_O}) + \mathtt{n}\right)^{-1}
\end{aligned}
$$

And so the the power method can be implemented using sparse matrices:

$$
\vec{p_{i+1}} = \mathrm{B_O}\ \operatorname{diag}\left(\vec{1}\mathbf{B_O} + \vec{1}n\right)\vec{p_i} + \vec{1}\vec{\delta}^{\mathrm{T}}\vec{p_i} \tag{46}
$$

in terms of ***R***:

```
p_new <- Bo %*% diag(colSums(B)+n) %*% p + rep(t( )
↪  %*% p, n)

# It would also be possible to sum the element-wise
↪  product
(t( ) %*% p) == sum(  * p)

# Because R treats vectors the same as a nX1 matrix
↪  we could also
# perform the dot product of the two vectors, meaning
↪  the following
# would be true in R but not true generally

(t( ) %*% p) == (  %*% p)
```

Solving the Background Probability In this case a vertical single column matrix will represent a vector and $\otimes$ will represent the outer product (i.e. the *Kronecker Product*):

Define $\vec{\delta}$ as the column sums of

$$
\begin{aligned}
\vec{\delta} &= \mathtt{colsum}(\mathbf{B})^{-1}\\
&= \frac{1}{\vec{1^T}\boldsymbol{B}}
\end{aligned}
$$

Then we have:

$$
\begin{aligned}
\mathbf{O}\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} &= \begin{pmatrix} 1 & 1 & 1 & & \\ 1 & 1 & 1 & \dots & \\ 1 & 1 & 1 & & \\ & \vdots & & \ddots & \end{pmatrix} \begin{pmatrix} \frac{1}{\delta_1} & 0 & 0 & & \\ 0 & \frac{1}{\delta_2} & 0 & \dots & \\ 0 & 0 & \frac{1}{\delta_{13}} & & \\ & \vdots & & \ddots & \end{pmatrix} \begin{pmatrix} p_{i,1} \\ p_{i,2} \\ p_{i,3} \\ \vdots \end{pmatrix} \\
&= \begin{pmatrix} \frac{p_{i,1}}{\delta 1} & + & \frac{p_{i,2}}{\delta_2} & + & \frac{p_{i,3}}{\delta_3} & \\ \frac{p_{i,1}}{\delta 1} & + & \frac{p_{i,2}}{\delta_2} & + & \frac{p_{i,3}}{\delta_3} & \dots \\ \frac{p_{i,1}}{\delta 1} & + & \frac{p_{i,2}}{\delta_2} & + & \frac{p_{i,3}}{\delta_3} & \\ & & \vdots & & & \ddots \end{pmatrix} \\
&= \begin{pmatrix} \sum_{k=1}^{n} [p_{i,k}\delta_i] \\ \sum_{k=1}^{n} [p_{i,k}\delta_i] \\ \sum_{k=1}^{n} [p_{i,k}\delta_i] \\ \vdots \end{pmatrix} \\
&= \begin{pmatrix} \overrightarrow{\delta^T}\overrightarrow{p_i} \\ \overrightarrow{\delta^T}\vec{p}_i \\ \overrightarrow{\delta^T}\vec{p}_i \\ \vdots \end{pmatrix} \\
&= \overrightarrow{\delta^T}\overrightarrow{p_i} \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix} \\
&= (\overrightarrow{\delta^T}\overrightarrow{p_i})\vec{1} \\
&= \texttt{repeat}(\overrightarrow{\delta}\,\overrightarrow{p_i}, \texttt{n})
\end{aligned}
$$

Observe also that If we let $\vec{\delta}$ and p_i be 1 dimensional vectors, this can also be expressed as a dot product:

Matrices	Vectors
$\vec{\delta}^{\mathrm{T}}\vec{p_i}$	$\vec{\delta}\vec{p_i}$

Practical; Implementing the Power Walk on Sparse Matrices

Inspect the newly created matrix and create constants

Setup

1. Define function to create DiagonalsSparse Diagonal Function

   Unlike the Random Surfer model the diagonal scaling matrix will always be given by $\mathbf{D}_B^{-1} = \mathbf{B} \enspace \mathrm{diag}\left(\frac{1}{\vec{1}\mathbf{B}}\right)$ because $\beta^{\mathbf{A}_{i,j}} \neq 0 \quad \forall \mathbf{A}_{i,j}$, this is convenient but in any case the `sparse_diag` function in listing **??** will still work.

Power Walk

1. Define B

```r
A       <- Matrix::Matrix(A, sparse = TRUE)
B       <- A
B@x     <-  ^(A@x)
B       <- A
B        <-  ^A

Bo      <- A

# These two approaches are equivalent
Bo@x    <-  ^(A@x) -1    # This in theory would be
↪  faster
# Bo      <-  ^(A) -1
# Bo      <- drop0(Bo)


 n <- nrow(A)
```

```
10 x 10 sparse Matrix of class "dgCMatrix"
   [[ suppressing 10 column names ‘1’, ‘2’, ‘8’ ... ]]

1  . 1 . . . . . . . .
2  1 . . . . . . . . .
8  1 1 . 1 1 1 . . . .
5  1 1 . . . . . 1 1 .
7  . 1 . . . . . . . .
6  . 1 . . . . . 1 . .
9  . 1 . . . . . 1 1 .
3  . . . . . . . . 1 .
4  . . . . . . . 1 . .
10 . . . . . . . 1 1 .
```

```r
print(round(B, 2))
```

```
10 x 10 Matrix of class "dgeMatrix"
      1    2 8    5    7    6 9    3    4 10
1  1.00 0.87 1 1.00 1.00 1.00 1 1.00 1.00  1
2  0.87 1.00 1 1.00 1.00 1.00 1 1.00 1.00  1
8  0.87 0.87 1 0.87 0.87 0.87 1 1.00 1.00  1
5  0.87 0.87 1 1.00 1.00 1.00 1 0.87 0.87  1
7  1.00 0.87 1 1.00 1.00 1.00 1 1.00 1.00  1
6  1.00 0.87 1 1.00 1.00 1.00 1 0.87 1.00  1
9  1.00 0.87 1 1.00 1.00 1.00 1 0.87 0.87  1
3  1.00 1.00 1 1.00 1.00 1.00 1 1.00 0.87  1
4  1.00 1.00 1 1.00 1.00 1.00 1 0.87 1.00  1
```

```
10 1.00 1.00 1 1.00 1.00 1.00 1 0.87 0.87  1
```

```r
print(Bo,2)
```

```
10 x 10 sparse Matrix of class "dgCMatrix"
   [[ suppressing 10 column names ‘1’, ‘2’, ‘8’ ... ]]

1  .     -0.13 . .     .     .     . .     .     .
2  -0.13 .     . .     .     .     . .     .     .
8  -0.13 -0.13 . -0.13 -0.13 -0.13 . .     .     .
5  -0.13 -0.13 . .     .     .     . -0.13 -0.13 .
7  .     -0.13 . .     .     .     . .     .     .
6  .     -0.13 . .     .     .     . -0.13 .     .
9  .     -0.13 . .     .     .     . -0.13 -0.13 .
3  .     .     . .     .     .     . .     -0.13 .
4  .     .     . .     .     .     . -0.13 .     .
10 .     .     . .     .     .     . -0.13 -0.13 .
```

2. Solve the Scaling Matrix We don't need to worry about any terms of $\delta_{\mathbf{B}} = \texttt{colsums}\left(\texttt{B\_o}\right) + \texttt{n}$ being 0:

```r
( B   <- 1/(colSums(Bo)+n))
```

```
        1         2         8         5         7         6         9         3 
0.1041558 0.1086720 0.1000000 0.1013479 0.1013479 0.1013479 0.1000000 0.1071237 
        4        10 
0.1056189 0.1000000 
```

```r
( B   <- 1/(colSums(B)))
```

```
        1         2         8         5         7         6         9         3 
0.1041558 0.1086720 0.1000000 0.1013479 0.1013479 0.1013479 0.1000000 0.1071237 
        4        10 
0.1056189 0.1000000 
```

3. Find the Transition Probability Matrix

```r
  DB   <- diag( B)
## ** Create the Transition Probability Matrix
## Create the Trans Prob Mat using Power Walk
```

```
  T <- Bo %*% DB
```

4. Implement the Loop

```
## ** Implement the Power Walk
## *** Set Initial Values
  p_new  <- rep(1/n, n)  # Uniform
  p      <- rep(0, n)    # Zero
        <- 10^(-6)
## *** Implement the Loop

 while (sum(abs(p_new - p)) >  ) {
    (p <- as.vector(p_new)) # P should remain a vector
    sum(p <- as.vector(p_new)) # P should remain a
    ↪  vector
     p_new  <- T %*% p + rep(t( B) %*% p, n)
  }
## ** Report the Values
print(paste("The stationary point is"))
print(p)
```

5 Creating a Package

In order to investigate the effect of the model parameters on the second Eigenvalue it will be necessary to use these functions, in order to document and work with them in a modular way they were placed into an ***R*** package and made available on *GitHub* [fn: https://github.com/RyanGreenup/PageRank], to load this package use the `devtools` library as shown in listing .

```
library(devtools)
library(Matrix)
library(tidyverse) # Maybe, TODO check if this is
↪  used, I don't think it is

  if (require("PageRank")) {
      library(PageRank)
    }else{
      devtools::install_github("ryangreenup/PageRank")
      library(PageRank)
    }
```

Listing 8: Load the *PageRank* package which consists of the functions from 4

```
Loading required package: usethis
```

```
Loading required package: PageRank

Attaching package: ‘PageRank’
```

Part II

Investigating ξ_2

6 Erdos Renyi Graphs

6.1 ER Graphs Plotting Various Values

6.1.1 Erdos Reny Game

The *Erdos Renyi* game, first published in 1959 [26] creates a graph by assuming that the number of nodes is constant and the probability of interlinking these nodes is equal.

This is implemented in ***R*** [16, IgraphManualPagesa]

The Erdos Renyi game does not produce graphs consistent with networks such as the web (see 7) or wikis, however, Sampling these graphs will provide a broader picture for the overall behaviour of ξ_2 over a broad range of graphs with respect to the parameters of the *Power Walk* method.

6.1.2 Correlation Plot

By looping over many random graphs for a variety of probabilities a data set can be constructed and a correlation plot generated. To implement this a data frame of input values was constructed in listing 9, a function that builds a data frame with the second eigenvalue, density, determinant and trace was constructed in listing **??** and finally a correlation plot was generated in listing 10 shown in figure .

```r
# Generate Constants
n         <- 20
p         <- 1:n/n
beta      <- 1:n/n
beta      <- runif(n)*100
#sz        <- 1:n/n+10
sz        <- (1:n/n)*100+10
input_var <- expand.grid("n" = n, "p" = p, "beta" =
  beta, "size" = sz)

# Print out a sample of all the rows
input_var[sample(1:nrow(input_var), 6),]
```

Listing 9: A data frame consisting of input variables to be used to generate *Erdos Renyi* graphs.

```
      n    p       beta size
7154 20 0.70  0.5872766  100
4103 20 0.15 82.8545866   65
```

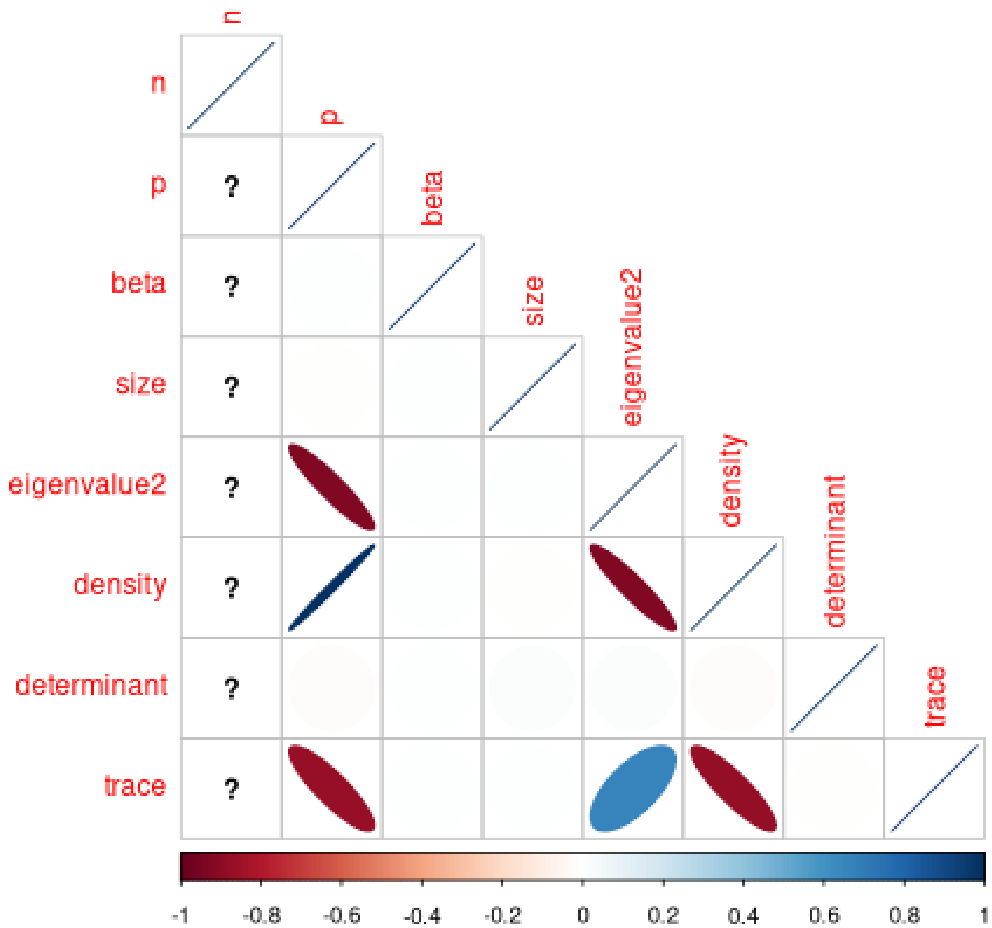
```
133   20 0.65 64.6700020   15
3887  20 0.35 76.8201442   60
1725  20 0.25 64.6700020   35
6071  20 0.55 26.0913073   90
```

mean$(\mathbf{A}), |A|, \mathrm{tr}(\mathbf{A})\rangle$\) corresponding to the *Power Walk* method using the PageRank package discussed at 5 .

```r
random_graph <- function(n, p, beta, size) {
      g1 <- igraph::erdos.renyi.game(n = sz, p)
      A <- igraph::get.adjacency(g1) # Row to column
      A <- Matrix::t(A)

      A_dens <- mean(A)
      T       <- PageRank::power_walk_prob_trans(A)
      T_tr     <- sum(diag(T))
      e2      <- eigen(T, only.values = TRUE)$values[2] # R orders by descending magnitude
      A_det   <- det(A)
      T_det   <- det(T)
      return(c(abs(e2), A_dens, T_det, T_tr)) # A_det and T_tr are uncorrelated
}
```

```r
filename <- "erdosData.rds"

if (file.exists(filename)) {

  data <- readRDS(filename)

  } else {

# Loop over the data
nc <- length(random_graph(1, 1, 1, 1))
Y  <- matrix(ncol = nc, nrow = nrow(input_var))

for (i in 1:nrow(input_var)) {
  X <- as.vector(input_var[i,])
  Y[i,] <-  random_graph(X$n, X$p, X$beta, X$size)
}

## Remove the 0i component
if (sum(abs(Y) != abs(Re(Y))) == 0) {
  Y <- Re(Y)
}



## Clean up the data frame
Y <- as.data.frame(Y); colnames(Y) <- c("eigenvalue2", "density", "determinant", "trace")
data <- cbind(input_var, Y)
data <- data[data$density!=0,]

## Save the data
saveRDS(data, filename)
}

corrplot(cor(data), method = "ellipse", type = "lower")
```

Listing 10: Produce a correlation plot Created from a dataframe constructed from the values assigned in listing 9 by using the function defined in listing ??, see figure .



6.1.3 Density of Adjacency Matrix

There appears to be a strong negative correlation between the eigenvalue and the density of the adjacency matrix.

This relationship is plotted in listing 11 and figure 6.

The relationship appears almost linear and so the data is log transformed and modelled against that in listing 12 with a corresponding plot generated in listing 13 and shown in figure 13 revealing a concave down relationship. The quartic model fits the data well and has the lowest *MSE*, however the logarithmic model is visually a good fit, significantly simpler and still has a low *MSE*, for this reason the logarithmic model will be used.

The coefficients of the logarithmic model, shown in listing 12, imply the following relationship:

$$\xi_2 = \left(1 - \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{A}_{i,j}}{n^2}\right)^{0.6} \cdot e^{-0.48} \pm \Delta \tag{47}$$

The maximum residual is given as 0.37 and so $\Delta = 0.4$ would provide a good indication for the value of the second eigenvalue by considering only the interconnectivity of the adjacency matrix.

This suggests that a more interlinked network will converge faster when using the *Power Walk* method.

```
ggplot(data) +
  geom_point(mapping = aes(x = density, y = eigenvalue2, size = beta, color = size )) +
  scale_size_continuous(range = c(0.1,1)) +
  labs(x = "Density of Adjacency Matrix", y = TeX("$\\xi_2$ of $T_{PW}$"), title = TeX("$\\xi_2$ of $T_{PW}$ given the Density of the Adjacency Matrix") ) +
  guides(size = FALSE, col = FALSE)
```

Listing 11: Create a Plot of ξ_2 given Adjacency Matrix, see plot in figure 6

```
   MSE Linear    MSE Quadratic      MSE Quartic MSE Logarithmic
        0.078            0.031            0.013           0.023

  (Intercept) log(1 - density)
   -0.4798300        0.6738175

[1] "Max Residual"      "0.371075096908937"
```

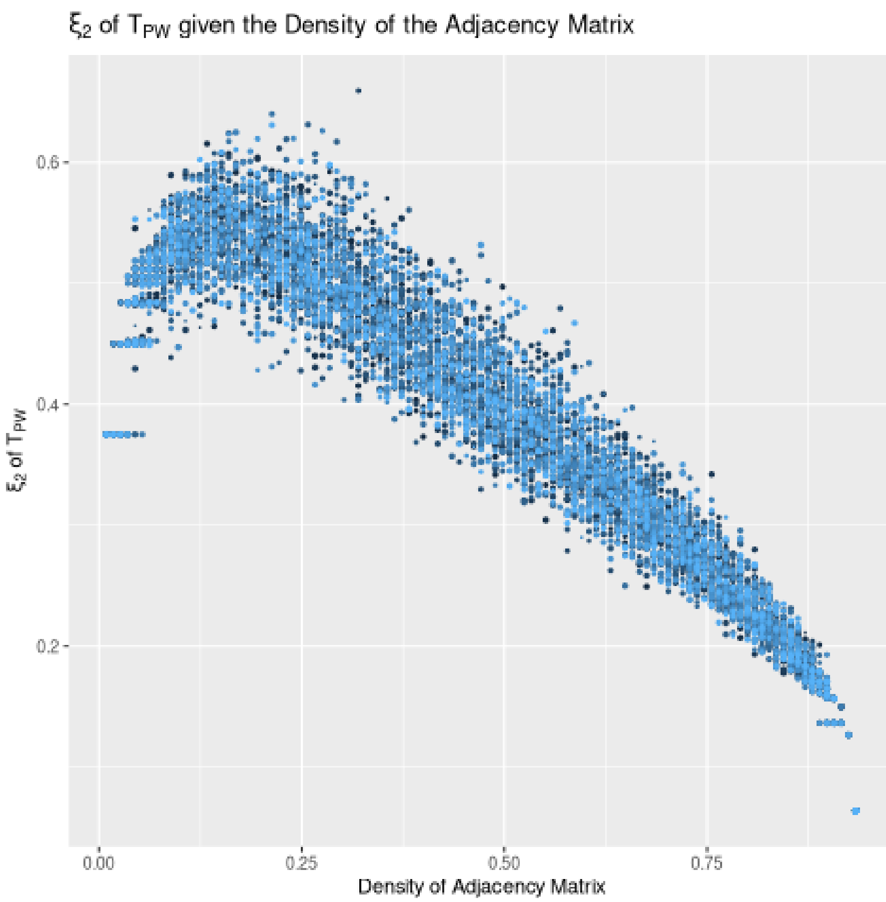


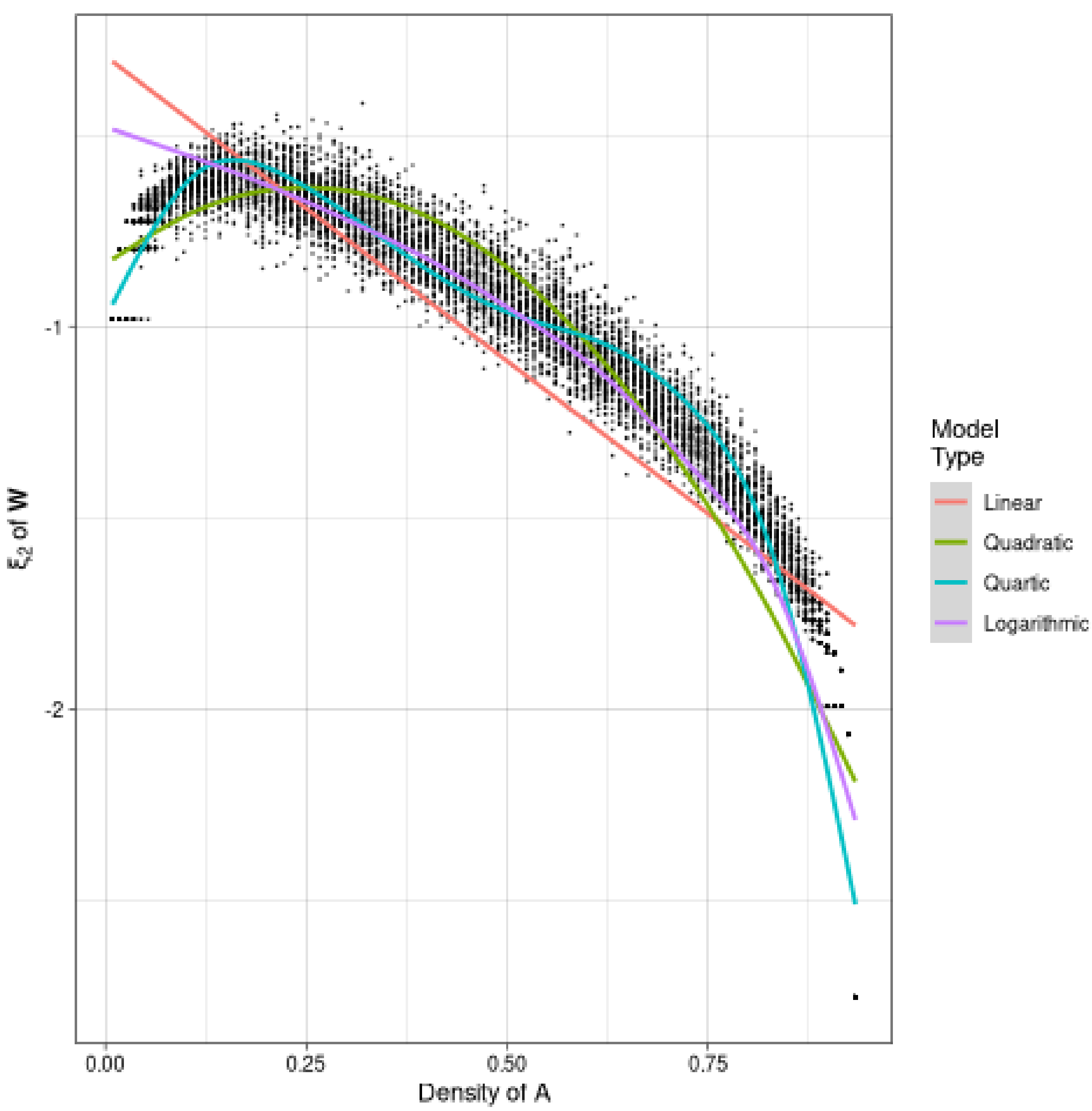
Figure 6: Plot of ξ_2 given Adjacency Matrix, see listing 11

```r
mod_x1 <- lm(log(eigenvalue2) ~ poly(density, 1), data = data)
data$x1 <- predict(mod_x1)

mod_x2 <- lm(log(eigenvalue2) ~ poly(density, 2), data = data)
data$x2 <- predict(mod_x2)

mod_x4 <- lm(log(eigenvalue2) ~ poly(density, 4), data = data)
data$x4 <- predict(mod_x4)

mod_xl <- lm(log(eigenvalue2) ~ log(1-density), data = data)
data$xl <- predict(mod_xl)

mod_df <- data
mod_df_long <- pivot_longer(mod_df, cols = c(x1, x2, x4, xl), names_to = "Model_Type", values_to = "eigenvalue2_mod")
mod_df_long$eigenvalue2_log <- log(mod_df_long$eigenvalue2)


print(c("MSE Linear"  = mean(mod_x1$residuals^2),
        "MSE Quadratic" = mean(mod_x2$residuals^2),
        "MSE Quartic" = mean(mod_x4$residuals^2),
        "MSE Logarithmic" = mean(mod_xl$residuals^2)
        ), 2)


cat("\n")
print(mod_xl$coefficients)
cat("\n")
print(c("Max Residual", max(mod_xl$residuals)))
```

Listing 12: Fit Models to log transofmred Density Comparison

```r
ggplot(mod_df_long, aes(x = density)) +
  geom_point(aes(y = eigenvalue2_log), fill =
  ↪  "lightblue", col = "black", size = 0.1, alpha =
  ↪  0.2) +
  geom_smooth(aes(y = eigenvalue2_mod, col =
  ↪  Model_Type), size = 0.9) +
  labs(col = c("Model \nType")) +
  scale_color_discrete(labels = c("Linear",
  ↪  "Quadratic", "Quartic", "Logarithmic")) +
  labs(x = "Density of A", y = TeX("$ \\xi_2 $ of
  ↪  $\\mathbf{W}$")) +
  theme_linedraw()
```

Listing 13: Plot of Log Transformed ξ_2 against density of Adjacency matrix, using the *Power Walk* algorithm applied to graphs randomly generated with the *Erdos-Renyi* game.



6.1.4 Trace of Transition Probability Matrix

The correlation plot suggests that there is some positive relationship between the trace of the transition probability matrix and the second eigenvalue, these values are plotted in listing 15 and figure 7, this relationship appears to be heteroskedastic and so it is log transformed in listing 14 and figure 8. This plot is still appears to have a non constant variance but this could be due to less data corresponding to lower trace values.

The plot suggests an exponential or hyperbolic model may be a good fit, this is performed in listing 16, **??** and 9. The hyperbolic model appears to be a reasonable fit for trace values less than half, giving the following relationship:

$$\xi_2 \approx \exp\left(\frac{0.2}{\mathrm{tr}\left(\mathbf{T}\right)}\right) \tag{48}$$

$$\approx \exp\left(\frac{0.2}{\mathrm{tr}\left(\mathbf{B}\mathbf{D}_{\mathbf{B}}^{-1}\right)}\right) \tag{49}$$

```r
ggplot(data, aes(x = trace , y = log(eigenvalue2))) +
  geom_point(mapping = aes(size = size, color = p,
  ↪ shape = factor(n))) +
#  stat_smooth() +
  scale_size_continuous(range = c(0.1,1.5)) +
  labs(x = "Trace of Transition Matrix", y =
  ↪ TeX("$\\log\\left( \\xi_2 \\right)$ of
  ↪ \\mathbf{W}"))
  labs(x = "Trace of Transition Matrix", y =
  ↪ TeX("$\\log\\left( \\xi_2 \\right)$ of
  ↪ \\mathbf{W}"))
```

Listing 14: Plot ξ_2 against the trace of the matrix of the *Power Walk* Transition Probability Matrix, see figure 8

```r
ggplot(data, aes(x = trace , y = eigenvalue2)) +
  geom_point(mapping = aes(size = size, color = p,
  ↪ shape = factor(n))) +
#  stat_smooth() +
  scale_size_continuous(range = c(0.1,1.5)) +
  labs(x = "Trace of Transition Matrix", y = TeX("$
  ↪ \\xi_2 $ of $\\mathbf{W}$"))
```

Listing 15: Plot ξ_2 against the trace of the matrix of the *Power Walk* Transition Probability Matrix

```
MSE Hyperbolic MSE Logarithmic
         0.081           0.127
```

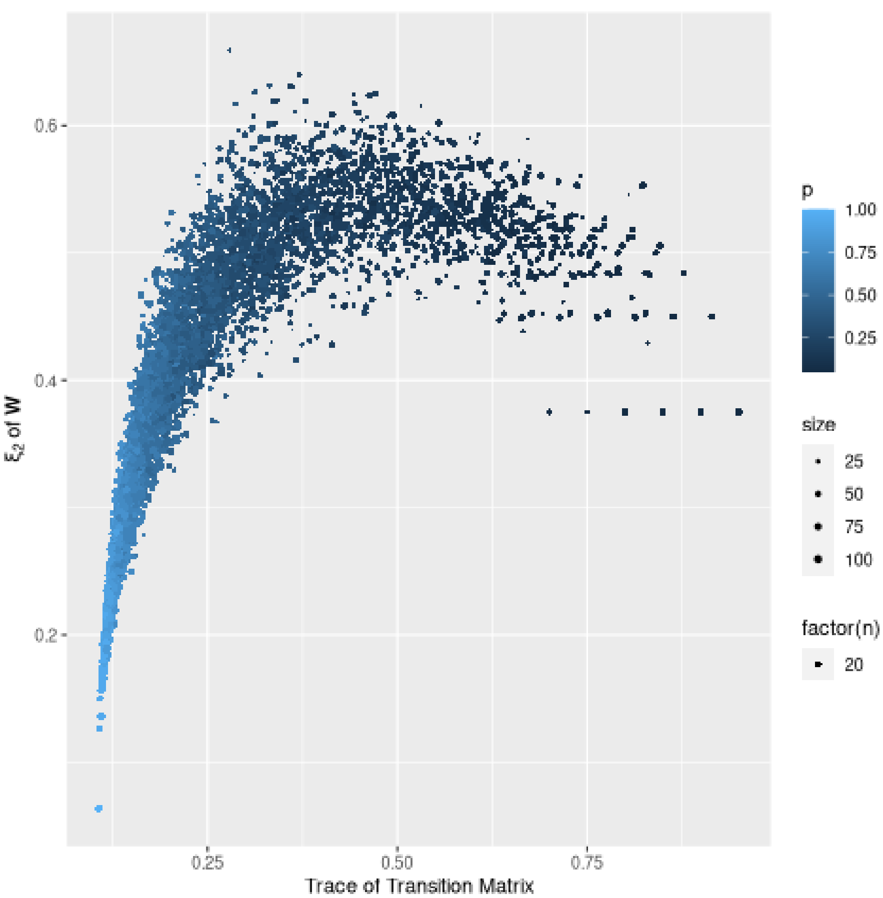


Figure 7: Plot of ξ_2 against the trace of the *Power Walk* probability transition matrix

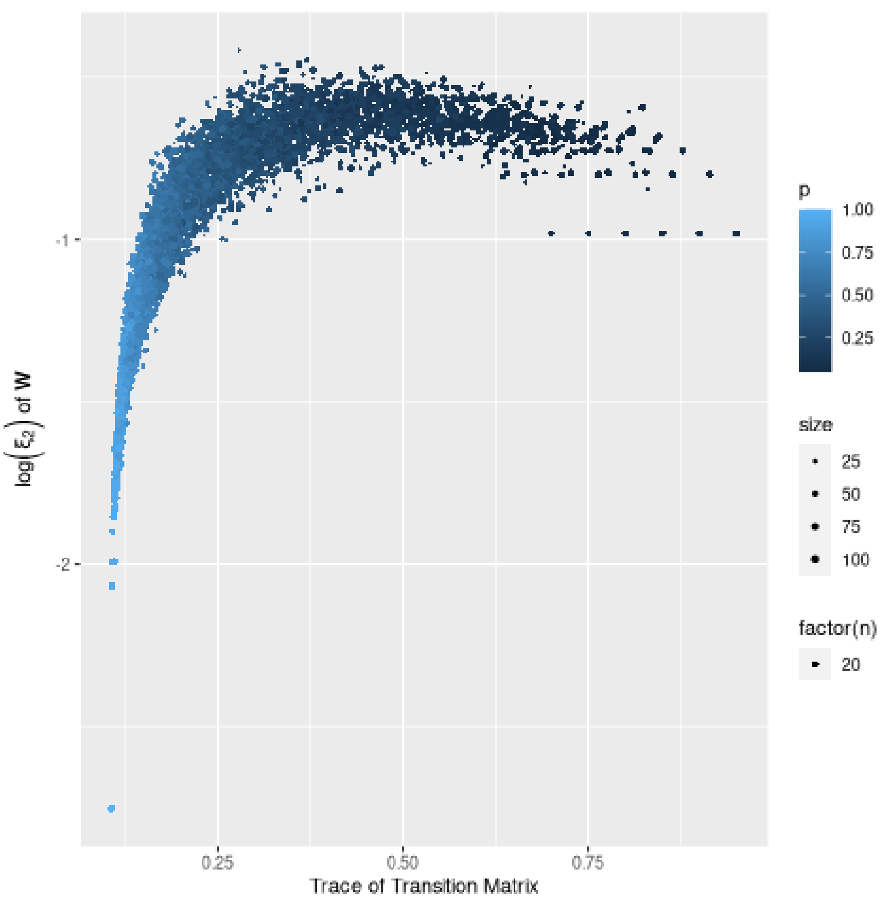


Figure 8: Log transformed plot of the trace of the *Power Walk* probability transition matrix

```
mod_df <- data


mod_hyp <- lm(log(eigenvalue2) ~ 0 + I(trace^(-1)),
 ↪  data = data)
mod_df$hyp <- predict(mod_hyp)


mod_log <- lm(log(eigenvalue2) ~ 0 + log(trace), data
 ↪  = data)
mod_df$log <- predict(mod_log)



mod_df_long <- pivot_longer(mod_df, cols = c(hyp,
mod_df_long$eigenvalue2_log <-
 ↪  log(mod_df_long$eigenvalue2)


print(c("MSE Hyperbolic"  = mean(mod_hyp$residuals^2),
        "MSE Logarithmic" =
         ↪  mean(mod_log$residuals^2)), 2)
cat("\n")
print(summary(mod_hyp)$coefficients)
```

Listing 16: Fit a Hyperbolic and Logarithmic model to the data, observe that a 0 intercept is set to fix the intercept as it would be expected that a 0 trace would correspond to a 0 eigenvector, the hyperbolic model has a slightly lower mean *MSE*.

```
                 Estimate   Std. Error   t value Pr(>|t|)
I(trace^(-1)) -0.1992157 0.0005471191 -364.1176        0
```

```r
ggplot(mod_df_long, aes(x = trace)) +
  geom_point(shape = 23, aes(y = eigenvalue2_log),
  ↪  fill = "lightblue", col = "black", size = 0.7,
  ↪  alpha = 0.4) +
  geom_line(aes(y = eigenvalue2_mod, col =
  ↪  Model_Type), size = 1) +
  labs(col = c("Model \nType")) +
  scale_color_manual(labels = c("Hyperbolic",
  ↪  "Logarithmic"),
                     values = c("indianred",
                     ↪  "royalblue"))  +
  labs(x = "Trace of Transition Matrix", y =
  ↪  TeX("$\\log\\left( \\xi_2 \\right)$ of
  ↪  \\mathbf{W}"), title = TeX("Models Fitted to
  ↪  Logarithmically scaled $\\xi_{2}$ given Matrix
  ↪  Trace")) +
  theme_linedraw()
```

6.1.5 Conclusion

The *Erdos Renyi* game produces a wide variety of all possible graphs, this provides two insights into the value of the second eigenvalue of the probability transition matrix corresponding to the power walk method:

$$\xi_2 \approx \exp\left(\frac{0.2}{\operatorname{tr}\left(\mathbf{B}\mathbf{D}_{\mathbf{B}}^{-1}\right)}\right) \tag{50}$$

$$\xi_2 = \left(1 - \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{A}_{i,j}}{n^2}\right)^{0.6} \cdot e^{-0.48} \pm 0.4 \tag{51}$$

These can be used to evaluate broadly the value of ξ_2 and in turn the rate of convergence of the *Power Walk* method corresponding to a given graph given only the method parameters and the adjacency matrix.

These are not however insightful of any direct relationships between the method parameters and ξ_2, this will be considered at 8 by trying to find a relationship between the *Power Walk* and the *Random Surfer* models.

6.2 Model the log transformed data using a linear regression or log(-x) regression

6.2.1 Change the colour of each model by using pivot$_{\text{longer}}$

6.3 Import wikipedia data

- ~~Import the wikipedia data~~

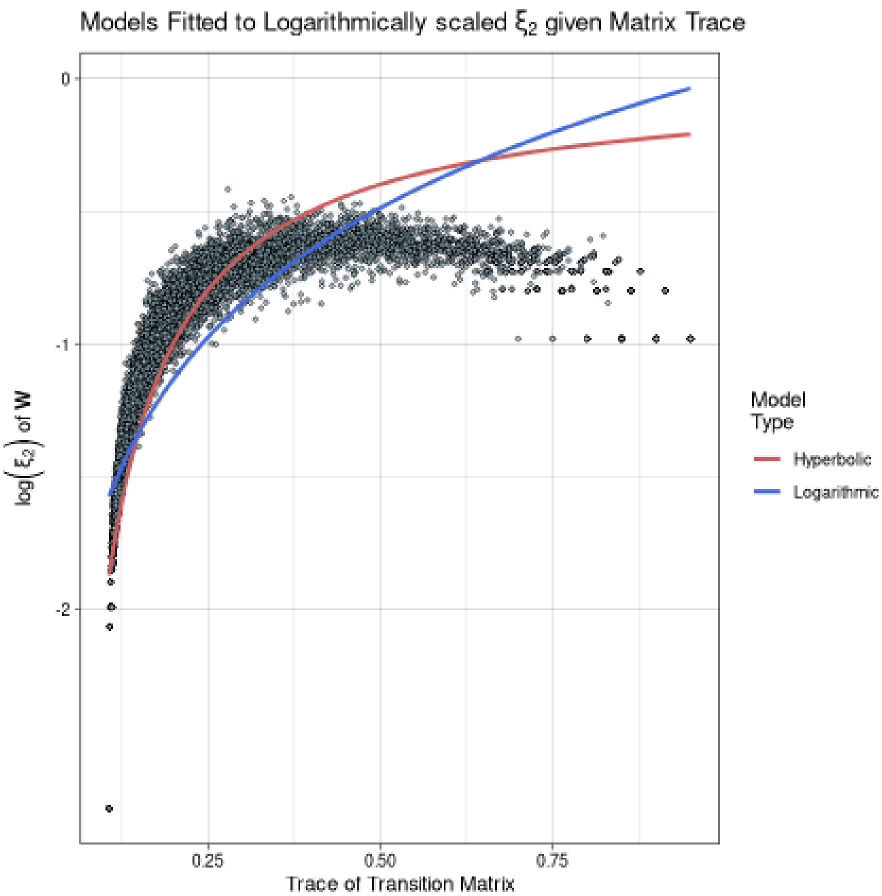


Figure 9: Plot of the second eigenvalue logarithmically scaled across the trace of a corresponding probability transition matrix created using the *Power Walk* method. The graphs were randomly generated using the *Erdos Renyi* game.

- ~~Measure the density~~
- ~~Use the density to guess the p of the game~~
  - ~~Justify the witht the scatterplot matrix~~
- ~~Measure the affect of different β values on λ_2 for graphs ov various sizes given that p value.~~
  - ~~Or atleast a range within that prob~~
    use a *Barabassi-Albert* Random Graph through the ~igraph::

6.4 Look at the Trace of the Matrix as a comparison point

6.5 Use BA Graphs

7 Barabasi Albert Graphs

A graph of the internet is *scale free*, this means that the number of nodes of a graph (n), having j edges is given by [20, §10.7.2]:

$$n \propto j^{-k}, \quad \exists k \in \mathbb{R} \tag{52}$$

The *Erdos Renyi* game is a random network, a superior approach to model the web is to use a scale free networks [4] such as the Barabasi-Albert graph [5]

The Erdos Renyi game assumes that the number of nodes is constant from beginning to end, clearly this is not true for networks such as the web. Consider a graph constructed node by node where each time a new node is introduced it is randomly connected to another with a constant probability. Despite the probability of connecting to any given node being constant as in the Erdos Renyi game, such a graph will favour nodes introduced earlier with respect to the number edges. This shows that the precense of network growth is an import feature in modelling networks.

Simply considering growth however is not sufficient to simulate graphs with a degree distribution consistent with the web [29, Ch. 7] (see figure).

When introducing a new node, the probability of linking to any other node is not uniformly random. When adding links to from one node to another it would be expected that links to more popular websited would be made (for example if somebody added a link to a personal website they might be more likely to link to *Wikipedia* than to the *Encyclopedia of Britannica* simply because it is more common). A simple approach is to presume that the probability of linking from one node to another is proportional to the number of links, i.e. a node with twice as many links will be twice as likely to receive a link from a new node.

These two distinguishing features departing from the *Erdos Renyi* model, known as *Growth* and *Preferential Attachment*, are what set the Barabassi-Albert model apart from the Erdos-Renyi model and why it is better suited to modelling networks such as the web. [3, Ch. 7]

A practical Simulation method for social networks simulate social network links, one possibility is this paper [29].

Actually there is a data set available [14], I should just analyse that, see how it was done in Visual Analytics as a reminder.

```r
layout(matrix(1:2, nrow = 2))
col  <- "Mediumpurple"

n <- 1000
hist(

  igraph::degree(igraph::sample_pa(n, 0.2)),
  binwidth = 0.3,

  xlab = "",
  main = "Barabassi-Albert Degree Distribution",

  col = col, freq = FALSE
)


hist(igraph::degree(igraph::erdos.renyi.game(n, 0.2)),

     main= "Erdos-Renyi Degree Distribution",
     col = col,

     binwidth = 0.3,
     xlab = "",

     freq = FALSE )
```

Listing 17: Simulate Erdos-Renyi and Barabassi-Albert graphs in order to measure the degree distribution, shown in 10

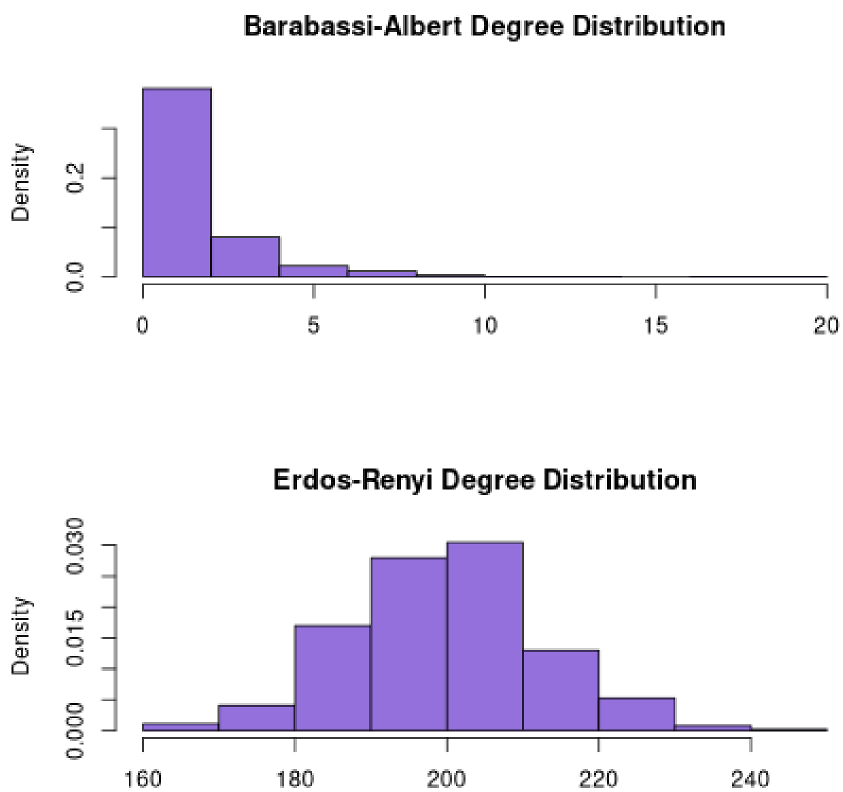


Figure 10: histograms of degree distribution of Erdos-Renyi and Barabassi-Albert graphs produced in listing 17

8 Relating the Power Walk to the Random Surfer

8.1 Introduction

These are notes relating to [25, §3.3], probably won't put this in the report, just arbitrary notes

So if a term in the Power Walk can be related to α in the random surfer, which is in turn ξ_2, I'll be able to understand it better. [8]

Consider the equation:

$$\begin{aligned}\mathbf{T} &= \mathbf{B}\mathbf{D}_{\mathbf{B}}^{-1} \\ &= (\mathbf{B}+\mathbf{O}-\mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\end{aligned}$$

Break this into to terms so that we can simplify it a bit:

$$\mathbf{T} = \left[(\mathbf{B}-\mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\right] + \left\{\mathbf{O}\mathbf{D}_{\mathbf{B}}^{-1}\right\}$$

8.2 Value of [1st Term]

Observe that for all $\forall i, j \in \mathbb{Z}^{+}$:

$$\begin{aligned}\mathbf{A}_{i,j} &\in \{0,1\} \\ \implies \mathbf{B}^{\mathbf{A}_{i,j}} &\in \left\{\beta^0, \beta^1\right\} \\ &= \{1, \beta\} \\ \implies \beta\mathbf{A} &= \{1,\beta\}\end{aligned}$$

Using this property we get the following

$$\begin{aligned}\mathbf{B}_{i,j} - \mathbf{O}_{i,j} = \left(\beta^{\mathbf{A}_{i,j}} - 1\right) &= \begin{cases} 0, & \mathbf{A}_{i,j} = 0 \\ \beta - 1, & \mathbf{A}_{i,j} = 1 \end{cases} \\ (\beta - 1)\,\mathbf{A}_{i,j} &= \begin{cases} 0, & \mathbf{A}_{i,j} = 0 \\ \beta - 1, & \mathbf{A}_{i,j} = 1 \end{cases}\end{aligned}$$

This means we have

$$\mathbf{A} \in \{0,1\}\,\forall i,j \implies \mathbf{B}_{i,j} - \mathbf{O}_{i,j} = (\beta - 1)\,\mathbf{A}_{i,j}$$

$$\begin{aligned}\mathbf{B} &= (\mathbf{B}+\mathbf{O}-\mathbf{O}) \\ &= (\mathbf{B}-1)\end{aligned}$$

[8] Although I'm not quite sure why α is ξ_2 either

8.3 Value of {2nd Term}

$$
\begin{aligned}
\mathbf{O}\mathbf{D}_{\mathbf{B}}^{-\mathbf{1}} &= \begin{pmatrix} 1 & 1 & 1 & \\ 1 & 1 & 1 & \cdots \\ 1 & 1 & 1 & \\ & \vdots & & \ddots \end{pmatrix} \begin{pmatrix} \frac{1}{\delta_1} & 1 & 1 & \\ 1 & \frac{1}{\delta_2} & 1 \cdots & \\ 1 & 1 & \frac{1}{\delta_3} & \\ & \vdots & & \ddots \end{pmatrix} \\
&= n \begin{pmatrix} \frac{1}{n} & \frac{1}{n} & \frac{1}{n} & \\ \frac{1}{n} & \frac{1}{n} & \frac{1}{n} & \cdots \\ \frac{1}{n} & \frac{1}{n} & \frac{1}{n} & \\ & \vdots & & \ddots \end{pmatrix} \begin{pmatrix} \frac{1}{\delta_1} & 1 & 1 & \\ 1 & \frac{1}{\delta_2} & 1 & \cdots \\ 1 & 1 & \frac{1}{\delta_3} & \\ & \vdots & & \ddots \end{pmatrix} \\
&= n\mathbf{E}\mathbf{D_B}^{-1}
\end{aligned}
$$

where the following definitions hold ($\forall i, j \in \mathbb{Z}^+$):

- $\mathbf{E}_{i,j} = \frac{1}{n}$
- $\mathbf{D_B}_{k,k}^{-1} = \frac{1}{\delta_k}$
- The value of δ is value that each term in a column must be divided by to become zero, in the case of the power walk that is just $\frac{1}{\texttt{colSums}(\mathbf{B})} = \vec{1}\mathbf{B}$, but if there were zeros in a column, it would be necessary to swap out the 0s for 1s and then sum in order to prevent a division by zero issue and because the 0s should be left.
- $\mathbf{A} \in \{0, 1\} \, \forall i, j$ is the unweighted adjacency matrix of the relevant graph.

putting this all together we can do the following:

$$
\begin{aligned}
\mathbf{T} &= \mathbf{B}\mathbf{D}_{\mathbf{B}}^{-1} \\
&= (\mathbf{B} + \mathbf{O} - \mathbf{O})\, \mathbf{D}_{\mathbf{B}}^{-1} \\
&= (\mathbf{B} - \mathbf{O})\, \mathbf{D}_{B}^{-1} + \mathbf{O}\mathbf{D}_{\mathbf{B}}^{-1}
\end{aligned}
$$

From above:

$$
\begin{aligned}
&= (\beta - 1)\, \mathbf{A}_{i,j} + n\mathbf{E}\mathbf{D}_{\mathbf{B}}^{-1} \\
&= \mathbf{A}_{i,j}\, (\beta - 1) + n\mathbf{E}\mathbf{D}_{\mathbf{B}}^{-1}
\end{aligned}
$$

because $\mathbf{D}\mathbf{D}^{-1} = \mathbf{I}$ we can multiply one side through:

$$
= \mathbf{D_A}\mathbf{D}_{\mathbf{A}}^{-1}\mathbf{A}_{i,j}\, (\beta - 1) + n\mathbf{E}\mathbf{D}_{\mathbf{B}}^{-1}
$$

But the next step requires showing that:

$$
(\beta - 1)\, \mathbf{D_A}\mathbf{D}_{\mathbf{B}}^{-1} = \mathbf{I} - n\mathbf{D}_{B}^{-1}
$$

8.4 Equate the Power Walk to the Random Surfer

Define the matrix $\mathbf{D_M}$:

$$\mathbf{D_M} = \mathrm{diag}\left(\texttt{colSum}\left(\mathbf{M}\right)\right) = \mathrm{diag}\left(\vec{1}\mathbf{M}\right) \tag{53}$$

To scale each column of that matrix to 1, each column will need to be divieded by the column sum, unless the column is already zero, this needs to be done to turn an adjacency matrix into a matrix of probabilities:

$$\mathbf{D_A^{-1}} : \left[\mathbf{D_A^{-1}}\right]_i = \begin{cases} 0, & [\mathbf{D_A}]_i = 0 \\ \left[\frac{1}{\mathbf{D_A}}\right], & [\mathbf{D_A}]_i \neq 0 \end{cases} \tag{54}$$

In the case of the power walk $\mathbf{B} = \beta^{\mathbf{A}} \neq 0$ so it is sufficient:

$$\mathbf{D_B^{-1}} = \frac{1}{\mathrm{diag}\left(\vec{1}\left(\beta^{\mathbf{A}}\right)\right)} \tag{55}$$

Recall that the *power walk* gives a transition probability matrix:

Power Walk

$$\mathbf{T} = \boxed{\mathbf{AD_A^{-1}}}\mathbf{D_A}\left(\beta - 1\right)\mathbf{D_B^{-1}} + \boxed{\mathbf{E}}n\mathbf{D_B^{-1}} \tag{56}$$

Random Surfer

$$\mathbf{T} = \alpha\boxed{\mathbf{AD_A^{-1}}} + (1-\alpha)\boxed{\mathbf{E}} \tag{57}$$

So these are equivalent when:

$$\mathbf{D_A}\left(\beta - 1\right)\mathbf{D_{B^{-1}}} = \mathbf{I}\alpha \tag{58}$$

$$\begin{aligned} \vec{1}\left(1-\alpha\right) &= -n\mathbf{D_B^{-1}} \\ \implies \vec{1}\alpha &= \vec{1} - n\mathbf{D_B^{-1}} \end{aligned} \tag{59}$$

Hence we have:

$$\mathbf{D_A}\left(\beta - 1\right)\mathbf{D_B^{-1}} = \vec{1}\alpha = \mathbf{I} - n\mathbf{D_B^{-1}} \tag{60}$$

Solving for β with (58) :

$$\beta = \frac{1-\Theta}{\Theta} \tag{61}$$

$$\tag{62}$$

where: [9]

- $\Theta = \mathbf{D_A} \mathbf{D_B}^{-1}$

but we can't really do this so instead:

$$\beta \mathbf{1}_{[n,n]} = (1 - \Theta)\, \Theta^{-1}$$

If β is set accordingly then by (60):

$$\begin{aligned} & \mathbf{A}\,(\beta - 1)\,\mathbf{D_B}^{-1} = \alpha = \mathbf{I} - n\mathbf{D_B}^{-1} \\ \implies & \mathbf{A}\,(\beta - 1)\,\mathbf{D_B}^{-1} = \mathbf{I} - n\mathbf{D_B}^{-1} \end{aligned} \tag{63}$$

And setting $\Gamma = \mathbf{I} - n\mathbf{D_B}^{-1}$ from (59) and putting in (56) we have:

$$\mathbf{T} = \boxed{\mathbf{A}\mathbf{D_A}^{-1}} \mathbf{D_A}\,(\beta - 1)\,\mathbf{D_B}^{-1} + \boxed{\mathbf{E}} n\mathbf{D_B}^{-1}$$
$$\mathbf{T} = \Gamma \boxed{\mathbf{A}\mathbf{D_A}^{-1}} + (1 - \Gamma)\,\boxed{\mathbf{E}}$$

$$\mathbf{T} = \Gamma \mathbf{A}\mathbf{D_A}^{-1} + (1 - \Gamma)\,\mathbf{E} \tag{64}$$

Where $\mathbf{E}$ is square matrix of $\frac{1}{n}$ as in (16) (23)

8.5 Conclusion

So when the adjacency matrix is stictly boolean, the power walk is equivalent to the random surfer.

8.6 The Second Eigenvalue

8.6.1 The Random Surfer

The Second eigenvalue ξ_2 of the Power Surfer is less than α (See 3.2; Stability and Concvergence, of proposal).

8.6.2 Power Walk

Because the Power Walk relates to the random surfer as demonstrated in section , what can be said about ξ_2

[9]NOTE: Similar to a signmoid function, which is a solution to $p \propto p(1-p)$, I wonder if this provides a connection to the exponential nature of the power walk ‘erdos.renyi‘erdos.renyi“

Applying this to Power Walk Let $\Lambda_{(2)}(\mathbf{T}) = \lambda_2$ return the second value of a transition, probability Matrix, then observe that:

$$\Lambda_{(2)}(\mathbf{T}_{\text{RS}}) \leq |\alpha| \implies \Lambda_{(2)}(\mathbf{T}_{\text{PW}}) \leq \left| \frac{\alpha - \mathbf{D_a D_B^{-1}}}{\mathbf{D_A D_B^{-1}}} \right| \tag{65}$$

where:

- $\lambda_{(2)}(\mathbf{T})$ refers to the transition probability matrix of the power walk and random surfer approaces as indicated.

My attempt

$$\beta \mathbf{1}_{[n,n]} = \frac{1 - \Theta}{\Theta} \tag{66}$$

$$\tag{67}$$

where:

- $\Theta = \mathbf{D_A D_B^{-1}}$

So I thought maybe if I could find a value of β that satisfied (66) then I could show circumstances under which $|\xi_2| < \alpha$.

Seemingly it's only satisfied where $\beta = 1$ though, using this simulation:

```r
g1 <- igraph::erdos.renyi.game(n = 9, 0.2)
A <- igraph::get.adjacency(g1) # Row to column
A <- t(A)
# plot(g1)

## * Finding beta values to behave like Random Surfer
  beta <- 10
  B <- beta^A

  DA     <- PageRank::create_sparse_diag_sc_inv_mat(A)
  DB_inv <-
   ↪  PageRank::create_sparse_diag_scaling_mat(B)

 THETA <- DA %*% DB_inv

THETA <- function(A, beta) {
  B  <- beta^A
  DA     <- PageRank::create_sparse_diag_sc_inv_mat(A)
  DB_inv <-
   ↪  PageRank::create_sparse_diag_scaling_mat(B)
  return(DA %*% DB_inv)
}

```

```
THETA_inv <- function(A, beta) {
  B  <- beta^A
  DB     <- PageRank::create_sparse_diag_sc_inv_mat(B)
  DA_inv <-
   ↪  PageRank::create_sparse_diag_scaling_mat(A)
  return(DA %*% DB_inv)
}

beta_func <- function(A, beta) {
    return(1-THETA(A, beta^A) %*% THETA_inv(A,
     ↪  beta^A))
}

THETA(A, 10) %*% THETA_inv(A, 10)


eta <- 10^-6
beta <- 1.01
while (mean(beta*matrix(1, nrow(A), ncol(A)) -
 ↪  beta_func(A, beta)) > eta) {
    beta <- beta + 0.01
    print(beta)
    print(diag(beta_func(A, beta)))
    print(beta*matrix(1, nrow(A), ncol(A)))
    print(beta_func(A, beta))
#    Sys.sleep(0.1)
}

beta


diag(beta_func(A, beta))
beta


## * blah
```

9 Appendix

9.1 Graph Diagrams

Graph Diagrams shown in 2.2.2 where produced using DOT (see [28, 10]).

```r
library(Matrix)
library(igraph)
n <- 200
m <- 5
power <- 1
g <- igraph::sample_pa(n = n, power = power, m = m, directed = FALSE)
plot(g)
A <- t(get.adjacency(g))
plot(A)
image(A)


# Create a Plotting Region
par(pty = "s", mai = c(0.1, 0.1, 0.4, 0.1))


# create the image

title=paste0("Undirected Barabassi Albert Graph with parameters:\n Power = ", power, "; size = ", n, "; Edges/step = ", round(m))
image(A, axes = FALSE, frame.plot = TRUE, main = title, xlab = "", ylab = "",  )
```

Listing 18: ***R*** code to produce an image illustrating the density of a simulated Barabasi-Albert graph, the *Barabasi-Albert* graph is a good analouge for the link structure of the internet [20, 4, 5] see the output in figure 11

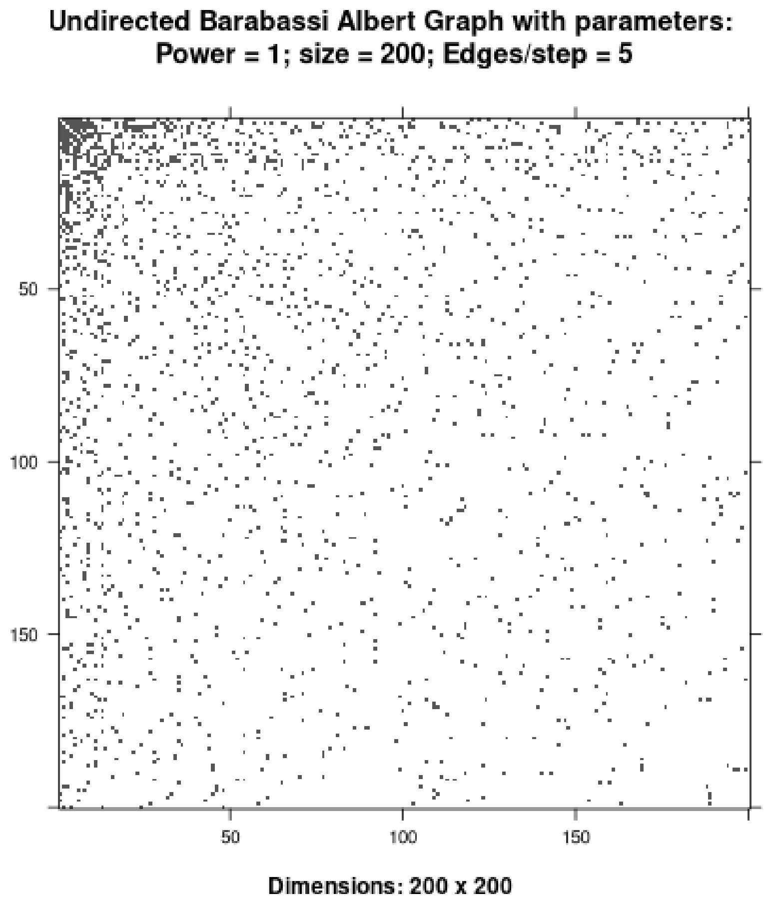


Figure 11: Plot of the adjacency matrix corresponding to a Barabassi-Albert (i.e. *Scale Free*) Graph produced by listing 18, observe the matrix is quite sparse.

9.2 Definitions

The following definitions are used in this report [10] :

Markov Chains are discrete mathematical model such that future values depend only on current values [11, §1.5]

Stochastic Matrices contain only positive values where each column sums to 1 [20, 11] (i.e. $\mathbf{T}$ is stochastic $\iff \vec{1}\mathbf{T} = \vec{1}$)

- some authors use rows (see e.g. [20, §15.3]), in this paper columns will be used, i.e. columns will add to one and an entry $\mathbf{A}_{i,j} \neq 0$ will indicate that travel is permitted from vertex j to vertex i.
  - *Column Stochastic* and *Row Stochastic* can be used to more clearly distinguish between which type of stochastic matrix is being used.
- Many programming languages return *unit-eigenvectors* $\vec{x}$ such that $||\vec{x}|| = 1$ as opposed to $\texttt{sum}\left(\vec{\mathtt{x}}\right) = 1$, so when solving for a stationary vector it can be necessary to perform $\vec{\mathtt{p}} \leftarrow \frac{\vec{\mathtt{p}}}{\sum \vec{p}}$

Irreducible graphs have a path from from any given vertex to another vertex. [20, §15.2]

Ergodic graphs are irreducible graphs with further constraints outside the scope of this report (see e.g. [24, 9])

- It is a necessary but not a sufficient condition of ergodic graphs that all vertices be reachable from any other vertices (see [27] for a counter example.)

Primitive Matrices are non-negative irreducible matrices that have only one eigenvalue on the unit circle.

- If a matrix is primitive it will approach a limit under exponentiation [20, §15.2]

Transition Probability Matrix is a stochastic matrix where each column is a vector of probabilities such that $\mathbf{T}_{i,j}$ represents the probability of travelling from vertex j to vertex i during a random walk.

- Some Authors consider the transpose (see e.g. [20]).

Aperiodic Markov chains are markov chains with an irreducible and primitive transition probability matrix.

- If the transition probability matrix is irreducible and imprimitive it is said to be a periodic Markov chain.

Regular Markov Chains are regular irreducible and aperiodic.

Sparse Matrices contain a majority of elements with values equal to 0 [20, §4.2]

Sparse Iterating the

PageRank A process of measuring graph centrality by using a random walk algorithm and measuring the most frequent node

- In the literature (see e.g. [15, 20]) the Random Surfer model is usually used to refer to the introduction of a probability of travelling to any other node, this is discussed in CROSSREF

[10] see generally [20, Ch. 15] for further reading

9.2.1 Notation

- $\mathbf{A}$
  - Is the adjacency matrix of a graph such that $\mathbf{A}_{i,j} = 1$ Indicates travel from j to i is possible.
- $\mathbf{T}$
  - Indicates the probability of a movement during a random walk, such that $T_{i,j}$ is equal to the probability of travelling $j \to i$ during a random walk.
- $\mathbf{D_A} = \mathrm{diag}\left(\vec{1}\mathbf{A}\right)$
- $\mathbf{D_A^{-1}} = \begin{cases} 0, & [\mathbf{D_A}]_i = 0 \\ \left[\frac{1}{\mathbf{D_A}}\right], & [\mathbf{D_A}]_i \neq 0 \end{cases}$
  - A diagonal scaling matrix such that $\mathbf{T} = \mathbf{A}\mathbf{D_A^{-1}}$, the piecewise definition is such that $\mathbf{D_A^{-1}}$ is still defined even if $\mathbf{A}$ is a reducible graph.
    - Where $\mathbf{D}^{-1}$ is a matrix such that multiplication with which scales each column of $\mathbf{A}$ to 1.
    - $\mathbf{D_A^{-1}} = \vec{1}\mathbf{D_A^{-1}} = \frac{1}{\vec{1}\mathbf{D_A}}$ for some stochastic matrix $\mathbf{A}$
- n
  - Refers to the number of vertices in a graph, $n = \texttt{nrow}\left(\mathbf{A}\right) = \texttt{ncol}\left(\mathbf{A}\right)$
- $\mathbf{E}_{i,j} = \frac{1}{n}$
  - A matrix of size $n \times n$ representing the background probability of jumping to any vertex of a graph.
- $\vec{1}$
  - a vector of length n containing only the value 1.
    - The convention that a vector behaves as a vertical $n \times 1$ matrix will be used here.
    - Some authors use $\mathbf{e}$, see e.g. [20]
- $\mathbf{J} = \vec{1} \cdot \vec{1}^{\mathrm{T}} \iff \mathbf{J}_{i,j} = 1$
  - A completely dense $n \times n$ matrix containing only 1
- α
  - The probability of teleporting from one vertex to another during a random walk.
    - In the literature α is often referred to as a damping factor (see e.g. [6, 8, 12, 17, 7])

or a smoothing constant (see e.g [18]).

10 my to do list

10.1 Look at the Trace of the Matrix as a comparison point

10.2 Use BA Graphs

**

10.3 TODO Diamater

Diamater of the web sounds like a fun read [2]

10.4 Improving the Performance of Page Rank

This:

> Another approach involves involves reordering the problem and taking advantage of the fact that the transition probability matrix is sparse in order to produce a new algorithm which cannot perform worse than the *power method* but has been shown to improve the rate of convergence in certain cases. [19].

There was also a book that I downloaded that mentioned it
Accellerating the Computatoin of Page Rank [20]

References

[1] *Adjacency Matrix*. In: *Wikipedia*. Sept. 20, 2020. URL: https://en.wikipedia.org/w/index.php?title=Adjacency_matrix&oldid=979433676 (visited on 10/10/2020).

[2] Réka Albert, Hawoong Jeong, and Albert-Lászió Barabási. "Diameter of the World-Wide Web". In: *Nature* 401.6749 (Sept. 1999), pp. 130–131. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/43601. URL: http://www.nature.com/articles/43601 (visited on 10/11/2020).

[3] Albert-László Barabási. *Linked: The New Science of Networks*. Cambridge, Mass: Perseus Pub, 2002. 280 pp. ISBN: 978-0-7382-0667-7.

[4] Albert-László Barabási. "The Physics of the Web". In: *Phys. World* 14.7 (July 2001), pp. 33–38. ISSN: 0953-8585, 2058-7058. DOI: 10.1088/2058-7058/14/7/32. URL: https://iopscience.iop.org/article/10.1088/2058-7058/14/7/32 (visited on 10/11/2020).

[5] Albert-László Barabási, Réka Albert, and Hawoong Jeong. "Scale-Free Characteristics of Random Networks: The Topology of the World-Wide Web". In: *Physica A: Statistical Mechanics and its Applications* 281.1-4 (June 2000), pp. 69–77. ISSN: 03784371. DOI: 10.1016/S0378-4371(00)00018-2. URL: https://linkinghub.elsevier.com/retrieve/pii/S0378437100000182 (visited on 10/11/2020).

[6] Joost Berkhout and Bernd F. Heidergott. "Ranking Nodes in General Networks: A Markov Multi-Chain Approach". In: *Discrete Event Dyn Syst* 28.1 (Mar. 1, 2018), pp. 3–33. ISSN: 1573-7594. DOI: 10.1007/s10626-017-0248-7. URL: https://doi.org/10.1007/s10626-017-0248-7 (visited on 08/19/2020).

[7] Monica Bianchini, Marco Gori, and Franco Scarselli. "Inside PageRank". In: *ACM Trans. Inter. Tech.* 5.1 (Feb. 1, 2005), pp. 92–128. ISSN: 15335399. DOI: 10.1145/1052934.1052938. URL: http://portal.acm.org/citation.cfm?doid=1052934.1052938 (visited on 08/18/2020).

[8] Michael Brinkmeier. "PageRank Revisited". In: *ACM Transactions on Internet Technology* 6.3 (Aug. 2006), pp. 282–301. ISSN: 15335399. DOI: 10.1145/1151087.1151090. URL: http://ezproxy.uws.edu.au/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=22173011&site=ehost-live&scope=site (visited on 08/19/2020).

[9] Mufa Chen. *Eigenvalues, Inequalities, and Ergodic Theory*. Probability and Its Applications. London: Springer, 2005. 228 pp. ISBN: 978-1-85233-868-8.

[10] *DOT (Graph Description Language)*. In: *Wikipedia*. June 11, 2020. URL: https://en.wikipedia.org/w/index.php?title=DOT_(graph_description_language)&oldid=961944797 (visited on 10/09/2020).

[11] François Fouss, Marco Saerens, and Masashi Shimbo. *Algorithms and Models for Network Data and Link Analysis*. New York, NY: Cambridge University Press, 2016. 521 pp. ISBN: 978-1-107-12577-3.

[12] Hwai-Hui Fu, Dennis K. J. Lin, and Hsien-Tang Tsai. "Damping Factor in Google Page Ranking". In: *Applied Stochastic Models in Business and Industry* 22.5-6 (2006), pp. 431–444. ISSN: 1526-4025. DOI: 10.1002/asmb.656. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.656 (visited on 08/19/2020).

[13] Gabor Csardi et al. *Igraph R Manual Pages*. May 9, 2019. URL: https://igraph.org/r/doc/as_adjacency_matrix.html (visited on 08/19/2020).

[14] Andrea Garritano. *Wikipedia Article Networks*. Dec. 2019. URL: https://kaggle.com/andreagarritano/wikipedia-article-networks (visited on 10/03/2020).

[15] Pankaj Gupta et al. "WTF: The Who to Follow Service at Twitter". In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW '13. New York, NY, USA: Association for Computing Machinery, May 13, 2013, pp. 505–514. ISBN: 978-1-4503-2035-1. DOI: 10.1145/2488388.2488433. URL: http://doi.org/10.1145/2488388.2488433 (visited on 10/09/2020).

[16] *Igraph R Manual Pages*. URL: https://igraph.org/r/doc/sample_pa.html (visited on 10/06/2020).

[17] Sepandar Kamvar, Taher Haveliwala, and Gene Golub. "Adaptive Methods for the Computation of PageRank". In: *Linear Algebra and its Applications*. Special Issue on the Conference on the Numerical Solution of Markov Chains 2003 386 (July 15, 2004), pp. 51–65. ISSN: 0024-3795. DOI: 10.1016/j.laa.2003.12.008. URL: http://www.sciencedirect.com/science/article/pii/S0024379504000023 (visited on 08/19/2020).

[18] Moshe Koppel and Nadav Schweitzer. "Measuring Direct and Indirect Authorial Influence in Historical Corpora". In: *Journal of the Association for Information Science and Technology* 65.10 (2014), pp. 2138–2144. ISSN: 2330-1643. DOI: 10.1002/asi.23118. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.23118 (visited on 08/21/2020).

[19] Amy N. Langville and Carl D. Meyer. "A Reordering for the PageRank Problem". In: *SIAM Journal on Scientific Computing; Philadelphia* 27.6 (2006), p. 9. ISSN: 10648275. DOI: http://dx.doi.org.ezproxy.uws.edu.au/10.1137/040607551. URL: http://search.proquest.com/docview/921138313/abstract/24AFC1417CF6412BPQ/1 (visited on 08/19/2020).

[20] Amy N. Langville and Carl D. Meyer. *Google's PageRank and beyond: The Science of Search Engine Rankings*. Neuaufl. Princeton: Princeton Univ. Press, 2012. 224 pp. ISBN: 978-0-691-15266-0.

[21] Larry Page and Sergey Brin. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". In: *Computer Networks and ISDN Systems* 30.1-7 (Apr. 1, 1998), pp. 107–117. ISSN: 0169-7552. DOI: 10.1016/S0169-7552(98)00110-X. URL: http://www.sciencedirect.com/science/article/pii/S016975529800110X (visited on 08/19/2020).

[22] Ron Larson and Bruce H. Edwards. *Elementary Linear Algebra*. 2nd ed. Lexington, Mass: D.C. Heath, 1991. 592 pp. ISBN: 978-0-669-24592-9.

[23] George Meghabghab and Abraham Kandel. *Search Engines, Link Analysis, and User's Web Behavior: A Unifying Web Mining Approach*. Studies in Computational Intelligence v. 99. Berlin: Springer, 2008. 269 pp. ISBN: 978-3-540-77468-6 978-3-540-77469-3.

[24] Nathanael Ackerman, Cameron Freer,Alex Kruckman, and Rehana Patel. *Properly Erodic Structures*. Oct. 25, 2017. URL: https://math.mit.edu/~freer/papers/properly-ergodic-structures.pdf (visited on 10/10/2020).

[25] Laurence A. F. Park and Simeon Simoff. “Power Walk: Revisiting the Random Surfer”. In: *Proceedings of the 18th Australasian Document Computing Symposium*. ADCS ’13. Brisbane, Queensland, Australia: Association for Computing Machinery, Dec. 5, 2013, pp. 50–57. ISBN: 978-1-4503-2524-0. DOI: 10.1145/2537734.2537749. URL: http://doi.org/10.1145/2537734.2537749 (visited on 07/31/2020).

[26] Alfred Renyi and Erdos Paul. “On Random Graphs I”. In: *Publ. math. debrecen* 6.290-297 (1959), p. 18. URL: http://www.leonidzhukov.net/hse/2016/networks/papers/erdos-1959-11.pdf.

[27] saz. *Probability Theory - Is This Graph Ergodic?* URL: https://math.stackexchange.com/questions/1327283/is-this-graph-ergodic (visited on 10/10/2020).

[28] *The DOT Language*. URL: https://graphviz.org/doc/info/lang.html (visited on 10/09/2020).

[29] Rui Zeng et al. “A Practical Simulation Method for Social Networks”. In: 144 (2013), p. 8.