

Page Rank

Ryan Greenup

October 18, 2020

Contents

1	Introduction	2
1.1	The PageRank Method	3
1.2	Power Walk and the Random Surfer	3
1.3	Stability and Convergence	3
I	Implementing PageRank	3
2	Definitions use in this report	4
2.1	Notation	5
3	Mathematics of Page Rank	6
3.1	The Stationary Distribution of a Probability Transition Matrix	6
3.2	Random Surfer Model	7
3.2.1	Problems with the Stationary Distribution	7
3.2.2	Markov Chains	7
3.2.3	Limitations	10
3.3	Power walk	10
4	Sparse Matrices	11
4.1	Solving the Stationary Distribution	11
5	Implementing the Models	12
5.1	Implementing the Random Surfer	14
5.1.1	Ordinary Matrices	14
5.1.2	Sparse Matrices	18
5.2	Power Walk Method	22
5.2.1	Ordinary Matrices	22
5.2.2	Sparse Matrices	24
6	Creating a Package	30
II	Investigating ξ_2	30

7 Erdos Renyi Graphs	31
7.1 Introduction	31
7.2 Correlation Plot	31
8 Barabasi Albert Graphs	42
8.1 Theory	42
8.2 Modelling	42
8.2.1 Model	45
9 Relating the Power Walk to the Random Surfer	53
9.1 Introduction	54
9.2 Value of [1st Term]	54
9.3 Value of {2nd Term}	55
9.4 Equate the Power Walk to the Random Surfer	56
9.5 Conclusion	57
10 Conclusion	57
11 Appendix	57
11.1 Graph Diagrams	57
12 my to do list	60
12.1 Use BA Graphs	60
12.2 Look at the ScatterPlot Matrix	60
12.3 Compare eigenvalue2 and iterations	60
12.4 Look at using an SVM/logReg or other classifier	60
12.4.1 Measure Iterations and E2 values	60
12.4.2 Are any values indicative of E2 Values?	60
12.4.3 Use Method Parameters with a log reg to predict a number of iterations	60
12.4.4 Use a ROC Curve to set a threshold	60
12.4.5 Discuss the results.	60
12.5 Tie the Report together.	60
12.6 Typeset the Report	60
12.7 TODO Diamater	60
12.8 Improving the Performance of Page Rank	60

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 *network 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 traversal during a random walk.

This approach relies on the assumption that the random walk can:

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

and so the *PageRank* method involves modifying a graph in some way to address these assumptions, specifically by modifying the corresponding probability transition matrix to be stochastic and primitive.

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 approach 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 means that the model treats any link as an endorsement of the destination node, this may not necessarily always be true (consider for example burned-in advertisements or negative reviews). In the past attributing weights to links was not particularly feasible, recent developments in sentiment analysis have 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 approaches are discussed in more detail at § 3.

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 § 5)

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 Definitions use in this report

The following definitions are used throughout this report ¹:

Markov Chains are discrete mathematical model such that future values depend only on current values [11, §1.5], this captures the concept of a random walk because the next destination depends only on the current location.

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 $\text{sum}(\vec{x}) = 1$, so when solving for a stationary vector it can be necessary to perform $\vec{p} \leftarrow \frac{\vec{p}}{\sum \vec{p}}$

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, 8])

- 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], hence the significance of this concept.

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 have 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 irreducible and aperiodic.

¹see generally [20, Ch. 15] for further reading

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

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 term *Random Surfer* is usually used to refer specifically to the smoothing algorithm shown in (5), but *PageRank* refers to the entire concept including the *Random Surfer*. In this report the term *PageRank* is used more generally to denote the concept the model *Random Surfer* or *Power Walk* is denoted specifically where necessary.

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}
 - Is the *transition probability matrix* of a graph, this matrix indicates the probability of a movement during a random walk, such that $T_{i,j}$ is equal to the probability of travelling $j \rightarrow i$ during a random walk.
- $\mathbf{D}_{\mathbf{A}} = \text{diag}(\vec{1}\mathbf{A})$
- $\mathbf{D}_{\mathbf{A}}^{-1} = \begin{cases} 0, & [\mathbf{D}_{\mathbf{A}}]_i = 0 \\ \left[\frac{1}{\mathbf{D}_{\mathbf{A}}}\right], & [\mathbf{D}_{\mathbf{A}}]_i \neq 0 \end{cases}$
 - A diagonal scaling matrix such that $\mathbf{T} = \mathbf{A}\mathbf{D}_{\mathbf{A}}^{-1}$, the piecewise definition is such that $\mathbf{D}_{\mathbf{A}}^{-1}$ 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.
 - * $\mathbf{D}_{\mathbf{A}}^{-1} = \vec{1}\mathbf{D}_{\mathbf{A}}^{-1} = \frac{1}{\vec{1}\mathbf{D}_{\mathbf{A}}}$ for some stochastic matrix \mathbf{A}
- n
 - Refers to the number of vertices in a graph, $n = \text{nrow}(\mathbf{A}) = \text{ncol}(\mathbf{A})$
- $\mathbf{E}_{i,j} = \frac{1}{n}$
 - 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, this size of which should be clear from the context.
 - * 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]
- $\mathbf{J} = \vec{1} \cdot \vec{1}^T \iff \mathbf{J}_{i,j} = 1$
 - A completely dense $n \times n$ matrix containing only 1

- ξ_n
 - The n^{th} largest eigenvalue of \mathbf{T} , the use of λ has been avoided because some authors use λ to represent damping factor α , given that $\alpha=\xi_2$ for certain graphs, this can be very ambiguous.
- α
 - A probability such that $(1 - \alpha)$ represents of teleporting from one vertex to another during a random walk, see 4.
 - * In the literature α is often referred to as a damping factor (see e.g. [5, 7, 12, 17, 6]) or a smoothing constant (see e.g [18]).

3 Mathematics of Page Rank

3.1 The Stationary Distribution of a Probability Transition Matrix

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

$$\mathbf{A}_{i,j} \in \{0, 1\}$$

Where each element of the matrix indicates whether or not travel from vertex j to vertex i is possible with a value of 1.²

During a random walk on a graph 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:

$$\mathbf{T} = \mathbf{A}\mathbf{D}_{\mathbf{A}}^{-1} \quad (1)$$

The value of $\mathbf{D}_{\mathbf{A}}^{-1}$ is such that under matrix multiplication \mathbf{A} will have columns that sum to 1³, this matrix is the *transition probability matrix* \mathbf{T} .

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

$$p_{i+1}^{\vec{}} = \mathbf{T}p_i^{\vec{}} \quad (2)$$

This relationship is a linear recurrence relation, more generally however it is a *Markov Chain* and [20, §4.4] and Finding the Stationary point for this relationship will give a frequency distribution for the nodes and a metric to measure the centrality of vertices.

²Some authors define an adjacency matrix transposed (see e.g. [1, 23]) this unfortunately includes the *igraph* library [13] but that convention will not be followed in this paper

³such a matrix is said to be a *column stochastic matrix*, for a reducible or non-stochastic graph the definition of $\mathbf{D}_{\mathbf{A}}^{-1}$ needs to be piecewise, as shown in § 2

3.2 Random Surfer Model

3.2.1 Problems with the Stationary Distribution

The approach to measuring centrality using a stationary distribution in § 3.1 has the following issues

1. Convergence of (2)
 - (a) Will this relationship converge or diverge?
 - (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 the resulting frequencies of node traversal are not meaningful.
3. Cycles
 - (a) A graph that is cyclical may not converge uniquely
 - i. Consider for example the graph $(A) \longleftrightarrow (B)$ or taking a directed edge into a closed loop.

3.2.2 Markov Chains

The relationship in (2) is a *Markov Chain*⁴ and it is known that the relationship will converge to a value (this is known as the power method):

- 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]

and so these concepts will be explored in order to address the issues with (2).

Stochastic If some 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 (2) will converge, the probability transition matrix must be made stochastic, to achieve this a uniform probability of teleporting from a dead end to any other vertex could be introduced:

$$S = T + \frac{\vec{a} \cdot \vec{1}^T}{n} \quad (3)$$

$$a_i = \begin{cases} 1, & \deg(V_i) = 0 \\ 0, & \deg(V_i) \neq 0 \end{cases} \quad (4)$$

This however would not be sufficient to ensure that (2) would converge, in addition the transition probability matrix must be made irreducible and aperiodic (i.e. primitive). [20]

⁴A *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]

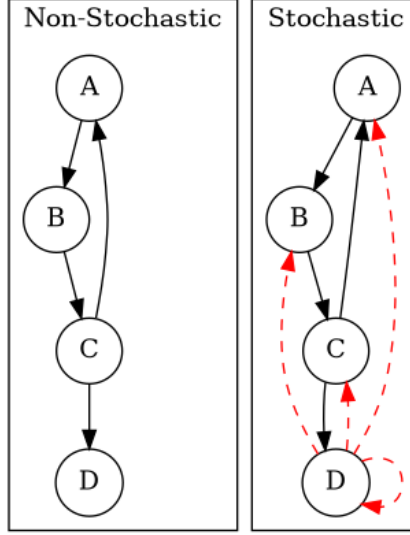


Figure 1: D is a *dangling node*, a dead end during a random walk, the corresponding probability transition matrix (\mathbf{T}) is hence non-stochastic (and also reducible), Introducing some probability of teleporting from a dead end to any other vertex as per (3) (denoted in red) will cause \mathbf{T} to be stochastic.

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 vertices be reached from any initial condition.

Aperiodic An aperiodic graph has only one eigenvalue that lies on the unit circle, this is important because $\lim_{k \rightarrow \infty} \left(\frac{\mathbf{A}^k}{r} \right)$ exists for a non-negative irreducible matrix \mathbf{A} if and only if \mathbf{A} is aperiodic. A graph that is a periodic can be made aperiodic by interlinking nodes ⁵

The Fix To ensure that the transition probability matrix is primitive (i.e. irreducible and aperiodic) as well as stochastic, instead of merely introducing the possibility to teleport out of dead ends, some probability of teleporting to any node at any time can be introduced ($1 - \alpha$), this approach is known as the *Random Surfer* model and the corresponding transition probability matrix is given by [21] :

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

This matrix is primitive and stochastic and so will converge [20, §4.5], it is also unfortunately completely dense, making it resource intensive to work with (see § 4.1).

Using this the relation ship in (2) can now be re expressed as:

$$p_{i+1}^{\rightarrow} \rightarrow \mathbf{S} \vec{p}_i \quad (6)$$

⁵Actually 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 because the graph may still be reducible or non-stochastic)

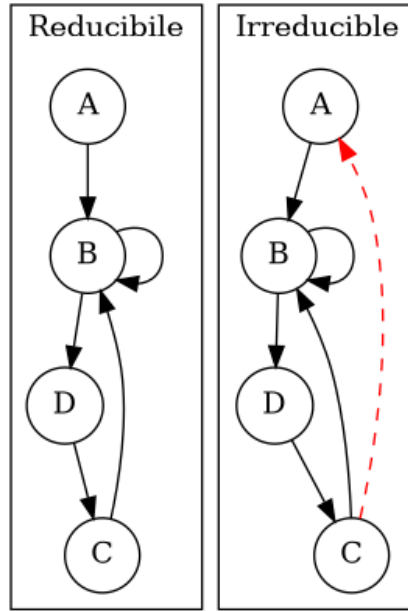


Figure 2: Example of a reducible graph, observe that although C is not a dead end as discussed in § 3.2.2 , there is no way to travel from C to A , by adding an edge such an edge in the resulting graph is irreducible. The resulting graph is also aperiodic (due to the loop on B) and stochastic, so there will be a stationary distribution corresponding to (2).

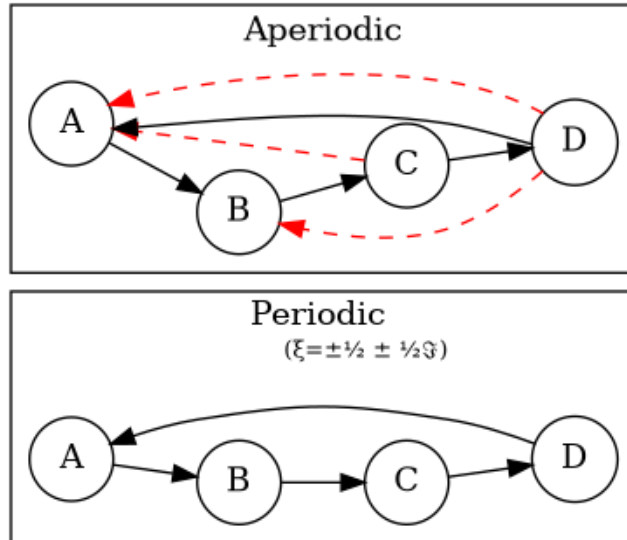


Figure 3: A periodic graph with all eigenvalues on the unit circle $\xi = \frac{\sqrt{2}}{2} e^{\frac{\pi i}{4} k}$, by adding in extra edges the graph is now aperiodic (this does not represent the random surfer or power walk models, which would in theory connect every vertex with some probability)

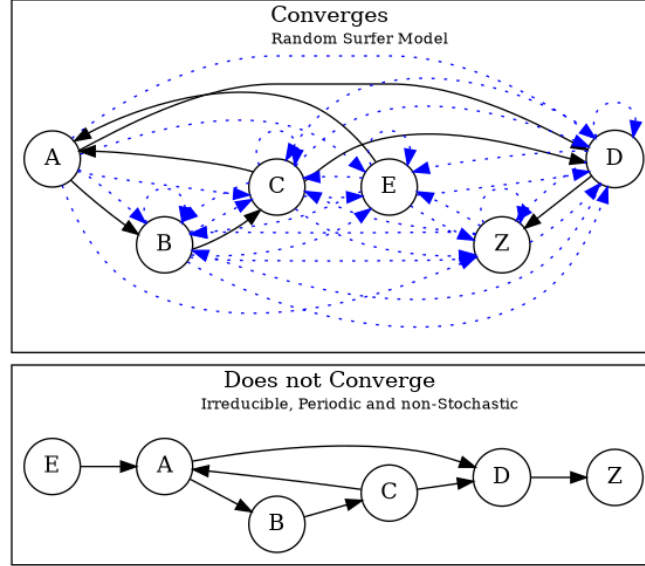


Figure 4: A graph that is aperiodic, reducible and non-stochastic, by applying the random surfer model (5) blue *teleportation* edges are introduced, these may be followed with a probability of $1 - \alpha$

3.2.3 Limitations

The *Random Surfer Model* can only consider positively weighted edges, it cannot take into account negatively weighted edges which might indicate that links promote aversion rather than endorsement.

3.3 Power walk

The *Power Walk* method is an alternative approach to develop a probability transition matrix to use in place of \mathbf{T} in (2) (and \mathbf{S} in (6)).

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 (\mathbf{W}) would be such that the probability of travelling to some vertex $j \rightarrow i$ would be :

$$\mathbf{W}_{i,j} = x\beta^{\mathbf{A}_{i,j}} \quad (7)$$

The random walk is constrained to the graph and so the probability of travelling to one of the vertices generally is 1, hence:

$$1 = \sum_{j=1}^n [x\beta^{\mathbf{A}_{i,j}}] \quad (8)$$

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

Substituting the value of x from (9) into (7) gives the probability as:

$$\mathbf{W}_{i,j} = \frac{\beta^{\mathbf{A}_{i,j}}}{\sum_{i=j}^n [\beta^{\mathbf{A}_{i,j}}]} \quad (10)$$

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 3.2.2 \mathbf{W} will be a primitive stochastic matrix and so if \mathbf{W} was substituted with \mathbf{T} in (2) a solution would exist.

4 Sparse Matrices

Most Adjacency matrices resulting from webpages and analagous networks result in sparse adjacency matrices (see figure 16), 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 alternative forms so as to reduce the memory footprint associated with that matrix, one such method is *Compressed Column Storage*, this involves listing only the non-zero elements as in (11) and (12), this is implemented in **R** with the **Matrix** package [10].

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \phi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \pi \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (11)$$

Row Index	Col Index	Value
1	1	1
3	2	ϕ
4	5	π

(12)

4.1 Solving the Stationary Distribution

The relationship in (2) ⁶ is equivalent to the eigenvalue problem, where $\vec{p} = \lim_{i \rightarrow \infty} (\vec{p}_i)$ is the eigenvector ⁷ \vec{x} that corresponds to the eigenvalue $\xi = 1$:

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

Solving eigenvectors for large matrices can be very resource intensive and so this approach isn't suitable for analysing large networks, it is however an appropriate method to check against and will be implemented in this report for that purpose.

Upon iteration (6) and (10) will converge to stable stationary points, as discussed in 3.2.2, this approach is known as the power method [22] and is what in practice must be implemented to solve the stationary point.

⁶This assumes that the transition probability matrix \mathbf{T} is stochastic and primitive as it would be for \mathbf{S} and \mathbf{W}

⁷More accurately the eigenvector scaled specifically to 1, so it would be more correct to say the eigenvector $\vec{x} / \sum \vec{x}$

As mentioned in §§ 3.2.2 and 3.3, the *Random Surfer* and *Power Walk* transition 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 Implementing the Models

To Implement the models via the power method, first they'll be implemented using an ordinary matrix and then improved to work with sparse matrices and algorithms, the results can be verified against ξ_1 .

The implementation has been performed with *R* and the preamble is provided in listing 1, an exemplar graph was created in listing 2 and shown in figure 5, and the corresponding adjacency matrix provided in listing 3, for the sake of comparison this graph was reproduced from [25].

```

1  if (require("pacman")) {
2      library(pacman)
3  }else{
4      install.packages("pacman")
5      library(pacman)
6  }
7
8  pacman::p_load(tidyverse, Matrix, igraph, plotly, plot3d, mise,
9  ↪ docstring, mise, corrplot, latex2exp)
10 # options(scipen=20) # Resist Scientific Notation

```

Listing 1: Implemented Packages used in this report

```

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

```

Listing 2: Produce exemplar graph in figure 5

	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

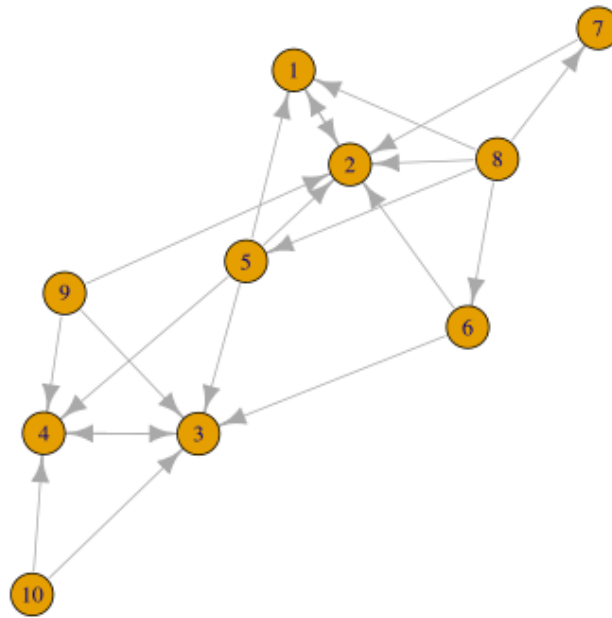


Figure 5: Exemplar graph for *PageRank* examples, produced in listing 2

```
1 A <- igraph::get.adjacency(g1, names = TRUE, sparse = FALSE)
2
3 ## igraph gives back the transpose
4 (A <- t(A))
```

Listing 3: Return the Adjacency Matrix corresponding to figure 5

```

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

```

5.1 Implementing the Random Surfer

5.1.1 Ordinary Matrices

Adjacency Matrix

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 **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 **T**. An alternative approach using sparse matrices will be presented below and in this case there exists corresponding **T** that is stochastic and so it is sufficient to use the relationship at (1), this is shown in listing 4.

```

1 (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

```

1 adj_to_probTrans <- function(A) {
2   A %*% diag(1/colSums(A))
3 }
4
5 (T <- adj_to_probTrans(A)) %>% round(2)

```

Loading required package: usethis
Loading required package: PageRank

Attaching package: 'PageRank'

The following object is masked _by_ '.GlobalEnv':

```
adj_to_probTrans
```

```
##      [,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 3.2.2 the following variables of the *Random Surfer* model:

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

$$(15)$$

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

$$n = ||V|| \quad (17)$$

$$\alpha \in [0, 1] \quad (18)$$

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

```
1 B <- matrix(rep(1/nrow(T), length.out = nrow(T)**2), nrow = nrow(T))
2 l <- 0.8123456789
3
4 (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

```
[1] 1.00000000e+00+0.0000000e+00i -8.12345679e-01+0.0000000e+00i
```

```

1 print(eigen(S, symmetric = FALSE, only.values = TRUE)$values, 9)
2 print(eigen(S, symmetric = FALSE)$vectors, 3)

```

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

```

[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):

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


```

1 (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
```

```

1 (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):

```

1 (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:

```

1 p_new <- p_new *123456789
2
3 while (sum(round(p, 9) != round(p_new, 9))) {
4   (p      <- p_new)
5   (p_new <- S %*% p)
6 }
7
8 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:

```

1 (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:

```

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

```

```

3
4   while (sum(round(p, 9) != round(p_new, 9))) {
5       (p      <- p_new)
6       (p_new <- S %*% p)
7   }
8
9   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

```

5.1.2 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 4

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

```

1 library(Matrix)
2 ## Create Example Matrix
3 n <- 20
4 m <- 10^6
5 i <- sample(1:m, size = n); j <- sample(1:m, size = n); x <- rpois(n,
  ↪ lambda = 90)
6 A <- sparseMatrix(i, j, x = x, dims = c(m, m))
7
8 summary(A)

```

```

1000000 x 1000000 sparse Matrix of class "dgCMatrix", with 20 entries
      i      j      x
1 141572 65888 92
2 547799 69135 87
3 368656 123865 87
4 881320 129763 111
5 53637 154979 92
6 193808 192238 100
7 467415 260074 86
8 28105 276311 79
9 481097 316591 102
10 559159 319674 93
11 927895 322174 77

```

```

12 562619 372818 82
13 123000 391022 80
14 75909 417462 70
15 309593 457917 78
16 434992 521070 101
17 617821 769436 93
18 673173 811478 103
19 860284 841473 104
20 734100 852938 83

```

As before in section 5.1.1, the probability transition matrix can be found by:

1. Creating adjacency matrix

- (a) Transposing as necessary such that $A_{i,j} \neq 0$ 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 (??) may be implemented, using sparse matrices has the advantage however that only non-zero elements will be operated on, meaning that columns that sum to zero can still be used to create a probability transition matrix⁸ practice 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:

```

1 sparse_diag <- function(mat) {
2
3   ## Get the Dimensions
4   n <- nrow(mat)
5
6   ## Make a Diagonal Matrix of Column Sums
7   D <- sparseMatrix(i = 1:n, j = 1:n, x = colSums(mat), dims = c(n,n))
8
9   ## Throw away explicit Zeroes
10  D <- drop0(D)
11
12  ## Inverse the Values
13  D@x <- 1/D@x
14
15  ## Return the Diagonal Matrix
16  return(D)
17 }

```

Applying this to the previously created sparse matrix:

⁸Although this matrix may still have columns that sum to zero and will hence be non-stochastic

```

1 D <- sparse_diag(t(A))
2 summary(D)

```

```

1000000 x 1000000 sparse Matrix of class "dgCMatrix", with 20 entries
      i      j      x
1  28105  28105 0.012658228
2  53637  53637 0.010869565
3  75909  75909 0.014285714
4 123000 123000 0.012500000
5 141572 141572 0.010869565
6 193808 193808 0.010000000
7 309593 309593 0.012820513
8 368656 368656 0.011494253
9 434992 434992 0.009900990
10 467415 467415 0.011627907
11 481097 481097 0.009803922
12 547799 547799 0.011494253
13 559159 559159 0.010752688
14 562619 562619 0.012195122
15 617821 617821 0.010752688
16 673173 673173 0.009708738
17 734100 734100 0.012048193
18 860284 860284 0.009615385
19 881320 881320 0.009009009
20 927895 927895 0.012987013

```

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

```

1 summary((T <- t(A) %*% D))

```

```

1000000 x 1000000 sparse Matrix of class "dgCMatrix", with 20 entries
      i      j x
1 276311  28105 1
2 154979  53637 1
3 417462  75909 1
4 391022 123000 1
5  65888 141572 1
6 192238 193808 1
7 457917 309593 1
8 123865 368656 1
9 521070 434992 1
10 260074 467415 1
11 316591 481097 1
12  69135 547799 1
13 319674 559159 1
14 372818 562619 1
15 769436 617821 1
16 811478 673173 1
17 852938 734100 1
18 841473 860284 1
19 129763 881320 1
20 322174 927895 1

```

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

However, creating a matrix of background probabilities (denoted by B in section 5.1.1) 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:

$$\vec{p} = \mathbf{S}\vec{p} \quad (19)$$

where:

$$S = \alpha \mathbf{T} + (1 - \alpha) \mathbf{B} \quad (20)$$

$$\vec{p} = (\alpha \mathbf{T} + (1 - \alpha) \mathbf{B}) \vec{p} \quad (21)$$

$$= \alpha \mathbf{T} \vec{p} + (1 - \alpha) \mathbf{B} \vec{p} \quad (22)$$

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

$$\mathbf{F} = \begin{bmatrix} \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \end{bmatrix} \begin{bmatrix} \vec{p}_1 \\ \vec{p}_2 \\ \vdots \\ \vec{p}_m \end{bmatrix} \quad (23)$$

$$= \begin{bmatrix} (\sum_{i=0}^m [p_i]) \times \frac{1}{N} \\ (\sum_{i=0}^m [p_i]) \times \frac{1}{N} \\ \vdots \\ (\sum_{i=0}^m [p_i]) \times \frac{1}{N} \end{bmatrix} \quad (24)$$

$$\text{Probabilities sum to 1 and hence:} \quad (25)$$

$$= \begin{bmatrix} \frac{1}{N} \\ \frac{1}{N} \\ \frac{1}{N} \\ \vdots \\ \frac{1}{N} \end{bmatrix} \quad (26)$$

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

```
1  ## Find Stationary point of random surfer
2  N      <- nrow(A)
3  alpha  <- 0.85
4  F      <- rep((1-alpha)/N, nrow(A))  ## A nx1 vector of (1-alpha)/N
5
6  ## Solve using the power method
7  p      <- rep(0, length.out = ncol(T)); p[1] <- 1
8  p_new  <- alpha*T %*% p + F
9
```

```

10  ## use a Counter to debug
11  i <- 0
12  while (sum(round(p, 9) != round(p_new, 9))) {
13      p      <- p_new
14      p_new <- alpha*T %*% p + F
15      (i <- i+1) %>% print()
16  }
17
18  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

```

5.2 Power Walk Method

Recall from 3.3 that the power walk is given by:

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

5.2.1 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

```

1  A <- igraph::get.adjacency(g1, names = TRUE, sparse = FALSE)
2
3  ## * Function to create Prob Trans Mat
4  adj_to_probTrans <- function(A, beta) {
5      B      <- A
6      B      <- beta^A      # Element Wise exponentiation
7      D      <- diag(colSums(B)) # B is completely dense so D = 0
8      D_in   <- solve(D)      # Solve returns inverse of matrix
9      W      <- B %*% D_in
10
11     return(as.matrix(W))
12 }
13
14 beta <- 0.867
15 (W <- adj_to_probTrans(A, beta = 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:

```
1 eigen(W, only.values = TRUE)$values %>% round(9)
2 eigen(W)$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 5.1.1 the relationship between the second eigenvalue and the model parameters is not as clear, this provides that the
Use the power method

```

1  ## * Power Method
2  p      <- rep(0, nrow(W))
3  p[1] <- 1
4  p_new  <- rep(0, nrow(W))
5  p_new[2] <- 1
6
7  while (sum(round(p, 9) != round(p_new, 9))) {
8      (p      <- p_new)
9      (p_new <- W %*% p)
10 }
11
12
13 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
```

5.2.2 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.

- $\mathbf{F} = \begin{bmatrix} \frac{1}{n} \\ \frac{1}{n} \\ \frac{1}{n} \\ \vdots \\ \frac{1}{n} \end{bmatrix}$

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 $\mathbf{D}_{\mathbf{A},i,j}^T := \text{diag}\left(\frac{1}{\text{colsums}(\mathbf{A})}\right)$ 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 $D_{(\mathbf{A}+\mathbf{Z})}^{-1}$ this will be consistent with the *Random Surfer* the method using \mathbf{F} in $[[\#eq:sparse-RS]]$ (5.2.2)

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:

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

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 (5.2.2) at 5.2.2):

$$\mathbf{O} \mathbf{D}_{\mathbf{B}}^{-1} \vec{p}_i = (\vec{\delta}^T \vec{p}_i) \vec{1} \quad (28)$$

$$= \text{repeat}(\vec{p} \bullet \vec{\delta}^T, \mathbf{n}) \quad (29)$$

$$(30)$$

where:

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

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

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

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

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

This means we can do:

$$\vec{p}_{i+1} = \mathbf{T}_{\text{pw}} \vec{p}_i \quad (31)$$

$$= \mathbf{B} \mathbf{D}_{\mathbf{B}}^{-1} \vec{p}_i \quad (32)$$

$$= (\mathbf{B} - \mathbf{O} + \mathbf{O}) \mathbf{D}_{\mathbf{B}}^{-1} \vec{p}_i \quad (33)$$

$$= \left((\mathbf{B} - \mathbf{O}) \mathbf{D}_{\mathbf{B}}^{-1} + \mathbf{O} \mathbf{D}_{\mathbf{B}}^{-1} \right) \vec{p}_i \quad (34)$$

$$= (\mathbf{B} - \mathbf{O}) \mathbf{D}_{\mathbf{B}}^{-1} \vec{p}_i + \mathbf{O} \mathbf{D}_{\mathbf{B}}^{-1} \vec{p}_i \quad (35)$$

$$= (\mathbf{B} - \mathbf{O}) \mathbf{D}_{\mathbf{B}}^{-1} \vec{p}_i + \vec{1} (\vec{\delta}^T \vec{p}_i) \quad (36)$$

$$= (\mathbf{B} - \mathbf{O}) \mathbf{D}_{\mathbf{B}}^{-1} \vec{p}_i + \text{rep}(\vec{\delta}^T \vec{p}_i) \quad (37)$$

where:

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

$$\vec{p_{i+1}} = \mathbf{B}_O \mathbf{D}_B^{-1} \vec{p_i} + \text{rep}(\vec{\delta^T} \vec{p_i})$$

Now solve D_B^{-1} in terms of \mathbf{B}_O :

$$\mathbf{B}_O = (\mathbf{B} - \mathbf{O}) \quad (38)$$

$$\mathbf{B} = \mathbf{B}_O + \mathbf{O} \quad (39)$$

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

$$\delta_B^{-1} = \vec{1} \mathbf{B} \quad (40)$$

$$= \vec{1} (\mathbf{B}_O + \mathbf{O}) \quad (41)$$

$$= \vec{1} \mathbf{B}_O + \vec{1} \mathbf{O} \quad (42)$$

$$= \vec{1} \mathbf{B}_O + \langle n, n, n, \dots n \rangle \quad (43)$$

$$= \vec{1} \mathbf{B}_O + \vec{1} n \quad (44)$$

$$\delta_B = 1 / (\text{colSums}(\mathbf{B}_O) + n) \quad (45)$$

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

$$\begin{aligned} D_B^{-1} &= \text{diag}(\delta_B^{-1}) \\ &= \text{diag}(\text{ColSums}(\mathbf{B}_O) + n)^{-1} \end{aligned}$$

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

$$p_{i+1} = \mathbf{B}_O \text{diag}(\vec{1} \mathbf{B}_O + \vec{1} n) \vec{p_i} + \vec{1} \delta^T \vec{p_i} \quad (46)$$

in terms of \mathbf{R} :

```
1 p_new <- B0 %*% diag(colSums(B)+n) %*% p + rep(t() %*% p, n)
2
3 # It would also be possible to sum the element-wise product
4 (t() %*% p) == sum( * p)
5
6 # Because R treats vectors the same as a nX1 matrix we could also
7 # perform the dot product of the two vectors, meaning the following
8 # would be true in R but not true generally
9
10 (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

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

Then we have:

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

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
$\vec{\delta}^T \vec{p}_i$	$\vec{\delta}^T \vec{p}_i$

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 $\mathbf{D}_B^{-1} = \mathbf{B} \text{ diag} \left(\frac{1}{\mathbf{1}\mathbf{B}} \right)$ because $\beta^{A_{i,j}} \neq 0 \quad \forall A_{i,j}$, this is convenient but in any case the `sparse_diag` function in listing ?? will still work.

Power Walk

1. Define B

```
1  A      <- Matrix::Matrix(A, sparse = TRUE)
2  B      <- A
3  B@x    <- ~(A@x)
4  B      <- A
5  B      <- ~A
6
7  Bo     <- A
8
9  # These two approaches are equivalent
10 Bo@x   <- ~(A@x) -1 # This in theory would be faster
11 # Bo    <- ~(A) -1
12 # Bo    <- drop0(Bo)
13
14
15 n <- nrow(A)
```

```
1  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
```

```
1  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 $\delta_B = \text{colSums}(B_o) + n$ being 0:

```
1 ( 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

```

```
1 ( 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

```

1 DB    <- diag( B)
2 ## ** Create the Transition Probability Matrix
3 ## Create the Trans Prob Mat using Power Walk
4 T <- Bo %*% DB

```

4. Implement the Loop

```

1 ## ** Implement the Power Walk
2 ## *** Set Initial Values
3 p_new <- rep(1/n, n) # Uniform
4 p      <- rep(0, n)   # Zero
5      <- 10^(-6)
6 ## *** Implement the Loop
7
8 while (sum(abs(p_new - p)) > ) {

```

```

9      (p <- as.vector(p_new)) # P should remain a vector
10     sum(p <- as.vector(p_new)) # P should remain a vector
11     p_new <- T %*% p + rep(t( B) %*% p, n)
12   }
13   ## ** Report the Values
14   print(paste("The stationary point is"))
15   print(p)

```

```

[1] "The stationary point is"
[1] 0.10153165 0.10159353 0.09609664 0.09725146 0.10153165 0.10008449
[7] 0.09865795 0.10157347 0.10155286 0.10012631

```

6 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 .

```

1  library(devtools)
2  library(Matrix)
3  library(tidyverse) # Maybe, TODO check if this is used, I don't think
   ↪ it is
4
5  if (require("PageRank")) {
6    library(PageRank)
7  }else{
8    devtools::install_github("ryangreenup/PageRank")
9    library(PageRank)
10 }

```

Listing 8: Load the *PageRank* package which consists of the functions from 5

```

Loading required package: usethis
Loading required package: PageRank

```

```

Attaching package: 'PageRank'

```

Part II

Investigating ξ_2

7 Erdos Renyi Graphs

7.1 Introduction

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 8) 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.

7.2 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 input values were specified listing 9, a function that builds a data frame with the second eigenvalue, density, determinant and trace was constructed in listing ??, a function to map this over the cartesian product of the input variables was created in listing 10 and finally a correlation plot was generated in listing 11 shown in figure 6. The correlation plot in figure 6 considers the Spearman correlation coefficient, which is concerned with a monotone relationship between the variables, this is appropriate here as the complex relationship between the variables is likely non-linear.

```
1 # Generate Constants
2 p      <- seq(from = 0.01, to = 0.99, length.out = 5)
3 beta   <- seq(from = 1, to = 20, length.out = 20)
4 size    <- seq(from = 100, to = 1000, length.out = 5) %>% rev() #
   ↪ Big First
5 input_var <- expand.grid("p" = p, "beta" = beta, "size" = size)
6
7 # Print out a sample of all the rows
8 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.

	p	beta	size
237	0.255	8	550
13	0.500	3	1000
438	0.500	8	100
456	0.010	12	100
209	0.745	2	550
384	0.745	17	325

$\text{mean}(\mathbf{A}), |\mathbf{A}|, \text{tr}(\mathbf{A})$) corresponding to the *Power Walk* method using the PageRank package discussed at 6 .

```

1 random_graph <- function(p, beta, size) {
2   g1      <- igraph::erdos.renyi.game(n = size, p)
3   A       <- igraph::get.adjacency(g1) # Row to column
4   A       <- Matrix::t(A)
5
6   # A_dens <- mean(A) # Very Slow, equal to p
7   T       <- PageRank::power_walk_prob_trans(A, beta = beta)
8   tr      <- sum(diag(T))
9   e2      <- eigen(T, only.values = TRUE)$values[2] # R orders by
10  ↪ descending magnitude
11  return(c(abs(e2), tr))
12 }

```

```

1 sim_graphs <- function(filename, p, beta, size) {
2
3   input_var <- expand.grid("p" = p , "beta" = beta, "size" = size)
4
5   nc      <- length(random_graph(1, 1, 1))
6   Y       <- matrix(ncol = nc, nrow = nrow(input_var))
7   for (i in 1:nrow(input_var)) {
8     X      <- as.vector(input_var[i,])
9     Y[i,] <- random_graph(X$p, X$beta, X$size)
10    print(i/nrow(input_var))
11  }
12  if (sum(abs(Y) != abs(Re(Y))) == 0) {
13    Y      <- Re(Y)
14  }
15  Y      <- as.data.frame(Y); colnames(Y) <- c("eigenvalue2", "trace")
16  data <- cbind(input_var, Y)
17  saveRDS(data, file = filename)
18  return(data)
19 }

```

Listing 10: A function to return a dataframe of simulated graphs using the `random_graph` function in listing ??.

The correlation plot shows a moderate relationship between size, the uniform probability of linking two nodes (p) and the size of the graph with ξ_2 . It's worth noting that $p = \text{mean}(\mathbf{A})$.

To inspect these relationships more closely a scatterplot matrix was produced in figure 12 and shown in figure 7. The matrix indicates a negative relationship between ξ_2 with p and size and a positive relationship between β and ξ_2 , these relationships also indicate some degree of interaction.

There also appears to be some relationship between β and the trace of the matrix, this won't be considered for the moment because the trace is not a parameter of the power walk method.

By looking at the ξ_2 value for a variety of p values with a constant size a pattern may emerge, a corresponding data set is produced in 13, a plot produced in listing 14 and the corresponding plot shown


```

1 filename <- "resources/erdosData.rds"
2
3 if (file.exists(filename)) {
4
5     data <- readRDS(filename)
6
7 } else {
8     data <- sim_graphs(filename, p, beta, size)
9
10 }
11 cormat = cor(data, method = 'spearman')
12 rownames(cormat) <- colnames(cormat) <- c("Prob\nEdges", " ", "Size",
13     ↪ " ", "Trace")
14 corplot(cormat, method = "ellipse", type = "lower")

```

Listing 11: 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.

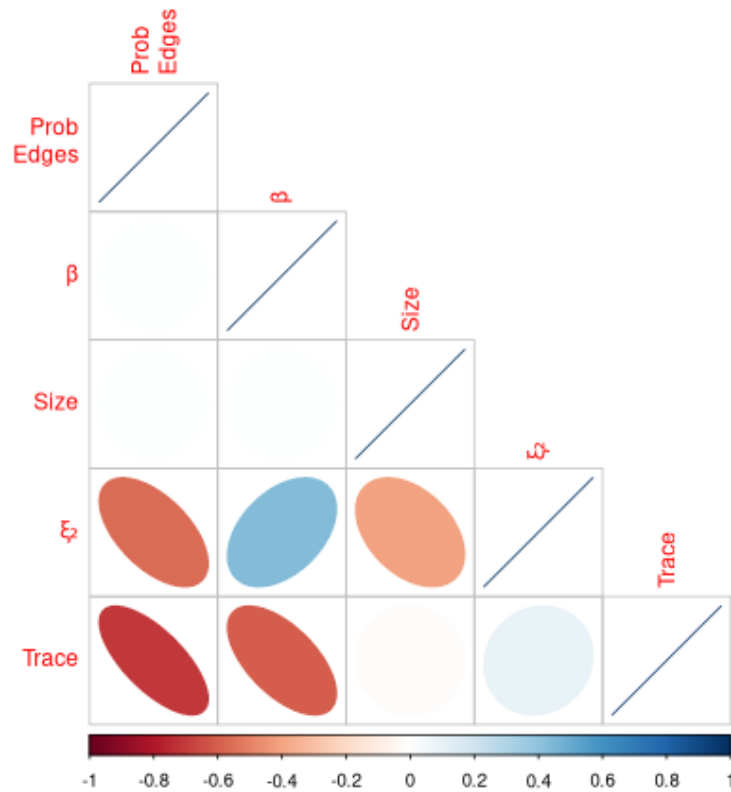


Figure 6: Correlation plot of parameters of *Power Walk* parameters and graph properties.

```
1 plot(data, labels = c("Mean\nEdges", " ", "Size", " ", "Trace"))
```

Listing 12: Density of Plot or something, see 7

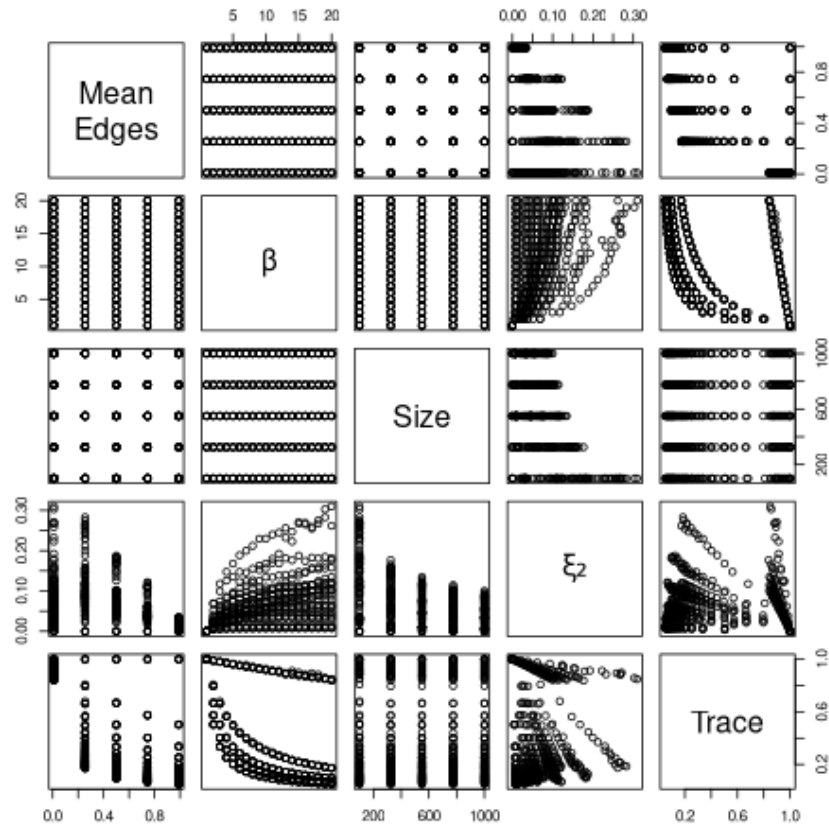


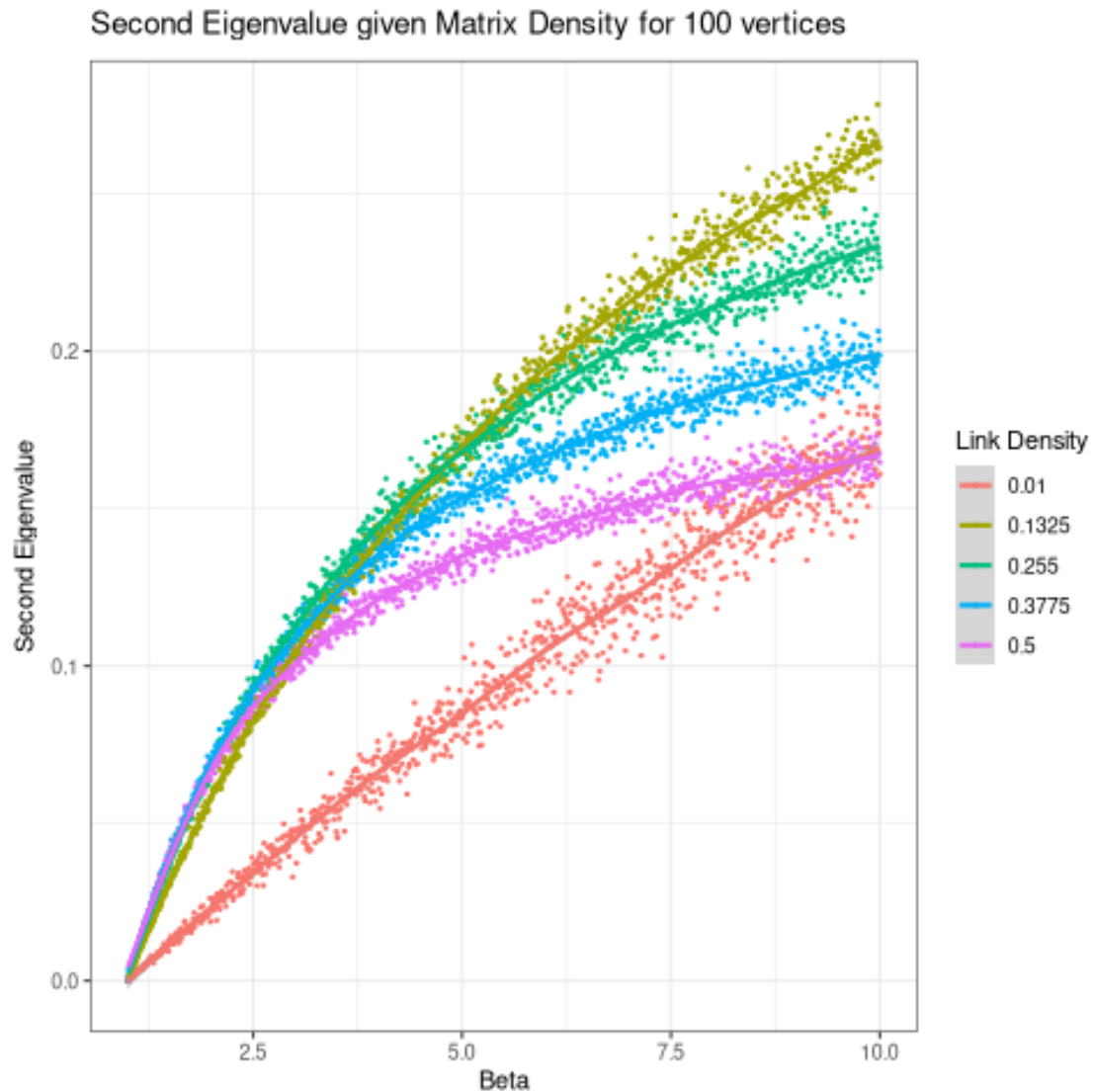
Figure 7: Plot of the Density of the Adjacency Matrix, Beta Value of the Power Walk method with ξ_2 represented by the vertical z axis.

in figure 8.

```
1 filename <- "resources/erdosData_constant_size.rds"
2
3 if (file.exists(filename)) {
4   data2 <- readRDS(filename)
5 } else {
6   p <- seq(from = 0.01, to = 0.5, length.out = 5 )
7   beta <- seq(from = 1 , to = 10 , length.out = 1000)
8   size <- 1000
9
10  data2 <- sim_graphs(filename, p, beta, size)
```

Listing 13: Produce a data frame of graphs corresponding to a constant size and link density.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
26
```



This resulting trends are well defined and positive, the data appears to have a concave down curvature and non constant variance, a sqrt transform is applied in listing 15 and shown in figure .

```

1 ggplot(data2[30:nrow(data2),], mapping = aes(col = factor(p), x =
  ↪ sqrt(beta), y = eigenvalue2)) +
2   geom_point(size = 0.5) +
3   stat_smooth() +
4   scale_size_continuous(range = c(0.1,1)) +
5   labs(x = "sqrt()", y = TeX("Second Eigenvalue"), title = TeX("Second
  ↪ Eigenvalue given Matrix Density for 100 vertices") ) +
6   guides(col = guide_legend("Link Density")) +
7   theme_bw()

```

Listing 15: listing:constant_{sizeerdosdensitysqrt}

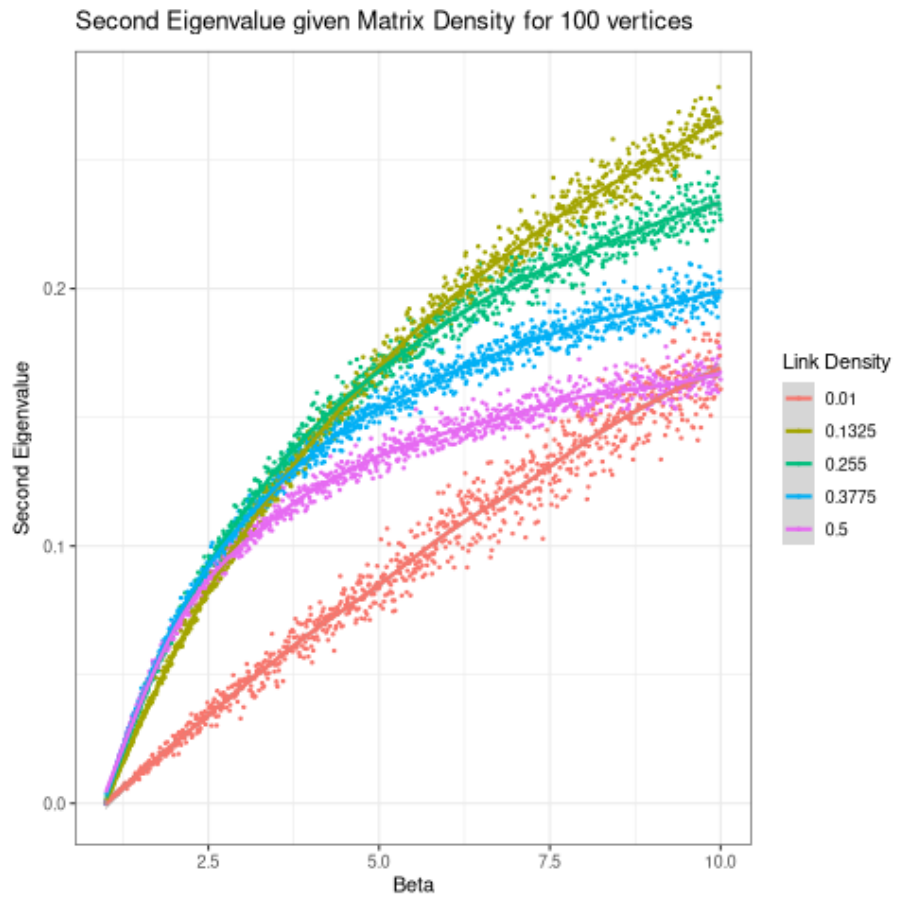


Figure 8: Constant Size Erdos Density

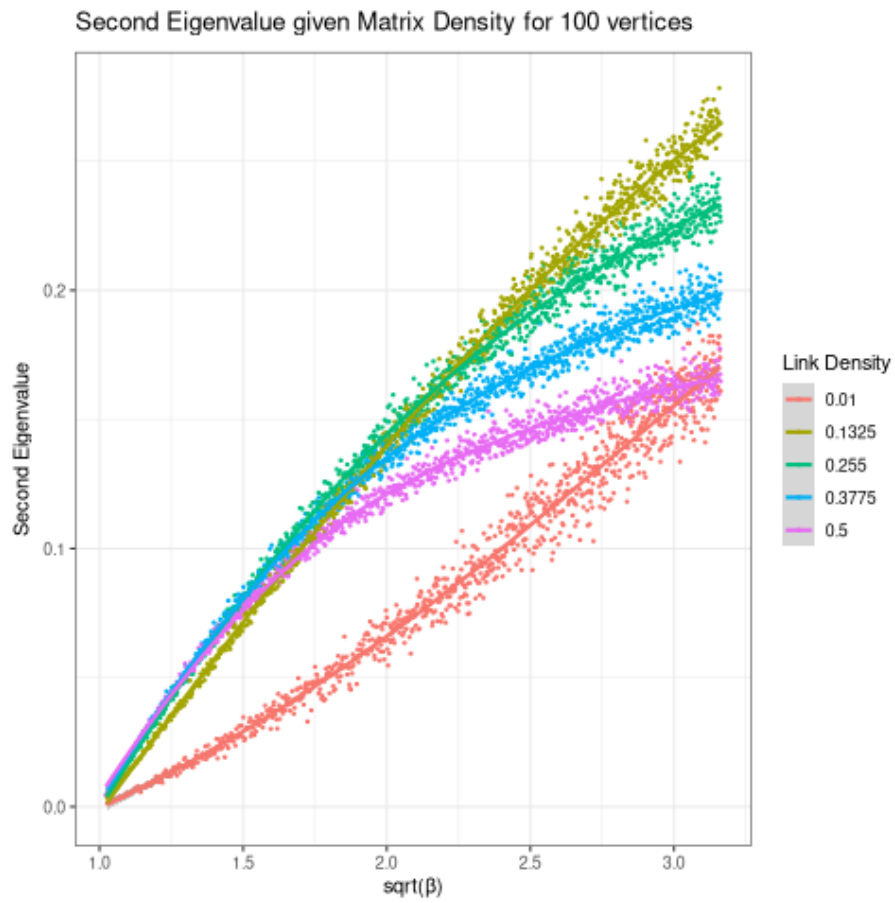


Figure 9: fig:constant_{sizeerdosdensitysqrt}

The root transform did not significantly normalise the variance or make the relationship much more linear. The noise would result from using p as a probability in forming the graphs as opposed to strictly enforcing the link density, this variance should be constant so this transform is not desirable.

Both plots in figures 9 8 appear to indicate a more linear relationship between ξ_2 and β for smaller link densities and β values.

In order to determine whether or not this relationship persists over many sizes a data set containing the relationship between ξ_2 and β was produced for sizes ranging from 100 to 1000 vertices in listing 16, a corresponding plot was produced in listing 17 and shown in figure 10.

```

1 filename <- "resources/erdosData_constant_dens.rds"
2
3 if (file.exists(filename)) {
4   data2 <- readRDS(filename)
5 } else {
6   p <- 5/100
7   beta <- seq(from = 1 , to = 10 , length.out = 1000)
8   size <- seq(from = 100, to = 1000, length.out = 5 )
9
10  data2 <- sim_graphs(filename, p, beta, size)
11 }

```

Listing 16: Produce a data set of a variety of sizes ranging from 100 to 1000 vertices.

```

1 ggplot(data2, mapping = aes(col = factor(size), x = (beta), y =
  ↪ eigenvalue2)) +
2   geom_point(size = 0.5) +
3   stat_smooth() +
4   scale_size_continuous(range = c(0.1,1)) +
5   labs(x = " ", y = TeX("Second Eigenvalue"), title = TeX("Second
  ↪ Eigenvalue for uniform degree"), subtitle = "mean degree ÷ size =
  ↪ 5%" ) +
6   guides(col = guide_legend("Size")) +
7   theme_bw()

```

Listing 17: listing:constant_{size}erdosdensity

This relationship appears to have a slight curvature, this curvature is very slight so a root transform is more appropriate than a log transform, this is implemented in listing 18 and shown in 11.

The plot shown in 11 demonstrates a very strong linear relationship between ξ_2 and $\sqrt{\beta}$ for a variety of constant link densities, the variance however is still non-constant over $\sqrt{\beta}$.

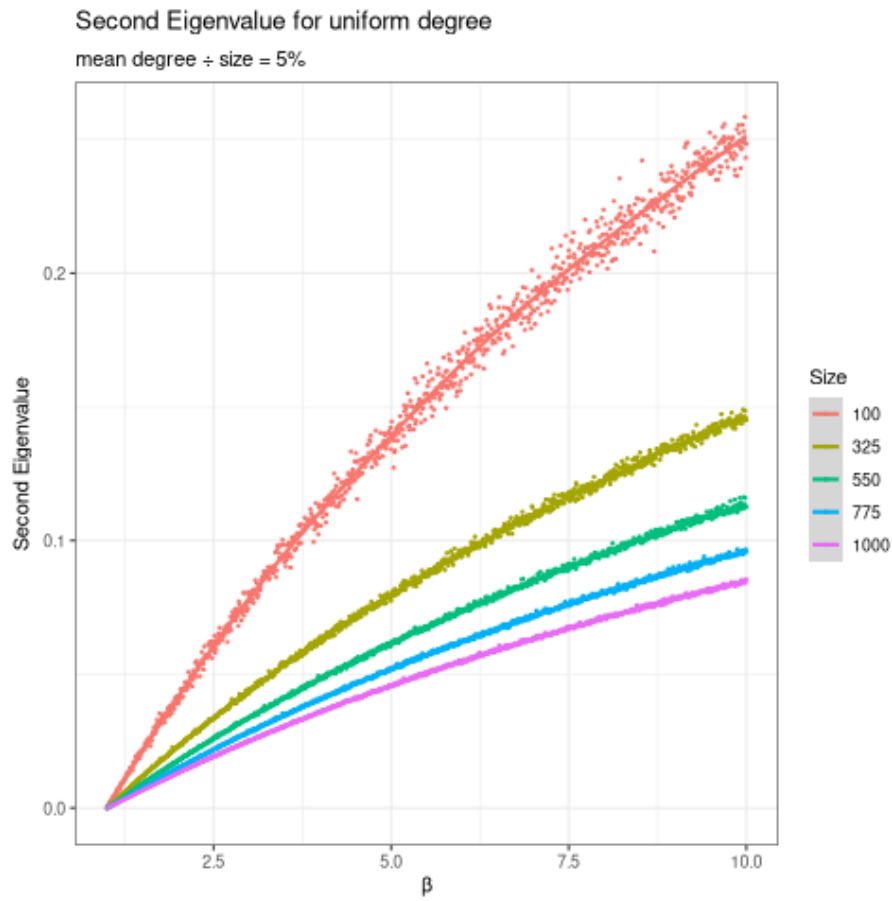


Figure 10: fig:constant_{denserdosdensity}

```

1  ggplot(data2, mapping = aes(col = factor(size), x = sqrt(beta), y =
   ↪ eigenvalue2)) +
2  geom_point(size = 0.5) +
3  stat_smooth() +
4  scale_size_continuous(range = c(0.1,1)) +

5  labs(x = TeX("\\sqrt{ }"), y = TeX("Second Eigenvalue"), title =
   ↪ TeX("Second Eigenvalue for uniform degree"), subtitle = "mean
   ↪ degree ÷ size = 5%" ) +
6  guides(col = guide_legend("Size")) +
7  theme_bw()

```

Listing 18: listing:constant_{sizeerdosdensitysqrt}

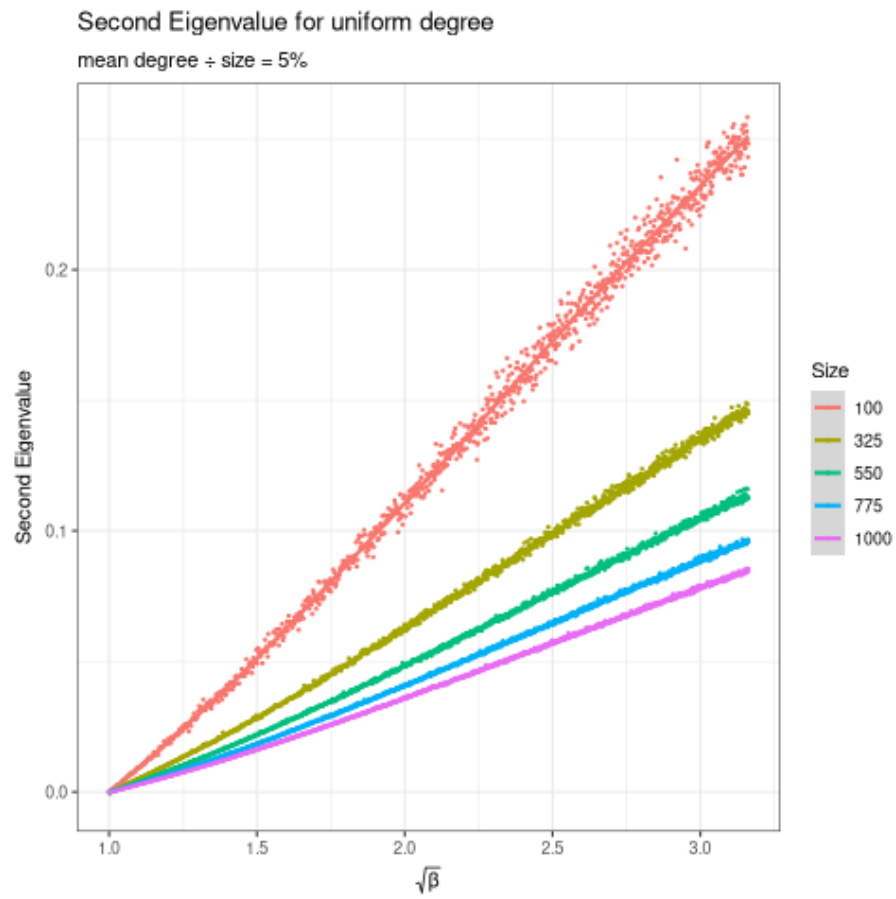


Figure 11: fig:constant_{denserdosdensitysqrt}

8 Barabasi Albert Graphs

8.1 Theory

The *Erdos Renyi* game is a random network, a superior approach to model the web is to use a scale free networks [3] such as the Barabasi-Albert graph [4]

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. [2, Ch. 7]

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

$$n \propto k^{-\gamma}, \quad \exists k \in \mathbb{R} \quad (47)$$

The Barabassi Albert model implements this scale free approach and predicts that γ values of 3 would fit the model, although, in practice values of in = 2.1 and out = 2.45 are observed. This value would be independent of m . also predicts that $A \propto m^2$: [4]

$$P(k) \propto m^2 k^{-\gamma}$$

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](#).

8.2 Modelling

At each step of the Barabassi-Albert algorithm, one node is added and linked to m other nodes with a probability proportional to the degree of that vertex.

This means that the average of the unweighted adjacency matrix would be given by $p = \text{mean}(\mathbf{A}) \in \left[\frac{m}{n}, \frac{m+1}{n}\right]$ and the relationship identified in 7 can be determined from the parameters of a scale-free network.

```

1 layout(matrix(1:2, nrow = 2))
2 col <- "Mediumpurple"
3 n <- 1000
4 hist(
5   igraph::degree(igraph::sample_pa(n, 0.2)),
6   binwidth = 0.3,
7   xlab = "",
8
9   main = "Barabassi-Albert Degree Distribution",
10  col = col, freq = FALSE
11 )
12 hist(igraph::degree(igraph::erdos.renyi.game(n, 0.2)),
13      main= "Erdos-Renyi Degree Distribution",
14      col = col,
15
16      binwidth = 0.3,
17      xlab = "",
18      freq = FALSE )

```

Listing 19: Simulate Erdos-Renyi and Barabassi-Albert graphs in order to measure the degree distribution, shown in 12

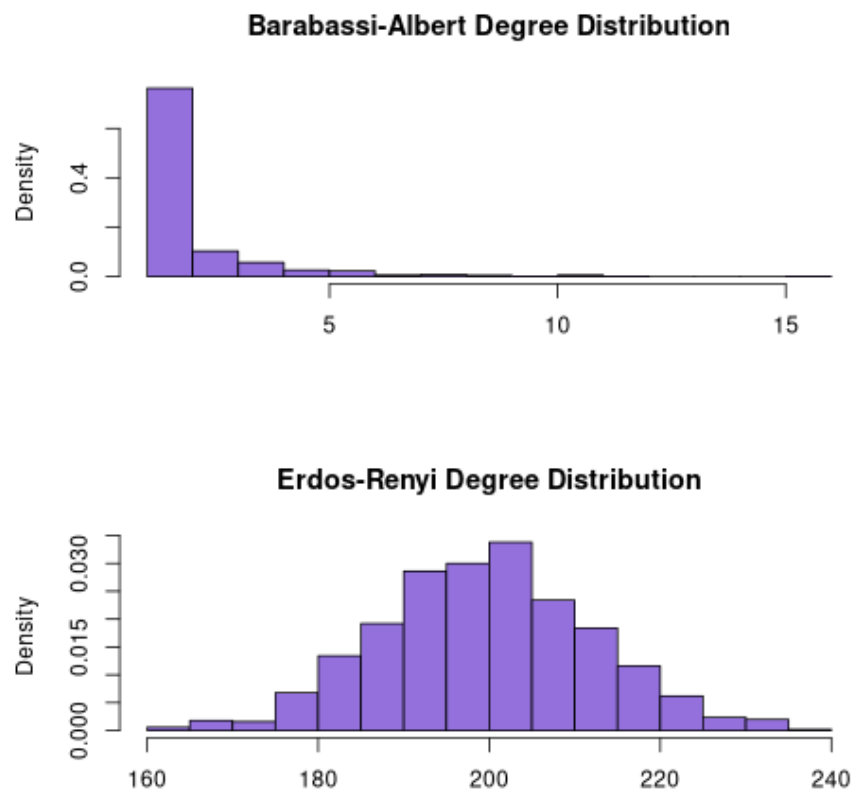


Figure 12: histograms of degree distribution of Erdos-Renyi and Barabassi-Albert graphs produced in listing 19

In practice though measuring m (as the mean out degree) will be imprecise and the adjacency matrix will be weighted any way, so taking the mean of the weighted adjacency matrix would still be necessary to implement this type of model in practice.

A function to generate a random Barabasi-Albert graph and return the value of ξ_2 corresponding to the Power Walk method is provided in listing 20, another function to map this over the cartesian product of input variables is provided in listing 21 and finally a dataset containing 9 different link densities was created in listing 8.2 .

This data plotted by listing 22 and shown in 13.

```

1 random_graph_pa <- function(m, beta, size) {
2   g1 <- igraph::sample_pa(n = size, power = 3, m = m)
3   A <- igraph::get.adjacency(g1) # Row to column
4   A <- Matrix::t(A)
5
6   # A_dens <- mean(A)
7   T <- PageRank::power_walk_prob_trans(A, beta = beta)
8   tr <- sum(diag(T))
9   e2 <- eigen(T, only.values = TRUE)$values[2] # R orders by
   ↪ descending magnitude
10  return(c(abs(e2), tr))
11 }
```

Listing 20: A function to build a random graph using the Barabassi-Albert Model and return the value of ξ_2 corresponding to the *Power Walk* method.

Due to the relationship between size and density in the barabassi albert model this approach seems to work even better for BA graphs, which is precisely the type of graph where we might anticipate the use of the power walk method.

The low link densities are such that there is no need for a log/root transform either, such a transform was found to be unsuitable in § 7.2 so this is encouraging.

To determine the relationship that the link density p may have, the value of β will be fixed and the behaviour of $\xi_2 \sim p$ can be investigated. A corresponding data set is produced in listing 23, plotted in listing 24 and shown in figure 14.

This seems to suggest that the effect on the eigenvalue that p has depends on the size of the graph, so p and size interact with respect to ξ_2 .

Hence an appropriate model will consider the value of p , $p * n$ and β , where n is the size of the graph.

8.2.1 Model

A simple model to consider the variables found to influence ξ_2 is multiple linear regression, although it is a very simple model it is easy to interpret and may offer insights into the behaviour of ξ_2 in response to different graphs.

More data was generated and a model fitted to training data in listing 25 a corresponding residual histogram for the testing data is provided in 15 and a summary of the model provided in listing 26.

In fitting the model in listing 25 the identity $p = \frac{m}{n}$ was implemented because taking the mean value of a large matrix is resource intensive and would have reduced the number of data points in the analysis.

Call:

```

1 sim_graphs_pa <- function(filename, m, beta, size) {
2
3   input_var <- expand.grid("m" = m, "beta" = beta, "size" = size)
4
5   nc <- length(random_graph_pa(1, 1, 1))
6   Y <- matrix(ncol = nc, nrow = nrow(input_var))
7   for (i in 1:nrow(input_var)) {
8     X <- as.vector(input_var[i,])
9     Y[i,] <- random_graph_pa(X$m, X$beta, X$size)
10    print(i/nrow(input_var))
11  }
12  if (sum(abs(Y) != abs(Re(Y))) == 0) {
13    Y <- Re(Y)
14  }
15  Y <- as.data.frame(Y); colnames(Y) <- c("eigenvalue2", "trace")
16  data2 <- cbind(input_var, Y)
17  saveRDS(data2, filename)
18  return(data2)
19 }

```

Listing 21: Return ξ_2 values by mapping the random_graph_pa function from listing 20 over the cartesian product of input variables.

```

1 filename <- "resources/BADData.rds"
2
3 if (file.exists(filename)) {
4   data2 <- readRDS(filename)
5 } else {
6   m <- seq(from = 1, to = 9, length.out = 3)
7   beta <- seq(from = 1, to = 6, length.out = 30)
8   sz <- seq(from = 100, to = 500, length.out = 3) %>% rev() # Big
9   ↪ numbers first
10
11   input_var <- expand.grid("m" = m, "beta" = beta, "size" = sz)
12
13   data2 <- sim_graphs_pa(filename, p, beta, size)
14 }

```

```

1 data2$p <- round((data2$m/data2$size), 2)
2
3 ggplot(data2, mapping = aes(col = factor(p), x = beta, y =
  ↪ eigenvalue2)) +
4   geom_point(size = 0.5) +
5   stat_smooth(method = 'lm', size = 0.4) +
6   scale_size_continuous(range = c(0.1,1)) +
7   labs(x = "Beta", y = TeX("Second Eigenvalue"), title = TeX("Second
  ↪ Eigenvalue given Matrix Density")) +
8   guides(col = guide_legend("Link Density (by m/n)")) +
9   theme_bw()

```

Listing 22: l:ba_{dataplot}

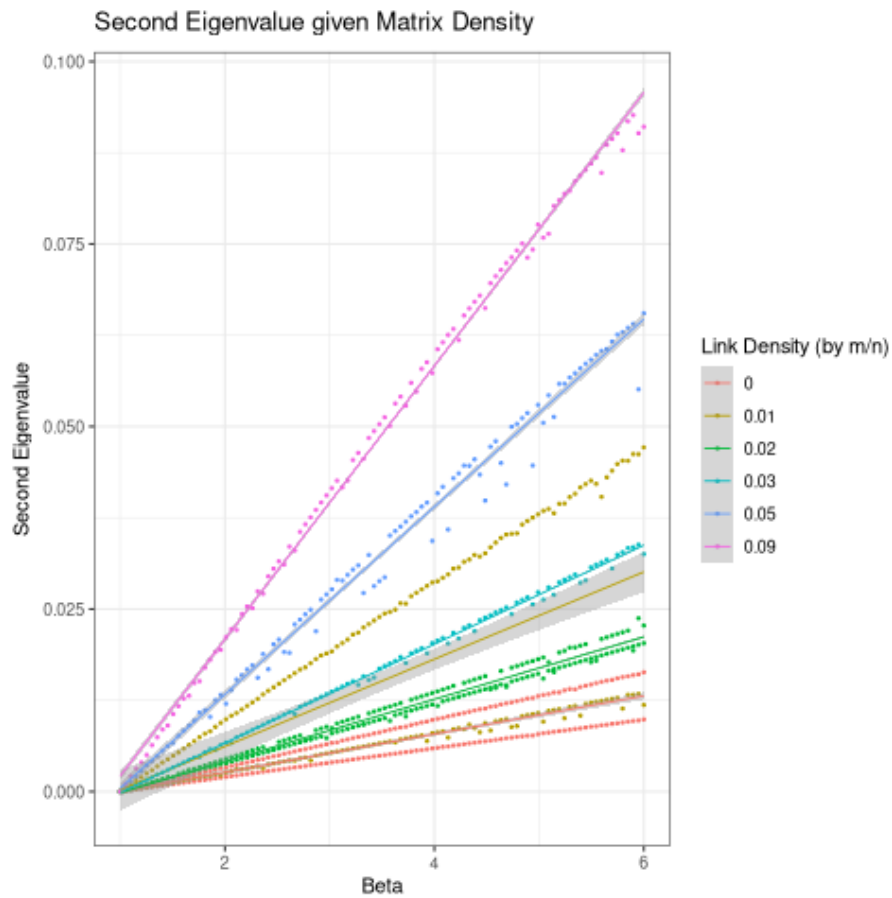


Figure 13: fig:ba_{dataplot}

```

1 filename <- "resources/BAData_constant_size.rds"
2
3 if (file.exists(filename)) {
4   data2 <- readRDS(filename)
5 } else {
6   m      <- seq(from = 1, to = 9, length.out = 15)
7   beta   <- 5
8   size   <- seq(from = 100, to = 500, length.out = 7)
9
10  data2 <- sim_graphs_pa(filename, m, beta, size)
11 }

```

Listing 23: l:baData_{constant size}

```

1 data2$p <- round((data2$m/data2$size), 2)
2
3 names(data2)
4 ggplot(data2, mapping = aes(col = factor(round(size)), x = p, y =
  ↪ eigenvalue2)) +
5   geom_point(size = 0.5) +
6   stat_smooth(method = 'lm', size = 0.4, se = FALSE) +
7   scale_size_continuous(range = c(0.1,1)) +
8   labs(x = "p", y = TeX("Second Eigenvalue"), title = TeX("Second
  ↪ Eigenvalue given Matrix Density") ) +
9   guides(col = guide_legend("Size")) +
10  theme_bw()

```

Listing 24: l:ba_{dataconstant size}plot

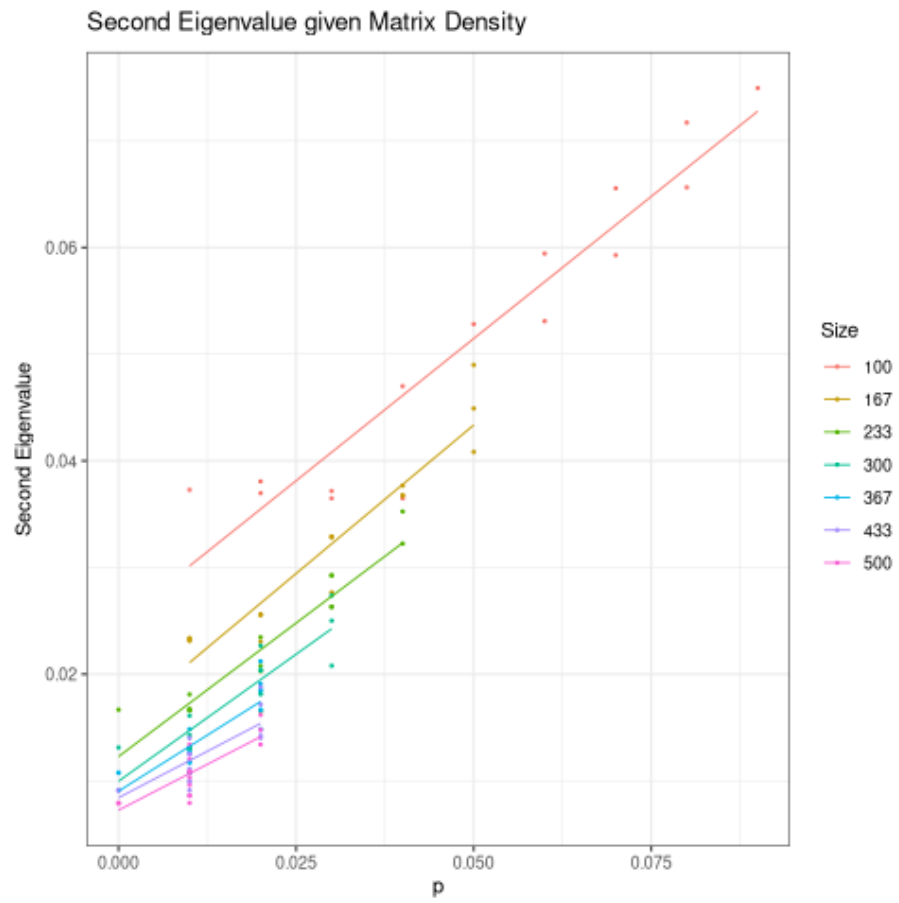


Figure 14: fig:ba_{dataconstant size}plot

```

1 filename <- "resources/BADData_lots.rds"
2
3 # Load the Data
4 if (file.exists(filename)) {
5     data_lots <- readRDS(filename)
6 } else {
7     m <- seq(from = 1, to = 9, length.out = 10)
8     beta <- seq(from = 1, to = 20, length.out = 40)
9     size <- seq(from = 100, to = 500, length.out = 5) %>% rev()
10    size <- c(size, 750, 1000)
11
12    data_lots <- sim_graphs_pa(filename, m, beta, size)
13 }
14
15 # Create the link density variable
16 data2 <- data_lots
17 data2$p <- data2$m/data2$size
18
19 # Use 80% testing data
20 n <- nrow(data2)
21 r <- 0.8
22 train <- sample(x = 1:nrow(data2), size = r*n)
23
24 # Fit the Model
25
26 test_pred <- predict(object = mod)[-train]
27 test_res <- test_pred-data2$eigenvalue2[-train]
28
29 # Make the Plot
30 q <- quantile(test_res, c(0.025, 0.5, 0.975))
31 quantile(test_res)

```

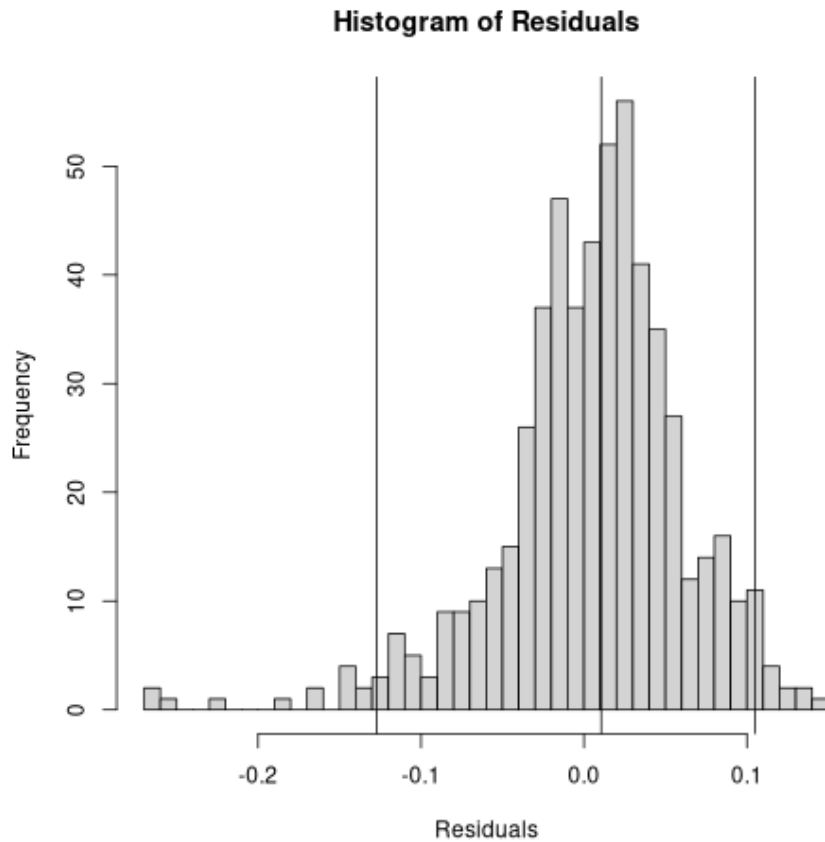


Figure 15: Histogram of the test set residuals for the multiple linear regression model fitted in listing 25, these residuals appear reasonably normal and are centred around zero, this suggests that the model may be an appropriate estimate on the domain that it was fitted.

```
1 summary(mod)
```

Listing 26: Summarise the Coefficients of the model.

```
lm(formula = eigenvalue2 ~ 0 + p * size + p + beta, data = data2)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.108811	-0.009673	-0.001948	0.007289	0.069467

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
p	1.348e+00	2.661e-02	50.651	<2e-16 ***
size	-3.404e-05	1.292e-06	-26.358	<2e-16 ***
beta	3.923e-03	5.205e-05	75.367	<2e-16 ***
p:size	-1.446e-03	1.683e-04	-8.592	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01805 on 2796 degrees of freedom

Multiple R-squared: 0.9089, Adjusted R-squared: 0.9088

F-statistic: 6975 on 4 and 2796 DF, p-value: < 2.2e-16

The residuals appear reasonably normal, this suggests that the model may be an appropriate estimation over the domain on which it was fitted, the corresponding model is given by:

$$|\xi_2| = 1.34p - \frac{3.4n}{10^5} + \frac{3.9\beta}{10^3} - \frac{1.4pn}{10^3} \quad (48)$$

$$p = \text{mean}(\mathbf{A}) \quad (49)$$

For an unweighted adjacency matrix from the internet we may expect:

$$p \approx \frac{m}{n} \quad (50)$$

$$= \frac{\text{mean}(\text{outdegree})}{n} \quad (51)$$

for larger scale free graphs $p \rightarrow 0$ so this model seems to suggest that the eigenvalue will be reduced for larger graphs.

To test the model, we can measure the eigenvalue resulting from values within the domain of the model and from outside the domain:

```
1 m      <- 1
2 beta   <- 3
3 size <- n <- 50
4
5
6 g <- igraph::sample_pa(power = 3, n = n, m = m)
7 A <- t(as.matrix(igraph::get.adjacency(g)))
8 W <- PageRank::power_walk_prob_trans(A, beta = beta)
9
10 p <- mean(A)
11 (e2 <- abs(eigen(W, only.values = TRUE)$values[2]))
12
13 abs(e2_pred <-
```

```

14   p      * 1.344
15 + size  * - 3.393e-05
16 + beta  * 3.918e-03
17 + p*size * - 1.437e-03
18   )

```

```

[1] 0.03689031
[1] 0.03499164

```

That's actually pretty close, but, how does it perform on graphs outside the range of size?

```

1  m      <- 1
2  beta   <- 3
3  size <- n <- 2000
4
5
6  g <- igraph::sample_pa(power = 3, n = n, m = m)
7  A <- t(as.matrix(igraph::get_adjacency(g)))
8  W <- PageRank::power_walk_prob_trans(A, beta = beta)
9
10 p <- mean(A)
11 (e2 <- abs(eigen(W, only.values = TRUE)$values[2]))
12
13 abs(e2_pred <-
14   p      * 1.344
15 + size  * - 3.393e-05
16 + beta  * 3.918e-03
17 + p*size * - 1.437e-03
18   )

```

```

[1] 0.001329781
[1] 0.03968164

```

Unfortunately this suggests that for large graphs this model is not a good fit, which means that this model can only provide insight into the behaviour of ξ_2 for graphs with less than 1000 vertices.

9 Relating the Power Walk to the Random Surfer

It has been shown that $\xi_2 \leq \alpha$ for a transition probability matrix corresponding the random surfer model and that $\xi_2 = \alpha$ if there are atleast two closed subgraphs in the initial network [haveliwalaSecondEigenvalueGoogle2003]. This is demonstrated by figure 5 and figure 5 where $\alpha = \xi_2 = 0.8123456789$.

Finding a connection between the random surfer and the the power walk methods could provide insight into the relationship of ξ_2 and the parameters of the power walk method.

9.1 Introduction

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:

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

9.2 Value of [1st Term]

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

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

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

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

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

9.3 Value of {2nd Term}

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

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

- $\mathbf{E}_{i,j} = \frac{1}{n}$
- $\mathbf{D}_{\mathbf{B}_{k,k}}^{-1} = \frac{1}{\delta_k}$
- 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 $\frac{1}{\text{colSums}(\mathbf{B})} = \vec{1}\mathbf{B}$, 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:

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

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:

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

But the next step requires showing that:

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

9.4 Equate the Power Walk to the Random Surfer

Define the matrix \mathbf{D}_M :

$$\mathbf{D}_M = \text{diag}(\text{colSum}(\mathbf{M})) = \text{diag}(\vec{1}\mathbf{M}) \quad (52)$$

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

$$\mathbf{D}_A^{-1} : [\mathbf{D}_A^{-1}]_i = \begin{cases} 0, & [\mathbf{D}_A]_i = 0 \\ \left[\frac{1}{\mathbf{D}_A}\right], & [\mathbf{D}_A]_i \neq 0 \end{cases} \quad (53)$$

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

$$\mathbf{D}_B^{-1} = \frac{1}{\text{diag}(\vec{1}(\beta \mathbf{A}))} \quad (54)$$

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

Power Walk

$$\mathbf{T} = \boxed{\mathbf{A}\mathbf{D}_A^{-1}} \mathbf{D}_A (\beta - 1) \mathbf{D}_B^{-1} + \boxed{\mathbf{E}} n \mathbf{D}_B^{-1} \quad (55)$$

Random Surfer

$$\mathbf{T} = \alpha \boxed{\mathbf{A}\mathbf{D}_A^{-1}} + (1 - \alpha) \boxed{\mathbf{E}} \quad (56)$$

So these are equivalent when:

$$\mathbf{D}_A (\beta - 1) \mathbf{D}_B^{-1} = \mathbf{I} \alpha \quad (57)$$

$$\begin{aligned} \vec{1}(1 - \alpha) &= -n \mathbf{D}_B^{-1} \\ \implies \vec{1} \alpha &= \vec{1} - n \mathbf{D}_B^{-1} \end{aligned} \quad (58)$$

Hence we have:

$$\mathbf{D}_A (\beta - 1) \mathbf{D}_B^{-1} = \vec{1} \alpha = \mathbf{I} - n \mathbf{D}_B^{-1} \quad (59)$$

Solving for β :

$$\beta \mathbf{J} = (1 - \Theta) \Theta^{-1}$$

where: ⁹

⁹This is similar to a sigmoid 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

- $\Theta = \mathbf{D}_A \mathbf{D}_B^{-1}$

If β is set accordingly then by (59):

$$\begin{aligned} \mathbf{A} (\beta - 1) \mathbf{D}_B^{-1} &= \alpha = \mathbf{I} - n \mathbf{D}_B^{-1} \\ \implies \mathbf{A} (\beta - 1) \mathbf{D}_B^{-1} &= \mathbf{I} - n \mathbf{D}_B^{-1} \end{aligned} \quad (60)$$

And setting $\Gamma = \mathbf{I} - n \mathbf{D}_B^{-1}$ from (58) and putting in (55) we have:

$$\begin{aligned} \mathbf{T} &= \boxed{\mathbf{A} \mathbf{D}_A^{-1}} \mathbf{D}_A (\beta - 1) \mathbf{D}_B^{-1} + \boxed{\mathbf{E}} n \mathbf{D}_B^{-1} \\ \mathbf{T} &= \Gamma \boxed{\mathbf{A} \mathbf{D}_A^{-1}} + (1 - \Gamma) \boxed{\mathbf{E}} \\ \mathbf{T} &= \Gamma \mathbf{A} \mathbf{D}_A^{-1} + (1 - \Gamma) \mathbf{E} \end{aligned} \quad (61)$$

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

9.5 Conclusion

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

More investigation will be required to determine further connections between these two methods.

10 Conclusion

In this report an approach to implement and investigate the random surfer and random walk *PageRank* methods was presented and a linear model was implemented to try and provide insight into the behaviour of the second eigenvalue of the probability transition matrix corresponding to the power walk method.

The model appears not to perform well for graphs over 1000 vertices, this is unfortunate and limits the amount of insight the model can provide into the behaviour of the power walk method.

A relationship between the random surfer and power walk method was also presented, this relationship appears to be restricted to unweighted adjacency matrices, it is not clear however if this could be expanded.

Further investigation is required to understand the relationship between the second eigenvalue and the method parameters of the power walk method, such further investigation should focus on investigating the relationship between the power walk and random surfer models and implementing non-linear models.

11 Appendix

11.1 Graph Diagrams

Graph Diagrams shown in 3.2.2 were produced using DOT (see [28, 9]).

```

1  library(Matrix)
2  library(igraph)
3  n <- 200
4  m <- 5
5  power <- 1
6  g <- igraph::sample_pa(n = n, power = power, m = m, directed = FALSE)
7  plot(g)

8  A <- t(get.adjacency(g))
9  plot(A)
10 image(A)
11
12
13 # Create a Plotting Region
14 par(pty = "s", mai = c(0.1, 0.1, 0.4, 0.1))
15
16
17 # create the image
18
19 title=paste0("Undirected Barabassi Albert Graph with parameters:\n
  ↳ Power = ", power, "; size = ", n, "; Edges/step = ", round(m))
20 image(A, axes = FALSE, frame.plot = TRUE, main = title, xlab = "", ylab
  ↳ = "", )

```

Listing 27: **R** code to produce an image illustrating the density of a simulated Barabasi-Albert graph, the *Barabasi-Albert* graph is a good analogue for the link structure of the internet [20, 3, 4] see the output in figure 16

**Undirected Barabassi Albert Graph with parameters:
Power = 1; size = 200; Edges/step = 5**

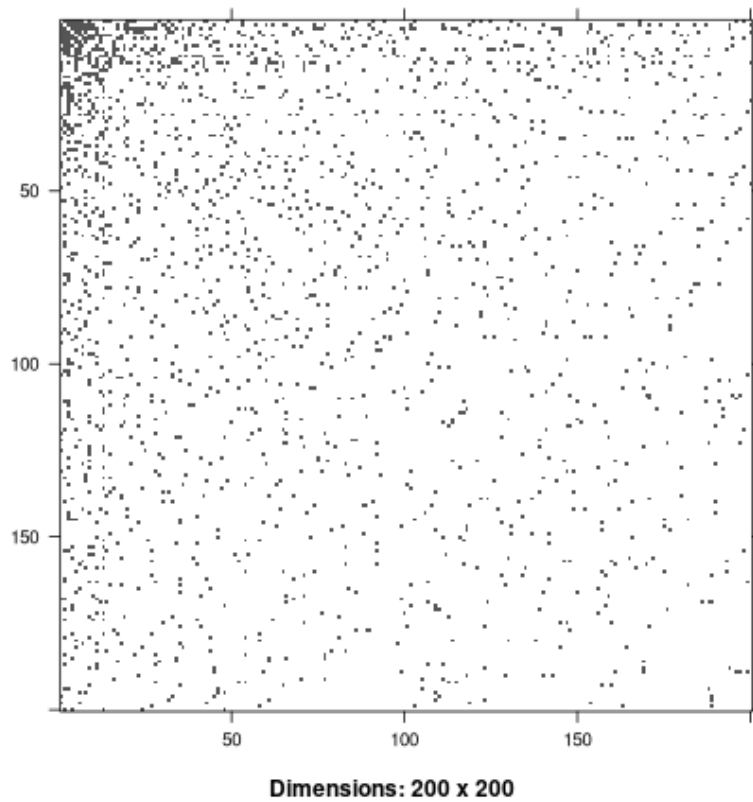


Figure 16: Plot of the adjacency matrix corresponding to a Barabassi-Albert (i.e. *Scale Free*) Graph produced by listing 27, observe the matrix is quite sparse.

12 my to do list

12.1 Use BA Graphs

12.2 Look at the ScatterPlot Matrix

12.3 Compare eigenvalue2 and iterations

12.4 Look at using an SVM/logReg or other classifier

Use a ROC curve to find which threshold of e2 predicts a below average number of iterations.
Predict that e2 value?

12.4.1 Measure Iterations and E2 values

12.4.2 Are any values indicative of E2 Values?

12.4.3 Use Method Parameters with a log reg to predict a number of iterations

12.4.4 Use a ROC Curve to set a threshold

12.4.5 Discuss the results.

12.5 Tie the Report together.

12.6 Typeset the Report

12.7 TODO Diamater

Diamater of the web sounds like a fun read [[albertDiameterWorldWideWeb1999](#)]

12.8 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) (cit. on p. 6).
- [2] Albert-László Barabási. *Linked: The New Science of Networks*. Cambridge, Mass: Perseus Pub, 2002. 280 pp. ISBN: 978-0-7382-0667-7 (cit. on p. 42).
- [3] 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](https://doi.org/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) (cit. on pp. 42, 58).

- [4] 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](https://doi.org/10.1016/S0378-4371(00)00018-2). URL: <https://linkinghub.elsevier.com/retrieve/pii/S0378437100000182> (visited on 10/11/2020) (cit. on pp. 42, 58).
- [5] 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](https://doi.org/10.1007/s10626-017-0248-7). URL: <https://doi.org/10.1007/s10626-017-0248-7> (visited on 08/19/2020) (cit. on p. 6).
- [6] 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](https://doi.org/10.1145/1052934.1052938). URL: <http://portal.acm.org/citation.cfm?doid=1052934.1052938> (visited on 08/18/2020) (cit. on p. 6).
- [7] Michael Brinkmeier. “PageRank Revisited”. In: *ACM Transactions on Internet Technology* 6.3 (Aug. 2006), pp. 282–301. ISSN: 15335399. DOI: [10.1145/1151087.1151090](https://doi.org/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) (cit. on p. 6).
- [8] Mufa Chen. *Eigenvalues, Inequalities, and Ergodic Theory*. Probability and Its Applications. London: Springer, 2005. 228 pp. ISBN: 978-1-85233-868-8 (cit. on p. 4).
- [9] DOT (Graph Description Language). In: *Wikipedia*. June 11, 2020. URL: [https://en.wikipedia.org/w/index.php?title=DOT_\(graph_description_language\)&oldid=961944797](https://en.wikipedia.org/w/index.php?title=DOT_(graph_description_language)&oldid=961944797) (visited on 10/09/2020) (cit. on p. 57).
- [10] Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*. Version 1.2-18. Nov. 27, 2019. URL: <https://CRAN.R-project.org/package=Matrix> (visited on 10/17/2020) (cit. on p. 11).
- [11] François Fouss, Marco Saelens, 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 (cit. on pp. 4, 7).
- [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](https://doi.org/10.1002/asmb.656). URL: <http://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.656> (visited on 08/19/2020) (cit. on p. 6).
- [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) (cit. on p. 6).
- [14] Andrea Garritano. *Wikipedia Article Networks*. Dec. 2019. URL: <https://kaggle.com/andreagarritano/wikipedia-article-networks> (visited on 10/03/2020) (cit. on p. 42).
- [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](https://doi.org/10.1145/2488388.2488433). URL: <http://doi.org/10.1145/2488388.2488433> (visited on 10/09/2020) (cit. on p. 5).
- [16] *Igraph R Manual Pages*. URL: https://igraph.org/r/doc/sample_pa.html (visited on 10/06/2020) (cit. on p. 31).

- [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](https://doi.org/10.1016/j.laa.2003.12.008). URL: <http://www.sciencedirect.com/science/article/pii/S0024379504000023> (visited on 08/19/2020) (cit. on p. 6).
- [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](https://doi.org/10.1002/asi.23118). URL: <https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.23118> (visited on 08/21/2020) (cit. on p. 6).
- [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) (cit. on p. 60).
- [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 (cit. on pp. 4–8, 11, 42, 58, 60).
- [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](https://doi.org/10.1016/S0169-7552(98)00110-X). URL: <http://www.sciencedirect.com/science/article/pii/S016975529800110X> (visited on 08/19/2020) (cit. on pp. 3, 8).
- [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 (cit. on p. 11).
- [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 (cit. on p. 6).
- [24] Nathanael Ackerman, Cameron Freer, Alex Kruckman, and Rehana Patel. *Properly Ergodic Structures*. Oct. 25, 2017. URL: <https://math.mit.edu/~freer/papers/properly-ergodic-structures.pdf> (visited on 10/10/2020) (cit. on p. 4).
- [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](https://doi.org/10.1145/2537734.2537749). URL: <http://doi.org/10.1145/2537734.2537749> (visited on 07/31/2020) (cit. on pp. 3, 12).
- [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> (cit. on p. 31).
- [27] saz. *Probability Theory - Is This Graph Ergodic?* URL: <https://math.stackexchange.com/questions/1327283/is-this-graph-ergodic> (visited on 10/10/2020) (cit. on p. 4).
- [28] *The DOT Language*. URL: <https://graphviz.org/doc/info/lang.html> (visited on 10/09/2020) (cit. on p. 57).
- [29] Rui Zeng et al. “A Practical Simulation Method for Social Networks”. In: 144 (2013), p. 8 (cit. on p. 42).