## Contents

# 1 Implementing the Power Walk and the Random Surfer

assess node centrality by perfor [25]

## 1.1 Page Rank Methods

### 1.1.1 Introduction

The centrality score of a graph is a metric that measures the importance and popularity of a vertex. [1]

The *PageRank* method asserts that the centrality of a vertex can be measured by the frequency of incidence with that vertex during a random walk.

### 1.1.2 Implementing PageRank Generally

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

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

Where each element of the matrix indicates whether or not travel from vertex $j$ to vertex $i$ is possible with a value of 1. [2]

During a random walk the probability of arriving at vertex $j$ from vertex $i$ can similarly be described as an element of a transition probability matrix $\mathbf{T}_{i,j}$, this matrix can be described by the following relationship [3]:

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

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

The value of $\mathbf{D}$ is such that under matrix multiplication $\mathbf{A}$ will have columns that sum to 1 (i.e. a *column stochastic matrix*, see § 1.1.4 ), for a non-ergodic graph the definition of $\mathbf{D}$ would need to be adjusted to acheive this, this is discussed below CROSSREF.

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

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

This relationship is a linear recurrence relation, more importantly however it is a *Markov Chain* [15, §4.4].

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

### 1.1.3 Issues

The approach in 1.1.2 has the following issues

1. Convergence of (3)
   (a) Will this relationship converge or diverge?
   (b) How quickly will it converge?
   (c) Will it converge uniquely?
2. Reducible graphs

[1]For a small graph drawn in a way to minimise overlapping edges the centremost geometric vertex will coincide with the highest centrality score, for example in figure 4 vertex $D$ is the vertex with the highest frequency during a random walk

[2]Some authors define an adjacency matrix transposed (see e.g. [23, 1, 18]) this unfourtunately includes the `igraph` library [9] but that convention will not be followed in this paper

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

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

3. Cycles

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

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

#### 1.1.4 Definitions

The following definitions are used in this report [4] :

**Markov Chains** are discrete mathematical model such that future values depend only on current values [7, §1.5]

**Stochastic Matrices** contain only positive values where each column sums to 1 [15, 17]

- some authors use rows (see e.g. [15, §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 $\mathtt{sum}\left(\vec{\mathtt{x}}\right) = 1$, so when solving for a stationary vector it can be necessary to perform $\vec{\mathtt{p}} \leftarrow \frac{\vec{\mathtt{p}}}{\sum \vec{p}}$

**Irreducible** graphs have a path from from any given vertex to another vertex. [15, §15.2]

**Ergodic** graphs are irreducible graphs with further constraints outside the scope of this report (see e.g. [20, 5])

- It is a necessary but not a sufficient condition of ergodic graphs that all vertices be reachable from any other vertices (see [24] 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 [15, §15.2]

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

- Some Authors consider the transpose (see e.g. [15]).

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

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

**Regular** Markov Chains are regular irreducible and aperiodic.

**Sparse** Matrices contain a majority of elements with values equal to 0 [15, §4.2]

**Sparse** Iterating the

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

- In the literature (see e.g. [12, 15]) the Random Surfer model is usually used to refer to the introduction of a probability of travelling to any other node, this is discussed in CROSSREF

[4] see generally [15, Ch. 15] for further reading

**Notation**

$\mathbf{A}$ Is the adjacency matrix of a graph

> $\mathbf{A}_{i,j} = 1$ Indicates that $j$ and $i$ are adjacent vertices.

$\mathbf{T}$ Is the transition probability matrix of a graph

- $T_{i,j}$ is equal to the probability of travelling $j \rightarrow i$ during a random walk.
  - $\mathbf{T} = \mathbf{A}\mathbf{D}_{\mathbf{A}}^{-1}$
    * 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 elements of a matrix

- $n = \texttt{nrow}(\mathbf{A}) = \texttt{ncol}(\mathbf{A})$

$\mathbf{B}_{i,j} = \frac{1}{n}$ Is a matrix of size $n \times n$ representing the background probability of uniformly selecting any vertex of a graph.

$\mathbf{J}_{i,j} = 1$ Is a completely dense $n \times n$ matrix.

- It's worth noting that $\mathbf{e}, \mathbf{E}, \mathbf{J}$ are common choices for this matrix.

$\vec{1}$ is a vector of length $n$ containing only the value 1.

$\alpha$ The probability of teleporting from one vertex to another during a random walk.

- In the literature $\alpha$ is often referred to as a damping factor (see e.g. [2, 4, 8, 13, 3])

or a smoothing constant (see e.g [14]).

### 1.1.5 Markov Chains

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

- for a stochastic irreducible markov chain [7, §1.5.5],
- regardless of the initial condition of the process for an *aperiodic* Markov chain [15, §4.4]

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

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

This however would not be sufficient to ensure that (3) would converge, in addition that transition probability matrix must be made irreducible and aperiodic (i.e. primitive). [15]

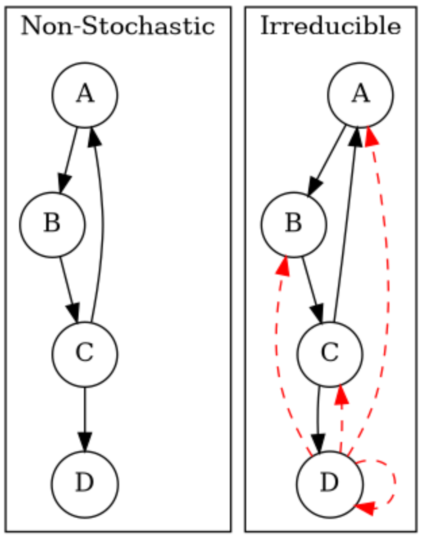


Figure 1: $D$ is a *dangling node*, a dead end during a random walk, the corresponding probability transition matrix ($\mathbf{T}$) is hence non-stochastic, Introducing some probability of teleporting from a dead end to any other vertex (as denoted in red) will cause the $\mathbf{T}$ to be stochastic.

**Irreducible** A graph that allows travel from any given vertex to any other vertex is said to be irreducible [15], see for example figure 4, this is important in the context of a random walk because only an irreducible graph can all vertexes be reached from any initial condition.

It must be primitive (i.e. irreducible and aperiodic) in order for the value to converge.

This can be acheived by introducing a probability of teleporting from one node to any other ( $\alpha$), this approach is known as the *Random Surfer* model and the transition probability matrix is given by [16] :

$$\mathbf{S} = \alpha \mathbf{T} + \frac{(1-\alpha)}{n} \mathbf{J} \tag{4}$$

This matrix is primitive and stochastic, it is also however completely dense (see CROSSREF) [15, §4.5]

**Aperiodic** An a periodic graph has only one eigenvalue that lies on the unit circle, this is important because such a graph

### 1.1.6 Irreducible Graphs

This issue occurs with non-ergodic graphs.

If not ergodic there are rank sinks from dangling nodes.

If the transition probability matrix is irreducible and stochastic then there will be a unique positive stationary distribution, under iteration (3) will converge to this stationery distribution if $\mathbf{T}$ is aperiodic. [15, §4.4]

Most Adjacency matrices resulting from webpages and analagous networks are sparse matrices (see figure x), this is a good thing because it requires far less computational resources to work with [15, §4.2]

The Barabassi Albert graph is a good analogue CITE see figure 7 for an example.

Random surfer adds random probability.

Power Walk redefines probability

The next issue however is that we need to use sparse matrices, otherwise it is too slow, these will be discussed below.

The definition used in this article is PageRank is the process of a random walk algorithm that ranks pages based on there frequency of incidence during a random walk, this has been used in the litereature (see e.g. [12, §3.1]),

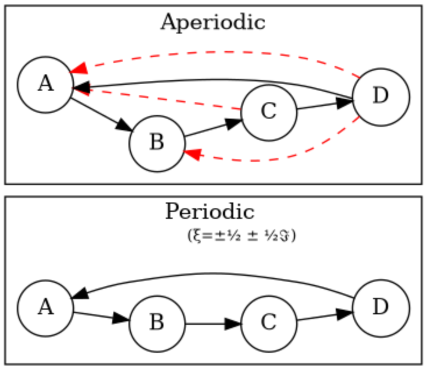


Figure 2: A periodic graph with all eigenvalues on the unit circle $\xi = \frac{\sqrt{2}}{2} e^{\frac{\pi i}{4} k}$, by adding in extra edges the graph is now aperiodic, this does not represent the random surfer moidel, which would in theory connect every vertex but with some probability.

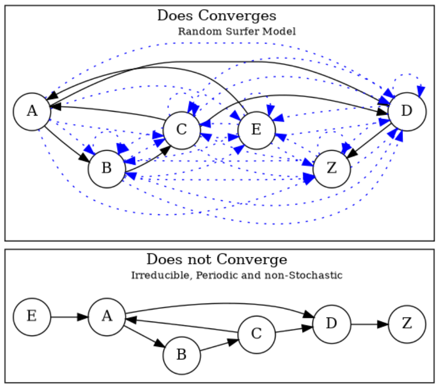


Figure 3: A graph that is aperiodic, reducible and non-stochastic, by applying the random surfer model (4) blue *teleportation* edges are introduced, these may be followed with a probability of $1 - \alpha$

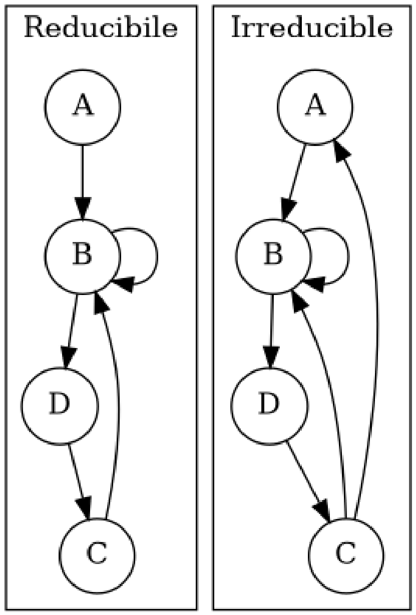


Figure 4: Example of a non-ergodic graph (produced using `DOT`, see [26, 6])

the class of random walk algorithm most often associated with this is the *Random Surfer* Model by page and brin [16], this implements a small probability of making ajumping to any vertex of the graph at each step.

This paper is concerned with a different approach to create the transition probability matrix.

$\sum_{p \in \vec{p}} [p]$

Using this transition matrix the state distribution at any point can be returned The issue with (1)

Ergodic is a proble8m

The *PageRank* method, developed by Larry Page and Sergey Brin in 1998 [16] is a method to measure node centrality by building on the concept of the stationary distribution of a graph.

A method to asses node centrality by performing a random walk across graph and recording the frequencies of This approach is relatively easy to implement if any vertex can reach any other vertex during a random walk (such a graph is said to be ergodic), if however a graph is not ergodic this method will not work.

named after larry page [11] If a graph is

The Page Rank methodrefers to the random surfer but I'm using it in this document to refer to any process that attributes a probability of landing on a vertex during a random walk to a graph that is not ergodic.

If each vertex is connected the graph is said to be ergodic and there is a closed solution for the limit values of the frequencies given this random walk:

- The eigenvalue equal to 1
- If the graph is not directed $\vec{p}$ is a vector of length $n$:
  - $n$ is the number of nodes in the graph $G$
  - $\vec{p}_i = \frac{\deg(v_1)}{\text{vol}(G)}$
    - $\text{vol}(G) = \sum_{i=1}^{n} [\text{indeg}(v)] = \sum_{i=1}^{n} [\text{outdeg}(v)] = \sum_{i=1}^{n} [\deg(v)]$

For large matrices calculating the eigenvalues will be expensive and so instead the power method is used, which is essentially looping over until the vector converges to a solution.

$$\vec{p} = \mathrm{T}\vec{p} \tag{5}$$

where:

A Is the adjacency Matrix, an element is 1 if movement from the row vertex to the column vertex is permitted.

- The matrix may be weighted in some way, for example 5 edges between vertices may be such that a 5 is used in the matrix not a 1
- An undirected graph will be such that $\mathbf{A} = \mathbf{A^T}$

T Is the transition probability matrix, an element in the matrix describes the probability of moving from the column-vertex to the row-vertex

- The transition matrix is intended to be such that for a given state distribution $\vec{p}$, the next iteration of a random walk will be $\mathrm{T}\vec{p}$
- Observe also that $\mathrm{T} = \mathrm{T} \cdot \text{diag}(\texttt{colsums}(\mathrm{A}^{\mathrm{T}}))$
  - i.e. the transpose of the adjacency matrix with each column scaled to 1.

### 1.1.7 Large Graphs

The relationship in (3) is equivelant to the eigenvalue value problem, where $\mathbf{T}$ is the eigenvector [5] $\vec{x}$ that corresponds to the eigenvalue $\xi = 1$:

$$\vec{p}(1) = \mathbf{T}\vec{p} \tag{6}$$

Solving eigenvecotrs for large matrices can be very difficult and so this approach isn't suitable for analysing large networks.

Upon iteration (3) will converge to stable stationary point equal (as shown in (6)) and this must be implemented.

considering the linear recurrence relation is a markov chain because any future progression depends only on the current state and not the past.

### 1.1.8 Random Surfer

**Introduction** For con If a graph is non-ergodic, then a random walk isn't as easy to implement because in escence there are multiple disconnected graphs, to address this, some value $\lambda$ is introduces which represents the probability of moving from one vertex to any other vertex. Essentially the difference here is

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

```r
  pacman::p_load(tidyverse, Matrix, igraph, plotly, mise, docstring)
```

**Small Graph, Ordinary Matrices**

**Example Graph** Consider the following Graph taken from the paper:

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

1. Adjacency Matrix The adjacency Matrix is given by:

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

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

## Adjust the Order
(A <- A[order(as.integer(row.names(A))),
↪  order(as.integer(colnames(A)))])
```

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

2. State Distribution The state distribution is the transpose of the adjacency matrix:

```r
(p0 <- t(A))
```

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

3. 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.

```r
p0 %*% diag(1/colSums(p0))
```

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

(a) Create a Function

```
adj_to_probTrans <- function(adjMat) {
  t(adjMat) %*% diag(1/colSums(t(adjMat)))
}

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

```
##     [,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** The random surfer page rank method modifies the probability transition matrix $T$ so that the method works also for non-ergodic graphs by introducing the possibility of a random jump, we'll call the surfer transition matrix $S$:

$$S = \lambda T + (1 - \lambda) B : \tag{7}$$

$$\tag{8}$$

$$B = \begin{bmatrix} \frac{1}{N} & \frac{1}{N} & \dots & \frac{1}{N} \\ \frac{1}{N} & \frac{1}{N} & \dots & \frac{1}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{N} & \frac{1}{N} & \dots & \frac{1}{N} \end{bmatrix} \tag{9}$$

$$N = ||V|| \tag{10}$$

$$\lambda \in [0, 1] \tag{11}$$

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

S <- l*T+(1-l)*B
```

1. Eigen Value Method The eigenvector corresponding to the the eigenvalue of 1 will be the stationary point:

```r
eigen(S, symmetric = FALSE)
```

```
eigen() decomposition
$values
 [1]  1.000000e+00 -8.123457e-01 -8.123457e-01  8.123457e-01 -3.407464e-09  3.407464e-09
 [7]  6.878591e-17 -4.393838e-17 -1.126771e-18 -1.292735e-32

$vectors
            [,1]          [,2]          [,3]          [,4]          [,5]          [,6]
 [1,] 0.48726141 -7.071005e-01  1.590774e-03  5.000000e-01  6.735753e-01 -6.735753e-01
 [2,] 0.52676629  7.071005e-01 -1.590774e-03  5.000000e-01  9.622504e-02 -9.622505e-02
 [3,] 0.49149620 -2.975837e-03  7.071050e-01 -5.000000e-01  9.622504e-02 -9.622505e-02
 [4,] 0.48044122  2.975837e-03 -7.071050e-01 -5.000000e-01  2.886751e-01 -2.886751e-01
 [5,] 0.04932738  1.463673e-18 -5.541166e-17  2.124631e-17 -3.849002e-01  3.849002e-01
 [6,] 0.04932738  1.463673e-18  5.541166e-17  2.124631e-17 -3.849002e-01  3.849002e-01
 [7,] 0.04932738  1.463673e-18 -2.077937e-17  2.124631e-17 -3.849002e-01  3.849002e-01
 [8,] 0.04243328 -6.484884e-18 -1.103904e-17  6.319692e-17  8.072508e-09  8.072508e-09
 [9,] 0.04243328  6.952446e-18 -9.740331e-18  6.005334e-17  8.072508e-09  8.072509e-09
[10,] 0.04243328  6.952446e-18 -9.740331e-18  6.005334e-17  8.072508e-09  8.072509e-09
               [,7]          [,8]          [,9]         [,10]
 [1,] -3.963430e-01  3.962600e-01  1.828019e-01 -1.752367e-01
 [2,] -1.291621e-01  2.027302e-01  2.199538e-01 -2.197680e-01
 [3,] -3.955284e-01  3.894308e-02  2.223048e-01 -2.248876e-01
 [4,] -4.215353e-01  1.043870e-01  2.747562e-01 -2.777266e-01
 [5,]  5.166485e-01 -8.109210e-01 -8.798152e-01  8.790721e-01
 [6,]  5.201366e-02 -1.308878e-01 -1.049028e-01  1.056778e-01
 [7,]  1.346275e-01 -1.936007e-01  9.054366e-02 -9.554811e-02
 [8,]  2.547528e-16 -1.352936e-16 -1.025353e-16  1.072771e-16
 [9,]  3.196396e-01  1.965446e-01 -2.821213e-03 -5.466313e-03
[10,]  3.196396e-01  1.965446e-01 -2.821213e-03  1.388344e-02
```

So in this case the stationary point is
$\langle -0.49, -0.53, -0.49, -0.48, -0.05, -0.05, -0.05, -0.04, -0.04, -0.04 \rangle$
which can be verified:

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

```r
(p     <- eigen(S)$values[1] * eigen(S)$vectors[,1])
```

```
## [1] -0.48531271 -0.52732002 -0.49152601 -0.47977477 -0.05288058 -0.05288058
## [7] -0.05288058 -0.04558671 -0.04558671 -0.04558671
```

```r
(p_new <- S %*% p)
```

```
##             [,1]
## 1  -0.48531271
## 2  -0.52732002
## 3  -0.49152601
## 4  -0.47977477
## 5  -0.05288058
## 6  -0.05288058
## 7  -0.05288058
## 8  -0.04558671
## 9  -0.04558671
## 10 -0.04558671
```

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

```r
(p_new <- p_new/sum(p_new))
```

```
##           [,1]
## 1  0.2129185
## 2  0.2313481
## 3  0.2156444
## 4  0.2104889
## 5  0.0232000
## 6  0.0232000
## 7  0.0232000
## 8  0.0200000
## 9  0.0200000
## 10 0.0200000
```

2. Power Value Method Using the power method should give the same result, which it indeed does, but for the scale:

```r
p_new <- p_new *123456789

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

p_new
```

```
##          [,1]
## 1  26286237
## 2   28561500
## 3   26622771
## 4   25986282
## 5    2864198
## 6    2864198
## 7    2864198
## 8    2469136
## 9    2469136
## 10   2469136
```

```
p
```

```
##          [,1]
## 1   26286237
## 2   28561500
## 3   26622771
## 4   25986282
## 5    2864198
## 6    2864198
## 7    2864198
## 8    2469136
## 9    2469136
## 10   2469136
```

This answer is however identical in direction, if it scaled to 1 the same value will be returned:

```
(p_new <- p_new/sum(p_new))
```

```
##           [,1]
## 1   0.2129185
## 2   0.2313481
## 3   0.2156444
## 4   0.2104889
## 5   0.0232000
## 6   0.0232000
## 7   0.0232000
## 8   0.0200000
## 9   0.0200000
## 10  0.0200000
```

3. Scaling However if the initial state sums to 1, then the scale of the stationary vector will also sum to 1.

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

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

cbind(p_new, p)
```

```
##         [,1]      [,2]
## 1  0.2129185 0.2129185
## 2  0.2313481 0.2313481
## 3  0.2156444 0.2156444
## 4  0.2104889 0.2104889
## 5  0.0232000 0.0232000
## 6  0.0232000 0.0232000
## 7  0.0232000 0.0232000
## 8  0.0200000 0.0200000
## 9  0.0200000 0.0200000
## 10 0.0200000 0.0200000
```

### Large Graph, Sparse Matrices using CRS

**Creating the Probability Transition Matrix** Implementing the page rank method on a larger graph requires the use of more efficient form of matrix storage.

An adjacency matrix (atleast in the context of graphs relating to webpages and social networks) will contain elements that are mostly zero because the number of edges leaving any vertex will tend to be significantly less than the total number of vertices.

A matrix exhibiting this property is known as a sparse matrix CITE

The properties of a sparse matrix can be implemented in order to improve performance, one such method to acheive this is *Compressed Sparse Row* (CSR) storage, which involves creating a seperate array of values and corresponding indices. CITE

This is implemented by the Matrix package in ***R***. CITE

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

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

summary(A)
```

As before in section 3, the probability transition matrix can be found by:

1. Transposing the adjacency matrix, then

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 may be implemented, however to create this new matrix, a new `sparseMatrix` will need to be created using the properties of the original matrix, this can be done like so:

```r
 sparse_diag <- function(mat) {
  #' Diagonal Factors of Sparse Matrix
  #'
  #' Return a Diagonal Matrix of the 1 / colsum() such that
  #' matrix multiplication with this matrix would have all column sums
  #' sum to 1
  #'
  #' This should take the transpose of an adjacency matrix in and the output
  #' can be multiplied by the original matrix to scale it to 1.
  #' i

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

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

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

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

  ## Return the Diagonal Matrix
  return(D)
}
D <- sparse_diag(t(A))
summary(D)
```

```
1000000 x 1000000 sparse Matrix of class "dgCMatrix", with 20 entries
        i      j           x
1   57981  57981 0.011235955
2   61426  61426 0.010309278
3  139900 139900 0.012048193
4  152229 152229 0.009615385
5  175521 175521 0.010204082
6  187782 187782 0.013513514
7  233553 233553 0.011904762
8  288279 288279 0.010309278
9  381442 381442 0.011494253
10 401058 401058 0.014084507
11 541818 541818 0.012820513
12 542888 542888 0.014492754
13 578270 578270 0.010101010
```

```
14 595348 595348 0.011764706
15 610432 610432 0.011904762
16 614645 614645 0.012195122
17 776459 776459 0.010989011
18 803589 803589 0.008474576
19 821120 821120 0.009708738
20 821769 821769 0.012658228
```

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

```r
summary(t(A) %*% D)
```

```
1000000 x 1000000 sparse Matrix of class "dgCMatrix", with 20 entries
        i      j x
1  809115  57981 1
2   83355  61426 1
3  649740 139900 1
4  451810 152229 1
5  775788 175521 1
6  364814 187782 1
7  631441 233553 1
8  630438 288279 1
9  681415 381442 1
10 103999 401058 1
11 976802 541818 1
12 217196 542888 1
13 755635 578270 1
14 993420 595348 1
15 206922 610432 1
16 462031 614645 1
17 566334 776459 1
18  66922 803589 1
19 809688 821120 1
20 291405 821769 1
```

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

However, creating a matrix of background probabilites (denoted by `B` is section 1.1.8 ) will not be feasible, it would simply be too large, instead some algebra can be used to reduce $B$ from a matrix into a vector containing only $\frac{1-\alpha}{N}$.

The power method is given by:

$$\vec{p} = \mathbf{S}\vec{p} \tag{12}$$

where:

$$
\begin{aligned}
S &= \alpha \mathbf{T} + (1-\alpha)\,\mathbf{B} && (13)\\
\vec{p} &= (\alpha \mathbf{T} + (1-\alpha)\,\mathbf{B})\,\vec{p} && (14)\\
&= \alpha \mathbf{T}\vec{p} + (1-\alpha)\,\mathbf{B}\vec{p} && (15)
\end{aligned}
$$

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

$$
\begin{aligned}
\mathbf{F} &= \begin{bmatrix} \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \end{bmatrix} \begin{bmatrix} \vec{p_1} \\ \vec{p_2} \\ \vdots \\ \vec{p_m} \end{bmatrix} && (16)\\
&= \begin{bmatrix} \left(\sum_{i=0}^{m} [p_i]\right) \times \frac{1}{N} \\ \left(\sum_{i=0}^{m} [p_i]\right) \times \frac{1}{N} \\ \vdots \\ \left(\sum_{i=0}^{m} [p_i]\right) \times \frac{1}{N} \end{bmatrix} && (17)\\
&\text{Probabilities sum to 1 and hence:} && (18)\\
&= \begin{bmatrix} \frac{1}{N} \\ \frac{1}{N} \\ \frac{1}{N} \\ \vdots \\ \frac{1}{N} \end{bmatrix} && (19)
\end{aligned}
$$

So instead the power method can be implemented by performing an algorithm to the effect of:

```r
## Find Stationary point of random surfer
N     <- nrow(A)
alpha <- 0.8
F     <- rep((1-alpha)/N, nrow(A))  ## A nx1 vector of (1-alpha)/N

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

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

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

### 1.1.9 Power Walk Method

#### Introduction

$$
\mathbf{T} = \mathbf{B}\mathbf{D}_B^{-1} \qquad (20)
$$

where:

- $\mathbf{B} = \beta^{\mathbf{A}}$

  $x\beta^1$ probability of following an edge of weight 1
  $x\beta^0$ probability of following an edge of weight 0
  $x\beta^{-1}$ probability of following an edge of weight -

- $D_B = \texttt{colsums}(\mathbf{B})$

**A** The Adjacency Matrix

**Ordinary Matrices** Solving the Power walk can be done pretty much the same as it is with the random surfer, but doing it with Sparse Matrices is a bit trickier.

**Sparse Matrices**

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

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

where:

- $\mathbf{T}$
  - is an $i \times j$ matrix that describes the probability of travelling from vertex $j$ to $i$
    - This is transpose from the way that `igraph` produces an adjacency matrix.

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

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

- No column of $\mathbf{A}$ sums to zero
  - If this does happen the question arises how to deal with $\mathbf{D}_{\mathbf{A}}^{-1}$
    - I've been doing $\mathbf{D}^{\mathrm{T}}_{\mathbf{A},i,j} := \texttt{diag}\left(\frac{1}{\texttt{colsums}(\mathbf{A})}\right)$ and then replacing any 0 on the diagonal with 1.
  - What is done in the paper is to make another matrix $\mathbf{Z}$ that is filled with 0, if a column sum of $\mathbf{A}$ adds to zero then that column in $\mathbf{Z}$ becomes $\frac{1}{n}$
    - This has the effect of making each row identical
    - The probability of going from an orphaned vertex to any other vertex would hence be $\frac{1}{n}$
    - The idea with this method is then to use $D^{-1}_{(\mathbf{A}+\mathbf{Z})}$ this will be consistent with the *Random Surfer* the method using $\mathbf{F}$ in [[#eq:sparse-RS][]] ( 1.1.9 )

  where each row is identical that is a 0

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

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

Be mindful that the use of exponentiation in ] is not an element wise exponentiation and not an actual matrix exponential (which would be defined by using power series and logs but is defined)

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 (1)):

$$\mathbf{O}\mathbf{D}_{\mathbf{B}}^{-1}\vec{p_i} = \texttt{repeat}(\vec{p} \bullet \vec{\delta}^{\mathrm{T}}, \texttt{n})$$

where:

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

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

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

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

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

This means we can do:

$$\begin{aligned}
\overrightarrow{p_{i+1}} &= \mathbf{T}_{\mathrm{pw}}\overrightarrow{p_i} && (21)\\
&= \mathbf{B}\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} && (22)\\
&= (\mathbf{B} - \mathbf{O} + \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} && (23)\\
&= \left((\mathbf{B} - \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1} + \mathbf{O}\mathbf{D}_{\mathbf{B}}^{-1}\right)\overrightarrow{p_i} && (24)\\
&= (\mathbf{B} - \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} + \mathbf{O}\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} && (25)\\
&= (\mathbf{B} - \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} + \vec{1}(\overrightarrow{\delta^{\mathrm{T}}}\overrightarrow{p_i}) && (26)\\
&= (\mathbf{B} - \mathbf{O})\,\mathbf{D}_{\mathbf{B}}^{-1}\overrightarrow{p_i} + \texttt{rep}(\overrightarrow{\delta^{\mathrm{T}}}\overrightarrow{p_i}) && (27)
\end{aligned}$$

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

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

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

$$\begin{aligned}
\mathbf{B}_{\mathbf{O}} &= (\mathbf{B} - \mathbf{O}) && (28)\\
\mathbf{B} &= \mathbf{B}_{\mathbf{O}} + \mathbf{O} && (29)
\end{aligned}$$

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

$$\begin{aligned}
\delta_{\mathbf{B}}^{-1} &= \vec{1}\mathbf{B} && (30)\\
&= \vec{1}\left(\mathbf{B_O} + \mathbf{O}\right) && (31)\\
&= \vec{1}\mathbf{B_O} + \vec{1}\mathbf{O} && (32)\\
&= \vec{1}\mathbf{B_O} + \langle n, n, n, ...n\rangle && (33)\\
&= \vec{1}\mathbf{B_O} + \vec{1}n && (34)\\
\delta_{\mathbf{B}} &= \texttt{1}/(\texttt{colSums}(\mathbf{B_O}) + \texttt{n}) && (35)
\end{aligned}$$

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

$$\begin{aligned}
D_B^{-1} &= \mathrm{diag}\left(\delta_{\mathbf{B}}^{-1}\right)\\
&= \texttt{diag}\left(\texttt{ColSums}(\texttt{B}_{\texttt{O}}) + \texttt{n}\right)^{-1}
\end{aligned}$$

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

$$\vec{p_{i+1}} = \mathrm{B_O}\ \ \mathrm{diag}\left(\vec{1}\mathbf{B_O} + \vec{1}n\right)\vec{p_i} + \vec{1}\vec{\delta}^{\mathrm{T}}\vec{p_i} \tag{36}$$

in terms of ***R***:

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

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

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

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

1. 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

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

   Then we have:

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

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

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

**Practical; Implementing the Power Walk on Sparse Matrices**

1. Inspect the newly created matrix and create constants
2. Setup
   (a) Load Packages

```r
if (require("pacman")) {
    library(pacman)
  }else{
    install.packages("pacman")
    library(pacman)
  }
  pacman::p_load(Matrix, igraph, plotly, mise, docstring, expm)
  mise()
```

```
Loading required package: pacman
```

(b) Define function to create DiagonalsSparse Diagonal Function This doesn't matter for the power walk, real exponents will always give non-zero values anyway

```r
sparse_diag <- function(mat) {
  #' Diagonal Factors of Sparse Matrix
  #'
  #' Return a Diagonal Matrix containing either 1 / colsum() or 0
  ↪  such that
  #' matrix multiplication with this matrix would have all columns
  #' sum to 1
  #'
  #' This should take the transpose of an adjacency matrix in and the
  ↪  output
  #' can be multiplied by the original matrix to scale it to 1.
  #' i
  # mat  <- A
  ## Get the Dimensions
  n <- nrow(mat)

  ## Make a Diagonal Matrix of Column Sums
      ## If a column sums to zero the diag can be zero iff the
      ↪  adjacency_matrix>=0
  D <- sparseMatrix(i = 1:n, j = 1:n, x = colSums(mat), dims = c(n,n))

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

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

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

(c) Make an Example Graph

```r
g1 <- igraph::erdos.renyi.game(n = 20, 0.2)
A <- igraph::get.adjacency(g1) # Row to column


beta = 0.843234
  = beta
```

(d) Plot

```r
plot(g1)
```

3. Power Walk

(a) Define B

```r
B       <- A
B@x     <-  ^(A@x)
B       <- A
B        <-  ^A

Bo      <- A

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


  n <- nrow(A)
```

```r
print(B)
```

```
20 x 20 Matrix of class "dgeMatrix"
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
 [1,] 1.000000 0.843234 1.000000 1.000000 1.000000 0.843234 1.000000 1.000000
 [2,] 0.843234 1.000000 1.000000 1.000000 0.843234 1.000000 1.000000 1.000000
 [3,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
 [4,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234
 [5,] 1.000000 0.843234 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
 [6,] 0.843234 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234
 [7,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
 [8,] 1.000000 1.000000 1.000000 0.843234 1.000000 0.843234 1.000000 1.000000
 [9,] 0.843234 1.000000 1.000000 1.000000 0.843234 1.000000 1.000000 1.000000
[10,] 0.843234 0.843234 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
[11,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
[12,] 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234 1.000000 0.843234
[13,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234
[14,] 1.000000 0.843234 1.000000 0.843234 1.000000 0.843234 1.000000 1.000000
[15,] 0.843234 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
[16,] 0.843234 1.000000 0.843234 1.000000 1.000000 1.000000 1.000000 1.000000
[17,] 1.000000 1.000000 0.843234 0.843234 1.000000 1.000000 0.843234 0.843234
[18,] 1.000000 1.000000 1.000000 1.000000 0.843234 1.000000 1.000000 1.000000
[19,] 1.000000 0.843234 0.843234 1.000000 1.000000 1.000000 1.000000 0.843234
[20,] 0.843234 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
          [,9]    [,10]    [,11]    [,12]    [,13]    [,14]    [,15]    [,16]
 [1,] 0.843234 0.843234 1.000000 1.000000 1.000000 1.000000 0.843234 0.843234
 [2,] 1.000000 0.843234 1.000000 1.000000 1.000000 0.843234 1.000000 1.000000
 [3,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234
 [4,] 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234 1.000000 1.000000
 [5,] 0.843234 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
 [6,] 1.000000 1.000000 1.000000 0.843234 1.000000 0.843234 1.000000 1.000000
 [7,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
 [8,] 1.000000 1.000000 1.000000 0.843234 0.843234 1.000000 1.000000 1.000000
 [9,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
[10,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234 1.000000
[11,] 1.000000 1.000000 1.000000 1.000000 0.843234 1.000000 0.843234 1.000000
[12,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234
[13,] 1.000000 1.000000 0.843234 1.000000 1.000000 1.000000 1.000000 1.000000
[14,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234
[15,] 1.000000 0.843234 0.843234 1.000000 1.000000 1.000000 1.000000 1.000000
[16,] 1.000000 1.000000 1.000000 0.843234 1.000000 0.843234 1.000000 1.000000
[17,] 1.000000 1.000000 0.843234 0.843234 0.843234 1.000000 1.000000 1.000000
[18,] 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
[19,] 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234 1.000000 1.000000
[20,] 1.000000 1.000000 1.000000 1.000000 1.000000 0.843234 1.000000 0.843234
         [,17]    [,18]    [,19]    [,20]
 [1,] 1.000000 1.000000 1.000000 0.843234
 [2,] 1.000000 1.000000 0.843234 1.000000
 [3,] 0.843234 1.000000 0.843234 1.000000
 [4,] 0.843234 1.000000 1.000000 1.000000
 [5,] 1.000000 0.843234 1.000000 1.000000
 [6,] 1.000000 1.000000 1.000000 1.000000
 [7,] 0.843234 1.000000 1.000000 1.000000
 [8,] 0.843234 1.000000 0.843234 1.000000
 [9,] 1.000000 1.000000 1.000000 1.000000
[10,] 1.000000 1.000000 1.000000 1.000000
[11,] 0.843234 1.000000 1.000000 1.000000
[12,] 0.843234 1.000000 1.000000 1.000000
[13,] 0.843234 1.000000 1.000000 1.000000
[14,] 1.000000 1.000000 0.843234 0.843234
[15,] 1.000000 1.000000 1.000000 1.000000
[16,] 1.000000 1.000000 1.000000 0.843234
[17,] 1.000000 0.843234 0.843234 1.000000
[18,] 0.843234 1.000000 0.843234 1.000000
[19,] 0.843234 0.843234 1.000000 1.000000
[20,] 1.000000 1.000000 1.000000 1.000000
```

```r
print(Bo)
```

```
20 x 20 sparse Matrix of class "dgCMatrix"

 [1,]  .         -0.156766  .         .         .         -0.156766  .
 [2,] -0.156766   .         .         .        -0.156766   .         .
 [3,]  .          .         .         .         .          .         .
 [4,]  .          .         .         .         .          .         .
 [5,]  .         -0.156766  .         .         .          .         .
 [6,] -0.156766   .         .         .         .          .         .
 [7,]  .         .         .         .         .         .         .
 [8,]  .         .         .        -0.156766  .        -0.156766  .
 [9,] -0.156766  .         .         .        -0.156766  .         .
[10,] -0.156766 -0.156766  .         .         .         .         .
[11,]  .         .         .         .         .         .         .
[12,]  .         .         .         .         .        -0.156766  .
[13,]  .         .         .         .         .         .         .
[14,]  .        -0.156766  .        -0.156766  .        -0.156766  .
[15,] -0.156766  .         .         .         .         .         .
[16,] -0.156766  .        -0.156766  .         .         .         .
[17,]  .         .        -0.156766 -0.156766  .         .        -0.156766
[18,]  .         .         .         .        -0.156766  .         .
[19,]  .        -0.156766 -0.156766  .         .         .         .
[20,] -0.156766  .         .         .         .         .         .

 [1,]  .        -0.156766 -0.156766  .         .         .         .
 [2,]  .         .        -0.156766  .         .         .        -0.156766
 [3,]  .         .         .         .         .         .         .
 [4,] -0.156766  .         .         .         .         .        -0.156766
 [5,]  .        -0.156766  .         .         .         .         .
 [6,] -0.156766  .         .         .        -0.156766  .        -0.156766
 [7,]  .         .         .         .         .         .         .
 [8,]  .         .         .         .        -0.156766 -0.156766  .
 [9,]  .         .         .         .         .         .         .
[10,]  .         .         .         .         .         .         .
[11,]  .         .         .         .         .        -0.156766  .
[12,] -0.156766  .         .         .         .         .         .
[13,] -0.156766  .         .        -0.156766  .         .         .
[14,]  .         .         .         .         .         .         .
[15,]  .         .        -0.156766 -0.156766  .         .         .
[16,]  .         .         .         .        -0.156766  .        -0.156766
[17,] -0.156766  .         .        -0.156766 -0.156766 -0.156766  .
[18,]  .         .         .         .         .         .         .
[19,] -0.156766  .         .         .         .         .        -0.156766
[20,]  .         .         .         .         .         .        -0.156766

 [1,] -0.156766 -0.156766  .         .         .        -0.156766
 [2,]  .         .         .         .        -0.156766  .
 [3,]  .        -0.156766 -0.156766  .        -0.156766  .
 [4,]  .         .        -0.156766  .         .         .
 [5,]  .         .         .        -0.156766  .         .
 [6,]  .         .         .         .         .         .
 [7,]  .         .        -0.156766  .         .         .
 [8,]  .         .        -0.156766  .        -0.156766  .
 [9,]  .         .         .         .         .         .
[10,] -0.156766  .         .         .         .         .
[11,] -0.156766  .        -0.156766  .         .         .
[12,]  .        -0.156766 -0.156766  .         .         .
[13,]  .         .        -0.156766  .         .         .
[14,]  .        -0.156766  .         .        -0.156766 -0.156766
```

```
[15,]  .         .         .         .         .         .
[16,]  .         .         .         .         .        -0.156766
[17,]  .         .         .        -0.156766 -0.156766  .
[18,]  .         .        -0.156766  .        -0.156766  .
[19,]  .         .        -0.156766 -0.156766  .         .
[20,]  .        -0.156766  .         .         .         .
```

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

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

```
 [1] 0.05290267 0.05203951 0.05120406 0.05120406 0.05120406 0.05161840
 [7] 0.05039501 0.05246754 0.05079631 0.05120406 0.05120406 0.05161840
[13] 0.05120406 0.05246754 0.05120406 0.05203951 0.05379495 0.05120406
[19] 0.05246754 0.05120406
```

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

```
 [1] 0.05290267 0.05203951 0.05120406 0.05120406 0.05120406 0.05161840
 [7] 0.05039501 0.05246754 0.05079631 0.05120406 0.05120406 0.05161840
[13] 0.05120406 0.05246754 0.05120406 0.05203951 0.05379495 0.05120406
[19] 0.05246754 0.05120406
```

(c) Find the Transition Probability Matrix

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

(d) Implement the Loop

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

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

```
[1] "The stationary point is"
 [1] 0.04882572 0.04963556 0.05044542 0.05044541 0.05044543 0.05004049
 [7] 0.05125527 0.04923064 0.05085035 0.05044543 0.05044542 0.05004049
[13] 0.05044542 0.04923064 0.05044543 0.04963557 0.04801586 0.05044542
[19] 0.04923063 0.05044542
```

#### 1.1.10 Creating a Package

# 2 Relating the Power Walk to the Random Surfer

## 2.1 Introduction

These are notes relating to [22, §3.3]

So if a term in the Power Walk can be related to $\alpha$ in the random surfer, which is in turn $\xi_2$, I'll be able to understand it better. [6]

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:

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

## 2.2 Value of [1st Term]

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

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

Using this property we get the following

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

$$(\beta - 1)\,\mathbf{A}_{i,j} = \begin{cases} 0, & \mathbf{A}_{i,j} = 0 \\ \beta - 1, & \mathbf{A}_{i,j} = 1 \end{cases}$$

[6] Although I'm not quite sure why $\alpha$ is $\xi_2$ either

This means we have

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

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

## 2.3 Value of {2nd Term}

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

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

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

putting this all together we can do the following:

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

From above:

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

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

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

But the next step requires showing that:

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

## 2.4 Equate the Power Walk to the Random Surfer

Define the matrix $\mathbf{D}_{\mathbf{M}}$:

$$\mathbf{D}_{\mathbf{M}} = \mathrm{diag}\left(\mathtt{colSum}\left(\mathbf{M}\right)\right) = \mathrm{diag}\left(\vec{1}\mathbf{M}\right) \tag{37}$$

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

$$\mathbf{D}_{\mathbf{A}}^{-1} : \left[\mathbf{D}_{\mathbf{A}}^{-1}\right]_i = \begin{cases} 0, & \left[\mathbf{D}_{\mathbf{A}}\right]_i = 0 \\ \left[\frac{1}{\mathbf{D}_{\mathbf{A}}}\right], & \left[\mathbf{D}_{\mathbf{A}}\right]_i \neq 0 \end{cases} \tag{38}$$

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

$$\mathbf{D}_{\mathbf{B}}^{-1} = \frac{1}{\mathrm{diag}\left(\vec{1}\left(\beta^{\mathbf{A}}\right)\right)} \tag{39}$$

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

**Power Walk**

$$\mathbf{T} = \boxed{\mathbf{A}\mathbf{D}_{\mathbf{A}}^{-1}}\mathbf{D}_{\mathbf{A}}\left(\beta - 1\right)\mathbf{D}_{\mathbf{B}}^{-1} + \boxed{\mathbf{E}}n\mathbf{D}_{\mathbf{B}}^{-1} \tag{40}$$

**Random Surfer**

$$\mathbf{T} = \alpha\boxed{\mathbf{A}\mathbf{D}_{\mathbf{A}}^{-1}} + \left(1 - \alpha\right)\boxed{\mathbf{E}} \tag{41}$$

So these are equivalent when:

$$\mathbf{D}_{\mathbf{A}}\left(\beta - 1\right)\mathbf{D}_{\mathbf{B}^{-1}} = \mathbf{I}\alpha \tag{42}$$

$$\begin{aligned} \vec{1}\left(1 - \alpha\right) &= -n\mathbf{D}_{\mathbf{B}}^{-1} \\ \implies \vec{1}\alpha &= \vec{1} - n\mathbf{D}_{\mathbf{B}}^{-1} \end{aligned} \tag{43}$$

Hence we have:

$$\mathbf{D}_{\mathbf{A}}\left(\beta - 1\right)\mathbf{D}_{\mathbf{B}}^{-1} = \vec{1}\alpha = \mathbf{I} - n\mathbf{D}_{\mathbf{B}}^{-1} \tag{44}$$

Solving for $\beta$ with (42) :

$$\beta = \frac{1-\Theta}{\Theta} \tag{45}$$

$$\tag{46}$$

where: [7]

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

but we can't really do this so instead:

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

If $\beta$ is set accordingly then by (44):

$$\begin{aligned} &\mathbf{A}\,(\beta-1)\,\mathbf{D_B}^{-1} = \alpha = \mathbf{I} - n\mathbf{D_B}^{-1} \\ \implies &\mathbf{A}\,(\beta-1)\,\mathbf{D_B}^{-1} = \mathbf{I} - n\mathbf{D_B}^{-1} \end{aligned} \tag{47}$$

And setting $\Gamma = \mathbf{I} - n\mathbf{D_B}^{-1}$ from (43) and putting in (40) we have:

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

$$\mathbf{T} = \Gamma\mathbf{A}\mathbf{D_A}^{-1} + (1-\Gamma)\,\mathbf{E} \tag{48}$$

Where $\mathbf{E}$ is square matrix of $\frac{1}{n}$ as in (9) (16)

## 2.5 Conclusion

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

## 2.6 The Second Eigenvalue

### 2.6.1 The Random Surfer

The Second eigenvalue $\xi_2$ of the Power Surfer is less than $\alpha$ (See 3.2; Stability and Concvergence, of proposal).

### 2.6.2 Power Walk

Because the Power Walk relates to the random surfer as demonstrated in section 2 , what can be said about $\xi_2$

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

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

$$\Lambda_{(2)}\left(\mathbf{T}_{\text{RS}}\right) \leq |\alpha| \implies \Lambda_{(2)}\left(\mathbf{T}_{\text{PW}}\right) \leq \left| \frac{\alpha - \mathbf{D_a D_B^{-1}}}{\mathbf{D_A D_B^{-1}}} \right| \tag{49}$$

where:

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

**My attempt**

$$\beta \mathbf{1}_{[n,n]} = \frac{1 - \Theta}{\Theta} \tag{50}$$

$$\tag{51}$$

where:

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

So I thought maybe if I could find a value of $\beta$ that satisfied (50) then I could show circumstances under which $|\xi_2| < \alpha$.

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

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

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

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

 THETA <- DA %*% DB_inv

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

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

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

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


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

beta


diag(beta_func(A, beta))
beta
```

```r

## * blah
```

# 3 Investigating the Second EigenValue

Maybe I should look at the most appropriate way to simulate social network links, one possibility is this paper [27].

Actually there is a data set available [10], I should just analyse that, see how it was done in Visual Analytics as a reminder.

Using the Wikipedia ArtYeah I think that's right, thaicle compare density and Determinant.

Is the determinant easily calculated for a large matrix? It appears to diverge

Will the determinant diverge for large matrices? Will the prob of making edges in the game just be the density?

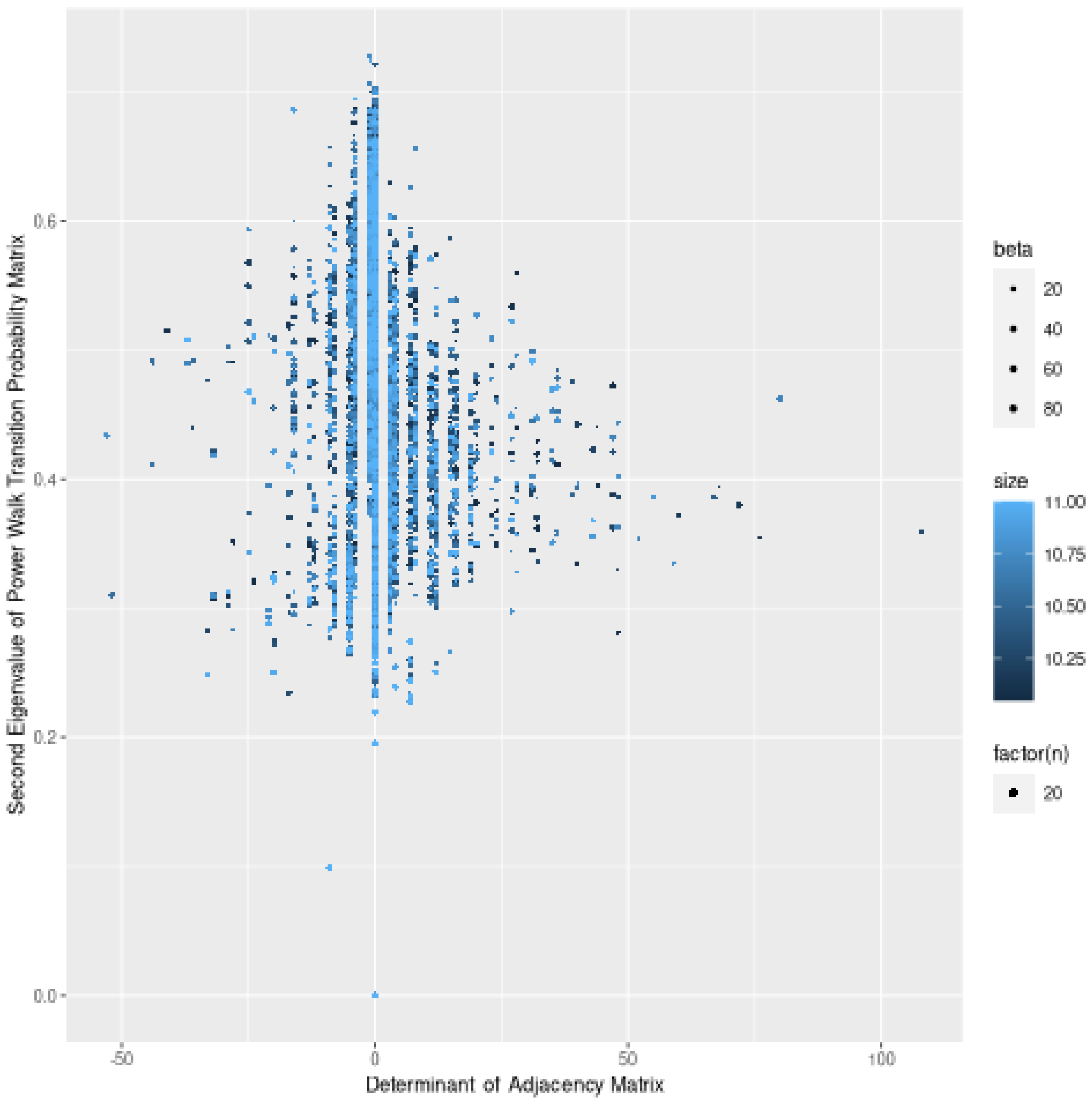
Look at comparing the determinant and the density of the wikipedia adjacency matrix.

What are some ways that we can model the second eigenvalue?

## 3.1 Plotting Various Values

There is some relationship between the determinant and the density, check out the pairs plot:

```r
  library(pacman)
  pacman::p_load(PageRank, devtools, Matrix, igraph, tidyverse)
n <- 20
p <- 1:n/n
beta <- 1:n/n
beta <- runif(n)*100
sz <- 1:n/n+10
input_var <- expand.grid("n" = n, "p" = p, "beta" = beta, "size" = sz)
input_var


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

      A_dens <- mean(A)
      T      <- PageRank::power_walk_prob_trans(A)
      e2     <- eigen(T, only.values = TRUE)$values[2] # R orders by
       ↪  descending magnitude
      A_det  <- det(A)
      return(c(abs(e2), A_det))
}

## TODO this should use pmap.
Y <- matrix(ncol = 2, nrow = nrow(input_var))
for (i in 1:nrow(input_var)) {
  X <- as.vector(input_var[i,])
  Y[i,] <-  random_graph(X$n, X$p, X$beta, X$size)
}
if (sum(abs(Y) != abs(Re(Y))) == 0) {
  Y <- Re(Y)
}
nrow(input_var)
nrow(Y)
Y <- as.data.frame(Y); colnames(Y) <- c("eigenvalue2", "determinant")

data <- cbind(input_var, Y)

ggplot(data, aes(x = determinant, y = eigenvalue2, size = beta, color = size,
 ↪  shape = factor(n))) +
  geom_point() +
  labs(x = "Determinant of Adjacency Matrix", y = "Second Eigenvalue of Power
   ↪  Walk Transition Probability Matrix") +
  scale_size_continuous(range = c(0.1,1))
```

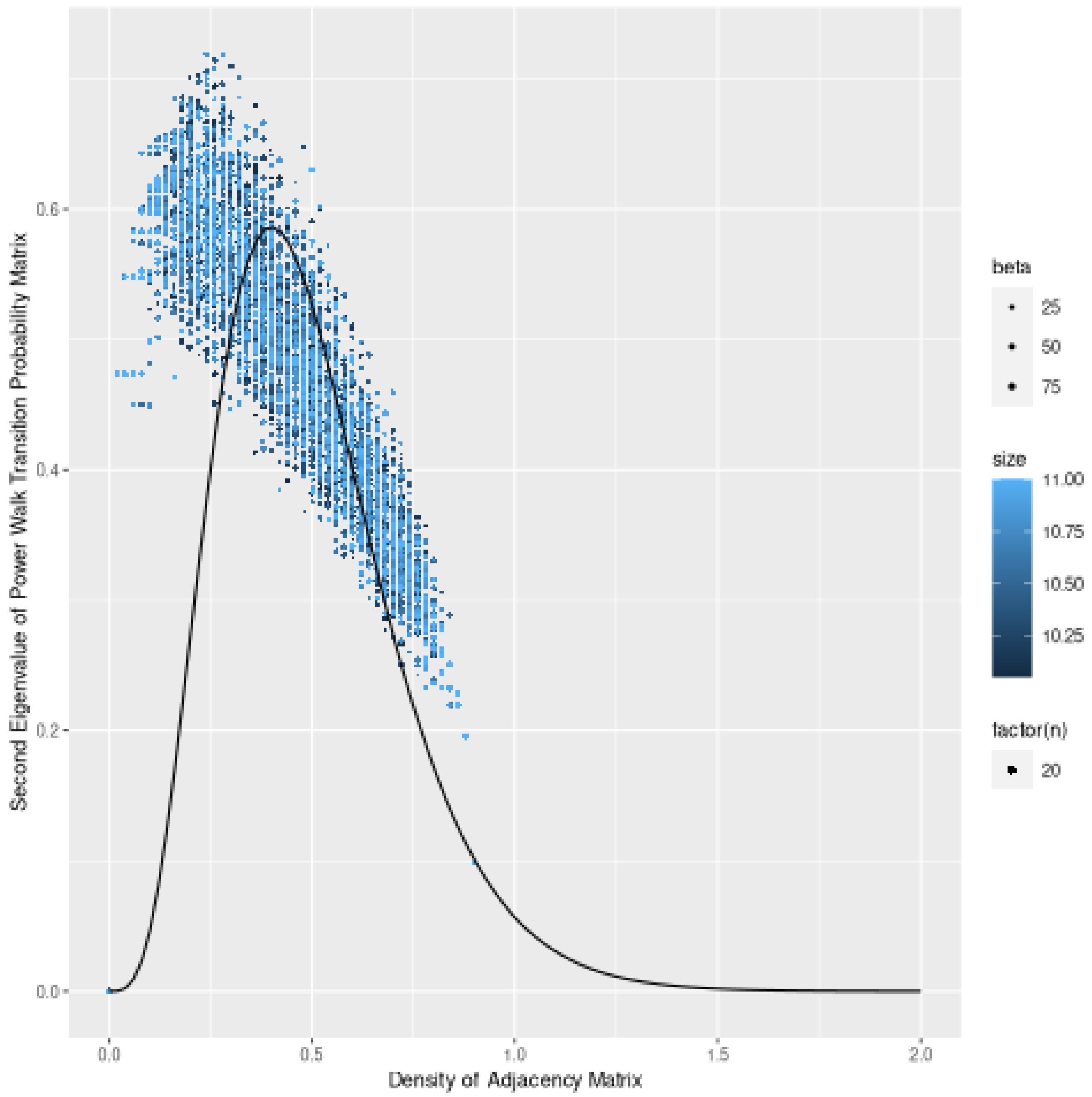


Second Eigenvalue of Power Walk Transition Probability Matrix
Determinant of Adjacency Matrix
0.0
0.2
0.4
0.6
-50
0
50
100
beta
20
40
60
80
size
11.00
10.75
10.50
10.25
factor(n)
20

```r
  library(pacman)
  pacman::p_load(PageRank, devtools, Matrix, igraph, tidyverse)
n <- 100
p <- 1:n/n
beta <- 1:n/n
beta <- runif(n)*100
sz <- 1:n/n+10
input_var <- expand.grid("n" = n, "p" = p, "beta" = beta, "size" = sz)
input_var


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

      A_dens <- mean(A)
      T      <- PageRank::power_walk_prob_trans(A)
      e2     <- eigen(T, only.values = TRUE)$values[2] # R orders by
       ↪  descending magnitude
      A_det  <- det(A)
      return(c(abs(e2), A_dens))
}

## TODO this should use pmap.
Y <- matrix(ncol = 2, nrow = nrow(input_var))
for (i in 1:nrow(input_var)) {
  X <- as.vector(input_var[i,])
  Y[i,] <-  random_graph(X$n, X$p, X$beta, X$size)
}
if (sum(abs(Y) != abs(Re(Y))) == 0) {
  Y <- Re(Y)
}
nrow(input_var)
nrow(Y)
Y <- as.data.frame(Y); colnames(Y) <- c("eigenvalue2", "determinant")

data <- cbind(input_var, Y)

ggplot(data, aes(x = determinant, y = eigenvalue2, color = size, shape =
 ↪  factor(n))) +
  geom_point(base_size = 99, aes(size = beta)) +
  labs(x = "Density of Adjacency Matrix", y = "Second Eigenvalue of Power Walk
   ↪  Transition Probability Matrix") +
  scale_size_continuous(range = c(0.1,1))
```

Maybe this looks like a Chi distribution?

```r
chival <- dchisq(seq(from = 0, to = 40, length.out = 100), df = 10)*6
index  <- seq(from = 0, to = 2, length.out = 100)
chidata  <- data.frame(index = index, chi = chival)
ggplot(data) +
  geom_point(mapping = aes(x = determinant, y = eigenvalue2, size = beta,
  ↪   color = size, shape = factor(n))) +
  geom_line(data = chidata, mapping = aes(x = index, y = chi)) +
  scale_size_continuous(range = c(0.1,1)) +
  labs(x = "Density of Adjacency Matrix", y = "Second Eigenvalue of Power Walk
  ↪   Transition Probability Matrix")
```



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

$$\xi_2 = \left(1 - \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{A}_{i,j}}{n^2}\right)^{0.6} \cdot e^{-0.48} \pm \Delta \tag{52}$$

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

### 3.3 Could I get better performance by also considering the determinant?

No not really, it terms of accuracy

### 3.4 Is the determinant faster or slower?

Significantly slower for large matrices.

### 3.5 Import wikipedia data

- ~~Import the wikipedia data~~
- ~~Measure the density~~
- ~~Use the density to guess the $p$ of the game~~
  - ~~Justify the witht the scatterplot matrix~~
- ~~Measure the affect of different $\beta$ values on $\lambda_2$ for graphs ov various sizes given that $p$ value.~~
  - ~~Or atleast a range within that prob~~
    use a *Barabassi-Albert* Random Graph through the ~igraph::

## 4 Cauchy Integral Formula

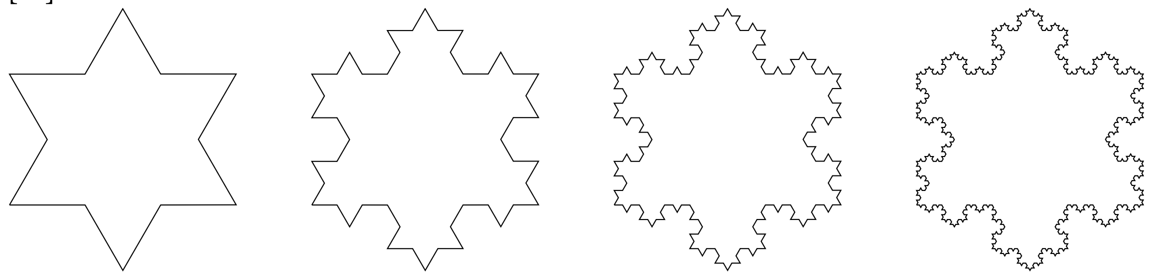
This is from section 54 of the book, isn't it nice that it more or less just works hey? [28]

$$f\left(a\right)\frac{1}{2\pi i}\oint\frac{f\left(z\right)}{z-a}\mathrm{d}z \tag{53}$$

In view of this equation then: [28]

$$\left|\int_C \frac{f\left(z\right)}{z-z_0}\mathrm{d}z - 2\pi i f\left(z_0\right)\right| < 2\pi\varepsilon$$

Some Images: [21]



@@latex: $n = 1$ $n = 2$ $n = 3$ $n = 4$

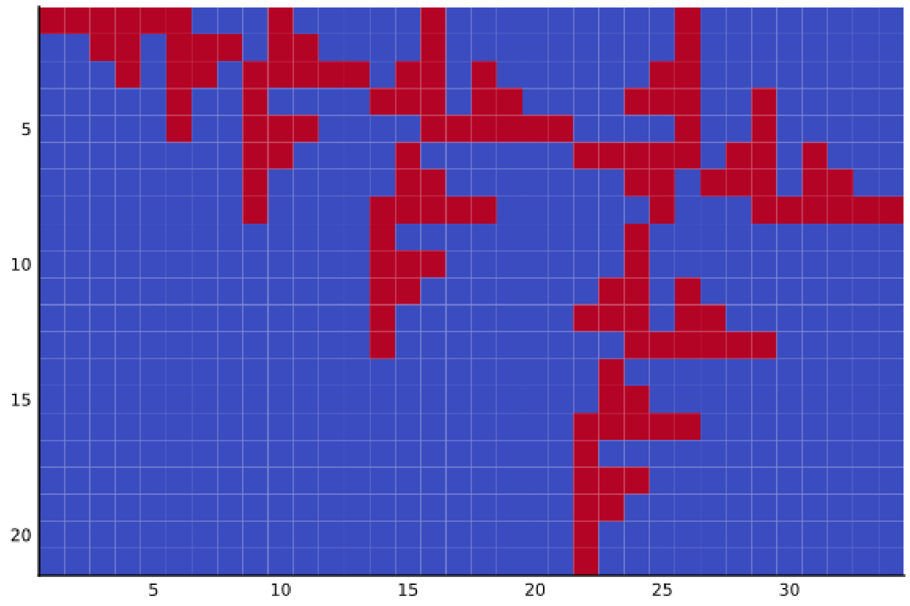


Figure 5: This image is for testing purposes [19]

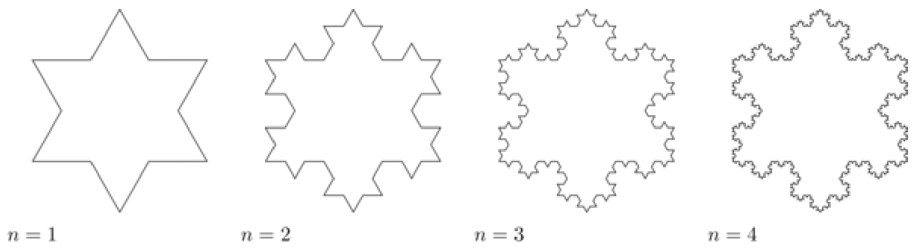


Figure 6: This is a tikz image inserted as a png from imagemagick

### 4.1 Heading 2

#### 4.1.1 Heading 3

```
echo "Hello World"
```

**Heading 4**

**Heading 5**

1. Heading 6 Arbitrary Code:

```
n/bash

# Print Help
if [ "$1" == "-h" ]; then
    echo "Usage: `basename $0` <Format> <CSS>"
    style=~/Dropbox/profiles/Emacs/org-css/github-org.css
    exit 0
fi

# Make a working File from clipboard
filename=lkjdskjjalkjkj392jlkj
xclip -o -selection clipboard >> $filename
LocalFile=$filename.org

pandoc -s  -f org -t gfm $filename -o $filename

echo "
This was converted from `org` to `md` using `pandoc -t gfm` at time:
$(date --utc +%FT%H-%M-%S)
" >> $filename

cat $filename | xclip -selection clipboard
rm $filename

nv & disown
echo "Conversion from Org Successful, MD is in Clipboard"

exit 0
```

## 5 Appendix

## 6 Extensions to this Report

Accellerating the Computatoin of Page Rank [15]

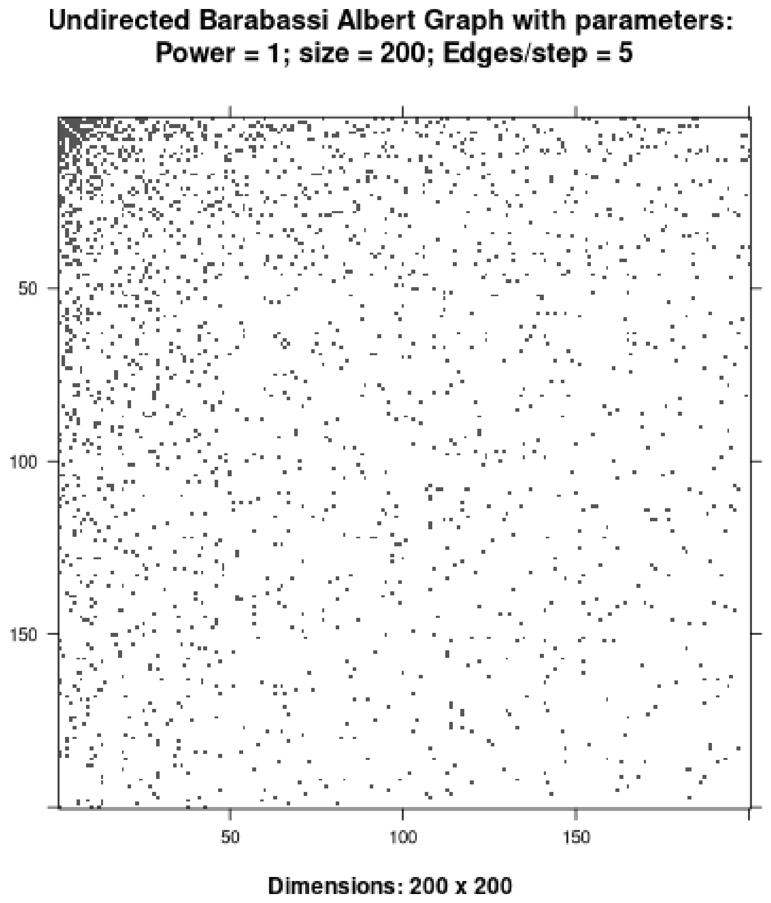


Figure 7: Plot of the adjacency matrix corresponding to a Barabassi-Albert Graph produced by listing 1, the matrix is quite sparse.

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


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


# create the image

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

Listing 1: Use ***R*** to produce an image illustrating the density of a simulated BA graph, see the output in figure 7

## 7 Is the power Walk transition prob matrix a stochastic because it may contain negatives?

## 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. 2).

[2] 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). The choice of damping factor of Googles page rank might have a large impact on the values given to vertices.
This suggests an approach that uses structural network dynatims to provide an appropriate score distribution.
The method implemented is not something I have come yet to understand, but it could be very interesting to see:
- how it relates to the power walk method
- whether or not it could offer insightts into the convergence and stability of the power walk method
- Whether or not the method would be compatible with negatively weighted edges., pp. 3–33. ISSN: 1573-7594. DOI: 10.1007/s10626-017-0248-7. URL: https://doi.org/10.1007/s10626-017-0248-7 (visited on 08/19/2020) (cit. on p. 4).

[3] Monica Bianchini, Marco Gori, and Franco Scarselli. "Inside PageRank". In: *ACM Trans. Inter. Tech.* 5.1 (Feb. 1, 2005). This is a discussion on the stability, complexity and critical role of parameters involved in the computation., pp. 92–128. ISSN: 15335399. DOI: 10.1145/1052934.1052938. URL: http://portal.acm.org/citation.cfm?doid=1052934.1052938 (visited on 08/18/2020) (cit. on p. 4).

[4] Michael Brinkmeier. “PageRank Revisited”. In: *ACM Transactions on Internet Technology* 6.3 (Aug. 2006), pp. 282–301. ISSN: 15335399. DOI: 10.1145/1151087.1151090. URL: http://ezproxy.uws.edu.au/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=22173011&site=ehost-live&scope=site (visited on 08/19/2020) (cit. on p. 4).

[5] 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. 3).

[6] *DOT (Graph Description Language)*. In: *Wikipedia*. June 11, 2020. URL: https://en.wikipedia.org/w/index.php?title=DOT_(graph_description_language)&oldid=961944797 (visited on 10/09/2020) (cit. on p. 6).

[7] François Fouss, Marco Saerens, and Masashi Shimbo. *Algorithms and Models for Network Data and Link Analysis*. New York, NY: Cambridge University Press, 2016. 521 pp. ISBN: 978-1-107-12577-3 (cit. on pp. 3, 4).

[8] Hwai-Hui Fu, Dennis K. J. Lin, and Hsien-Tang Tsai. “Damping Factor in Google Page Ranking”. In: *Applied Stochastic Models in Business and Industry* 22.5-6 (2006), pp. 431–444. ISSN: 1526-4025. DOI: 10.1002/asmb.656. URL: http://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.656 (visited on 08/19/2020) (cit. on p. 4).

[9] 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. 2).

[10] Andrea Garritano. *Wikipedia Article Networks*. Dec. 2019. URL: https://kaggle.com/andreagarritano/wikipedia-article-networks (visited on 10/03/2020) (cit. on p. 33).

[11] *Google Press Center: Fun Facts*. July 15, 2001. URL: https://web.archive.org/web/20010715123343/https://www.google.com/press/funfacts.html (visited on 10/09/2020) (cit. on p. 7).

[12] Pankaj Gupta et al. “WTF: The Who to Follow Service at Twitter”. In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW ’13. New York, NY, USA: Association for Computing Machinery, May 13, 2013, pp. 505–514. ISBN: 978-1-4503-2035-1. DOI: 10.1145/2488388.2488433. URL: http://doi.org/10.1145/2488388.2488433 (visited on 10/09/2020) (cit. on pp. 3, 5).

[13] Sepandar Kamvar, Taher Haveliwala, and Gene Golub. “Adaptive Methods for the Computation of PageRank”. In: *Linear Algebra and its Applications*. Special Issue on the Conference on the Numerical Solution of Markov Chains 2003 386 (July 15, 2004), pp. 51–65. ISSN: 0024-3795. DOI: 10.1016/j.laa.2003.12.008. URL: http://www.sciencedirect.com/science/article/pii/S0024379504000023 (visited on 08/19/2020) (cit. on p. 4).

[14] Moshe Koppel and Nadav Schweitzer. “Measuring Direct and Indirect Authorial Influence in Historical Corpora”. In: *Journal of the Association for Information Science and Technology* 65.10 (2014), pp. 2138–2144. ISSN: 2330-1643. DOI: 10.1002/asi.23118. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.23118 (visited on 08/21/2020) (cit. on p. 4).

[15] 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. 2–5, 40).

[16] Larry Page and Sergey Brin. “The Anatomy of a Large-Scale Hypertextual Web Search Engine”. In: *Computer Networks and ISDN Systems* 30.1-7 (Apr. 1, 1998), pp. 107–117. ISSN: 0169-7552. DOI: 10.1016/S0169-7552(98)00110-X. URL: http://www.sciencedirect.com/science/article/pii/S016975529800110X (visited on 08/19/2020) (cit. on pp. 5, 7).

[17] Ron Larson and Bruce H. Edwards. *Elementary Linear Algebra*. 2nd ed. Includes index. Lexington, Mass: D.C. Heath, 1991. 592 pp. ISBN: 978-0-669-24592-9 (cit. on p. 3).

[18] 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. 2).

[19] Paula Moskowitz. *Library Guides: Wikipedia: Should You Use Wikipedia?* URL: https://mville.libguides.com/c.php?g=370066&p=2500344 (visited on 08/19/2020) (cit. on p. 39).

[20] Nathanael Ackerman, Cameron Freer,Alex Kruckman, and Rehana Patel. *Properly Erodic Structures.* Oct. 25, 2017. URL: https://math.mit.edu/~freer/papers/properly-ergodic-structures.pdf (visited on 10/10/2020) (cit. on p. 3).

[21] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. "Stable Algorithms for Link Analysis". In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '01. New York, NY, USA: Association for Computing Machinery, Sept. 1, 2001, pp. 258–266. ISBN: 978-1-58113-331-8. DOI: 10.1145/383952.384003. URL: http://doi.org/10.1145/383952.384003 (visited on 08/19/2020) (cit. on p. 38).

[22] Laurence A. F. Park and Simeon Simoff. "Power Walk: Revisiting the Random Surfer". In: *Proceedings of the 18th Australasian Document Computing Symposium.* ADCS '13. Brisbane, Queensland, Australia: Association for Computing Machinery, Dec. 5, 2013, pp. 50–57. ISBN: 978-1-4503-2524-0. DOI: 10.1145/2537734.2537749. URL: http://doi.org/10.1145/2537734.2537749 (visited on 07/31/2020) (cit. on p. 27).

[23] Kenneth H. Rosen. *Discrete Mathematics and Its Applications.* 6th ed. The foundations : logic and proofs – Basic structures : sets, functions, sequences, and sums – The fundamentals : algorithms, the integers, and matrices – Induction and recursion – Counting – Discrete probability – Advanced counting techniques – Relations – Graphs – Trees – Boolean algebra – Modeling computation. Boston: McGraw-Hill Higher Education, 2007. 843 pp. ISBN: 978-0-07-288008-3 978-0-07-322972-0 (cit. on p. 2).

[24] 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. 3).

[25] Stan Development Team. *6 Sparse Matrix Operations | Stan Functions Reference.* URL: https://mc-stan.org/docs/2_22/functions-reference/sparse-matrices.html (visited on 08/07/2020) (cit. on p. 1).

[26] *The DOT Language.* URL: https://graphviz.org/doc/info/lang.html (visited on 10/09/2020) (cit. on p. 6).

[27] Rui Zeng et al. "A Practical Simulation Method for Social Networks". In: 144 (2013), p. 8 (cit. on p. 33).

[28] Hui Zhang et al. "Making Eigenvector-Based Reputation Systems Robust to Collusion". In: *Algorithms and Models for the Web-Graph.* Ed. by Stefano Leonardi. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 92–104. ISBN: 978-3-540-30216-2. DOI: 10.1007/978-3-540-30216-2_8 (cit. on p. 38).