

# R Notebook

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

```
## Loading required package: pacman
```

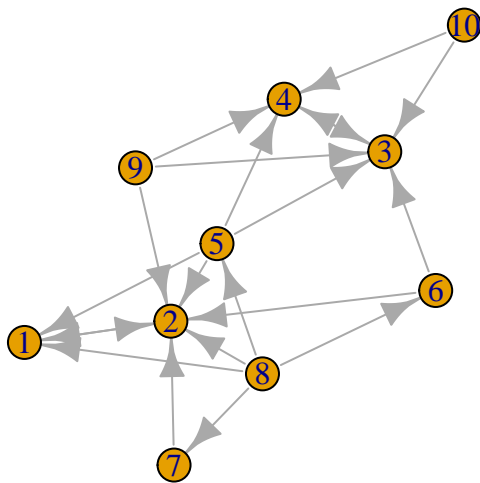
```
pacman::p_load(tidyverse)
```

## Implementing Page Rank Methods

### Example Graph

Consider the following Graph taken from the paper:

```
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,  
plot(g1)
```



### Adjacency Matrix

The adjacency Matrix is given by:

```
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
```

## State Distribution

The state distribution is the transpose of the adjacency matrix:

```
(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 0 1 0
## 6  0 0 0 0 0 0 0 0 1 0
## 7  0 0 0 0 0 0 0 0 1 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
```

## 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.

```
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
```

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

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

## Create a Function

```
##      [,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  $S$ :

$$S = \lambda T + (1 - \lambda) B : \quad (1)$$

(2)

$$B = \begin{bmatrix} \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \end{bmatrix} \quad (3)$$

$$N = ||V|| \quad (4)$$

$$\lambda \in [0, 1] \quad (5)$$

```
B <- matrix(rep(1/nrow(T), length.out = nrow(T)**2), nrow = nrow(T))
l <- 0.8

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

## Eigen Value Method

The eigenvector corresponding to the the eigenvalue of 1 will be the stationary point:

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

```
## eigen() decomposition
## $values
## [1] 1.000000e+00 8.000000e-01 -8.000000e-01 -8.000000e-01 3.117428e-09
## [6] -3.117428e-09 1.233252e-17 -6.831762e-18 1.351981e-18 1.827902e-34
##
## $vectors
##          [,1]          [,2]          [,3]          [,4]          [,5]
## [1,] -0.48531271 5.000000e-01 1.074642e-03 7.070996e-01 6.735753e-01
## [2,] -0.52732002 5.000000e-01 -1.074642e-03 -7.070996e-01 9.622505e-02
## [3,] -0.49152601 -5.000000e-01 7.071060e-01 3.197134e-03 9.622505e-02
## [4,] -0.47977477 -5.000000e-01 -7.071060e-01 -3.197134e-03 2.886751e-01
## [5,] -0.05288058 7.620569e-17 1.212133e-16 -3.419297e-18 -3.849002e-01
## [6,] -0.05288058 7.620569e-17 -7.099634e-17 -3.419297e-18 -3.849002e-01
## [7,] -0.05288058 7.620569e-17 2.943750e-17 -3.419297e-18 -3.849002e-01
## [8,] -0.04558671 6.926804e-17 -1.948070e-18 -4.183296e-17 -7.499367e-09
```

```
## [9,] -0.04558671 6.926804e-17 7.359376e-18 -4.183296e-17 -7.499367e-09
## [10,] -0.04558671 6.926804e-17 7.359376e-18 -4.183296e-17 -7.499367e-09
##           [,6]           [,7]           [,8]           [,9]           [,10]
## [1,] -6.735753e-01 -2.658112e-01 5.357798e-01 -4.171123e-01 2.122431e-01
## [2,] -9.622504e-02 1.896313e-01 -1.330801e-01 -7.334472e-02 1.241240e-01
## [3,] -9.622504e-02 1.990137e-01 -1.665251e-01 2.591495e-01 -1.598069e-04
## [4,] -2.886751e-01 -3.972519e-02 1.099005e-02 -1.886035e-02 6.690506e-02
## [5,] 3.849002e-01 -7.585250e-01 5.323206e-01 2.933789e-01 -4.964959e-01
## [6,] 3.849002e-01 4.774778e-01 -3.550303e-01 5.560197e-01 -1.341297e-01
## [7,] 3.849002e-01 2.204565e-01 -5.047228e-01 1.987554e-01 1.561493e-01
## [8,] -7.499367e-09 -1.208463e-16 1.668414e-16 -2.685569e-16 3.416862e-17
## [9,] -7.499367e-09 -1.125898e-02 4.013399e-02 -3.989931e-01 -5.316106e-01
## [10,] -7.499367e-09 -1.125898e-02 4.013399e-02 -3.989931e-01 6.029746e-01
```

So in this case the a 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:

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

```
(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

(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
```

## Power Value Method

Using the power method should give the same result:

```
p <- c(1, 0, 0, 0, 0, 0, 0, 0, 0, 0)

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

p_new

##           [,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
```

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
```

### Comparison

So why doesn't the eigen value

\$