Ginorio_Final

MGinorio

12/11/2021

Problem 1

Environment Setup

```
library(matrixcalc)
library(igraph)
```

Form the A matrix

```
p1 <- c(0, 1/2, 1/2, 0, 0, 0)
p2 <- rep(1/6, 6) #dangling node if leave 0 -  fix probability
p3 <- c(1/3, 1/3, 0, 0, 1/3, 0)
p4 <- c(0, 0, 0, 0, 1/2, 1/2)
p5 <- c(0, 0, 0, 1/2, 0, 1/2)
p6 <- c(0, 0, 0, 1, 0, 0)

A <- matrix(c(p1, p2, p3, p4, p5, p6), 6)

A
```

```
##      [,1]      [,2]      [,3] [,4] [,5] [,6]
## [1,]  0.0 0.1666667 0.3333333  0.0  0.0    0
## [2,]  0.5 0.1666667 0.3333333  0.0  0.0    0
## [3,]  0.5 0.1666667 0.0000000  0.0  0.0    0
## [4,]  0.0 0.1666667 0.0000000  0.0  0.5    1
## [5,]  0.0 0.1666667 0.3333333  0.5  0.0    0
## [6,]  0.0 0.1666667 0.0000000  0.5  0.5    0
```

```
# check and adjust probability for each column (row)
colSums(A)
```

```
## [1] 1 1 1 1 1 1
```

Introduce Decay

```
B <- 0.85 * A + 0.15/nrow(A)
```

Uniform Rank Vector

$r = B^n * r$

```
r <- rep(1/nrow(A), nrow(A))
```

Power Iterations

```
cbind(matrix.power(B,10) %*% r,
      matrix.power(B,20) %*% r,
      matrix.power(B,30) %*% r,
      matrix.power(B,40) %*% r,
      matrix.power(B,50) %*% r,
      matrix.power(B,60) %*% r)
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] 0.05205661 0.05170616 0.05170475 0.05170475 0.05170475 0.05170475
## [2,] 0.07428990 0.07368173 0.07367927 0.07367926 0.07367926 0.07367926
## [3,] 0.05782138 0.05741406 0.05741242 0.05741241 0.05741241 0.05741241
## [4,] 0.34797267 0.34870083 0.34870367 0.34870369 0.34870369 0.34870369
## [5,] 0.19975859 0.19990313 0.19990381 0.19990381 0.19990381 0.19990381
## [6,] 0.26810085 0.26859408 0.26859607 0.26859608 0.26859608 0.26859608
```

Page Rank PR1 with Power Iterations

```
PR1 <- matrix.power(B, 40) %*% r   #convergence happens at 40
```

Eigen Decomposition

Compute the eigen-decomposition of B and verify that you indeed get an eigenvalue of 1 as the largest eigenvalue and that its corresponding eigenvector is the same vector that you obtained in the previous power iteration method. Further, this eigenvector has all positive entries and it sums to 1

Page Rank PR2 with Eigen Decomposition

```
eigen_decom <- eigen(B)
PR2 <- as.numeric(eigen_decom$vectors[,which.max(eigen_decom$values)])
#get vectors associated with largest eigenvalue == 1
#locate max values of each vector of numeric (as.numeric) transform

PR2 <- (1/sum(PR2))*PR2 #normalize
PR2
```

```
## [1] 0.05170475 0.07367926 0.05741241 0.34870369 0.19990381 0.26859608
```

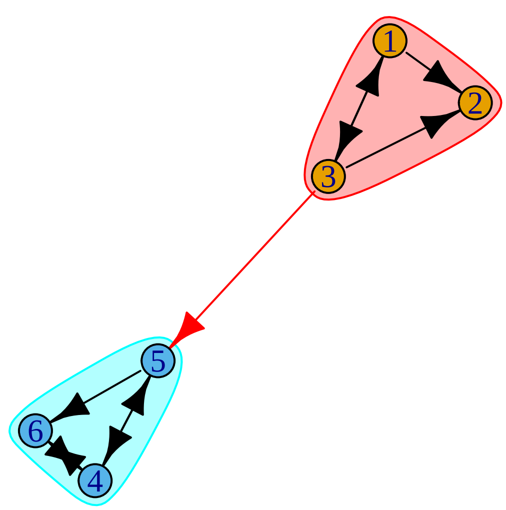
```
# return to A_0 matrix without the normalized row
p2_0 <- rep(0, 6)
A_0 <- matrix(c(p1, p2_0, p3, p4, p5, p6), 6)
A_0
```

```
##      [,1] [,2]      [,3] [,4] [,5] [,6]
## [1,]  0.0    0 0.3333333  0.0  0.0    0
## [2,]  0.5    0 0.3333333  0.0  0.0    0
## [3,]  0.5    0 0.0000000  0.0  0.0    0
## [4,]  0.0    0 0.0000000  0.0  0.5    1
## [5,]  0.0    0 0.3333333  0.5  0.0    0
## [6,]  0.0    0 0.0000000  0.5  0.5    0
```

Igraph Library

Use the graph package in R and its page.rank method to compute the Page Rank of the graph as given in A. Note that you don't need to apply decay. The package starts with a connected graph and applies decay internally. Verify that you do get the same PageRank vector as the two approaches above.

```
a <- graph.adjacency(t(A_0), weighted = TRUE, mode = 'directed')
ceb <- cluster_edge_betweenness(a)
plot(ceb,a)
```



```r
PR3_info <- page.rank(a)
PR3 <- page.rank(a)$vector
PR3_info
```

```
## $vector
## [1] 0.05170475 0.07367926 0.05741241 0.34870369 0.19990381 0.26859608
##
## $value
## [1] 1
##
## $options
## NULL
```

```r
results <- cbind(PR1, PR2, PR3)
results <- rbind(results, colSums(results))

colnames(results) <- c('PowerIteration', 'EigenDecomp', 'IgraphTool')
row.names(results) <- c('p1', 'p2', 'p3','p4', 'p5','p6', 'colSum')

knitr::kable(results)
```

	PowerIteration	EigenDecomp	IgraphTool
p1	0.0517047	0.0517047	0.0517047
p2	0.0736793	0.0736793	0.0736793
p3	0.0574124	0.0574124	0.0574124
p4	0.3487037	0.3487037	0.3487037
p5	0.1999038	0.1999038	0.1999038
p6	0.2685961	0.2685961	0.2685961
colSum	1.0000000	1.0000000	1.0000000

Problem 2

Problem 3