Page Rank

STA 325: Lab 4, Fall 2017

ATTN: I'm having trouble finding a link matrix that's non trivial. Fix this.

Today's agenda: Page rank

Programming partner's: You should have a programming partner for each lab, and you should switch off who is programming, and use each other for help. We will spend about 30–50 minutes per week on lab exercises and you will be expected to bring you laptops to class to work on these exercises in class. Myself and the TA's will be in class to help you.

Lab Tasks

1. Read in the file titled pageRank.RData. Use the ls() command and ensure that the link matrix myData is present in the environment.

```
# Load the data #
load(file='pageRank.RData')

# Create a Link Count #
M <- diag(apply(myData, 2, sum))
```

2. Count the number of links for each page (at each row) and place these into a diagonal matrix. Call this matrix M.} \item{Invert the matrix M and store the inverse into the variable MInv.

```
# Invert the matrix #
MInv <- solve(M)
```

3. Based on the link matrix and the matrix M compute the Broken Rank. What is the problem here? Hint: Before performing the calculation remember to transpose your link matrix since the you need the links to go from $j \to i$.

```
# Broken Rank #
(BR <- t(myData) %*% MInv)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    0    0    0
## [2,]    0    0    1    0    0
## [3,]    1    0    0    0    0
## [4,]    0    0    0    0    1
## [5,]    0    0    0    1    0
```

```
# Calculate the eigen values/vectors #
(BRresults <- eigen(BR))
```

```
## eigen() decomposition
## $values
## [1] -0.5+0.8660254i -0.5-0.8660254i  1.0+0.0000000i -1.0+0.0000000i
## [5]  1.0+0.0000000i
##
## $vectors
##                 [,1]            [,2]         [,3]          [,4]
## [1,]  0.5773503+0.0i  0.5773503+0.0i 0.0000000+0i  0.0000000+0i
## [2,] -0.2886751+0.5i -0.2886751-0.5i 0.0000000+0i  0.0000000+0i
## [3,] -0.2886751-0.5i -0.2886751+0.5i 0.0000000+0i  0.0000000+0i
## [4,]  0.0000000+0.0i  0.0000000+0.0i 0.7071068+0i -0.7071068+0i
## [5,]  0.0000000+0.0i  0.0000000+0.0i 0.7071068+0i  0.7071068+0i
##                [,5]
## [1,] -0.5773503+0i
## [2,] -0.5773503+0i
## [3,] -0.5773503+0i
## [4,]  0.0000000+0i
## [5,]  0.0000000+0i
```

```
# Isolate the results #
(BrokenRank <- BRresults$vectors[,which(signif(BRresults$values, 6) == 1)])
```

```
##               [,1]          [,2]
## [1,] 0.0000000+0i -0.5773503+0i
## [2,] 0.0000000+0i -0.5773503+0i
## [3,] 0.0000000+0i -0.5773503+0i
## [4,] 0.7071068+0i  0.0000000+0i
## [5,] 0.7071068+0i  0.0000000+0i
```

```
# Normalize #
# Cannot do eigen on this because its not a square matrix
(normalizedBrokenRank <- scale(BrokenRank, center=FALSE,
    scale = as.numeric(apply(as.matrix(BrokenRank), 2, sum))))
```

```
##        [,1]         [,2]
## [1,] 0.0+0i 0.3333333+0i
## [2,] 0.0+0i 0.3333333+0i
## [3,] 0.0+0i 0.3333333+0i
## [4,] 0.5+0i 0.0000000+0i
## [5,] 0.5+0i 0.0000000+0i
## attr(,"scaled:scale")
## [1]  1.414214 -1.732051
```

Notice, that the main issue with Broken Rank is the fact that there is more that one unique eigenvector of `BR` and these give opposite rankings. Thus, BrokenRank does not tell us how we should order our pages.

4. Create a dampening parameter `d` and set this to 0.85. Next, count the number of webpages present in the link matrix and store this in the variable `n`. Next, initialize a matrix `E` with dimensions equal to the link matrix that contains only 1s. Now combine these elements to compute the Page Rank vector for the link matrix. How are this different from the Broken Rank. Hint: Before performing the calculation remember to transpose your link matrix since the you need the links to go from $j \to i$.

```
#Page Rank #

# Dampening Parameter #
d <- 0.85

# The Number of Pages in the Dataset #
n <- dim(myData)[1]

# A matrix of 1s #
E <- matrix(rep(1, n*n), nrow= n, ncol = n)

# Page Rank #
PR <- ((1-d)/n)*E + d * t(myData) %*% MInv
# Calculate the results #
PRresults <- eigen(PR)
# Isolate the results #
```

```r
PageRank <- PRresults$vectors[,which(signif(PRresults$values, 6) == 1)]
# Normalize #
normalizedPageRank <- scale(PageRank, center=FALSE,
    scale = as.numeric(apply(as.matrix(PageRank), 2, sum)))
```

5. Let us now we analyze the PageRank algorithm using a collaboration network data set. This data set is a weighted co-authorship network of authors who posted preprints in the Astrophysics e-print archive between the dates January 1st, 1995 and December 31st, 1999. The data set has been pruned and consists of 101 vertices representing 101 unique authors with 125 edges denoting the collaborations among these authors.

You can read in the data using the following command:

```r
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```r
astrogr <- read.graph("astro-ph.gml","gml")
```

The data set as obtained is very huge and for our analysis we will prune the data set to consist only of vertices whose id values are less than 100 using the the following command:

```r
#delete.vertices(astrogr, #V(astrogr)$id[V(astrogr)$id])

astrogr_sub <- delete.vertices(astrogr, V(astrogr)$id[V(astrogr)$id > 100])
length(astrogr_sub)
```

```
## [1] 10
```

```r
E(astrogr_sub)$value
```

```
##   [1] 1.0000000 1.0000000 1.5000000 1.0000000 1.5000000 0.5000000 0.5000000
##   [8] 0.5000000 1.0000000 0.5000000 0.5000000 0.3333330 0.3333330 0.3333330
##  [15] 0.3333330 2.0833300 0.3333330 0.5866010 1.0032700 1.6084300 0.6421570
##  [22] 2.4306500 3.4373900 0.2500000 0.2500000 0.2500000 0.2500000 2.2500000
##  [29] 0.5000000 0.5000000 2.0000000 0.5000000 1.5000000 2.0000000 0.5400000
##  [36] 0.5000000 1.0000000 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [43] 0.5000000 0.2500000 1.8333300 0.5000000 0.5000000 0.0416667 2.4989200
##  [50] 3.3322500 2.6655800 0.2361110 0.5000000 0.5000000 0.5000000 1.0000000
##  [57] 4.7857100 4.7857100 5.3690500 1.4095200 1.9928600 1.9928600 1.4095200
##  [64] 1.6595200 1.6595200 2.4261900 1.4095200 1.4095200 1.4095200 3.9761900
##  [71] 1.4095200 1.0000000 1.5000000 0.5000000 1.5000000 0.7000000 0.8333330
##  [78] 0.8333330 2.6166700 0.1250000 0.8333330 0.8333330 0.8333330 2.5000000
##  [85] 1.5000000 0.5000000 1.0000000 3.1666700 0.5000000 0.5000000 0.5000000
##  [92] 0.5000000 0.5000000 0.5000000 0.0769231 0.0384615 2.5000000 0.5000000
##  [99] 1.0000000 0.5000000 0.5000000 1.4500000 0.5000000 2.5000000 0.5000000
## [106] 0.3333330 0.5000000 2.8333300 2.0000000 0.3333330 0.3333330 0.3333330
## [113] 0.3333330 0.8333330 0.3333330 0.5000000 1.2500000 1.0000000 1.0000000
## [120] 0.5000000 0.5000000 1.0000000 0.6666670
```

Write a function myPageRank that takes as its inputs a link matrix and a dampening parameter and outputs the Page Rank vector. Now test this on the the link matrix myNewData and the dampening parameter d. Hint: Before performing the calculation remember to transpose your link matrix since the you need the links to go from $j \to i$.

```r
# Function To Calculate Page Rank #
# Input: Link Matrix, Dampening Parameter #
# Output: Page Rank #
# Summary: The function takes as its inputs a dampening parameter #
# and a linking matrix to compute the page rank vector #

myPageRank <- function(linkMatrix, dampeningParameter){
    #Count the number of web pages #
    n <- dim(linkMatrix)[1]
    # Count the number of pages that each page links to #
    M <- diag(apply(linkMatrix, 1, sum))
    # Invert the matrix #
    MInv <- solve(M)
    # Set up a matrix of 1s to make the Page Rank Calculation Possible #
    E <- matrix(rep(1, n*n), nrow=n, ncol=n)
    # Create the Matrix A #
    myCalc <- ((1-dampeningParameter)/n)*E +
        dampeningParameter * t(linkMatrix) %*% MInv
    # Calculate the eigen vectors of the matrix #
    myEigenResults <- eigen(myCalc)
    # Find Correct Eigen Vector #
    myPageRankIndex <- which(signif(myEigenResults$values, 6) == 1)
    PageRank <- myEigenResults$vectors[,myPageRankIndex]
    # Normalize the Page Rank #
    normalizedPageRank <- scale(PageRank, center = FALSE,
        scale = as.numeric(apply(as.matrix(PageRank), 2, sum)))
    return(normalizedPageRank)
}
```

Let's test out page rank on a simple adjacency matrix.

```r
myNewData <- matrix(c(0,1,1,0,0,0,1,0,1,1,1,1,0,1,0,0,0,0,0,0,1,0,1,0,0,1,1,0,0,0,0,0,1,0,0,0),nrow=6,n
myPageRank(myNewData, d)
```

```
##                [,1]
## [1,] 0.08426215+0i
## [2,] 0.34860090+0i
## [3,] 0.27853908+0i
## [4,] 0.08426215+0i
## [5,] 0.12007357+0i
## [6,] 0.08426215+0i
## attr(,"scaled:scale")
## [1] -2.063613
```