Markov Chains Demonstration Using PageRank

Giancarlo Dibonaventi, Hope Foster-Reyes, Haroon Janjua, Mallika Thanky, Ethan Ton

May 28, 2016

This example demonstrates Markov Chains using the PageRank algorithm.

Functions

```r
load.graph <- function(graph.file) {
  # Loads graph used by other functions in demo. Assumes file is in working directory.
  #
  # Arguments:
  #   graph.file: Name of xlsx file containing matrix of a graph of interlinked web pages.

  file.data <- read.csv(graph.file)
  data.matrix(file.data) # Convert data to matrix, transpose, and return
}


check.markov <- function(graph, fix.dangling) {
  # Checks that the sum of every column = 1.
  #   If it adds to 1, return the graph unchanged.
  #   If it adds to 0 (dangling node), return an adjusted graph, create each value
  #     as 1/nx where nx is total number of nodes.
  #   If it does not add to 1 and is not a dangling node, return FALSE. This represents
  #     an error in the matrix itself.
  #
  # Arguments:
  #   graph: Matrix of a graph of interlinked web pages

  nx <- nrow(graph)

  # Loop through columns and normalize.
  adjusted <- FALSE
  for (i in 1:nx) {
    colsum <- sum(graph[,i])
    if (!isTRUE(all.equal(colsum,1, tolerance=0.0001))) {
      if (colsum == 0) {
        if (fix.dangling) {
          graph[,i] <- 1/nx
          adjusted <- TRUE
        }
      } else {
        return(FALSE)
      }
    }
  }
```

```r
  if (adjusted) {
    message("Graph adjusted to correct dangling nodes:")
    print(graph)
  }

  return(graph)
}


markov.demo <- function(graph, initial, damping.factor=0.85,
                        print.skip=3, fix.dangling=TRUE) {
  # Demonstrates iterations of Markov Chain using PageRank algorithm
  #
  # Arguments:
  #   graph: Matrix of a graph of interlinked web pages, forming the transition matrix
  #     representing the probability of state change from j to i, i.e. the probability
  #     of a hypothetical web surfer following a link from the jth page to the ith page.
  #   initial: Initial probability vector.
  #   damping.factor: Damping constant accounts for isolated pages and ensures a regular
  #     matrix is used. As written, this factor is the probability that a random surfer
  #     will *not* make a jump to a random page but will follow links.
  #     Set damping.factor to 1 to simulate basic Markov Chain without damping.
  #   print.skip: Skip count when printing graphs to demonstrate iterations.

  nx <- nrow(graph) # number of nodes/pages
  probability <- initial

  message("Graph input, representing original transition matrix:")
  print(graph)

  # Check if truly Markov, if not, change problem columns to sum to 1
  graph <- check.markov(graph, fix.dangling)
  if (graph[1] == FALSE & length(graph) == 1) {
    stop("ERROR: Data is not properly formatted.")
  }

  # Minimum difference between iteration probability values
  delta_threshold <- 1e-7

  # Iterate until PageRank probability vector is stable to threshold delta,
  # or max 1000 iterations
  for (i in 1:1000) {
    previous <- probability

    # PageRank formula
    probability <- (1 - damping.factor) / nx + damping.factor * (graph %*% probability)

    # Print alternate iterations.
    if (i %in% 1:3 | (i %% print.skip == 0)) {
      message("Iteration ", i, " PageRank (probability) vector: ")
      print(probability)
    }
```

```r
    # Check difference between probability and previous probability iteration.
    check_vector <- abs(previous - probability)

    # If all values in check_vector are less than delta_threshold, print result and end.
    if (all(check_vector < delta_threshold)) {
      message("Probabilities converge to steady state vector at iteration number ", i-1, ": ")
      return(probability)
      break
    }
  }

  message("Did not reach steady state within 1000 iterations.")
  print(probability)
}


eigen.demo <- function(graph, damping.factor=0.85, fix.dangling=TRUE) {

  nx <- nrow(graph)

  # Check if truly Markov, if not, change problem columns to sum to 1
  graph <- check.markov(graph, fix.dangling)
  if (graph[1] == FALSE & length(graph) == 1) {
    stop("ERROR: Data is not properly formatted.")
  }

  # Create Perturbation Matrix (B)
  B <- matrix(1/nx,nrow=nx,ncol=nx)

  # Create PageRank Matrix based off Transition Matrix (graph) and Perturbation Matrix (B)
  M <- (damping.factor * graph) + ((1 - damping.factor) * B)

  eigen_output <- eigen(M)

  # Create eigenvector with an eigenvalue of 1 and change type to double (by default,
  # it is complex type)
  where.eigenvalue.1 <- abs(as.double(eigen_output$values) - 1) < 1e-6
  if (!any(where.eigenvalue.1)) {
    stop("There is no eigenvalue of 1.")
  }
  eigen_vector <- as.double(eigen_output$vectors[, where.eigenvalue.1])

  # Normalize vector such that entire column sum = 1
  steady_state_vector <- eigen_vector / sum(eigen_vector)

  # Run check to see if Steady State Vector actually sums to 1
  check <- sum(steady_state_vector)

  if (isTRUE(all.equal(check,1))) {
    message("Steady state vector is:")
    return(steady_state_vector)
  } else {
    warning("Normalized eigenvector DOES NOT sum to 1.")
```

```
    print(steady_state_vector)
  }
}
```

Simple Example

Let's start with a simple example. This example confirms that if every page points equally to every other page, the PageRank will be evenly distributed.

```
graph <- load.graph("graph-massive-ball.csv")
nx <- nrow(graph)
```

We'll start with an initial probabiity vector with an even probability of starting on any node:

```
(initial <- rep(1 / nx, nx))
```

```
##  [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

```
markov.demo(graph, initial, print.skip = 5)
```

```
## Graph input, representing original transition matrix:
```

```
##         A   B   C   D   E   F   G   H   I   J
##  [1,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [2,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [3,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [4,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [5,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [6,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [7,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [8,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [9,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
## [10,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

```
## Iteration 1 PageRank (probability) vector:
```

```
##       [,1]
##  [1,]  0.1
##  [2,]  0.1
##  [3,]  0.1
##  [4,]  0.1
##  [5,]  0.1
##  [6,]  0.1
##  [7,]  0.1
##  [8,]  0.1
##  [9,]  0.1
## [10,]  0.1
```

```
## Probabilities converge to steady state vector at iteration number 0:
```

```
##        [,1]
##  [1,]  0.1
##  [2,]  0.1
##  [3,]  0.1
##  [4,]  0.1
##  [5,]  0.1
##  [6,]  0.1
##  [7,]  0.1
##  [8,]  0.1
##  [9,]  0.1
## [10,]  0.1
```

```
eigen.demo(graph)
```

```
## Steady state vector is:

##  [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

We can see above the transition graph already matches the initial matrix, so no iterations are necessary; the initial vector is the same as the steady state vector.

What if we start on a specific node instead? The steady state is reached in one interation.

```
(initial <- c(1, rep(0, nx-1)))
```

```
##  [1] 1 0 0 0 0 0 0 0 0 0
```

```
markov.demo(graph, initial, print.skip = 5)
```

```
## Graph input, representing original transition matrix:

##          A   B   C   D   E   F   G   H   I   J
##  [1,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [2,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [3,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [4,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [5,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [6,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [7,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [8,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
##  [9,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
## [10,] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1

## Iteration 1 PageRank (probability) vector:

##        [,1]
##  [1,]  0.1
##  [2,]  0.1
##  [3,]  0.1
##  [4,]  0.1
##  [5,]  0.1
```

```
##  [6,]  0.1
##  [7,]  0.1
##  [8,]  0.1
##  [9,]  0.1
## [10,]  0.1
```

```
## Iteration 2 PageRank (probability) vector:
```

```
##       [,1]
##  [1,]  0.1
##  [2,]  0.1
##  [3,]  0.1
##  [4,]  0.1
##  [5,]  0.1
##  [6,]  0.1
##  [7,]  0.1
##  [8,]  0.1
##  [9,]  0.1
## [10,]  0.1
```

```
## Probabilities converge to steady state vector at iteration number 1:
```

```
##       [,1]
##  [1,]  0.1
##  [2,]  0.1
##  [3,]  0.1
##  [4,]  0.1
##  [5,]  0.1
##  [6,]  0.1
##  [7,]  0.1
##  [8,]  0.1
##  [9,]  0.1
## [10,]  0.1
```

```
eigen.demo(graph)
```

```
## Steady state vector is:
```

```
##  [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

One of the requirements of using a Markov chain model is a matrix whose columns sum to 1. Here is an example of a matrix which does not, producing an error message.

```
(fail <- matrix(c(.5, .5, .75, .5), nrow = 2, ncol = 2))
```

```
##      [,1] [,2]
## [1,]  0.5 0.75
## [2,]  0.5 0.50
```

```r
markov.demo(fail, rep(1/2, 2))
```

```
## Graph input, representing original transition matrix:
```

```
##      [,1] [,2]
## [1,]  0.5 0.75
## [2,]  0.5 0.50
```

```
## Error in markov.demo(fail, rep(1/2, 2)): ERROR: Data is not properly formatted.
```

Single Hub Example

This example represents a "single hub". Note that all pages point to one hub.

Our first output in these examples is the graph itself, followed by the Markov iterations which calculate PageRank and gradually reach a steady state.

```r
graph <- load.graph("graph-single-hub.csv")
nx <- nrow(graph)
initial <- rep(1 / nx, nx)
markov.demo(graph, initial, print.skip = 5)
```

```
## Graph input, representing original transition matrix:
```

```
##       A B C D E F G H I J
##  [1,] 1 1 1 1 1 1 1 1 1 1
##  [2,] 0 0 0 0 0 0 0 0 0 0
##  [3,] 0 0 0 0 0 0 0 0 0 0
##  [4,] 0 0 0 0 0 0 0 0 0 0
##  [5,] 0 0 0 0 0 0 0 0 0 0
##  [6,] 0 0 0 0 0 0 0 0 0 0
##  [7,] 0 0 0 0 0 0 0 0 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
```

```
## Iteration 1 PageRank (probability) vector:
```

```
##        [,1]
##  [1,] 0.865
##  [2,] 0.015
##  [3,] 0.015
##  [4,] 0.015
##  [5,] 0.015
##  [6,] 0.015
##  [7,] 0.015
##  [8,] 0.015
##  [9,] 0.015
## [10,] 0.015
```

```
## Iteration 2 PageRank (probability) vector:

##        [,1]
##  [1,] 0.865
##  [2,] 0.015
##  [3,] 0.015
##  [4,] 0.015
##  [5,] 0.015
##  [6,] 0.015
##  [7,] 0.015
##  [8,] 0.015
##  [9,] 0.015
## [10,] 0.015

## Probabilities converge to steady state vector at iteration number 1:

##        [,1]
##  [1,] 0.865
##  [2,] 0.015
##  [3,] 0.015
##  [4,] 0.015
##  [5,] 0.015
##  [6,] 0.015
##  [7,] 0.015
##  [8,] 0.015
##  [9,] 0.015
## [10,] 0.015
```

```
eigen.demo(graph)
```

```
## Steady state vector is:

##  [1] 0.865 0.015 0.015 0.015 0.015 0.015 0.015 0.015 0.015 0.015
```

Above, you see the impact of the damping factor, as the single hub retains approximately 85% of its importance, while the remainder is shared by the rest of the sites.

Transfer of Influence

Next, we vary the single hub example so that the hub points only to another page instead of itself. As shown, the hub has the largest PageRank, but the page it points to has nearly as much as the hub "passes" its PageRank to the other page.

```
graph <- load.graph("graph-hub-transfer.csv")
markov.demo(graph, initial, print.skip = 30)
```

```
## Graph input, representing original transition matrix:
```

```
##       A B C D E F G H I J
##  [1,] 0 1 1 1 1 1 1 1 1 1
##  [2,] 1 0 0 0 0 0 0 0 0 0
##  [3,] 0 0 0 0 0 0 0 0 0 0
##  [4,] 0 0 0 0 0 0 0 0 0 0
##  [5,] 0 0 0 0 0 0 0 0 0 0
##  [6,] 0 0 0 0 0 0 0 0 0 0
##  [7,] 0 0 0 0 0 0 0 0 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
```

```
## Iteration 1 PageRank (probability) vector:
```

```
##        [,1]
##  [1,] 0.780
##  [2,] 0.100
##  [3,] 0.015
##  [4,] 0.015
##  [5,] 0.015
##  [6,] 0.015
##  [7,] 0.015
##  [8,] 0.015
##  [9,] 0.015
## [10,] 0.015
```

```
## Iteration 2 PageRank (probability) vector:
```

```
##        [,1]
##  [1,] 0.202
##  [2,] 0.678
##  [3,] 0.015
##  [4,] 0.015
##  [5,] 0.015
##  [6,] 0.015
##  [7,] 0.015
##  [8,] 0.015
##  [9,] 0.015
## [10,] 0.015
```

```
## Iteration 3 PageRank (probability) vector:
```

```
##         [,1]
##  [1,] 0.6933
##  [2,] 0.1867
##  [3,] 0.0150
##  [4,] 0.0150
##  [5,] 0.0150
##  [6,] 0.0150
##  [7,] 0.0150
##  [8,] 0.0150
##  [9,] 0.0150
## [10,] 0.0150
```

```
## Iteration 30 PageRank (probability) vector:

##            [,1]
##  [1,] 0.4647627
##  [2,] 0.4152373
##  [3,] 0.0150000
##  [4,] 0.0150000
##  [5,] 0.0150000
##  [6,] 0.0150000
##  [7,] 0.0150000
##  [8,] 0.0150000
##  [9,] 0.0150000
## [10,] 0.0150000

## Iteration 60 PageRank (probability) vector:

##            [,1]
##  [1,] 0.4675462
##  [2,] 0.4124538
##  [3,] 0.0150000
##  [4,] 0.0150000
##  [5,] 0.0150000
##  [6,] 0.0150000
##  [7,] 0.0150000
##  [8,] 0.0150000
##  [9,] 0.0150000
## [10,] 0.0150000

## Iteration 90 PageRank (probability) vector:

##            [,1]
##  [1,] 0.4675674
##  [2,] 0.4124326
##  [3,] 0.0150000
##  [4,] 0.0150000
##  [5,] 0.0150000
##  [6,] 0.0150000
##  [7,] 0.0150000
##  [8,] 0.0150000
##  [9,] 0.0150000
## [10,] 0.0150000

## Probabilities converge to steady state vector at iteration number 97:

##            [,1]
##  [1,] 0.4675675
##  [2,] 0.4124325
##  [3,] 0.0150000
##  [4,] 0.0150000
##  [5,] 0.0150000
##  [6,] 0.0150000
##  [7,] 0.0150000
```

```
##  [8,] 0.0150000
##  [9,] 0.0150000
## [10,] 0.0150000
```

```
eigen.demo(graph)
```

```
## Steady state vector is:
```

```
##  [1] 0.4675676 0.4124324 0.0150000 0.0150000 0.0150000 0.0150000 0.0150000
##  [8] 0.0150000 0.0150000 0.0150000
```

Dual Hub – More Complex Example

This more complex example demonstrates two hubs.

```
graph <- load.graph("graph-dual-hub.csv")
nx <- nrow(graph)
initial <- rep(1 / nx, nx)
markov.demo(graph, initial, print.skip = 20)
```

```
## Graph input, representing original transition matrix:
```

```
##       A B C   D   E   F   G   H   I J K
##  [1,] 0 0 0 0.5 0.0 0.0 0.0 0.0 0.0 0 0
##  [2,] 0 0 1 0.5 0.0 0.5 0.5 0.5 0.5 0 0
##  [3,] 0 1 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
##  [4,] 0 0 0 0.0 0.5 0.0 0.0 0.0 0.0 0 0
##  [5,] 0 0 0 0.0 0.0 0.5 0.5 0.5 0.5 1 1
##  [6,] 0 0 0 0.0 0.5 0.0 0.0 0.0 0.0 0 0
##  [7,] 0 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
##  [8,] 0 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
##  [9,] 0 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
## [10,] 0 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
## [11,] 0 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
```

```
## Graph adjusted to correct dangling nodes:
```

```
##                A B C   D   E   F   G   H   I J K
##  [1,] 0.09090909 0 0 0.5 0.0 0.0 0.0 0.0 0.0 0 0
##  [2,] 0.09090909 0 1 0.5 0.0 0.5 0.5 0.5 0.5 0 0
##  [3,] 0.09090909 1 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
##  [4,] 0.09090909 0 0 0.0 0.5 0.0 0.0 0.0 0.0 0 0
##  [5,] 0.09090909 0 0 0.0 0.0 0.5 0.5 0.5 0.5 1 1
##  [6,] 0.09090909 0 0 0.0 0.5 0.0 0.0 0.0 0.0 0 0
##  [7,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
##  [8,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
##  [9,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
## [10,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
## [11,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
```

```
## Iteration 1 PageRank (probability) vector:

##             [,1]
##  [1,] 0.05929752
##  [2,] 0.29111570
##  [3,] 0.09793388
##  [4,] 0.05929752
##  [5,] 0.32975207
##  [6,] 0.05929752
##  [7,] 0.02066116
##  [8,] 0.02066116
##  [9,] 0.02066116
## [10,] 0.02066116
## [11,] 0.02066116

## Iteration 2 PageRank (probability) vector:

##             [,1]
##  [1,] 0.04341989
##  [2,] 0.17820811
##  [3,] 0.26566679
##  [4,] 0.15836307
##  [5,] 0.10488683
##  [6,] 0.15836307
##  [7,] 0.01821844
##  [8,] 0.01821844
##  [9,] 0.01821844
## [10,] 0.01821844
## [11,] 0.01821844

## Iteration 3 PageRank (probability) vector:

##             [,1]
##  [1,] 0.08429584
##  [2,] 0.40064544
##  [3,] 0.16846843
##  [4,] 0.06156844
##  [5,] 0.13849572
##  [6,] 0.06156844
##  [7,] 0.01699154
##  [8,] 0.01699154
##  [9,] 0.01699154
## [10,] 0.01699154
## [11,] 0.01699154

## Iteration 20 PageRank (probability) vector:

##             [,1]
##  [1,] 0.04007887
##  [2,] 0.34902122
##  [3,] 0.32751232
##  [4,] 0.05492841
```

```
##  [5,] 0.08986215
##  [6,] 0.05492841
##  [7,] 0.01673372
##  [8,] 0.01673372
##  [9,] 0.01673372
## [10,] 0.01673372
## [11,] 0.01673372
```

```
## Iteration 40 PageRank (probability) vector:
```

```
##             [,1]
##  [1,] 0.04007476
##  [2,] 0.35636797
##  [3,] 0.32019309
##  [4,] 0.05492167
##  [5,] 0.08985558
##  [6,] 0.05492167
##  [7,] 0.01673305
##  [8,] 0.01673305
##  [9,] 0.01673305
## [10,] 0.01673305
## [11,] 0.01673305
```

```
## Iteration 60 PageRank (probability) vector:
```

```
##             [,1]
##  [1,] 0.04007476
##  [2,] 0.35665232
##  [3,] 0.31990874
##  [4,] 0.05492167
##  [5,] 0.08985558
##  [6,] 0.05492167
##  [7,] 0.01673305
##  [8,] 0.01673305
##  [9,] 0.01673305
## [10,] 0.01673305
## [11,] 0.01673305
```

```
## Iteration 80 PageRank (probability) vector:
```

```
##             [,1]
##  [1,] 0.04007476
##  [2,] 0.35666335
##  [3,] 0.31989772
##  [4,] 0.05492167
##  [5,] 0.08985558
##  [6,] 0.05492167
##  [7,] 0.01673305
##  [8,] 0.01673305
##  [9,] 0.01673305
## [10,] 0.01673305
## [11,] 0.01673305
```

```
## Probabilities converge to steady state vector at iteration number 93:

##             [,1]
##  [1,] 0.04007476
##  [2,] 0.35666375
##  [3,] 0.31989732
##  [4,] 0.05492167
##  [5,] 0.08985558
##  [6,] 0.05492167
##  [7,] 0.01673305
##  [8,] 0.01673305
##  [9,] 0.01673305
## [10,] 0.01673305
## [11,] 0.01673305
```

```r
eigen.demo(graph)
```

```
## Graph adjusted to correct dangling nodes:

##                A B C   D   E   F   G   H   I J K
##  [1,] 0.09090909 0 0 0.5 0.0 0.0 0.0 0.0 0.0 0 0
##  [2,] 0.09090909 0 1 0.5 0.0 0.5 0.5 0.5 0.5 0 0
##  [3,] 0.09090909 1 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
##  [4,] 0.09090909 0 0 0.0 0.5 0.0 0.0 0.0 0.0 0 0
##  [5,] 0.09090909 0 0 0.0 0.0 0.5 0.5 0.5 0.5 1 1
##  [6,] 0.09090909 0 0 0.0 0.5 0.0 0.0 0.0 0.0 0 0
##  [7,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
##  [8,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
##  [9,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
## [10,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0
## [11,] 0.09090909 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0

## Warning in eigen.demo(graph): imaginary parts discarded in coercion

## Steady state vector is:

##  [1] 0.04007476 0.35666379 0.31989727 0.05492167 0.08985558 0.05492167
##  [7] 0.01673305 0.01673305 0.01673305 0.01673305 0.01673305
```

Demonstration of Case for Dangling Node Correction

What if we were to attempt to locate our steady state vector without correcting for the dangling node, Page A which has no outgoing links? This is not a true stochastic matrix, and does not have and eigenvalue of 1 or its corresponding eigenvector.

```r
eigen.demo(graph, fix.dangling = FALSE)
```

```
## Warning in eigen.demo(graph, fix.dangling = FALSE): imaginary parts
## discarded in coercion

## Error in eigen.demo(graph, fix.dangling = FALSE): There is no eigenvalue of 1.
```

Disconnected Pages Example

Our final example contains a selection of pages that are completely disconnected from the others.

```r
graph <- load.graph("graph-disconnected.csv")
nx <- nrow(graph)
initial <- rep(1 / nx, nx)
markov.demo(graph, initial, print.skip = 20)
```

```
## Graph input, representing original transition matrix:
```

```
##       A B C D E F G H I J K L M N O P Q R S T
##  [1,] 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
##  [2,] 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
##  [3,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
##  [4,] 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0
##  [5,] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [6,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [7,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [8,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [9,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [10,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [11,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [12,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [13,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [14,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [15,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [16,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [17,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [18,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [19,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [20,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
## Iteration 1 PageRank (probability) vector:
```

```
##         [,1]
##  [1,] 0.2625
##  [2,] 0.2200
##  [3,] 0.1775
##  [4,] 0.1775
##  [5,] 0.0500
##  [6,] 0.0075
##  [7,] 0.0075
##  [8,] 0.0075
##  [9,] 0.0075
## [10,] 0.0075
## [11,] 0.0075
## [12,] 0.0075
## [13,] 0.0075
## [14,] 0.0075
## [15,] 0.0075
## [16,] 0.0075
```

```
## [17,] 0.0075
## [18,] 0.0075
## [19,] 0.0075
## [20,] 0.0075
```

```
## Iteration 2 PageRank (probability) vector:
```

```
##           [,1]
##  [1,] 0.370875
##  [2,] 0.256125
##  [3,] 0.033000
##  [4,] 0.069125
##  [5,] 0.158375
##  [6,] 0.007500
##  [7,] 0.007500
##  [8,] 0.007500
##  [9,] 0.007500
## [10,] 0.007500
## [11,] 0.007500
## [12,] 0.007500
## [13,] 0.007500
## [14,] 0.007500
## [15,] 0.007500
## [16,] 0.007500
## [17,] 0.007500
## [18,] 0.007500
## [19,] 0.007500
## [20,] 0.007500
```

```
## Iteration 3 PageRank (probability) vector:
```

```
##             [,1]
##  [1,] 0.27875625
##  [2,] 0.34824375
##  [3,] 0.03300000
##  [4,] 0.16124375
##  [5,] 0.06625625
##  [6,] 0.00750000
##  [7,] 0.00750000
##  [8,] 0.00750000
##  [9,] 0.00750000
## [10,] 0.00750000
## [11,] 0.00750000
## [12,] 0.00750000
## [13,] 0.00750000
## [14,] 0.00750000
## [15,] 0.00750000
## [16,] 0.00750000
## [17,] 0.00750000
## [18,] 0.00750000
## [19,] 0.00750000
## [20,] 0.00750000
```

```
## Iteration 20 PageRank (probability) vector:
```

```
##            [,1]
##  [1,] 0.3237523
##  [2,] 0.3032477
##  [3,] 0.0330000
##  [4,] 0.1162477
##  [5,] 0.1112523
##  [6,] 0.0075000
##  [7,] 0.0075000
##  [8,] 0.0075000
##  [9,] 0.0075000
## [10,] 0.0075000
## [11,] 0.0075000
## [12,] 0.0075000
## [13,] 0.0075000
## [14,] 0.0075000
## [15,] 0.0075000
## [16,] 0.0075000
## [17,] 0.0075000
## [18,] 0.0075000
## [19,] 0.0075000
## [20,] 0.0075000
```

```
## Iteration 40 PageRank (probability) vector:
```

```
##            [,1]
##  [1,] 0.3211846
##  [2,] 0.3058154
##  [3,] 0.0330000
##  [4,] 0.1188154
##  [5,] 0.1086846
##  [6,] 0.0075000
##  [7,] 0.0075000
##  [8,] 0.0075000
##  [9,] 0.0075000
## [10,] 0.0075000
## [11,] 0.0075000
## [12,] 0.0075000
## [13,] 0.0075000
## [14,] 0.0075000
## [15,] 0.0075000
## [16,] 0.0075000
## [17,] 0.0075000
## [18,] 0.0075000
## [19,] 0.0075000
## [20,] 0.0075000
```

```
## Iteration 60 PageRank (probability) vector:
```

```
##            [,1]
##  [1,] 0.3210851
##  [2,] 0.3059149
##  [3,] 0.0330000
##  [4,] 0.1189149
```

```
##  [5,] 0.1085851
##  [6,] 0.0075000
##  [7,] 0.0075000
##  [8,] 0.0075000
##  [9,] 0.0075000
## [10,] 0.0075000
## [11,] 0.0075000
## [12,] 0.0075000
## [13,] 0.0075000
## [14,] 0.0075000
## [15,] 0.0075000
## [16,] 0.0075000
## [17,] 0.0075000
## [18,] 0.0075000
## [19,] 0.0075000
## [20,] 0.0075000
```

```
## Iteration 80 PageRank (probability) vector:
```

```
##            [,1]
##  [1,] 0.3210812
##  [2,] 0.3059188
##  [3,] 0.0330000
##  [4,] 0.1189188
##  [5,] 0.1085812
##  [6,] 0.0075000
##  [7,] 0.0075000
##  [8,] 0.0075000
##  [9,] 0.0075000
## [10,] 0.0075000
## [11,] 0.0075000
## [12,] 0.0075000
## [13,] 0.0075000
## [14,] 0.0075000
## [15,] 0.0075000
## [16,] 0.0075000
## [17,] 0.0075000
## [18,] 0.0075000
## [19,] 0.0075000
## [20,] 0.0075000
```

```
## Probabilities converge to steady state vector at iteration number 87:
```

```
##            [,1]
##  [1,] 0.3210811
##  [2,] 0.3059189
##  [3,] 0.0330000
##  [4,] 0.1189189
##  [5,] 0.1085811
##  [6,] 0.0075000
##  [7,] 0.0075000
##  [8,] 0.0075000
##  [9,] 0.0075000
```

```
## [10,] 0.0075000
## [11,] 0.0075000
## [12,] 0.0075000
## [13,] 0.0075000
## [14,] 0.0075000
## [15,] 0.0075000
## [16,] 0.0075000
## [17,] 0.0075000
## [18,] 0.0075000
## [19,] 0.0075000
## [20,] 0.0075000
```

```r
eigen.demo(graph)
```

```
## Warning in eigen.demo(graph): imaginary parts discarded in coercion
```

```
## Steady state vector is:
```

```
##  [1] 0.3210811 0.3059189 0.0330000 0.1189189 0.1085811 0.0075000 0.0075000
##  [8] 0.0075000 0.0075000 0.0075000 0.0075000 0.0075000 0.0075000 0.0075000
## [15] 0.0075000 0.0075000 0.0075000 0.0075000 0.0075000 0.0075000
```

Demonstration of Case for Damping Factor Correction

Google's damping factor corrects for disconnected pages. Let's run our Markov chain and eigenvector calculation *without* using it.

```r
markov.demo(graph, initial, damping.factor = 1, print.skip = 249)
```

```
## Graph input, representing original transition matrix:
```

```
##       A B C D E F G H I J K L M N O P Q R S T
##  [1,] 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
##  [2,] 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
##  [3,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
##  [4,] 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0
##  [5,] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [6,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [7,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [8,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [9,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [10,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [11,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [12,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [13,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [14,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [15,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [16,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [17,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [18,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [19,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [20,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
## Iteration 1 PageRank (probability) vector:
```

```
##       [,1]
##  [1,] 0.30
##  [2,] 0.25
##  [3,] 0.20
##  [4,] 0.20
##  [5,] 0.05
##  [6,] 0.00
##  [7,] 0.00
##  [8,] 0.00
##  [9,] 0.00
## [10,] 0.00
## [11,] 0.00
## [12,] 0.00
## [13,] 0.00
## [14,] 0.00
## [15,] 0.00
## [16,] 0.00
## [17,] 0.00
## [18,] 0.00
## [19,] 0.00
## [20,] 0.00
```

```
## Iteration 2 PageRank (probability) vector:
```

```
##       [,1]
##  [1,] 0.45
##  [2,] 0.30
##  [3,] 0.00
##  [4,] 0.05
##  [5,] 0.20
##  [6,] 0.00
##  [7,] 0.00
##  [8,] 0.00
##  [9,] 0.00
## [10,] 0.00
## [11,] 0.00
## [12,] 0.00
## [13,] 0.00
## [14,] 0.00
## [15,] 0.00
## [16,] 0.00
## [17,] 0.00
## [18,] 0.00
## [19,] 0.00
## [20,] 0.00
```

```
## Iteration 3 PageRank (probability) vector:
```

```
##       [,1]
##  [1,] 0.30
##  [2,] 0.45
```

```
##  [3,] 0.00
##  [4,] 0.20
##  [5,] 0.05
##  [6,] 0.00
##  [7,] 0.00
##  [8,] 0.00
##  [9,] 0.00
## [10,] 0.00
## [11,] 0.00
## [12,] 0.00
## [13,] 0.00
## [14,] 0.00
## [15,] 0.00
## [16,] 0.00
## [17,] 0.00
## [18,] 0.00
## [19,] 0.00
## [20,] 0.00

## Iteration 249 PageRank (probability) vector:

##       [,1]
##  [1,] 0.30
##  [2,] 0.45
##  [3,] 0.00
##  [4,] 0.20
##  [5,] 0.05
##  [6,] 0.00
##  [7,] 0.00
##  [8,] 0.00
##  [9,] 0.00
## [10,] 0.00
## [11,] 0.00
## [12,] 0.00
## [13,] 0.00
## [14,] 0.00
## [15,] 0.00
## [16,] 0.00
## [17,] 0.00
## [18,] 0.00
## [19,] 0.00
## [20,] 0.00

## Iteration 498 PageRank (probability) vector:

##       [,1]
##  [1,] 0.45
##  [2,] 0.30
##  [3,] 0.00
##  [4,] 0.05
##  [5,] 0.20
##  [6,] 0.00
##  [7,] 0.00
```

```
##  [8,] 0.00
##  [9,] 0.00
## [10,] 0.00
## [11,] 0.00
## [12,] 0.00
## [13,] 0.00
## [14,] 0.00
## [15,] 0.00
## [16,] 0.00
## [17,] 0.00
## [18,] 0.00
## [19,] 0.00
## [20,] 0.00
```

```
## Iteration 747 PageRank (probability) vector:
```

```
##       [,1]
##  [1,] 0.30
##  [2,] 0.45
##  [3,] 0.00
##  [4,] 0.20
##  [5,] 0.05
##  [6,] 0.00
##  [7,] 0.00
##  [8,] 0.00
##  [9,] 0.00
## [10,] 0.00
## [11,] 0.00
## [12,] 0.00
## [13,] 0.00
## [14,] 0.00
## [15,] 0.00
## [16,] 0.00
## [17,] 0.00
## [18,] 0.00
## [19,] 0.00
## [20,] 0.00
```

```
## Iteration 996 PageRank (probability) vector:
```

```
##       [,1]
##  [1,] 0.45
##  [2,] 0.30
##  [3,] 0.00
##  [4,] 0.05
##  [5,] 0.20
##  [6,] 0.00
##  [7,] 0.00
##  [8,] 0.00
##  [9,] 0.00
## [10,] 0.00
## [11,] 0.00
## [12,] 0.00
```

```
## [13,] 0.00
## [14,] 0.00
## [15,] 0.00
## [16,] 0.00
## [17,] 0.00
## [18,] 0.00
## [19,] 0.00
## [20,] 0.00
```

```
## Did not reach steady state within 1000 iterations.
```

```
##       [,1]
##  [1,] 0.45
##  [2,] 0.30
##  [3,] 0.00
##  [4,] 0.05
##  [5,] 0.20
##  [6,] 0.00
##  [7,] 0.00
##  [8,] 0.00
##  [9,] 0.00
## [10,] 0.00
## [11,] 0.00
## [12,] 0.00
## [13,] 0.00
## [14,] 0.00
## [15,] 0.00
## [16,] 0.00
## [17,] 0.00
## [18,] 0.00
## [19,] 0.00
## [20,] 0.00
```

```
eigen.demo(graph, damping.factor = 1)
```

```
## Steady state vector is:
```

```
##  [1] 0.25 0.25 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
## [15] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.25 0.25 0.00 0.00 0.00
## [29] 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

As is seen above, while we can solve for our eigenvector with an eigenvalue of 1, our Markov chain does not appear to converge on that vector and instead seems to vacillate between probability vectors – demonstrating that our transition matrix, when not correcting with our damping factor, is stocastic, but not regular.

One last demonstration shows that this will further vary based on our initial probability vector. If instead of beginning with an even probability of starting at any node, we begin at a specific node, such a B, our Markov chain becomes trapped in recursive loops without a damping factor.

```
(initial <- c(0, 1, rep(0, nx-2)))
```

```
##  [1] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```r
markov.demo(graph, initial, damping.factor = 1, print.skip = 500)
```

```
## Graph input, representing original transition matrix:

##       A B C D E F G H I J K L M N O P Q R S T
##  [1,] 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
##  [2,] 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0
##  [3,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
##  [4,] 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0
##  [5,] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [6,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [7,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [8,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [9,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [10,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [11,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [12,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [13,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [14,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [15,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [16,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [17,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [18,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [19,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [20,] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

## Iteration 1 PageRank (probability) vector:

##       [,1]
##  [1,]    1
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0
## [17,]    0
## [18,]    0
## [19,]    0
## [20,]    0

## Iteration 2 PageRank (probability) vector:
```

```
##       [,1]
##  [1,]    0
##  [2,]    1
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0
## [17,]    0
## [18,]    0
## [19,]    0
## [20,]    0
```

```
## Iteration 3 PageRank (probability) vector:
```

```
##       [,1]
##  [1,]    1
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0
## [17,]    0
## [18,]    0
## [19,]    0
## [20,]    0
```

```
## Iteration 500 PageRank (probability) vector:
```

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

```
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0
## [17,]    0
## [18,]    0
## [19,]    0
## [20,]    0
```

```
## Iteration 1000 PageRank (probability) vector:
```

```
##       [,1]
##  [1,]    0
##  [2,]    1
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0
## [17,]    0
## [18,]    0
## [19,]    0
## [20,]    0
```

```
## Did not reach steady state within 1000 iterations.
```

```
##       [,1]
##  [1,]    0
##  [2,]    1
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
```

```
## [10,]    0
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0
## [17,]    0
## [18,]    0
## [19,]    0
## [20,]    0
```