Markov Chains Demonstration Using PageRank

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

May 28, 2016

This example demonstrates Markov Chains using the PageRank algorithm.

```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)
      }
    }
  }

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

```r
    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 simulates random walk accounting for isolated pages.
  #     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)
    }

    # 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)) {
```

```r
      message("Probabilities converge to steady state vector at iteration number ", i, ": ")
      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 Random Walk Matrix (B)
  B <- matrix(1/nx,nrow=nx,ncol=nx)

  # Create PageRank Matrix based off Transition Matrix (graph) and Random Walk 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 typ
  eigen_vector <- as.double(eigen_output$vectors[, abs(as.double(eigen_output$values) - 1) < 1e-6])
  #eigen_vector <- as.double(eigen_output$vectors[, which.min(abs(eigen_output$values - 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)
  }
}
```

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.

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

```
graph <- load.graph("graph-massive-ball.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,] 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 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
```

```
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.
```

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

```
graph <- load.graph("graph-single-hub.csv")
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 2:

##          [,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
```

```r
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
```

In the above example you see the impact of the damping factor in force. What if we ran the Markov Chain without the damping factor? As you can see below, the entire weight of probability in the steady state is placed on the single hub. This is true even though we started with an even probability of starting on each page.

```r
markov.demo(graph, initial, damping.factor=1, 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,]    1
## [2,]    0
## [3,]    0
## [4,]    0
## [5,]    0
## [6,]    0
## [7,]    0
## [8,]    0
## [9,]    0
## [10,]   0


## Iteration 2 PageRank (probability) vector:

##      [,1]
## [1,]    1
## [2,]    0
## [3,]    0
## [4,]    0
## [5,]    0
## [6,]    0
## [7,]    0
## [8,]    0
## [9,]    0
## [10,]   0


## Probabilities converge to steady state vector at iteration number 2:

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

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 98:


##            [,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
```

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 0.5 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 0.0 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 0.5 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 0.0 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,] 0.32975207
##  [3,] 0.09793388
##  [4,] 0.02066116
##  [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,] 0.02699944
##  [2,] 0.30193229
##  [3,] 0.29850770
##  [4,] 0.01821844
##  [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,] 0.02346552
##  [2,] 0.41230680
##  [3,] 0.27236513
##  [4,] 0.01572268
##  [5,] 0.13722686
##  [6,] 0.06029959
##  [7,] 0.01572268
##  [8,] 0.01572268
##  [9,] 0.01572268
## [10,] 0.01572268
## [11,] 0.01572268


## Iteration 20 PageRank (probability) vector:


##              [,1]
##  [1,] 0.02183629
##  [2,] 0.39676595
##  [3,] 0.35687201
##  [4,] 0.01532371
##  [5,] 0.08228753
##  [6,] 0.05029594
```

```
##  [7,] 0.01532371
##  [8,] 0.01532371
##  [9,] 0.01532371
## [10,] 0.01532371
## [11,] 0.01532371


## Iteration 40 PageRank (probability) vector:


##              [,1]
##  [1,] 0.02183629
##  [2,] 0.39899877
##  [3,] 0.35463923
##  [4,] 0.01532371
##  [5,] 0.08228752
##  [6,] 0.05029591
##  [7,] 0.01532371
##  [8,] 0.01532371
##  [9,] 0.01532371
## [10,] 0.01532371
## [11,] 0.01532371


## Iteration 60 PageRank (probability) vector:


##              [,1]
##  [1,] 0.02183629
##  [2,] 0.39908531
##  [3,] 0.35455268
##  [4,] 0.01532371
##  [5,] 0.08228752
##  [6,] 0.05029591
##  [7,] 0.01532371
##  [8,] 0.01532371
##  [9,] 0.01532371
## [10,] 0.01532371
## [11,] 0.01532371


## Iteration 80 PageRank (probability) vector:


##              [,1]
##  [1,] 0.02183629
##  [2,] 0.39908867
##  [3,] 0.35454933
##  [4,] 0.01532371
##  [5,] 0.08228752
##  [6,] 0.05029591
##  [7,] 0.01532371
##  [8,] 0.01532371
##  [9,] 0.01532371
## [10,] 0.01532371
## [11,] 0.01532371


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

```
##             [,1]
##  [1,] 0.02183629
##  [2,] 0.39908884
##  [3,] 0.35454915
##  [4,] 0.01532371
##  [5,] 0.08228752
##  [6,] 0.05029591
##  [7,] 0.01532371
##  [8,] 0.01532371
##  [9,] 0.01532371
## [10,] 0.01532371
## [11,] 0.01532371
```

```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 0.5 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 0.0 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
```

```
## Steady State Vector is:
```

```
##  [1] 0.02183629 0.39908880 0.35454919 0.01532371 0.08228752 0.05029591
##  [7] 0.01532371 0.01532371 0.01532371 0.01532371 0.01532371
```

What if we were to attempt to run our Markov chain on the above example without correcting for the dangling node, Page A which has no outgoing links?

```r
markov.demo(graph, initial, print.skip = 20, fix.dangling = FALSE)
```

```
## 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 0.5 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 0.0 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
```

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

##              [,1]
##  [1,] 0.05227273
##  [2,] 0.32272727
##  [3,] 0.09090909
##  [4,] 0.01363636
##  [5,] 0.32272727
##  [6,] 0.05227273
##  [7,] 0.01363636
##  [8,] 0.01363636
##  [9,] 0.01363636
## [10,] 0.01363636
## [11,] 0.01363636


## Iteration 2 PageRank (probability) vector:

##              [,1]
##  [1,] 0.01943182
##  [2,] 0.27346591
##  [3,] 0.28795455
##  [4,] 0.01363636
##  [5,] 0.07642045
##  [6,] 0.15079545
##  [7,] 0.01363636
##  [8,] 0.01363636
##  [9,] 0.01363636
## [10,] 0.01363636
## [11,] 0.01363636


## Iteration 3 PageRank (probability) vector:

##              [,1]
##  [1,] 0.01943182
##  [2,] 0.37814631
##  [3,] 0.24608239
##  [4,] 0.01363636
##  [5,] 0.11829261
##  [6,] 0.04611506
##  [7,] 0.01363636
##  [8,] 0.01363636
##  [9,] 0.01363636
## [10,] 0.01363636
## [11,] 0.01363636


## Iteration 20 PageRank (probability) vector:

##              [,1]
##  [1,] 0.01943182
##  [2,] 0.35271525
##  [3,] 0.31793690
##  [4,] 0.01363636
```

```
##  [5,] 0.07322655
##  [6,] 0.04475767
##  [7,] 0.01363636
##  [8,] 0.01363636
##  [9,] 0.01363636
## [10,] 0.01363636
## [11,] 0.01363636


## Iteration 40 PageRank (probability) vector:


##              [,1]
##  [1,] 0.01943182
##  [2,] 0.35504956
##  [3,] 0.31560262
##  [4,] 0.01363636
##  [5,] 0.07322654
##  [6,] 0.04475765
##  [7,] 0.01363636
##  [8,] 0.01363636
##  [9,] 0.01363636
## [10,] 0.01363636
## [11,] 0.01363636


## Iteration 60 PageRank (probability) vector:


##              [,1]
##  [1,] 0.01943182
##  [2,] 0.35514003
##  [3,] 0.31551214
##  [4,] 0.01363636
##  [5,] 0.07322654
##  [6,] 0.04475765
##  [7,] 0.01363636
##  [8,] 0.01363636
##  [9,] 0.01363636
## [10,] 0.01363636
## [11,] 0.01363636


## Iteration 80 PageRank (probability) vector:


##              [,1]
##  [1,] 0.01943182
##  [2,] 0.35514354
##  [3,] 0.31550863
##  [4,] 0.01363636
##  [5,] 0.07322654
##  [6,] 0.04475765
##  [7,] 0.01363636
##  [8,] 0.01363636
##  [9,] 0.01363636
## [10,] 0.01363636
## [11,] 0.01363636
```

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

```
##              [,1]
##  [1,] 0.01943182
##  [2,] 0.35514373
##  [3,] 0.31550845
##  [4,] 0.01363636
##  [5,] 0.07322654
##  [6,] 0.04475765
##  [7,] 0.01363636
##  [8,] 0.01363636
##  [9,] 0.01363636
## [10,] 0.01363636
## [11,] 0.01363636
```

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

```
## Warning in eigen.demo(graph, fix.dangling = FALSE): Normalized eigenvector
## DOES NOT sum to 1.
```

```
## numeric(0)
```

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 1 1 1 0 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
```

18

```
##  [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 88:


##              [,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
```

```
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
```

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

```
markov.demo(graph, initial, damping.factor = 1, print.skip = 250)
```

```
## 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 250 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 500 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 750 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 1000 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
```