Problem 1

Let's put $v^{(n)} = \begin{bmatrix} g_1^{(n)} \\ g_2^{(n)} \end{bmatrix}$ where $g_1^{(n)}$ and $g_2^{(n)}$ are respectively the sizes of group 1 and group 2 at day **n**.

For day n+1:

$$g_1^{(n+1)} = 0.7 * g_1^{(n)} + 0.2 * g_2^{(n)}$$

$$g_2^{(n+1)} = 0.3 * g_1^{(n)} + 0.8 * g_2^{(n)}$$

In matrix representation:

$$\begin{bmatrix} g_1^{(n+1)} \\ g_2^{(n+1)} \end{bmatrix} = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \begin{bmatrix} g_1^{(n)} \\ g_2^{(n)} \end{bmatrix}$$

Therefore:

$$\mathbf{M} = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \quad \text{and} \quad v^{(n+1)} = \mathbf{M} v^{(n)}$$

Also we can write:

$$v^{(n)} = \mathbf{M}^n v^{(0)}$$

To find $\mathbf{M}^n$, let's diagonalize **M**.

a. Eigenvalues of **M**:

$$\det\left(\begin{bmatrix} 0.7 - \lambda & 0.2 \\ 0.3 & 0.8 - \lambda \end{bmatrix}\right) = 0$$

$$\Rightarrow (0.7 - \lambda)(0.8 - \lambda) - 0.2 * 0.3 = 0$$

$$\Rightarrow \lambda^2 + 0.5 - 1.5\lambda = 0$$

$$\Rightarrow \lambda = 1 \text{ or } \lambda = \frac{1}{2}$$

$$\Rightarrow \lambda_1 = 1 \text{ and } \lambda_2 = \frac{1}{2} \text{ are eigenvalues of } \mathbf{M}.$$

b. Eigenvectors of **M**:

$\lambda_1 = 1 \Rightarrow x_1 \in N(M - I)$	$\lambda_2 = \frac{1}{2} \Rightarrow x_2 \in N(M - \frac{1}{2} I)$
$M - I = \begin{bmatrix} -0.3 & 0.2 \\ 0.3 & 0.2 \end{bmatrix} \rightarrow \begin{bmatrix} -0.3 & 0.2 \\ 0 & 0 \end{bmatrix}$ $\Rightarrow x_1 = \begin{bmatrix} \frac{2}{3} \\ 1 \end{bmatrix}$	$M - \frac{1}{2} I = \begin{bmatrix} 0.2 & 0.2 \\ 0.3 & 0.3 \end{bmatrix} \rightarrow \begin{bmatrix} 0.2 & 0.2 \\ 0 & 0 \end{bmatrix}$ $\Rightarrow x_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$
Check: $M x_1 = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \begin{bmatrix} \frac{2}{3} \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} \\ 1 \end{bmatrix} = \lambda_1 x_1$	Check: $M x_2 = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{-1}{2} \\ \frac{1}{2} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \lambda_2 x_2$

c. Diagonalization of **M**:

Let $\mathbf{S} = \begin{bmatrix} \frac{2}{3} & -1 \\ 1 & 1 \end{bmatrix}$ and $\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$

$\mathbf{S^{-1}} = \frac{3}{5}\begin{bmatrix} 1 & 1 \\ -1 & \frac{2}{3} \end{bmatrix}$

$\mathbf{M} = \mathbf{S}\,\mathbf{\Lambda}\,\mathbf{S^{-1}} = \frac{3}{5}\begin{bmatrix} \frac{2}{3} & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & \frac{2}{3} \end{bmatrix}$

$\mathbf{M^n} = \mathbf{S}\,\mathbf{\Lambda^n}\,\mathbf{S^{-1}} = \frac{3}{5}\begin{bmatrix} \frac{2}{3} & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2^n} \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & \frac{2}{3} \end{bmatrix}$

1. Probability

To know the probability **P(2)**, **P(3)** and **P(n)** that a person who purchased a paper today will purchase a paper on Day 2, Day 3 and Day n, we need to compute $\mathbf{M^2}$, $\mathbf{M^3}$ and $\mathbf{M^n}$ respectively.

$\mathbf{M^2} = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix}\begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} = \begin{bmatrix} 0.55 & 0.3 \\ 0.45 & 0.7 \end{bmatrix}$

$\mathbf{M^3} = \begin{bmatrix} 0.55 & 0.3 \\ 0.45 & 0.7 \end{bmatrix}\begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} = \begin{bmatrix} 0.475 & 0.35 \\ 0.525 & 0.65 \end{bmatrix}$

$\mathbf{M^n} = \frac{3}{5}\begin{bmatrix} \frac{2}{3} & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2^n} \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & \frac{2}{3} \end{bmatrix} = \begin{bmatrix} \frac{2}{5} + \frac{3}{5*2^n} & \frac{2}{5} - \frac{1}{5*2^{n-1}} \\ \frac{3}{5} - \frac{3}{5*2^n} & \frac{3}{5} + \frac{1}{5*2^{n-1}} \end{bmatrix}$

We know that a person who purchased a paper today belongs to group 1; therefore, the probability that he will also be in group 1 (i.e. will buy a paper) after n days is given by entry $\boldsymbol{m_{1,1}^n}$ in $\mathbf{M^n}$.

So **P(2) = 0.55**, **P(3) = 0.475** and **P(n)** = $\frac{2}{5} + \frac{3}{5*2^n}$.

2. Sales figures

To find the expected sales figure for Day 2, Day 3 and Day n, we can compute, respectively, $\boldsymbol{v^{(2)}}$, $\boldsymbol{v^{(3)}}$ and $\boldsymbol{v^{(n)}}$.

We have: $\boldsymbol{v^{(0)}} = \begin{bmatrix} g_1^{(0)} \\ g_2^{(0)} \end{bmatrix} = \begin{bmatrix} 750 \\ 250 \end{bmatrix}$ and we know that: $\boldsymbol{v^{(n)}} = \mathbf{M^n}\,\boldsymbol{v^{(0)}}$

So:

$\boldsymbol{v^{(2)}} = \mathbf{M^2}\,\boldsymbol{v^{(0)}} = \begin{bmatrix} 0.55 & 0.3 \\ 0.45 & 0.7 \end{bmatrix}\begin{bmatrix} 750 \\ 250 \end{bmatrix} = \begin{bmatrix} 487.5 \\ 512.5 \end{bmatrix}$

$\boldsymbol{v^{(3)}} = \mathbf{M^3}\,\boldsymbol{v^{(0)}} = \begin{bmatrix} 0.475 & 0.35 \\ 0.525 & 0.65 \end{bmatrix}\begin{bmatrix} 750 \\ 250 \end{bmatrix} = \begin{bmatrix} 443.75 \\ 556.25 \end{bmatrix}$

$$\boldsymbol{v^{(n)}} = \mathbf{M^n}\,\boldsymbol{v^{(0)}} = \begin{bmatrix} \frac{2}{5} + \frac{3}{5*2^n} & \frac{2}{5} - \frac{1}{5*2^{n-1}} \\ \frac{3}{5} - \frac{3}{5*2^n} & \frac{3}{5} + \frac{1}{5*2^{n-1}} \end{bmatrix} \begin{bmatrix} 750 \\ 250 \end{bmatrix}$$

So, about $\mathbf{487}$, $\mathbf{443}$ and $\mathbf{750} * \frac{2^{n+1}+3}{5*2^n} + \mathbf{250} * \frac{2^n-1}{5*2^{n-1}}$ papers are expected to be sold, respectively, on Day 2, Day 3 and Day n.

3. Sales figures future trend

We already know that we can compute the expected number of papers that will be sold (or not sold) on a certain day by the formula:

$$\boldsymbol{v^{(n)}} = \mathbf{M^n}\,\boldsymbol{v^{(0)}} = \begin{bmatrix} \frac{2}{5} + \frac{3}{5*2^n} & \frac{2}{5} - \frac{1}{5*2^{n-1}} \\ \frac{3}{5} - \frac{3}{5*2^n} & \frac{3}{5} + \frac{1}{5*2^{n-1}} \end{bmatrix} \begin{bmatrix} 750 \\ 250 \end{bmatrix}$$

If n → ∞, then $\frac{3}{5*2^n}$ → 0 and $\frac{1}{5*2^{n-1}}$ → 0.

Therefore, when n is large enough, we can expect the sales figures to stabilize.

$$\boldsymbol{v^{(n)}} = \begin{bmatrix} \frac{2}{5} & \frac{2}{5} \\ \frac{3}{5} & \frac{3}{5} \end{bmatrix} \begin{bmatrix} 750 \\ 250 \end{bmatrix} = \begin{bmatrix} 400 \\ 600 \end{bmatrix}$$

The trend after a while will hence become about 2 out of 3 papers being sold everyday.

Problem 2

1. The PageRank method

The idea is that the ranking of a webpage should depend on the ranking of all webpages linking to it. If many high rank webpages recommend a webpage, it will have a higher rank. Algebraically, the PageRank method computes a vector **v** where each entry is the ranking of a page in the web. In a simpler scenario, **v** is found by solving the eigenvalue problem **v = Pv** where **v** is the eigenvector of the Markov matrix P with eigenvalue 1.

However, in a real scenario where there are many groups of webpages that are not connected, solving **v = Pv** leads to more than 1 distinct eigenvector and hence the ranking vector won't be unique. Another issue is that some pages will have scores of 0. Additionally, a web surfer may stop following hyperlinks and instead go to another random page in the web ("teleporting").

The PageRank method tackles these issues and models the behavior of a random web surfer by creating a new matrix Q and introducing a parameter r = 0.85 which is the probability that a web surfer will visit a new page through a hyperlink in a previous page.

The equation for **Q** is: **Q = r * P + (1 – r) * T**.

T is the "teleportation" matrix; and because a surfer may choose any random page out of the total N webpages in the web, all the elements in **T** are $\frac{1}{N}$.

Q is also a Markov matrix but unlike **P**, it has no 0 entry. This ensures that solving **v = Qv** will lead to a unique ranking vector **v** and that no element in **v** is 0.

2. Solution to the problem

When the new Markov matrix Q is found, the eigenvector **v** can be found by solving **v = Qv**. Gaussian elimination would of course work but it is not efficient when we know there are billions of webpages. A faster way exploits theorem 2 and finds **v** by calculating $\mathbf{Q^k w}$ where **k** is a large number and **w** is an initial ranking vector with entries adding to 1. However, these repeated multiplications of a very large matrix and vector will still be time consuming. The idea is to use the formula of Q instead of involving the computed Q itself.

Since **Q = r * P + (1 – r) * T**, multiplying both sides by vector **w** gives:

Q * w = r * P * w + (1 – r) * T * w = **r * P * w + (1 – r) * T * w**

and $\mathbf{T * w} = \frac{1}{N} * \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix} * \begin{bmatrix} w1 \\ \vdots \\ wN \end{bmatrix} = \frac{1}{N} * \begin{bmatrix} w1 + \cdots + wN \\ \vdots \\ w1 + \cdots + wN \end{bmatrix} = \frac{1}{N} * \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \frac{1}{N} * \mathbf{u}$

So, $\mathbf{Q * w} = \mathbf{r * P * w} + \frac{1 - r}{N} * \mathbf{u}$.

We can now use the right-hand side which, though it involves a large matrix and vector multiplication, will take much less time to compute as **P**, unlike **Q**, would contain many 0 entries.

So $\mathbf{w_1 = r * P * w_0 + \frac{1-r}{N} * u,}$

$\mathbf{w_2 = r * P * w_1 + \frac{1-r}{N} * u}$

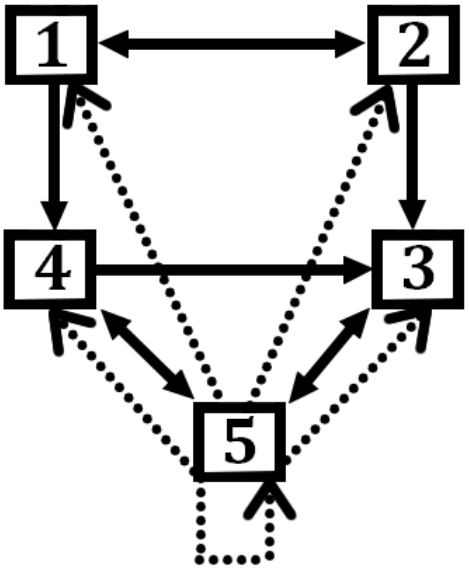
$\mathbf{\dots}$

$\mathbf{w_{k+1} = r * P * w_k + \frac{1-r}{N} * u}$

If $\mathbf{w_{k+1} = w_k}$, then we take $\mathbf{v = w_k}$.

3. Applied example

Let's consider the following scenario of 5 webpages (page 5 is a dead-end):



$$P = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 & \frac{1}{5} \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{5} \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{5} \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{5} \\ 0 & 0 & 1 & \frac{1}{2} & \frac{1}{5} \end{bmatrix}, \quad T = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}, \quad r = 0.85 \,,\; 1 - r = 0.15$$

$$\mathbf{Q = r * P + (1 - r) * T}$$

$$= \begin{bmatrix} 0 & 0.425 & 0 & 0 & 0.17 \\ 0.425 & 0 & 0 & 0 & 0.17 \\ 0 & 0.425 & 0 & 0.425 & 0.17 \\ 0.425 & 0 & 0 & 0 & 0.17 \\ 0 & 0 & 0.85 & 0.425 & 0.17 \end{bmatrix} + \begin{bmatrix} 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \end{bmatrix}$$

$$= \begin{bmatrix} 0.03 & 0.455 & 0.03 & 0.03 & 0.2 \\ 0.455 & 0.03 & 0.03 & 0.03 & 0.2 \\ 0.03 & 0.455 & 0.03 & 0.455 & 0.2 \\ 0.455 & 0.03 & 0.03 & 0.03 & 0.2 \\ 0.03 & 0.03 & 0.88 & 0.455 & 0.2 \end{bmatrix}$$

Let's choose $\mathbf{w} = \begin{bmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{bmatrix}$

Then $\mathbf{v} = \mathbf{Q}^{\mathbf{k}}\ \mathbf{w}$ for large $\mathbf{k}$. Eventually $\mathbf{v} = \begin{bmatrix} 0.150687511772458.. \\ 0.150687511772458.. \\ 0.214729704275757.. \\ 0.150687511772458.. \\ 0.333207760406870.. \end{bmatrix}$

(using Matlab). As expected, the entries in **v** adds to 1 (or almost, if we reduce the precision).

However, we can make the calculation of **Q * w** faster by following the formula:

$$\mathbf{Q\ w = r * P * w + \frac{1-r}{N}\ u}$$

At initial state (iteration#0): $\mathbf{w_0 = w} = \begin{bmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{bmatrix}$

We have: $\mathbf{r * P} = \begin{bmatrix} 0 & 0.425 & 0 & 0 & 0.17 \\ 0.425 & 0 & 0 & 0 & 0.17 \\ 0 & 0.425 & 0 & 0.425 & 0.17 \\ 0.425 & 0 & 0 & 0 & 0.17 \\ 0 & 0 & 0.85 & 0.425 & 0.17 \end{bmatrix}$ and $\frac{1-r}{N}\mathbf{u} = \begin{bmatrix} 0.03 \\ 0.03 \\ 0.03 \\ 0.03 \\ 0.03 \end{bmatrix}$

So: $\mathbf{w_{k+1} = Q * w_k}$

$$= \mathbf{r * P * w_k + \frac{1-r}{N}\ u}$$

$$= \begin{bmatrix} 0 & 0.425 & 0 & 0 & 0.17 \\ 0.425 & 0 & 0 & 0 & 0.17 \\ 0 & 0.425 & 0 & 0.425 & 0.17 \\ 0.425 & 0 & 0 & 0 & 0.17 \\ 0 & 0 & 0.85 & 0.425 & 0.17 \end{bmatrix} * w_k + \begin{bmatrix} 0.03 \\ 0.03 \\ 0.03 \\ 0.03 \\ 0.03 \end{bmatrix}$$

The matrix-vector multiplication is now faster because the matrix contains many 0's compared to Q which has none.

Iteration#1	Iteration#2	Iteration#3
$w_1 = Q * w_0 = \begin{bmatrix} 0.149 \\ 0.149 \\ 0.234 \\ 0.149 \\ 0.319 \end{bmatrix}$	$w_2 = Q * w_1 = \begin{bmatrix} 0.147555 \\ 0.147555 \\ 0.210880 \\ 0.147555 \\ 0.346455 \end{bmatrix}$	$w_3 = Q * w_2 = \begin{bmatrix} 0.151608225 \\ 0.151608225 \\ 0.214319100 \\ 0.151608225 \\ 0.330856225 \end{bmatrix}$
Iteration#4	Iteration#5	….
$w_4 = Q * w_3 = \begin{bmatrix} 0.150679053875 \\ 0.150679053875 \\ 0.215112549500 \\ 0.150679053875 \\ 0.332850288875 \end{bmatrix}$	$w_5 = Q * w_4 = \begin{bmatrix} 0.150623147005625 \\ 0.150623147005625 \\ 0.214661744902500 \\ 0.150623147005625 \\ 0.333468814080625 \end{bmatrix}$	….

So we can take the eigenvector/ranking vector to be $\mathbf{v} = w_5 = \begin{bmatrix} 0.150623147005625 \\ 0.150623147005625 \\ 0.214661744902500 \\ 0.150623147005625 \\ 0.333468814080625 \end{bmatrix}$.

4. Matlab code

```
function [v, iteration] = eigenvector_func (P, w0)
  % if matrix P not given, create one
  if ~exist('P')
    P = [0    0.5   0   0     0.2;
         0.5  0     0   0     0.2;
         0    0.5   0   0.5   0.2;
         0.5  0     0   0     0.2;
         0    0     1   0.5   0.2];
  end
  [M,N] = size(P);
  % check if P is a square matrix
  if M != N
    error('The input matrix is not a square matrix.')
  end
  % check if each column of P sums to 1
  if ~isequal(sum(P),ones(1,N))
    error('The input matrix is not a Markov matrix.')
  end
  % if initial vector w0 not given, create one
  if ~exist('w0')
    w0 = (1/N) * [ones(N,1)];
  end

  r = .85; % porbability parameter
  s = 1 - r; % compelment of probability parameter
  u = ones(N,1); % vector of 1's
  rP = r * P; % first part of equation of w
  n = (s / N) * u; % second part of equation of w
  count = 0; % keep track of repetitions
  max = 100; % max number of iterations
  w = w0; % keep track of latest computed w

  while (count < max)
    w_new = rP * w + n; % compute new w
    % check if reached a stable state
    if isequal(w, w_new)
      break; % stop the loop if we found the final w
    end
    w = w_new; % update w for the next iteration
    count++;
  end

  v = w; % return eigenvector
  iteration = count; % return number of iterations to find eigenvector
endfunction
```

Sample output (no arguments was passed so a default **P** and $\mathbf{w_0}$ was used):

```
>> [v, iterations] = eigenvector_func()
v =

   0.150687511772462
   0.150687511772462
   0.214729704275758
   0.150687511772462
   0.333207760406856

iterations =  29
>> sum(v)
ans =  1.00000000000000
```

5. Comments

People who understand how the PageRank works, may "hack" it for the purpose of boosting a targeted website's PageRank score. They may create many pages that links to a certain website for the only reason to make it appear in the top search results. This is known as link spamming. Also, there is only one ranking score computed for each page in the web regardless of what is queried. This is not optimal as some webpages that may be very relevant in a topic may have a lesser score than other webpages which deal little on the same topic.

However, several researches were done to improve the PageRank algorithm or overcome its weaknesses such as the TrustRank algorithm that reduces the effect of link spamming and the Topic Sensitive PageRank that optimizes the accuracy of the modeling assumption to provide a relatively better (or lower) raking for a page depending on the topic searched for.

Ultimately, these issues and more are already tackled by Google's search engine, but most importantly, it is to be known that, nowadays, the PageRank algorithm is just one part of the mechanism built by Google to displays search results.