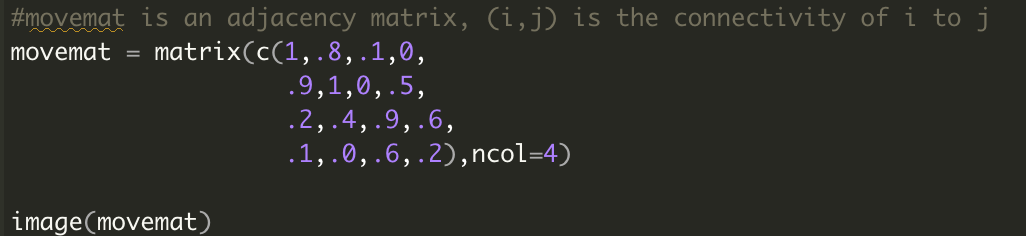
**Network models**

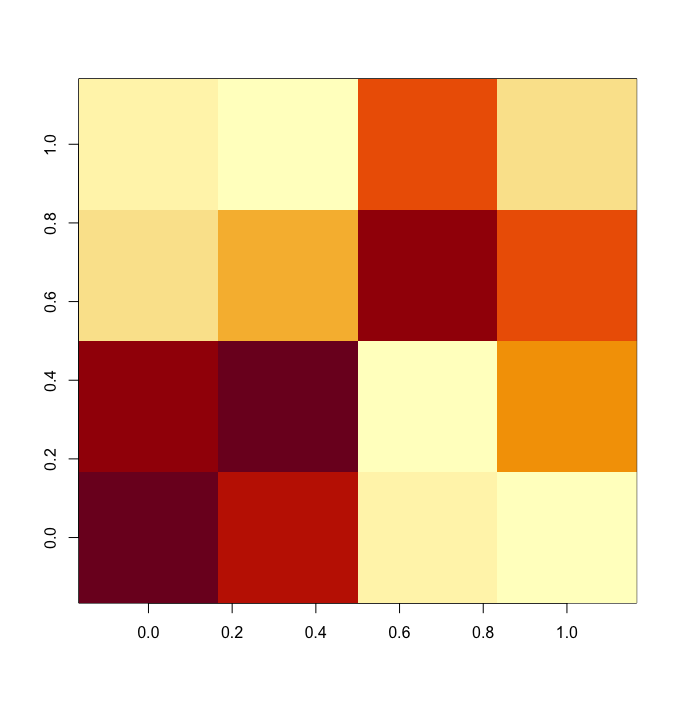
We will continue working with real data, and instead of running simulations of disease spread today, we will be familiarizing ourselves with various network analyses that you could use on any mobility matrix.

The main package we will be using today is called “igraph”; make sure it is installed first using install.packages(“igraph”), then load it into your workspace using library(igraph). This package does a lot of different network analyses, including the centrality measures and the community detection we discussed in class.

First, generally, igraph uses adjacency matrices. This can be constructed from scratch, which is what the first few lines of code does:

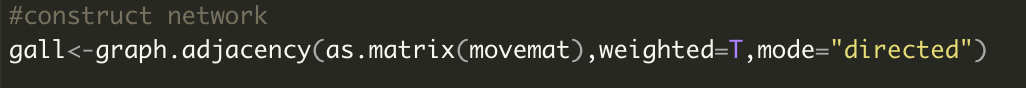


Which should return the following adjacency matrix heatmap:



Essentially, this is similar to a 4 patch system, where patches 1 and 2 are relatively strongly connected to each other. Note that the sums of the rows and columns do not add up to 1 (I just made up the values)—when we use adjacency matrices in igraph, we don’t have to worry about the weights summing up to 1, or anything. They simply mean a relative weight on the connection between two patches, and could be numbers of travelers per day, or proportion of time spent, or any metric.

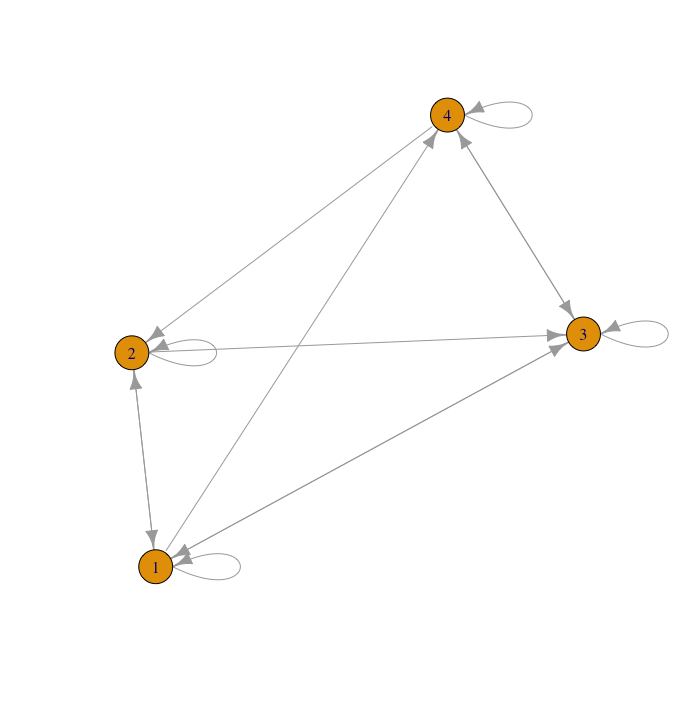
The first thing we need to do with this adjacency matrix is create a “graph” object in igraph using the “graph.adjacency” function:



Here, we are telling igraph to create a graph using the supplied adjacency matrix. We are also telling it that it is a “weighted” network, which means there are weights corresponding to each connection. An unweighted network would be one where there is either a connection or not (1 or 0), which would be useful if we were mapping a friendship network where someone is friends with another, or not.

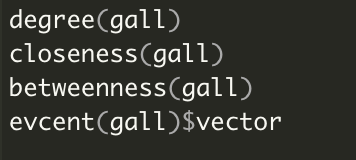
Finally, we are telling igraph that this is a “directed” network, which means that the connection in one direction might be different from the connection in the other direction. This is generally true for mobility networks; for example, more people may visit Kigali from Entebbe than the other way around.

Now that this graph is constructed, we can plot it using plot.igraph():



Once you’ve done that, then we can start calculating the various network metrics. First, let’s figure out centrality: Which nodes are most central to this network, based on the different definitions of centrality?

The next few lines of code calculate these values for the 4 different patches in our theoretical network.

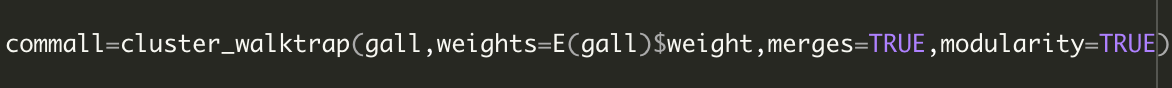


These functions calculate the degree for each patch, the closeness centrality, the betweenness centrality, and the eigenvector centrality for each graph.

TASK 1: Which patch has the highest eigenvector centrality? This would imply that if an outbreak began in this patch, it would have the highest potential to spread across the network.

Next, we are going to calculate the clustering coefficient and identify the community structure in this graph. We can do this using cluster\_walktrap(). There are a number of clustering algorithms we can use (see the igraph page for more information), but for the purposes of this worksheet we will use the walktrap algorithm.

The following line runs the walktrap clustering algorithm:



Here, we are telling it the graph to use (“gall”), and then we are telling it what the weights for each of the connections are (here, igraph stores that within “gall”, and it can be called using E(gall)$weight). We are telling it to keep track of merges, so that we can look at other clusters (keeping in mind that clustering is intrinsically hierarchical). Finally, we are telling it to calculate a modularity score, which is an overall measure of clustering.

The modularity score for a community structure for a network is calculated as the difference in connections within communities and between communities. The value for a modularity score varies from 0 – 1, where 0 indicates a network with no clustering whatsoever, and 1 denotes an entirely clustered network (where communities are not connected to each other at all).

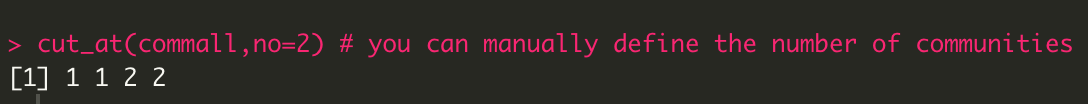
If we run this walktrap algorithm for our theoretical graph, we get these results:

A black background with white text

Description automatically generated

What this output tells us is there is optimally only one group of nodes (every node is in the same community), and the clustering within this network is 0.24, which is fairly low from the spectrum of 0 – 1. It also notes that the first group contains all 4 nodes (named 1, 2, 3, and 4).

Finally, you can define the number of communities you want manually. This can be useful if you know how many partitions you’d like to divide a country or region into, and you want to hard-code that into your analysis.



This line says we want 2 communities, and it tells us 1 and 2 are in the first group, and 3 and 4 are in the second group.

For the second part of this assignment, you will be using the Virginia dataset and doing the following:

* Create an igraph object
* Use this igraph object to calculate degree centrality, betweenness centrality, and eigenvector centrality. I have added code to plot the eigenvector centrality, modify this to show the betweenness centrality instead.
* Finally, plot the community structure.

Please post the figures below: