CMLC Random Graph Example

H. Robert Frost

This vignette illustrates the use of the CMLC R package to compute constrained multilayer centrality values for a 2 layer network with varying inter-layer dependency structures. Each layer of this network is generated as an interconnected group of 5 Erdos-Renyi random graphs that each have 20 nodes.

1 Load packages

Load the CMLC and igraph R packages:

```
> library(CMLC)
> library(igraph)
```

Set seed for repeatability:

```
> set.seed(1)
```

2 Create multilayer network

Create the graph for the first layer as a collection of 5 islands that are each an Erdos-Renyi random graph with 20 nodes. Create a layout for this graph using the Fruchterman-Reingold algorithm and compute the eigenvector centrality values.

```
> g1 = sample_islands(islands.n=5, islands.size=20, islands.pin=0.2, n.inter=1)
> lo1 = layout_with_fr(g1)
> g1.evc = eigen_centrality(g1, scale=FALSE)
```

Create the graph for layer 2 using similar logic.

```
> g2 = sample_islands(islands.n=5, islands.size=20, islands.pin=0.2, n.inter=1)
> lo2 = layout_with_fr(g2)
> g2.evc = eigen_centrality(g2, scale=FALSE)
```

Get the adjacency matrices for both graphs and combine into a multilayer list.

```
> adj.1 = get.adjacency(g1)
> adj.2 = get.adjacency(g2)
> ML=list(adj.1, adj.2)
```

3 Both layers are independent (case A)

Compute centrality values for the no dependency case. For all scenarios, set max.iter to 200 to ensure convergence at default threshold.

```
> max.iter = 200
> mlc.indep = constrainedMultiplePowerIteration(X=ML,max.iter=max.iter,A=diag(c(1,1)))
```

4 Combination of independence (case A) and mixed dependency (case C) using CMLC

Compute centrality values when layer 1 is independent and dependent node importance for layer 2 is based on a mixture of 10% layer 1 and 90% layer 2.

```
> A.2to1.part = matrix(c(1,0,
+                   0.1,0.9),
+                 nrow=2, byrow=TRUE)
> mlc.2to1.part = constrainedMultiplePowerIteration(X=ML,max.iter=max.iter,A=A.2to1.part)
```

Compute centrality values when layer 2 is independent and dependent node importance for layer 1 is based on a mixture of 10% layer 2 and 90% layer 1.

```
> A.1to2.part = matrix(c(0.9,0.1,
+                   0,1),
+                 nrow=2, byrow=TRUE)
> mlc.1to2.part = constrainedMultiplePowerIteration(X=ML,max.iter=max.iter,A=A.1to2.part)
```

5 Combination of independence (case A) and dependent (case B) using CMLC

Compute centrality values when layer 1 is independent and dependent node importance for layer 2 is based completely on layer 1.

```
> A.2to1 = matrix(c(1,0,
+                   1,0),
+            nrow=2, byrow=TRUE)
> mlc.2to1 = constrainedMultiplePowerIteration(X=ML,max.iter=max.iter,A=A.2to1)
```

Compute centrality values when layer 2 is independent and dependent node importance for layer 2 is based completely on layer 1.

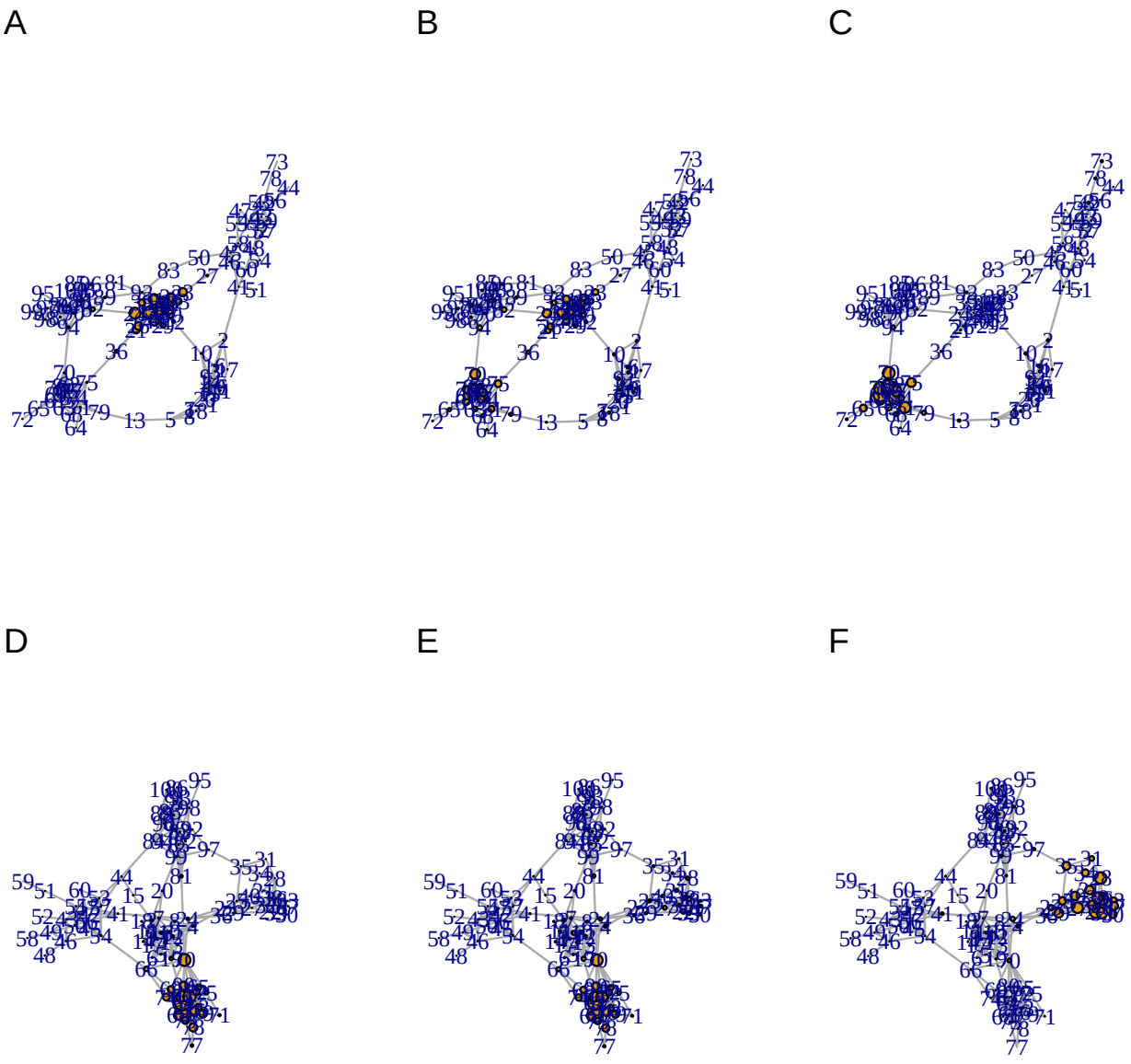
```
> A.1to2 = matrix(c(0,1,
+                   0,1),
+                 nrow=2, byrow=TRUE)
> mlc.1to2 = constrainedMultiplePowerIteration(X=ML,max.iter=max.iter,A=A.1to2)
```

Create a multi-panel plot where each row corresponds to one layer and node size is based on eigenvector centrality values. The first column is the independent case, the second column is the scenario where one layer is independent and there other has a 90/10 split dependency, and the last column is the scenario where one layer is independent and the other is fully dependent.

6 Visualize CMLC centrality values

```
> par(mfrow=c(2,3), mar=c(2,2,2,2))
> size.scaling=30
> plot(g1, layout=lo1, vertex.size=mlc.indep$V1[,1]*size.scaling)
> mtext("A", 3, adj=0)
> plot(g1, layout=lo1, vertex.size=mlc.1to2.part$V1[,1]*size.scaling)
> mtext("B", 3, adj=0)
> plot(g1, layout=lo1, vertex.size=mlc.1to2$V1[,1]*size.scaling)
```

```
> mtext("C", 3, adj=0)
> plot(g2, layout=lo2, vertex.size=mlc.indep$V1[,2]*size.scaling)
> mtext("D", 3, adj=0)
> plot(g2, layout=lo2, vertex.size=mlc.2to1.part$V1[,2]*size.scaling)
> mtext("E", 3, adj=0)
> plot(g2, layout=lo2, vertex.size=mlc.2to1$V1[,2]*size.scaling)
> mtext("F", 3, adj=0)
```



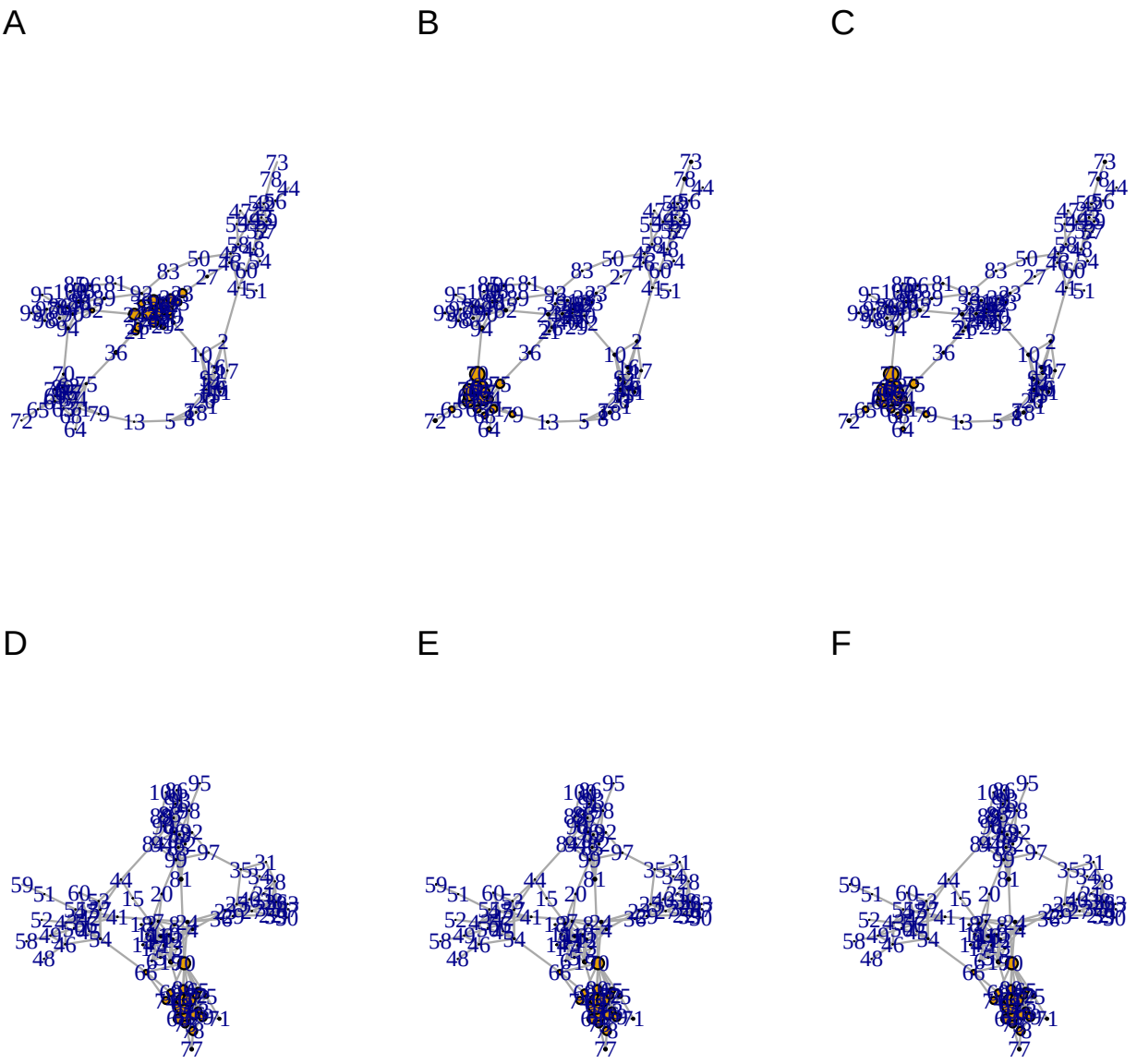
7 Compute and visualize centrality values using inter-layer edge model

Compute centrality values using the same dependency adjacency matrices using the inter-layer edge approach.

```
> interlayer.edge.2to1.part = interlayerEdgeCentrality(X=ML, A=A.2to1.part)
> interlayer.edge.1to2.part = interlayerEdgeCentrality(X=ML,A=A.1to2.part)
> interlayer.edge.2to1 = interlayerEdgeCentrality(X=ML,A=A.2to1)
> interlayer.edge.1to2 = interlayerEdgeCentrality(X=ML,A=A.1to2)
```

Create a multi-panel plot where each row corresponds to one layer and node size is based on eigenvector centrality values. The first column is the independent case, the second column is the scenario where one layer is independent and there other has a 90/10 split dependency, and the last column is the scenario where one layer is independent and the other is fully dependent.

```
> par(mfrow=c(2,3), mar=c(2,2,2,2))
> size.scaling=30
> plot(g1, layout=lo1, vertex.size=mlc.indep$V1[,1]*size.scaling)
> mtext("A", 3, adj=0)
> plot(g1, layout=lo1, vertex.size=interlayer.edge.1to2.part[,1]*size.scaling)
> mtext("B", 3, adj=0)
> plot(g1, layout=lo1, vertex.size=interlayer.edge.1to2[,1]*size.scaling)
> mtext("C", 3, adj=0)
> plot(g2, layout=lo2, vertex.size=mlc.indep$V1[,2]*size.scaling)
> mtext("D", 3, adj=0)
> plot(g2, layout=lo2, vertex.size=interlayer.edge.2to1.part[,2]*size.scaling)
> mtext("E", 3, adj=0)
> plot(g2, layout=lo2, vertex.size=interlayer.edge.2to1[,2]*size.scaling)
> mtext("F", 3, adj=0)
```



8 Combination of independence (case A) and mixed dependency (case C) with negative weight

Compute centrality values when layer 1 is independent and dependent node importance for layer 2 is based on a mixture of -10% layer 1 and 110% layer 2.

```
> A.2to1.neg.part = matrix(c(1,0,
+                    -0.1,1.1),
+                  nrow=2, byrow=TRUE)
> mlc.2to1.neg.part = constrainedMultiplePowerIteration(X=ML,max.iter=max.iter,A=A.2to1.neg.part)
```

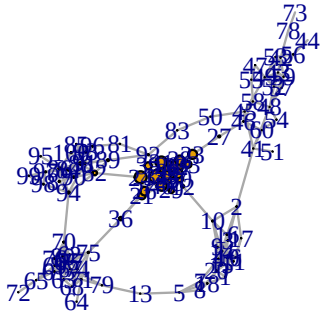
Compute centrality values when layer 2 is independent and dependent node importance for layer 1 is based on a mixture of -10% layer 2 and 110% layer 1.

```
> A.1to2.neg.part = matrix(c(1.1,-0.1,
+                    0,1),
+                  nrow=2, byrow=TRUE)
> mlc.1to2.neg.part = constrainedMultiplePowerIteration(X=ML,max.iter=max.iter,A=A.1to2.neg.part)
```

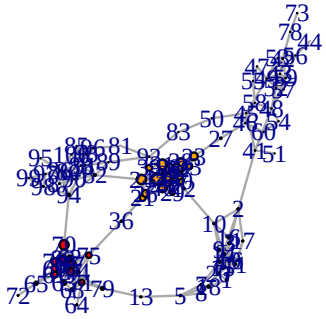
Create a multi-panel plot where each row corresponds to one layer and node size is based on eigenvector centrality values. The first column is the independent case, the second column is the scenario where one layer is independent and there other has a 90/10 split dependency, and the last column is the scenario where one layer is independent and the other is fully dependent.

```
> par(mfrow=c(2,3), mar=c(2,2,2,2))
> size.scaling=30
> plot(g1, layout=lo1, vertex.size=mlc.indep$V1[,1]*size.scaling)
> mtext("A", 3, adj=0)
> neg.vals = which(mlc.1to2.neg.part$V1[,1] < 0)
> n = length(mlc.1to2.neg.part$V1[,1])
> colors = rep("orange", n)
> colors[neg.vals] = "red"
> plot(g1, layout=lo1, vertex.size=abs(mlc.1to2.neg.part$V1[,1]*size.scaling), vertex.color=colors)
> mtext("B", 3, adj=0)
> plot(g1, layout=lo1, vertex.size=mlc.1to2$V1[,1]*size.scaling)
> mtext("C", 3, adj=0)
> plot(g2, layout=lo2, vertex.size=mlc.indep$V1[,2]*size.scaling)
> mtext("D", 3, adj=0)
> neg.vals = which(mlc.2to1.neg.part$V1[,2] < 0)
> colors = rep("orange", n)
> colors[neg.vals] = "red"
> plot(g2, layout=lo2, vertex.size=abs(mlc.2to1.neg.part$V1[,2]*size.scaling), vertex.color=colors)
> mtext("E", 3, adj=0)
> plot(g2, layout=lo2, vertex.size=mlc.2to1$V1[,2]*size.scaling)
> mtext("F", 3, adj=0)
```

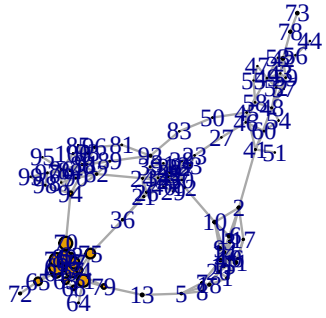
A



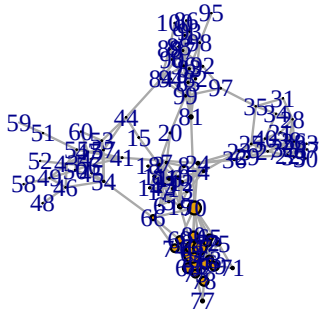
B



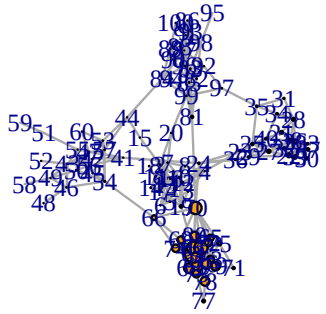
C



D



E



F