Project_8

Contents

1. Install and load the R packages

```
#Load packages
library(ggplot2)
library(igraph)
library(mnem)
```

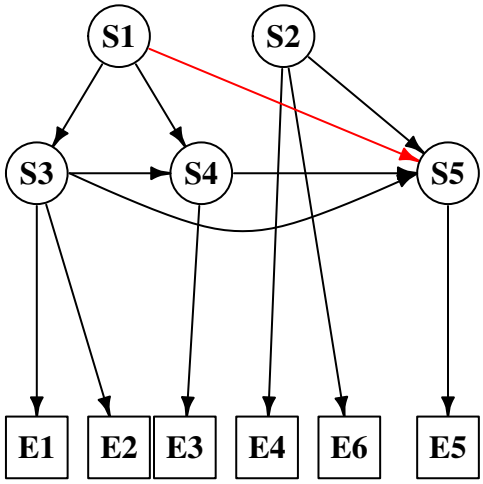
1.1

Problem 20 : Classical NEMs

1. For each model, construct the transitive closure (by adding edges) and define the corresponding adjacency matrices ϕ and θ, which represent the signalling pathways and the E-gene attachments. Determine the corresponding expected effect patterns (F).

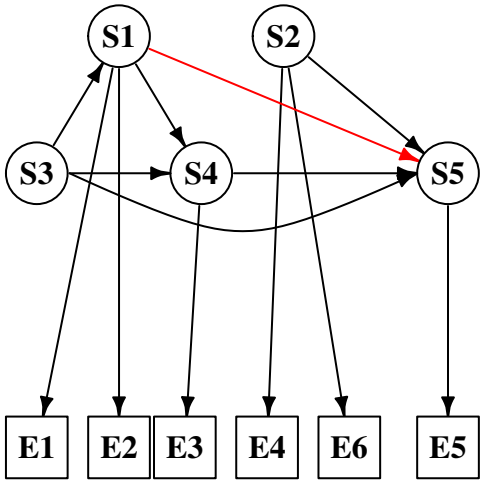
```
# Define the graph for Model 1
edges_model1 <- c('S1', 'S3',
                  "S1", "S4",
                  "S2", "E4",
                  "S2", "E6",
                  "S2", "S5",
                  "S3", "E1",
                  "S3", "E2",
                  "S3", "S4",
                  "S3", "S5",
                  "S4", "E3",
                  "S4", "S5",
                  "S5", "E5", #from here transitive closure edges
                  "S1", "S5"
                  )
graph_model1 <- graph(edges_model1, directed = TRUE)

# Define vertex names
V(graph_model1)$name <- c('S1', 'S3', 'S4', 'S2', 'E4', 'E6', 'S5', 'E1', 'E2', 'E3', 'E5')

# Define vertex shapes
V(graph_model1)$shape <- ifelse(grepl("^S", V(graph_model1)$name), "circle", "square")
```

```r
# Define curved edge (S3 to S5)
E(graph_model1)$curved <- 0
E(graph_model1, path = c('S3', 'S5'))$curved <- -0.5

# Set the original edges color to black
E(graph_model1)$color <- "black"

# Set the color of the transitive closure edges to the red

E(graph_model1, path = c("S1", "S5"))$color <- "red"


# Manual layout
layout_matrix <- matrix(c(
  # S genes (circles)
  1, 1,   # S1
  0, 0,   # S3
  2, 0,   # S4
  3, 1,   # S2
  2.8, -2,  # E4
  3.8, -2, # E6
  5, 0,  # S5
  0, -2,  # E1
  1, -2,  # E2
  1.8, -2,  # E3
  5, -2   # E5
), ncol = 2, byrow = TRUE)

# Normalize layout for plotting
max_coord <- apply(layout_matrix, 2, max)
min_coord <- apply(layout_matrix, 2, min)
layout_matrix <- sweep(layout_matrix, 2, min_coord, "-")
layout_matrix <- sweep(layout_matrix, 2, max_coord - min_coord, "/")

# Plot the graph
plot(graph_model1,
     layout = layout_matrix,
     edge.arrow.size = 0.5,
     vertex.label = V(graph_model1)$name,
     vertex.size = 30,
     vertex.color = "white",
     vertex.frame.color = "black",
     vertex.label.color = "black",
     vertex.label.font = 2, # Using a bold font for the vertex labels
     vertex.shape = V(graph_model1)$shape
)
```




```
# Define the graph for Model 2 (change edges as per model specifics)
edges_model2 <- c("S1", "S4", "S1", "E1", "S1", "E2", "S2", "E4","S2", "E6","S2", "S5",
                  "S3", "S1", "S3", "S5", "S3", "S4", "S4", "E3", "S4", "S5", "S5", "E5", #from here tr
                  "S1", "S5")
graph_model2 <- graph(edges_model2, directed = TRUE)

# Define vertex names
V(graph_model2)$name <- c('S1', 'S4', 'E1', 'E2', 'S2', 'E4', 'E6', 'S5', 'S3', 'E3', 'E5')

# Define vertex shapes
V(graph_model2)$shape <- ifelse(grepl("^S", V(graph_model2)$name), "circle", "square")

# Define curved edge (S3 to S5)
E(graph_model2)$curved <- 0
E(graph_model2, path = c('S3', 'S5'))$curved <- -0.5

# Set the original edges color to black
E(graph_model2)$color <- "black"


# Set the color of the transitive closure edges to the red

E(graph_model2, path = c("S1", "S5"))$color <- "red"


# Manual layout
```

```r
layout_matrix <- matrix(c(
  # S genes (circles)
  1, 1,   # S1
  2, 0,   # S4
  0, -2,  # E1
  1, -2,  # E2
  3, 1,   # S2
  2.8, -2,  # E4
  3.8, -2, # E6
  5, 0,  # S5
  0, 0,   # S3
  1.8, -2,  # E3
  5, -2   # E5
), ncol = 2, byrow = TRUE)

# Normalize layout for plotting
max_coord <- apply(layout_matrix, 2, max)
min_coord <- apply(layout_matrix, 2, min)
layout_matrix <- sweep(layout_matrix, 2, min_coord, "-")
layout_matrix <- sweep(layout_matrix, 2, max_coord - min_coord, "/")

# Plot the graph
plot(graph_model2,
     layout = layout_matrix,
     edge.arrow.size = 0.5,
     vertex.label = V(graph_model2)$name,
     vertex.size = 30,
     vertex.color = "white",
     vertex.frame.color = "black",
     vertex.label.color = "black",
     vertex.label.font = 2, # Using a bold font for the vertex labels
     vertex.shape = V(graph_model2)$shape
)
```



We added the edges to ensure the transitive closure in red

```
# Create a subgraph with only S vertices and their edges for the adjanc matrix phi
s_vertices1 <- V(graph_model1)[grepl("^S", V(graph_model1)$name)]
subgraph_s1 <- induced_subgraph(graph_model1, s_vertices1)

# # Plot the subgraph
# plot(subgraph_s1,
#      edge.arrow.size = 0.5,
#      vertex.label = V(subgraph_s1)$name,
#      vertex.size = 30,
#      vertex.color = "white",
#      vertex.frame.color = "black",
#      vertex.label.color = "black",
#      vertex.label.font = 2)

# Adjacency matrix
phi_model1 <- as_adjacency_matrix(subgraph_s1)
diag(phi_model1) <- 1 # Set the diagonal elements to 1
phi1<- as.matrix(phi_model1)

# Define new order for the rows and columns
new_order_s <- c("S1", "S2", "S3", "S4","S5")

# Reorder rows and columns
phi1 <- phi1[new_order_s, new_order_s]
```

```r
s_vertices2 <- V(graph_model2)[grepl("^S", V(graph_model2)$name)]
subgraph_s2 <- induced_subgraph(graph_model2, s_vertices2)

# # Plot the subgraph
# plot(subgraph_s2,
#      edge.arrow.size = 0.5,
#      vertex.label = V(subgraph_s2)$name,
#      vertex.size = 30,
#      vertex.color = "white",
#      vertex.frame.color = "black",
#      vertex.label.color = "black",
#      vertex.label.font = 2)

# Adjacency matrix
phi_model2 <- as_adjacency_matrix(subgraph_s2)
diag(phi_model2) <- 1 # Set the diagonal elements to 1
phi2<- as.matrix(phi_model2)

# Reorder rows and columns
phi2 <- phi2[new_order_s, new_order_s]


# Display adjancy matrices
print(phi1)
```

```
##    S1 S2 S3 S4 S5
## S1  1  0  1  1  1
## S2  0  1  0  0  1
## S3  0  0  1  1  1
## S4  0  0  0  1  1
## S5  0  0  0  0  1
```

```r
print(phi2)
```

```
##    S1 S2 S3 S4 S5
## S1  1  0  0  1  1
## S2  0  1  0  0  1
## S3  1  0  1  1  1
## S4  0  0  0  1  1
## S5  0  0  0  0  1
```

Above are the two adjacency matrices.

```r
# Get the names of S and E nodes
s_names <- V(graph_model1)$name[grepl("^S", V(graph_model1)$name)]
e_names <- V(graph_model1)$name[grepl("^E", V(graph_model1)$name)]

# Create an empty matrix with S nodes as rows and E nodes as columns
theta_matrix1 <- matrix(0, nrow = length(s_names), ncol = length(e_names),
                    dimnames = list(s_names, e_names))
```

```r
# Fill the matrix with edges information
for (s in s_names) {
  for (e in e_names) {
    if (are_adjacent(graph_model1, s, e)) {
      theta_matrix1[s, e] <- 1
    }
  }
}

new_order_e <-c("E1", "E2", "E3", "E4","E5","E6")

theta1 <- theta_matrix1[new_order_s, new_order_e]

# View the adjacency matrix
print(theta1)
```

```
##    E1 E2 E3 E4 E5 E6
## S1  0  0  0  0  0  0
## S2  0  0  0  1  0  1
## S3  1  1  0  0  0  0
## S4  0  0  1  0  0  0
## S5  0  0  0  0  1  0
```

```r
theta_matrix2 <- matrix(0, nrow = length(s_names), ncol = length(e_names),
                    dimnames = list(s_names, e_names))

# Fill the matrix with edges information
for (s in s_names) {
  for (e in e_names) {
    if (are_adjacent(graph_model2, s, e)) {
      theta_matrix2[s, e] <- 1
    }
  }
}
theta2 <- theta_matrix2[new_order_s, new_order_e]

# View the adjacency matrix
print(theta2)
```

```
##    E1 E2 E3 E4 E5 E6
## S1  1  1  0  0  0  0
## S2  0  0  0  1  0  1
## S3  0  0  0  0  0  0
## S4  0  0  1  0  0  0
## S5  0  0  0  0  1  0
```

Above are the two E-gene attachments matrices.

```r
#Compute the Expected effect pattern F

F1 <- phi1%*%theta1
F2 <- phi2%*%theta2
```

```
print(F1)
```

```
##    E1 E2 E3 E4 E5 E6
## S1  1  1  1  0  1  0
## S2  0  0  0  1  1  1
## S3  1  1  1  0  1  0
## S4  0  0  1  0  1  0
## S5  0  0  0  0  1  0
```

```
print(F2)
```

```
##    E1 E2 E3 E4 E5 E6
## S1  1  1  1  0  1  0
## S2  0  0  0  1  1  1
## S3  1  1  1  0  1  0
## S4  0  0  1  0  1  0
## S5  0  0  0  0  1  0
```

Above are the two expected effect pattern matrices.

2. Assuming no noise, determine the discrete data D1 and D2 from both models. Given only the data, can you tell apart the two models?

```
D1 <- t(F1)
D2 <- t(F2)

print(D1)
```

```
##    S1 S2 S3 S4 S5
## E1  1  0  1  0  0
## E2  1  0  1  0  0
## E3  1  0  1  1  0
## E4  0  1  0  0  0
## E5  1  1  1  1  1
## E6  0  1  0  0  0
```

```
print(D2)
```

```
##    S1 S2 S3 S4 S5
## E1  1  0  1  0  0
## E2  1  0  1  0  0
## E3  1  0  1  1  0
## E4  0  1  0  0  0
## E5  1  1  1  1  1
## E6  0  1  0  0  0
```

Assuming no noise, the discrete data matrix is the transpose of the expected pattern matrix. Given only these two data matrices which are identical, we cannot tell the two models apart.

3.Use the mnem package for this question: Take D1 and D2 from the previous question. For each model, calculate the marginal log-likelihood ratio (network score) given the data by setting the false positive rate to be 5% and the false negative rate to be 1%.

```
#Put phi and theta into the correct format (from matrix to list:)
phi_1 = t(array(c(c(1,0,1,1,1),
                  c(0,1,0,0,1),
                  c(0,0,1,1,1),
                  c(0,0,0,1,1),
                  c(0,0,0,0,1)),
                dim = c(5, 5),
                dimnames = list(c("S1", "S2", "S3", "S4", "S5"),
                                c("S1", "S2", "S3", "S4", "S5"))))

theta_1 = t(array(c(c(0,0,0,0,0,0),
                  c(0,0,0,1,0,1),
                  c(1,1,0,0,0,0),
                  c(0,0,1,0,0,0),
                  c(0,0,0,0,1,0)),
                dim = c(5, 6),
                dimnames = list(c("S1", "S2", "S3", "S4", "S5"),c("E1", "E2", "E3", "E4", "E5", "E6"))))


phi_2 = t(array(c(c(1,0,0,1,1),
                  c(0,1,0,0,1),
                  c(1,0,1,1,1),
                  c(0,0,0,1,1),
                  c(0,0,0,0,1)),
                dim = c(5, 5),
                dimnames = list(c("S1", "S2", "S3", "S4", "S5"),
                                c("S1", "S2", "S3", "S4", "S5"))))
theta_2 = t(array(c(c(1,1,0,0,0,0),
                  c(0,0,0,1,0,1),
                  c(0,0,0,0,0,0),
                  c(0,0,1,0,0,0),
                  c(0,0,0,0,1,0)),
                dim = c(5, 6),
                dimnames = list(c("S1", "S2", "S3", "S4", "S5"),c("E1", "E2", "E3", "E4", "E5", "E6"))))

#Compute networks scores (Log-likelihood ratios)
network_score_1 = scoreAdj(D1,adj = phi_1,method="disc",logtype=exp(1), fpfn=c(0.05,0.01))$score
network_score_2 = scoreAdj(D2,adj = phi_2,method="disc",logtype=exp(1), fpfn=c(0.05,0.01))$score

print(network_score_1)
```

```
## [1] 41.79955
```

```
print(network_score_2)
```

```
## [1] 41.79955
```

9.1

Problem 21: Hidden Markov NEMs

1.Using the definitions for HM-NEMs from the lecture, compute the transition probabilities from $G_t = u$ *to* $G_{t+1} \in v_1, v_2$ for different smoothness parameter $\lambda \in 0.1, ..., 0.9$.

```r
##Without transitive closure:
#Define variables
u = t(array(c(c(1,1,1,0),
              c(0,1,1,1),
              c(0,0,1,1),
              c(0,0,0,1)),
              dim = c(4, 4), dimnames = list(c("S1", "S2", "S3", "S4"),
                                             c("S1", "S2", "S3", "S4"))))
v1 = t(array(c(c(1,1,1,0),
              c(0,1,1,1),
              c(0,0,1,0),
              c(0,0,0,1)),
              dim = c(4, 4), dimnames = list(c("S1", "S2", "S3", "S4"),
                                             c("S1", "S2", "S3", "S4"))))
v2 = t(array(c(c(1,0,0,0),
              c(1,1,1,0),
              c(1,0,1,0),
              c(1,0,0,1)),
              dim = c(4, 4), dimnames = list(c("S1", "S2", "S3", "S4"),
                                             c("S1", "S2", "S3", "S4"))))

lambda = seq(0.1, 0.9, by=0.1)

s_uv1 = sum(u!=v1)
s_uv2 = sum(u!=v2)
W = mnem:::enumerate.models(c("S1","S2","S3","S4"),trans.close = FALSE)
```

```
## Generated 4096 unique models ( out of 4096 )
```

```r
s_uw = array(dim = c(length(W), 1))
for(i in 1:length(W)){
  s_uw[i] = sum(u!=W[[i]])
}

T = array(dim = c(9,2), dimnames = list(lambda,c("v1", "v2")))
C = array(dim = c(9,1), dimnames = list(lambda,c("C")))

for(i in lambda){
  C[as.character(i),] = sum((1-i)^s_uw*i)
  T[as.character(i),"v1"] = (1/C[as.character(i),])*((1-i)^s_uv1)*i
  T[as.character(i),"v2"] = (1/C[as.character(i),])*((1-i)^s_uv2)*i
}

print(T)
```

```
##                v1           v2
## 0.1 0.0004066299 2.160998e-04
## 0.2 0.0006915442 1.812842e-04
## 0.3 0.0012014646 1.413511e-04
## 0.4 0.0021316282 9.945325e-05
## 0.5 0.0038536733 6.021365e-05
## 0.6 0.0070554312 2.889905e-05
## 0.7 0.0128765947 9.387038e-06
```

```
## 0.8 0.0224313310 1.435605e-06
## 0.9 0.0318630818 3.186308e-08
```

```
## With transitives closure:
#Define variables
u_bis = t(array(c(c(1,1,1,1),
              c(0,1,1,1),
              c(0,0,1,1),
              c(0,0,0,1)),
              dim = c(4, 4), dimnames = list(c("S1", "S2", "S3", "S4"),
                                             c("S1", "S2", "S3", "S4"))))
v1_bis = t(array(c(c(1,1,1,1),
              c(0,1,1,1),
              c(0,0,1,0),
              c(0,0,0,1)),
              dim = c(4, 4), dimnames = list(c("S1", "S2", "S3", "S4"),
                                             c("S1", "S2", "S3", "S4"))))
v2_bis = t(array(c(c(1,0,0,0),
              c(1,1,1,0),
              c(1,0,1,0),
              c(1,0,0,1)),
              dim = c(4, 4), dimnames = list(c("S1", "S2", "S3", "S4"),
                                             c("S1", "S2", "S3", "S4"))))
lambda = seq(0.1, 0.9, by=0.1)

s_uv1_bis = sum(u_bis!=v1_bis)
s_uv2_bis = sum(u_bis!=v2_bis)

w_bis = mnem:::enumerate.models(c("S1","S2","S3","S4"),trans.close = TRUE)
```

```
## Generated 355 unique models ( out of 4096 )
```

```
s_uw_bis = array(dim = c(length(w_bis), 1))
for(i in 1:length(w_bis)){
  s_uw_bis[i] = sum(u_bis!=w_bis[[i]])
}

T_bis = array(dim = c(9,2), dimnames = list(lambda,c("v1", "v2")))
C_bis = array(dim = c(9,1), dimnames = list(lambda,c("C")))

for(i in lambda){
  C_bis[as.character(i),] = sum((1-i)^s_uw_bis*i)
  T_bis[as.character(i),"v1"] = (1/C_bis[as.character(i),])*((1-i)^s_uv1_bis)*i
  T_bis[as.character(i),"v2"] = (1/C_bis[as.character(i),])*((1-i)^s_uv2_bis)*i
}

print(T_bis)
```

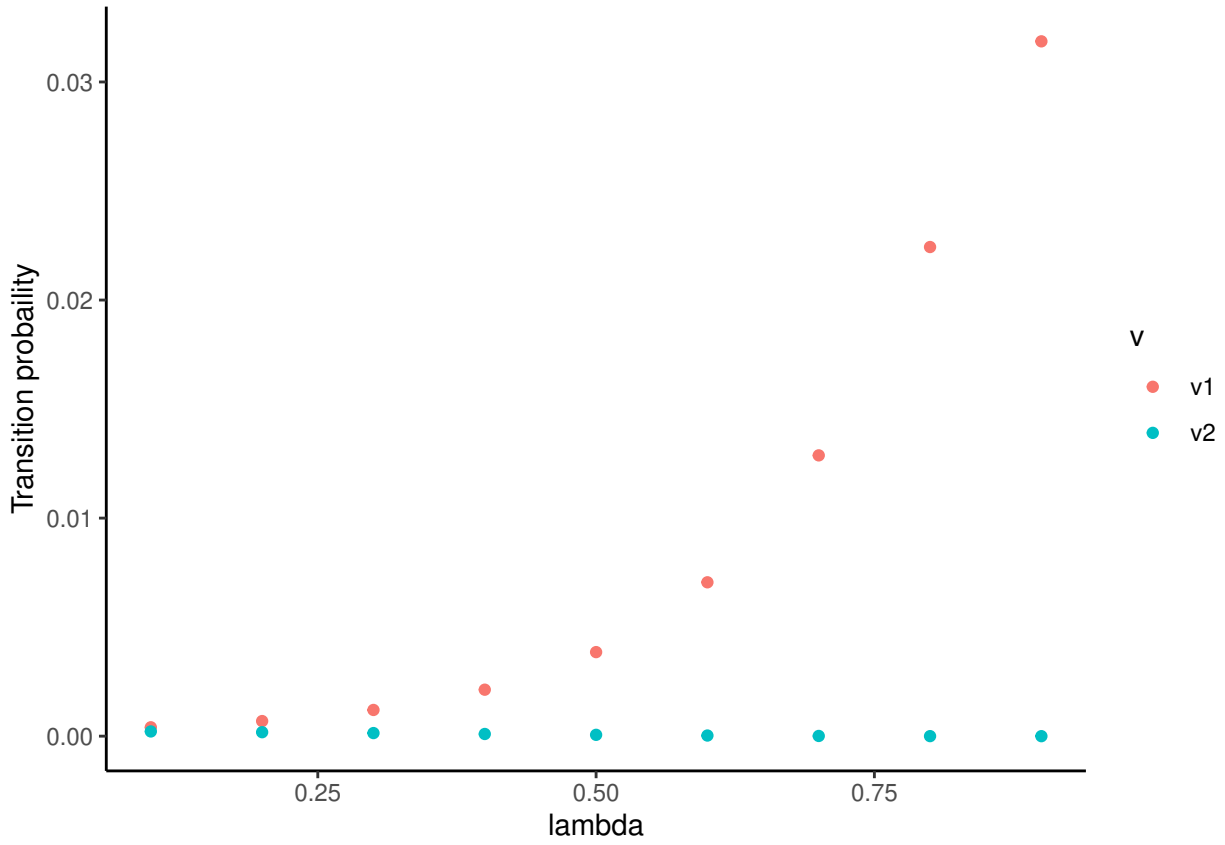
```
##              v1           v2
## 0.1 0.004640039 2.219316e-03
## 0.2 0.007597382 1.593287e-03
## 0.3 0.012270196 1.010503e-03
## 0.4 0.019353227 5.417665e-04
```

```
## 0.5 0.029428247 2.299082e-04
## 0.6 0.042393843 6.945807e-05
## 0.7 0.056328212 1.231898e-05
## 0.8 0.065478114 8.381199e-07
## 0.9 0.056637363 5.663736e-09
```

We computed two different types of transition probabilities: on one hand, by setting the transitive closure parameter of the *mnem* :: *enumerate.models* function to $False$, allowing us to go through all possible existing networks (4096); on the other hand, by setting the same parameter to $True$, thus grouping networks with the same transitive closure together (355 unique models among 4096), given that we consider them equivalent in other NEM analyses.

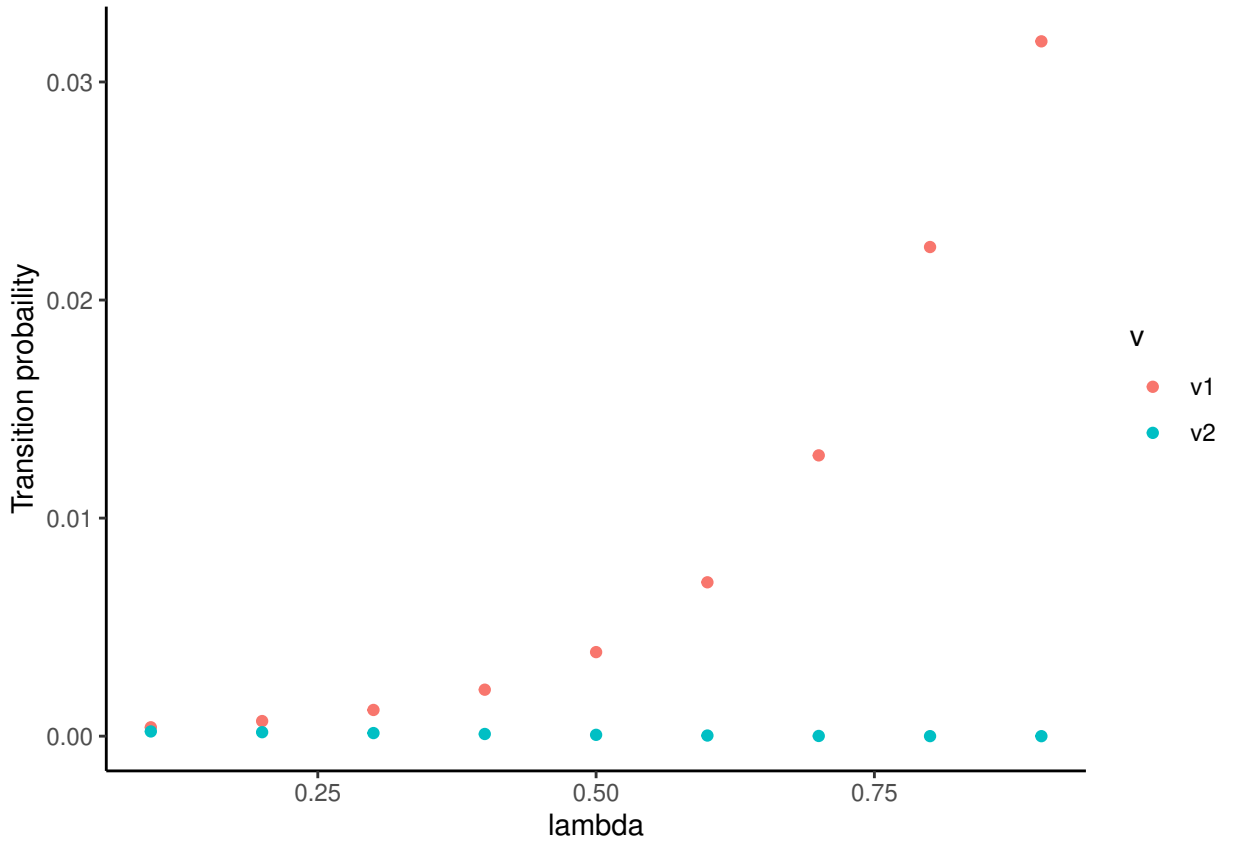
2. Plot the transition probabilities for v_1 and v_2 as a function of λ. Describe the transition probabilities as a function of λ.

```
library(reshape2)
library(ggplot2)

data = data.frame(melt(T))

colnames(data)<-c("lambda","v","T")

plot<-ggplot(data,aes(x=lambda,y=T,color=v))+
  geom_point()+
  theme_classic()+
  ylab("Transition probaility")

print(plot)
```



13.1

```
data_bis = data.frame(melt(T_bis))

colnames(data)<-c("lambda","v","T_bis")

plot<-ggplot(data,aes(x=lambda,y=T_bis,color=v))+
  geom_point()+
  theme_classic()+
  ylab("Transition probaility")

print(plot)
```



As λ increases, it places greater emphasis on the similarity between networks when calculating the transition probabilities, following the formula $P(v|u, \lambda)$. Networks that are less similar to the initial network u are penalized more heavily, resulting in a decrease in their transition probabilities as λ rises. Because network v_1 is more similar to u than v_2, the likelihood of transitioning to v_1 increases as λ increases. Conversely, when λ approaches zero, the impact of network similarity on the transition probabilities becomes negligible, causing the transition probabilities of v_1 and v_2 to become more equal.

14.1

Problem 22: Mixture NEMs

1. Determine the the cellular perturbation map ρ, where $\rho_{ic} = 1$ if cell c is perturbed by a knock-down of S-gene i.

```
rho = array(dim = c(2,4), dimnames = list(c("S1","S2"),
                                          c("C1", "C2", "C3", "C4")))
rho["S1",] = c(1,0,1,0)
rho["S2",] = c(0,1,1,1)
print(rho)
```

```
##    C1 C2 C3 C4
## S1  1  0  1  0
## S2  0  1  1  1
```

2.Assume that {C1,C2} are generated from F_1 and $C3, C4$ are generated from F_2, compute the noiseless log odds matrix R, where $R_{jc} > 0$ means that the perturbation on cell c has an effect on E-gene j:

(a) For each component k, compute the expected effect pattern $(\rho^T \phi_k \theta_k)^T$. Replace all non-zeros by 1.

```
phi_F1 = array(dim = c(2,2), dimnames = list(c("S1","S2"),
                                             c("S1","S2")))
phi_F1["S1",] = c(1,1)
phi_F1["S2",] = c(0,1)

phi_F2 = array(dim = c(2,2), dimnames = list(c("S1","S2"),
                                             c("S1","S2")))
phi_F2["S1",] = c(1,0)
phi_F2["S2",] = c(1,1)

theta_F1 = array(dim = c(2,2), dimnames = list(c("S1","S2"),
                                               c("E1","E2")))
theta_F1["S1",] = c(1,0)
theta_F1["S2",] = c(0,1)

theta_F2 = array(dim = c(2,2), dimnames = list(c("S1","S2"),
                                               c("E1","E2")))
theta_F2["S1",] = c(0,1)
theta_F2["S2",] = c(1,0)

# Expected effect pattern of F1 and F2
EEP_F1 = t(t(rho)%*%phi_F1%*%theta_F1)
EEP_F1[EEP_F1>0] = 1
print(EEP_F1)
```

```
##    C1 C2 C3 C4
## E1  1  0  1  0
## E2  1  1  1  1
```

```
EEP_F2 = t(t(rho)%*%phi_F2%*%theta_F2)
EEP_F2[EEP_F2>0] = 1
print(EEP_F2)
```

```
##    C1 C2 C3 C4
## E1  0  1  1  1
## E2  1  1  1  1
```

(b) Based on the component assignment for each cell, extract the corresponding column from the expected effect patterns computed above and put it into R. Replace all zeros by -1.

```
R= cbind(EEP_F1[,1:2],EEP_F2[,3:4])
R[R==0] = -1
print(R)
```

```
##    C1 C2 C3 C4
## E1  1 -1  1  1
## E2  1  1  1  1
```

3. Take R from the previous question. Given the vector of mixture weights $\pi = (0.44, 0.56)$, calculate the responsibilities Γ. Then, update the mixture weights.

```r
L1 = t(EEP_F1)%*%R
L2 = t(EEP_F2)%*%R

print(L1)
```

```
##    C1 C2 C3 C4
## C1  2  0  2  2
## C2  1  1  1  1
## C3  2  0  2  2
## C4  1  1  1  1
```

```r
print(L2)
```

```
##    C1 C2 C3 C4
## C1  1  1  1  1
## C2  2  0  2  2
## C3  2  0  2  2
## C4  2  0  2  2
```

```r
pi = c(0.44,0.56)

#Calculate the responsibilities
gamma = array(dim = c(2,4), dimnames = list(c("F1","F2"),
                                            c("C1", "C2", "C3", "C4")))
gamma["F1",] = pi[1]*exp(diag(L1))/(pi[1]*exp(diag(L1))+pi[2]*exp(diag(L2)))
gamma["F2",] = pi[2]*exp(diag(L2))/(pi[2]*exp(diag(L2))+pi[1]*exp(diag(L1)))

print(gamma)
```

```
##           C1        C2   C3        C4
## F1 0.6811014 0.6811014 0.44 0.2242338
## F2 0.3188986 0.3188986 0.56 0.7757662
```

```r
# Updated the mixture weights
pi[1] = sum(gamma["F1",])/(sum(gamma["F1",])+sum(gamma["F2",]))
pi[2] = sum(gamma["F2",])/(sum(gamma["F1",])+sum(gamma["F2",]))
print("Updated mixture weights :")
```

```
## [1] "Updated mixture weights :"
```

```r
print(pi)
```

```
## [1] 0.5066091 0.4933909
```

```r
library(rmarkdown)
render("project8.Rmd", pdf_document(TRUE), "Project8_Indilewitsch_Toidze_Houhamdi_Pudziuvelyte.pdf")
```

Index of comments