Homework 5

Naga Kartheek Peddisetty, 50538422

05/10/2024

Question 1) Consider random variables X1,X2,X3,X4. In each of the following cases draw a graph that has the given independence relations:

(a) X1 ⊥ X3|X2 and X2 ⊥ X4|X3.

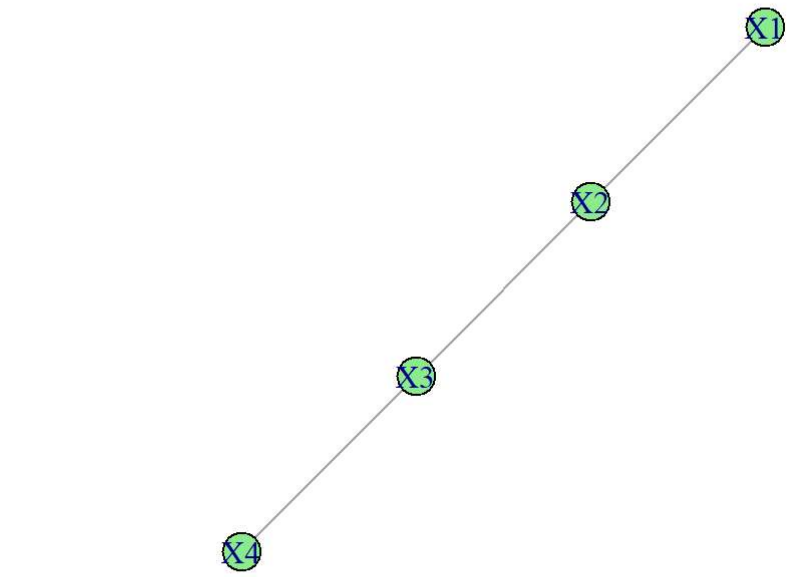
```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

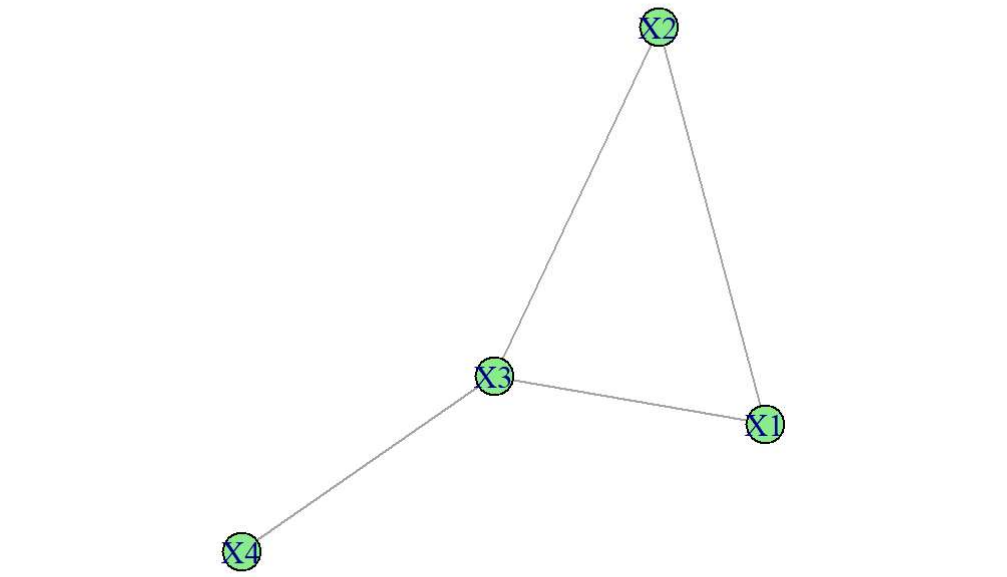
```
## The following object is masked from 'package:base':
##
##     union
```

```
a <- c("X1","X2")
b <- c("X2","X3")
d <- c("X3","X4")

gra <- graph(c(a, b, d), directed = FALSE)
plot(gra, layout = layout_with_kk, vertex.color = "lightgreen", main = "Undirected graphical mod
el with given independencies")
```



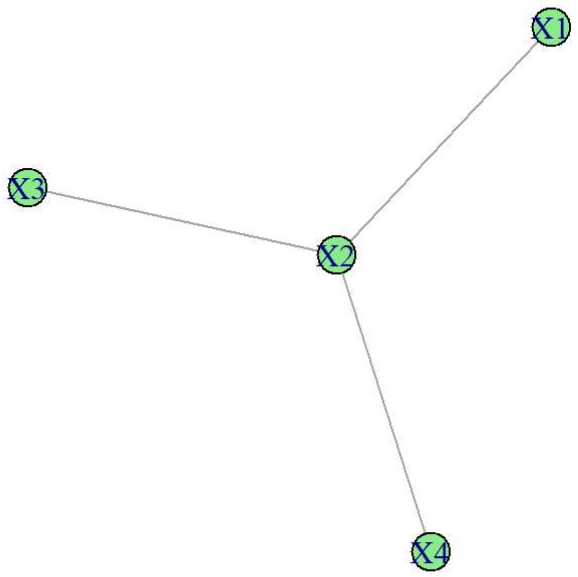
(b) X1 ⊥ X4|X2,X3 and X2 ⊥ X4|X1,X3.

```
a <- c("X1","X2","X3")
b <- c("X2","X1","X3")
d <- c("X3","X4")

gra1 <- graph(c(a, b, d), directed = FALSE)
plot(gra1, layout = layout_with_kk, vertex.color = "lightgreen", main = "Undirected graphical mo
del with given independencies")
```



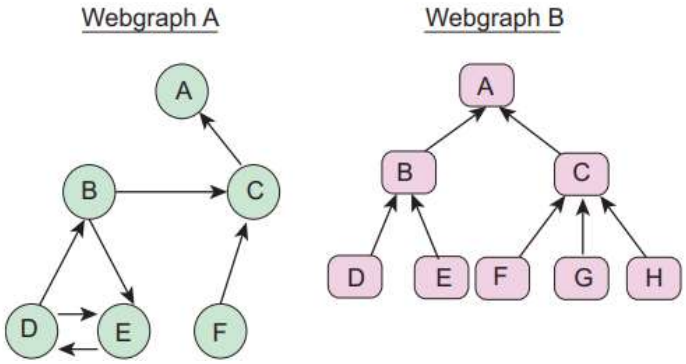
(c) X1 ⊥ X4|X2,X3, X1 ⊥ X3|X2,X4 and X3 ⊥ X4|X1,X2.

```
a <- c("X1","X2")
b <- c("X2","X3","X2","X4")

gra2 <- graph(c(a, b), directed = FALSE)
plot(gra2, layout = layout_with_kk, vertex.color = "lightgreen", main = "Undirected graphical mo
del with given independencies")
```



Question 2) Consider the following webgraphs A and B.

```
knitr::include_graphics("Q2.png")
```

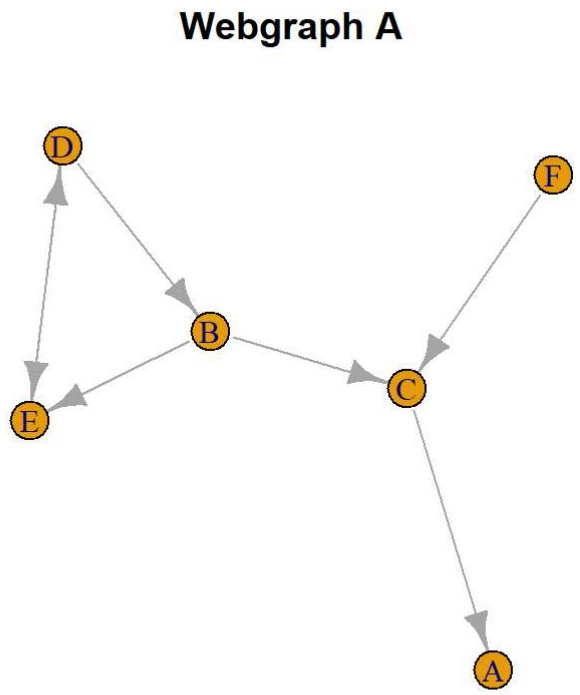


a) Compute the PageRank vector of Webgraph A for damping constants p = 0.05, 0.25, 0.50, 0.75, and 0.95. How sensitive is the PageRank vector, and overall ranking of importance, to the damping constant? Does the relative ranking of importance according to PageRank support your intuition?

```
library(igraph)

vertices_A <- data.frame(names = c("A", "B", "C", "D", "E", "F"))
edges_A <- data.frame(
           from = c("F", "C", "B", "B", "D", "D", "E"),
           to = c("C", "A", "C", "E", "B", "E", "D"))
gra_A <- graph.data.frame(edges_A, directed = TRUE, vertices = vertices_A)
```

```
## Warning: `graph.data.frame()` was deprecated in igraph 2.0.0.
## i Please use `graph_from_data_frame()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
plot(gra_A, layout = layout_with_kk, main = "Webgraph A")
```




```
damping_factors <- c(0.05, 0.25, 0.50, 0.75, 0.95)

for (i in damping_factors){
  val <- page.rank(gra_A, damping = i)$vector
  print(paste("for p = ", i))
  print(val)
}
```

```
## Warning: `page.rank()` was deprecated in igraph 2.0.0.
## i Please use `page_rank()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## [1] "for p =  0.05"
##         A         B         C         D         E         F
## 0.1683271 0.1639395 0.1718214 0.1681380 0.1680380 0.1597361
## [1] "for p =  0.25"
##         A         B         C         D         E         F
## 0.1786588 0.1544288 0.1848587 0.1758772 0.1737324 0.1324441
## [1] "for p =  0.5"
##          A          B          C          D          E          F
## 0.19227231 0.14719411 0.18583257 0.19135235 0.18399264 0.09935603
## [1] "for p =  0.75"
##          A          B          C          D          E          F
## 0.19399617 0.14778661 0.17077331 0.21832113 0.20320659 0.06591619
## [1] "for p =  0.95"
##          A          B          C          D          E          F
## 0.17305017 0.15761096 0.14454445 0.25658531 0.23247617 0.03573294
```

• Let's compare the PageRank values for each vertex at the lowest (0.05) and highest (0.95) damping constant to see how much they change.

– Vertex A: From 0.1683 to 0.1730 (change: +0.0047)

– Vertex B: From 0.1639 to 0.1576 (change: -0.0063)

– Vertex C: From 0.1718 to 0.1445 (change: -0.0273)

– Vertex D: From 0.1681 to 0.2565 (change: +0.0884)

– Vertex E: From 0.1680 to 0.2324 (change: +0.0644)

– Vertex F: From 0.1597 to 0.0357 (change: -0.1240)

• These changes indicate that there is indeed sensitivity in the PageRank vector to the damping constant, but the level of sensitivity varies per vertex. Some vertices (like D and E) significantly increase in their ranking with a higher damping constant, while others (like C and F) decrease. Vertices A & B appears the least sensitive with a small change.

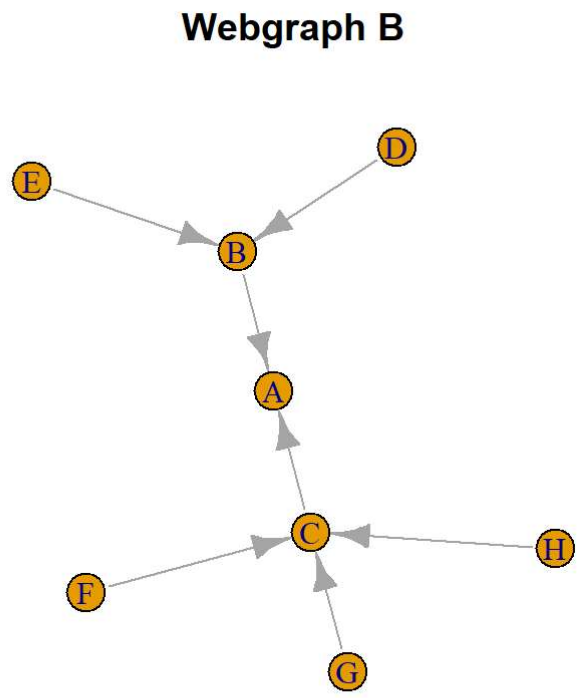
• A lower damping constant distributes the importance more uniformly across the nodes, whereas a higher damping constant leads to a higher concentration of importance on fewer nodes.

• In terms of sensitivity, the PageRank vector and overall ranking of importance are sensitive to the damping constant, and it can significantly change the interpretation of the importance of each node in the network.

• Assuming my intuition is "more incoming edges, more importance," PageRank algorithm generally aligns with that intuition. PageRank algorithm indeed rewards nodes with more incoming links. However, it not only considers the quantity of the incoming links but also their quality. This means that a node can be highly ranked not just by having a lot of incoming links, but also by being linked from other highly ranked nodes. This factor is not captured in a simpler "more incoming edges, more importance" intuition.

b) Compute the PageRank vector of Webgraph B for damping constant p = 0.15. Interpret your results in terms of the relationship between the number of incoming links that each node has. Does the relative ranking of importance according to PageRank support your intuition?

```
vertices_B <- data.frame(names = c("A", "B", "C", "D", "E", "F", "G", "H"))
edges_B <- data.frame(
           from = c("B", "C", "D", "E", "F", "G", "H"),
           to = c("A", "A", "B", "B", "C", "C", "C"))
gra_B <- graph.data.frame(edges_B, directed = TRUE, vertices = vertices_B)

plot(gra_B, layout = layout_with_kk, main = "Webgraph B")
```




```
pg_B <- page.rank(gra_B, damping = 0.15)
pg_B$vector
```

```
##         A         B         C         D         E         F         G         H
## 0.1541610 0.1418827 0.1582538 0.1091405 0.1091405 0.1091405 0.1091405 0.1091405
```

When damping factor is set to 0.15

• Node A (0.1541): Node A has the second highest PageRank score, indicating that it is considered as one of the most important according to the PageRank algorithm. This suggests that although Node A has two (B & C) incoming links compared to some other nodes, the quality and significance of the incoming links (nodes with higher PageRank scores) contribute to its importance.

• Nodes B, C (0.1418 and 0.1582): Nodes B and C also have relatively high PageRank scores, indicating their importance. Node C has the highest PageRank score among all nodes, likely due to its strong connectivity and receiving three incoming links from nodes like F, G, H.

• Nodes D, E, F, G, H (0.1091 each): These nodes have lower PageRank scores compared to A, B, and C. They are not receiving any incoming links, resulting in lower importance scores.

• According to PageRank algorithm the webpages that receive more incoming links tend to be more significant. It's similar to a popularity contest - the more votes or "links" a page gets from others, the higher it's considered in importance. But it's not just about quantity. 'Quality' also counts. If a link comes from a well-established and popular webpage, it generally gives a stronger boost to the PageRank score, raising the relevance of the linked webpage in the eyes of the algorithm.

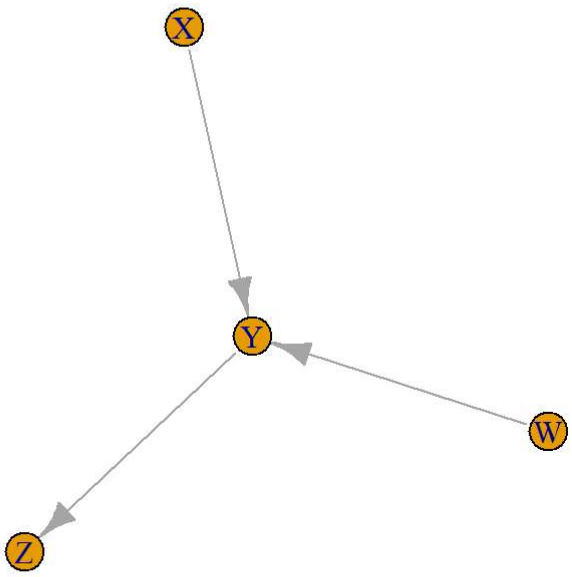
• However, it's crucial to keep in mind that the PageRank is just part of a bigger picture. The way that all the webpages are connected together in what's known as a webgraph can also influence rankings. Certain characteristics of this network, like the way pages are linked together and other factors, can have a substantial impact on how the ranks are distributed. So while more and better links can boost a webpage ranking, there's a complex web of other elements at play that can shape the final results.

Question 3) Specify the structure of a Bayesian Network that contains four nodes {W,X,Y,Z} and has satisfies the following set of independencies.

```
knitr::include_graphics("Q3.png")
```

$$
\begin{aligned}
W &\perp X \\
W &\not\perp Z \mid X \\
Z &\perp W \mid Y \\
W &\not\perp Y \\
X &\not\perp Y \\
W &\not\perp X \mid Z \\
X &\perp Z \mid W, Y
\end{aligned}
$$

```
library(igraph)

vertices <- data.frame(names = c("W", "X", "Y", "Z"))
edge <- data.frame(
           from = c("W", "X", "Y"),
           to = c("Y", "Y", "Z"))
gr <- graph.data.frame(edge, directed = TRUE, vertices = vertices)

plot(gr, layout = layout_with_kk, main = "Bayesian Network with given independencies")
```



1. W is independent of X. Because they are d-separated by v-structure node Y which is not conditioned and it's trail Z is also not conditioned.
2. W is not independent of Z given X. Because there is dependency flow from W to Z via node Y which is not conditioned.
3. Z is independent of W given Y. When Y is conditioned then dependency flow from W to Z is blocked and hence Z is independent of W when conditioned on Y.
4. W is not independent of Y since they are directly connected.
5. X is not independent of Y since they are directly connected.
6. W is not independent of X given Z. Since Y is a v-structured node between W and X. When node Y or it's trail Z is conditioned then X and W will no more be independent.
7. X is independent of Z given W and Y. Node X and node Z are d-separated by node Y and when Y and W are conditioned then X and Z will be independent.

Question 4) Data released from the US department of Commerce, Bureau of the Census is available in R. data(state) ?state.

```
library(kohonen)

data(state)
dat <- data.frame(state.x77)
dim(dat)
```

```
## [1] 50  8
```

```
summary(dat)
```

```
##    Population        Income       Illiteracy       Life.Exp
##  Min.   :  365   Min.   :3098   Min.   :0.500   Min.   :67.96
##  1st Qu.: 1080   1st Qu.:3993   1st Qu.:0.625   1st Qu.:70.12
##  Median : 2838   Median :4519   Median :0.950   Median :70.67
##  Mean   : 4246   Mean   :4436   Mean   :1.170   Mean   :70.88
##  3rd Qu.: 4968   3rd Qu.:4814   3rd Qu.:1.575   3rd Qu.:71.89
##  Max.   :21198   Max.   :6315   Max.   :2.800   Max.   :73.60
##      Murder          HS.Grad          Frost             Area
##  Min.   : 1.400   Min.   :37.80   Min.   :  0.00   Min.   :  1049
##  1st Qu.: 4.350   1st Qu.:48.05   1st Qu.: 66.25   1st Qu.: 36985
##  Median : 6.850   Median :53.25   Median :114.50   Median : 54277
##  Mean   : 7.378   Mean   :53.11   Mean   :104.46   Mean   : 70736
##  3rd Qu.:10.675   3rd Qu.:59.15   3rd Qu.:139.75   3rd Qu.: 81163
##  Max.   :15.100   Max.   :67.30   Max.   :188.00   Max.   :566432
```

```
str(dat)
```

```
## 'data.frame':    50 obs. of  8 variables:
##  $ Population: num  3615 365 2212 2110 21198 ...
##  $ Income    : num  3624 6315 4530 3378 5114 ...
##  $ Illiteracy: num  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2 ...
##  $ Life.Exp  : num  69 69.3 70.5 70.7 71.7 ...
##  $ Murder    : num  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10.7 13.9 ...
##  $ HS.Grad   : num  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 52.6 40.6 ...
##  $ Frost     : num  20 152 15 65 20 166 139 103 11 60 ...
##  $ Area      : num  50708 566432 113417 51945 156361 ...
```

```
head(dat)
```

```
##            Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
## Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
## Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
## Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
## Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
## California      21198   5114        1.1    71.71   10.3    62.6    20 156361
## Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
```

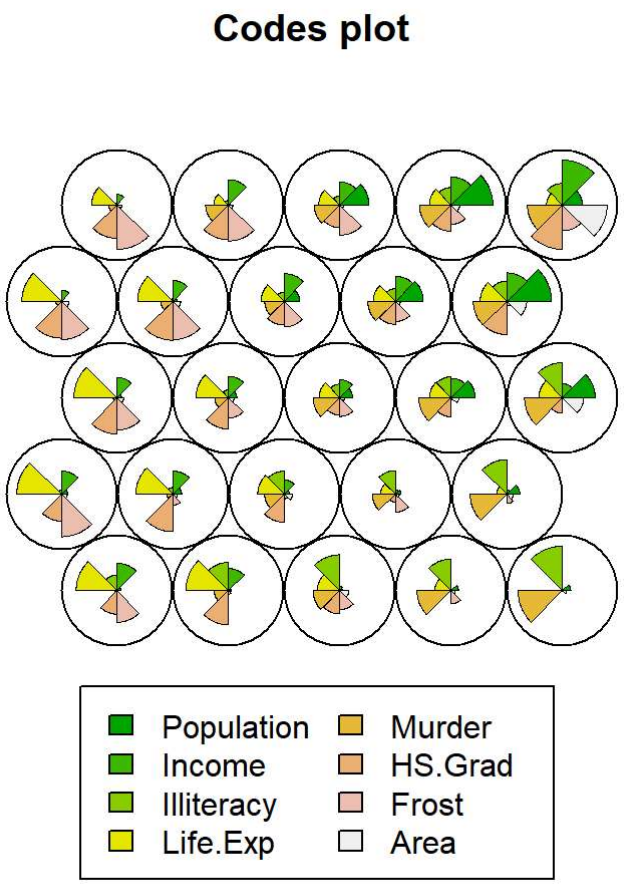
```
dat_scaled <- scale(dat)
```

Build a Gaussian Graphical Model using the Graphical Lasso for the 8 predictors (Population, Income, Illiteracy, Life Exp, Murder, HS Grad, Frost, Area) using a range of penalties. What do you find for different penalties, and how does it compliment (and/or contradict) a model fit with SOM?

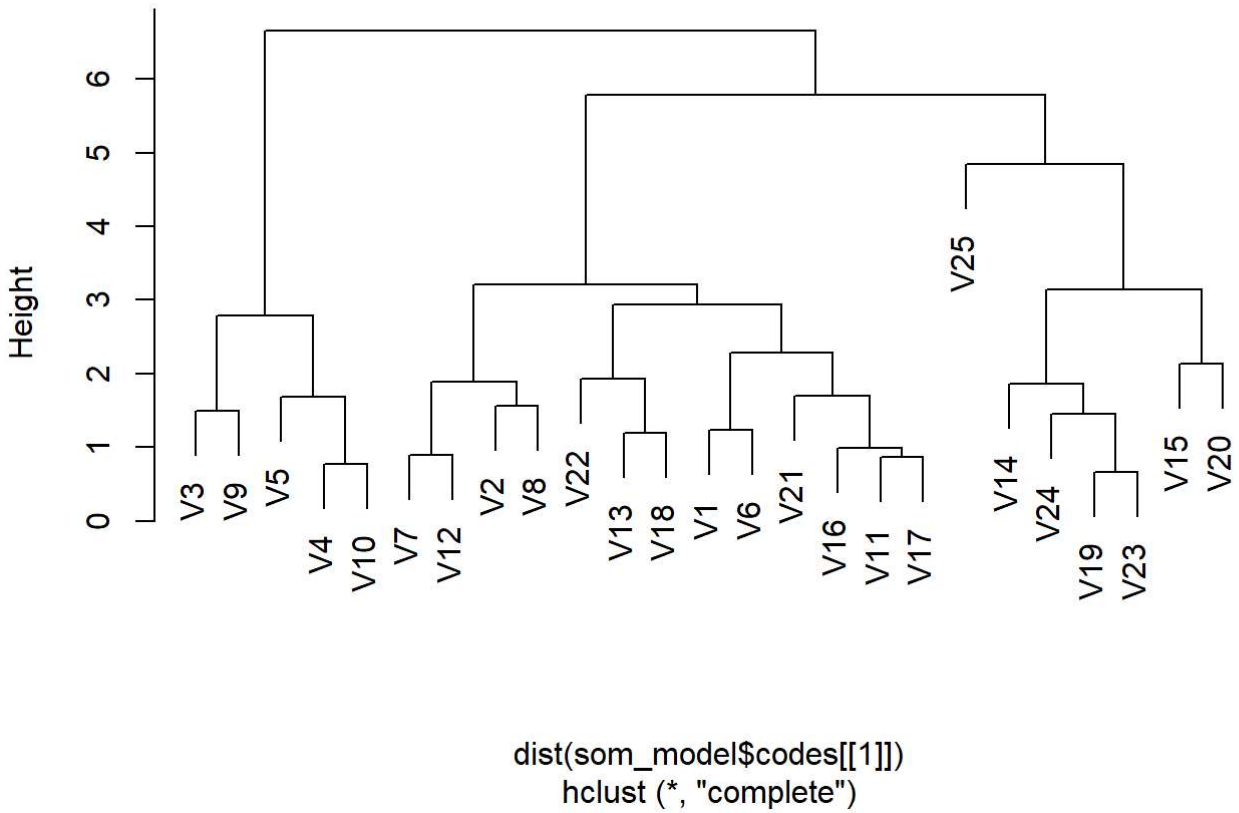
```
set.seed(123)
# Define the SOM grid dimensions
grid_size <- somgrid(xdim = 5, ydim = 5, topo = "hexagonal")
# Train the SOM model
som_model <- som(dat_scaled, grid = grid_size, rlen = 100)
som_model
```

```
## SOM of size 5x5 with a hexagonal topology.
## Training data included.
```

```
plot(som_model)
```




```
# plot the hierarchical clustering
hc <- hclust(dist(som_model$codes[[1]]))
plot(hc)
```




```
# get SOM clusters
cluster_assignments <- cutree(hc, h = 6)
cluster_assignments
```

```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20
##   1   1   2   2   2   1   1   1   2   2   1   1   1   1   1   1   1   1   1   1
## V21 V22 V23 V24 V25
##   1   1   1   1   1
```

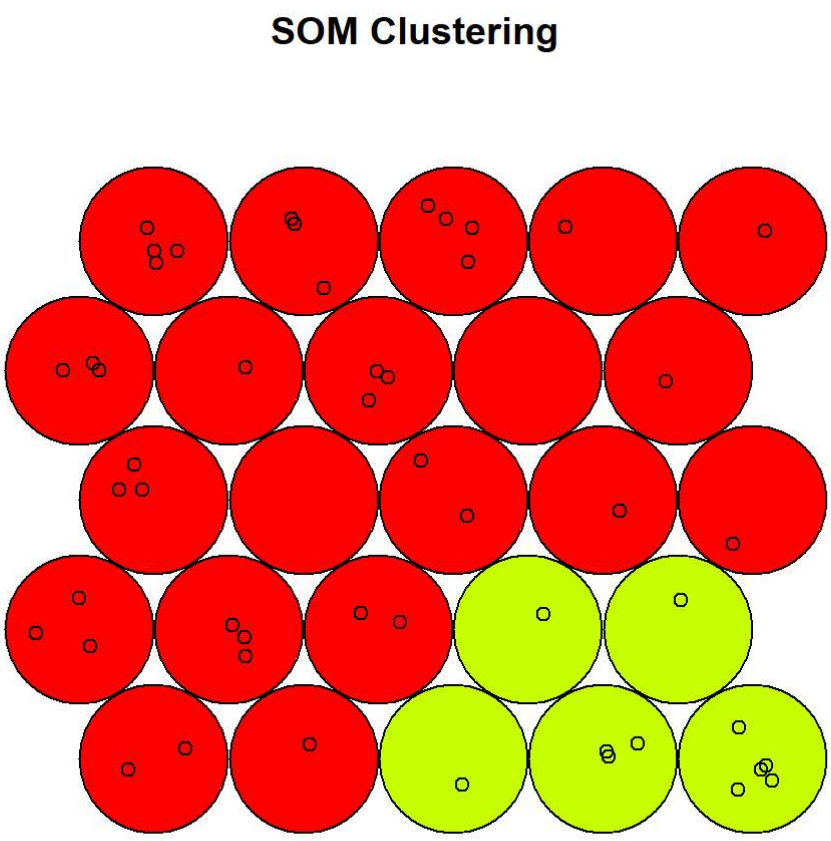
```
# Add the cluster assignments to the original data
dat <- cbind(dat, Cluster = cluster_assignments)
dim(dat)
```

```
## [1] 50  9
```

```
head(dat)
```

```
##            Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
## Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
## Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
## Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
## Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
## California      21198   5114        1.1    71.71   10.3    62.6    20 156361
## Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
##            Cluster
## Alabama          1
## Alaska           1
## Arizona          2
## Arkansas         2
## California       2
## Colorado         1
```

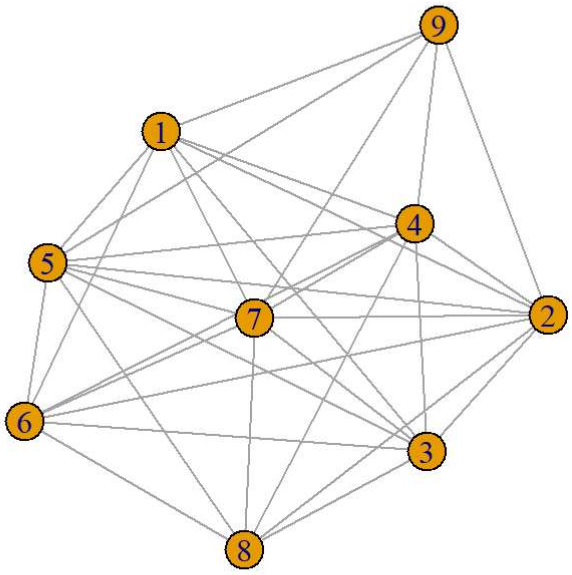
```
# Plot the SOM with cluster colors
plot(som_model, type = "mapping", bgcol = rainbow(5)[cluster_assignments], main = "SOM Clusterin
g")
```



```r
library(glasso)
library(igraph)

# let's take 5 different L1 penalties
p1 <- 0.0001
p2 <- 0.01
p3 <- 0.1
p4 <- 0.5
p5 <- 0.9

# Compute the empirical covariance matrix
cov_mat <- cov(scale(dat))
cov_mat
```

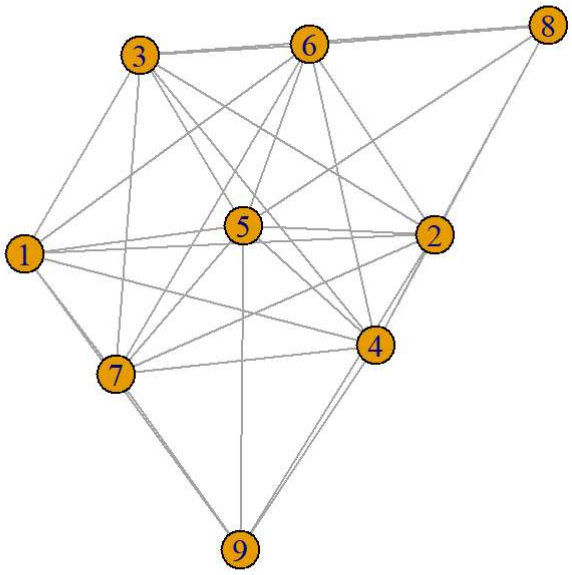
```
##             Population     Income  Illiteracy    Life.Exp     Murder
## Population  1.00000000  0.2082276  0.10762237 -0.06805195  0.3436428
## Income      0.20822756  1.0000000 -0.43707519  0.34025534 -0.2300776
## Illiteracy  0.10762237 -0.4370752  1.00000000 -0.58847793  0.7029752
## Life.Exp   -0.06805195  0.3402553 -0.58847793  1.00000000 -0.7808458
## Murder      0.34364275 -0.2300776  0.70297520 -0.78084575  1.0000000
## HS.Grad    -0.09848975  0.6199323 -0.65718861  0.58221620 -0.4879710
## Frost      -0.33215245  0.2262822 -0.67194697  0.26206801 -0.5388834
## Area        0.02254384  0.3633154  0.07726113 -0.10733194  0.2283902
## Cluster     0.18520592  0.1549415  0.02485884 -0.07058460  0.1069932
##                 HS.Grad       Frost        Area      Cluster
## Population -0.098489748 -0.33215245  0.02254384  0.185205925
## Income      0.619932323  0.22628218  0.36331544  0.154941473
## Illiteracy -0.657188609 -0.67194697  0.07726113  0.024858843
## Life.Exp    0.582216204  0.26206801 -0.10733194 -0.070584598
## Murder     -0.487971022 -0.53888344  0.22839021  0.106993200
## HS.Grad     1.000000000  0.36677970  0.33354187  0.009504966
## Frost       0.366779702  1.00000000  0.05922910 -0.084145617
## Area        0.333541871  0.05922910  1.00000000 -0.021775733
## Cluster     0.009504966 -0.08414562 -0.02177573  1.000000000
```

```r
# writing function to plot the ggm
ggm <- function(cov_mat, penalty) {
  result <- glasso(cov_mat, rho = penalty)
  # create adjacency matrix
  adj_matrix <- ifelse(abs(result$w) > 0.05, 1, 0)
  # Create a graph from the adjacency matrix
  g <- graph_from_adjacency_matrix(adj_matrix, mode = "undirected", diag = FALSE)
  # Plot the graph
  plot(g)
  #print edge count
  print(paste("Number of edges:",ecount(g)))
}

ggm(cov_mat, p1)
```



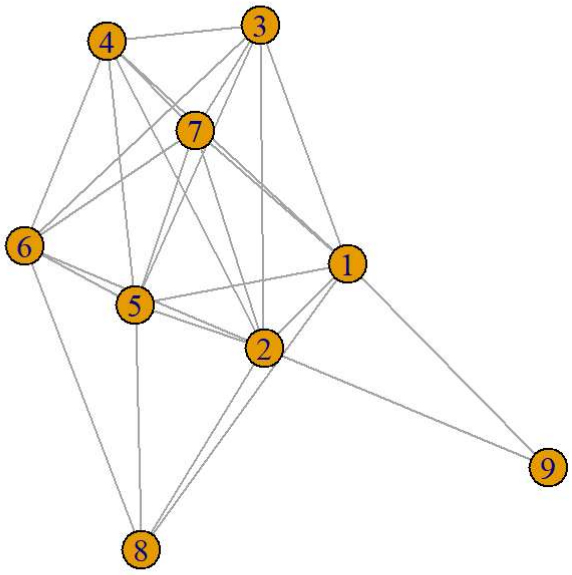

```
## [1] "Number of edges: 32"
```

```
ggm(cov_mat, p2)
```



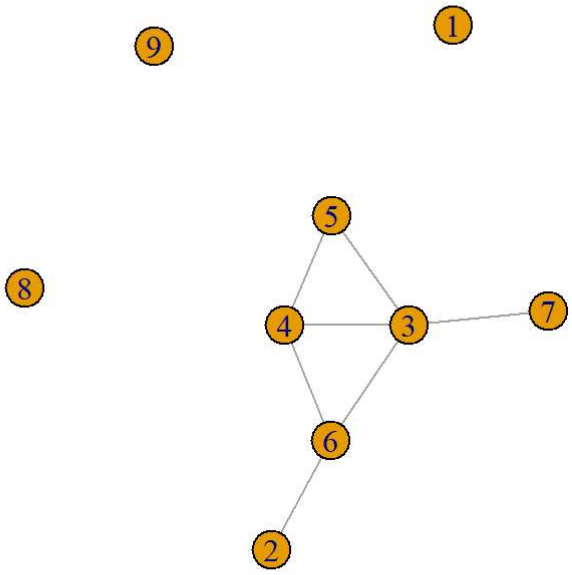

```
## [1] "Number of edges: 31"
```

```
ggm(cov_mat, p3)
```



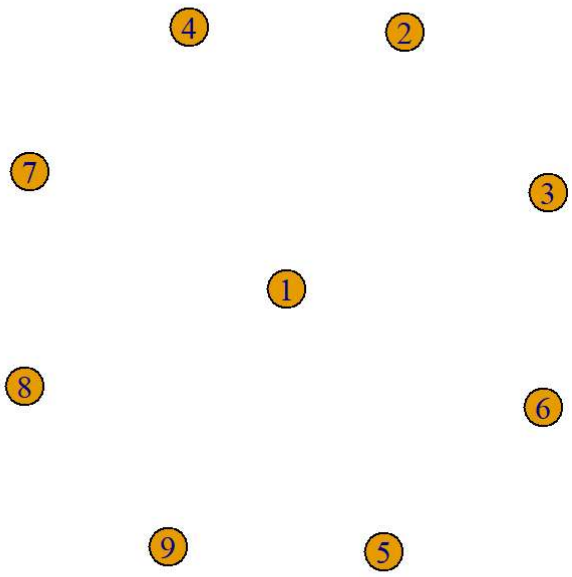

```
## [1] "Number of edges: 26"
```

```
ggm(cov_mat, p4)
```




```
## [1] "Number of edges: 7"
```

```
ggm(cov_mat, p5)
```




```
## [1] "Number of edges: 0"
```

• As the L1 penalities (ρ) in the Graphical Lasso go up, the solution becomes more sparse. In the context of the Gaussian Graphical Model (GGM), this means that more entries in the estimated precision (inverse covariance) matrix become zero.

• From the above graph plots it is clear that as we increase the regularization penalty (ρ):

– The number of non-zero entries in the precision matrix decreases.

– The number of edges in the graph decreases.

– The graph becomes sparser.

• As the penalty gets closer to 1 we can see that number of edges in the graph become 0.

• It's important to choose a good ρ, as setting it too high will lead to an overly simplified model (underfitting), while setting it too low may fail to induce the necessary sparsity to clarify the underlying network structure (overfitting).

Question 5) Consider the "cad1" data set in the package gRbase. These observations are from individuals in the Danish Heart Clinic.

```
library(gRbase)
```

```
##
## Attaching package: 'gRbase'
```

```
## The following objects are masked from 'package:igraph':
##
##     edges, is_dag, topo_sort
```

```
data(cad1)
dim(cad1)
```

```
## [1] 236  14
```

```
str(cad1)
```

```
## 'data.frame':    236 obs. of  14 variables:
##  $ Sex        : Factor w/ 2 levels "Female","Male": 2 2 1 2 2 2 2 2 1 2 ...
##  $ AngPec     : Factor w/ 3 levels "Atypical","None",..: 2 1 2 2 2 2 2 2 2 1 ...
##  $ AMI        : Factor w/ 2 levels "Definite","NotCertain": 2 2 1 2 2 2 2 2 2 2 ...
##  $ QWave      : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 2 2 1 1 ...
##  $ QWavecode  : Factor w/ 2 levels "Nonusable","Usable": 2 2 2 2 2 2 2 2 1 2 ...
##  $ STcode     : Factor w/ 2 levels "Nonusable","Usable": 2 2 2 1 1 1 1 1 1 2 ...
##  $ STchange   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 2 ...
##  $ SuffHeartF : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Hypertrophi: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Hyperchol  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Smoker     : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Inherit    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Heartfail  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ CAD        : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(cad1)
```

```
##      Sex           AngPec            AMI      QWave         QWavecode
##  Female: 47   Atypical: 30   Definite  : 63   No :153   Nonusable: 13
##  Male  :189   None    : 85   NotCertain:173   Yes: 83   Usable   :223
##               Typical :121
##        STcode    STchange  SuffHeartF Hypertrophi Hyperchol Smoker    Inherit
##  Nonusable: 79   No :133   No :167    No :172     No :108   No : 51   No :162
##  Usable   :157   Yes:103   Yes: 69    Yes: 64     Yes:128   Yes:185   Yes: 74
##
##  Heartfail  CAD
##  No :177   No :129
##  Yes: 59   Yes:107
##
```

```
colSums(is.na(cad1))
```

```
##         Sex      AngPec         AMI       QWave   QWavecode      STcode
##           0           0           0           0           0           0
##    STchange  SuffHeartF Hypertrophi   Hyperchol      Smoker     Inherit
##           0           0           0           0           0           0
##   Heartfail         CAD
##           0           0
```

```
head(cad1)
```

```
##      Sex   AngPec        AMI QWave QWavecode    STcode STchange SuffHeartF
## 1   Male     None NotCertain    No    Usable    Usable       No         No
## 2   Male Atypical NotCertain    No    Usable    Usable       No         No
## 3 Female     None   Definite    No    Usable    Usable       No         No
## 4   Male     None NotCertain    No    Usable Nonusable       No         No
## 5   Male     None NotCertain    No    Usable Nonusable       No         No
## 6   Male     None NotCertain    No    Usable Nonusable       No         No
##   Hypertrophi Hyperchol Smoker Inherit Heartfail CAD
## 1          No        No     No      No        No  No
## 2          No        No     No      No        No  No
## 3          No        No     No      No        No  No
## 4          No        No     No      No        No  No
## 5          No        No     No      No        No  No
## 6          No        No     No      No        No  No
```

```
cad1_data <- cad1
```

a) Learn a Bayesian Network using a structural learning knowledge, and prior knowledge obtained through the definitions of the variables in the helpfiles. You do not have to use all of the variables. Make sure to detail your network construction process.

```
library(bnlearn)
```

```
##
## Attaching package: 'bnlearn'
```

```
## The following objects are masked from 'package:gRbase':
##
##     ancestors, children, parents
```

```
## The following objects are masked from 'package:igraph':
##
##     as.igraph, compare, degree, subgraph
```

```
library(Rgraphviz)
```

```
## Loading required package: graph
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following object is masked from 'package:bnlearn':
##
##     score
```

```
## The following objects are masked from 'package:igraph':
##
##     normalize, path, union
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, table,
##     tapply, union, unique, unsplit, which.max, which.min
```

```
##
## Attaching package: 'graph'
```

```
## The following objects are masked from 'package:bnlearn':
##
##     degree, nodes, nodes<-
```

```
## The following objects are masked from 'package:gRbase':
##
##     addEdge, adj, connComp, edges, nodes, removeEdge, subGraph
```

```
## The following objects are masked from 'package:igraph':
##
##     degree, edges, intersection
```

```
## Loading required package: grid
```

```
# Learn the structure of the Bayesian network using hill climbing algorithm
cad_bn <- hc(cad1_data)
cad_bn
```

```
##
##   Bayesian network learned via Score-based methods
##
##   model:
##    [AngPec][STcode|AngPec][SuffHeartF|STcode][Hypertrophi|SuffHeartF]
##    [QWavecode|STcode:Hypertrophi][Heartfail|STcode:Hypertrophi]
##    [CAD|AngPec:Heartfail][Sex|CAD][AMI|CAD][STchange|STcode:Hypertrophi:CAD]
##    [Hyperchol|CAD][Smoker|CAD][Inherit|CAD][QWave|AMI:CAD]
##   nodes:                                 14
##   arcs:                                  19
##     undirected arcs:                     0
##     directed arcs:                       19
##   average markov blanket size:           3.29
##   average neighbourhood size:            2.71
##   average branching factor:              1.36
##
##   learning algorithm:                    Hill-Climbing
##   score:                                 BIC (disc.)
##   penalization coefficient:              2.731916
##   tests used in the learning procedure:  338
##   optimized:                             TRUE
```

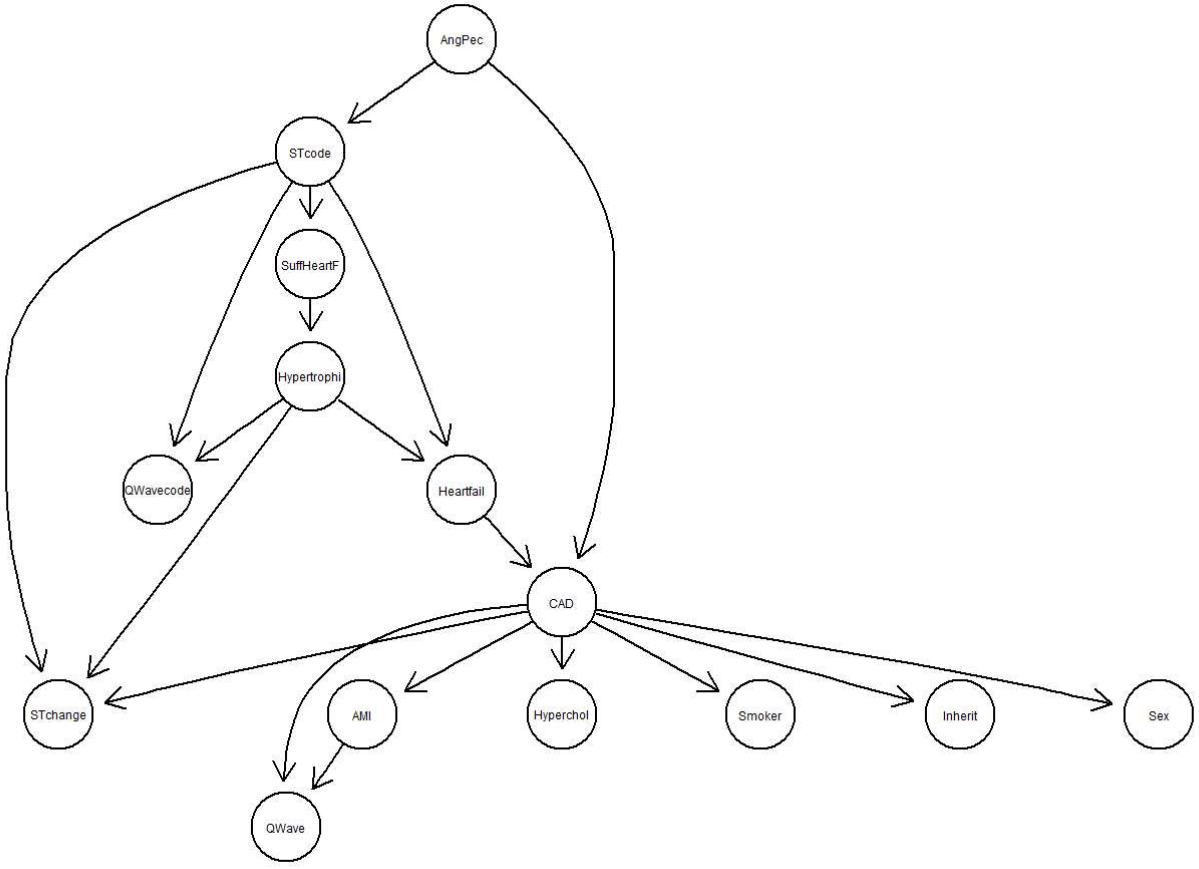
```
# creating adjacency matrix
adj_mat <- amat(cad_bn)
adj_mat
```

```
##             Sex AngPec AMI QWave QWavecode STcode STchange SuffHeartF
## Sex           0      0   0     0         0      0        0          0
## AngPec        0      0   0     0         0      1        0          0
## AMI           0      0   0     1         0      0        0          0
## QWave         0      0   0     0         0      0        0          0
## QWavecode     0      0   0     0         0      0        0          0
## STcode        0      0   0     0         1      0        1          1
## STchange      0      0   0     0         0      0        0          0
## SuffHeartF    0      0   0     0         0      0        0          0
## Hypertrophi   0      0   0     0         1      0        1          0
## Hyperchol     0      0   0     0         0      0        0          0
## Smoker        0      0   0     0         0      0        0          0
## Inherit       0      0   0     0         0      0        0          0
## Heartfail     0      0   0     0         0      0        0          0
## CAD           1      0   1     1         0      0        1          0
##             Hypertrophi Hyperchol Smoker Inherit Heartfail CAD
## Sex                    0         0      0       0         0   0
## AngPec                 0         0      0       0         0   1
## AMI                    0         0      0       0         0   0
## QWave                  0         0      0       0         0   0
## QWavecode              0         0      0       0         0   0
## STcode                 0         0      0       0         1   0
## STchange               0         0      0       0         0   0
## SuffHeartF             1         0      0       0         0   0
## Hypertrophi            0         0      0       0         1   0
## Hyperchol              0         0      0       0         0   0
## Smoker                 0         0      0       0         0   0
## Inherit                0         0      0       0         0   0
## Heartfail              0         0      0       0         0   1
## CAD                    0         1      1       1         0   0
```

```
# grapnel object encode the graph as a list in which each element refer to one of the nodes in t
he graph and contains a vector with its children;
net <- as(adj_mat, "graphNEL")
net
```

```
## A graphNEL graph with directed edges
## Number of Nodes = 14
## Number of Edges = 19
```

```
plot(net)
```



• hc is used for hill climbing structure learning. It searches for the optimal network structure that best fits the given data (cad1_data).

• cad_bn displays the learned Bayesian network structure, showing the nodes and their connections based on the hill climbing algorithm's output.

• amat creates an adjacency matrix (adj_mat) from the learned Bayesian network structure (cad_bn). This matrix represents the conditional dependencies between nodes in the network.

• as() converts the adjacency matrix (adj_mat) into a graphNEL object (net). This object is a representation of the Bayesian network graph.

b) Construct the above network in R, and infer the Conditional Probability Tables using the cad1 data. Identify any d-separations in the graph.

```
# Build a model string from a Bayesian network
cad_model_string <- modelstring(cad_bn)
cad_model_string
```

```
## [1] "[AngPec][STcode|AngPec][SuffHeartF|STcode][Hypertrophi|SuffHeartF][QWavecode|STcode:Hype
rtrophi][Heartfail|STcode:Hypertrophi][CAD|AngPec:Heartfail][Sex|CAD][AMI|CAD][STchange|STcode:H
ypertrophi:CAD][Hyperchol|CAD][Smoker|CAD][Inherit|CAD][QWave|AMI:CAD]"
```

```
# model to network
bn <- model2network(cad_model_string)
bn
```

```
##
##   Random/Generated Bayesian network
##
##   model:
##    [AngPec][STcode|AngPec][SuffHeartF|STcode][Hypertrophi|SuffHeartF]
##    [Heartfail|Hypertrophi:STcode][QWavecode|Hypertrophi:STcode]
##    [CAD|AngPec:Heartfail][AMI|CAD][Hyperchol|CAD][Inherit|CAD][Sex|CAD]
##    [Smoker|CAD][STchange|CAD:Hypertrophi:STcode][QWave|AMI:CAD]
##   nodes:                                 14
##   arcs:                                  19
##     undirected arcs:                     0
##     directed arcs:                       19
##   average markov blanket size:           3.29
##   average neighbourhood size:            2.71
##   average branching factor:              1.36
##
##   generation algorithm:                  Empty
```

```
# fitting the bn model, Conditional probability tables using cad1 data
bn_fit <- bn.fit(bn, data = cad1_data)
bn_fit
```

```
##
##   Bayesian network parameters
##
##   Parameters of node AMI (multinomial distribution)
##
## Conditional probability table:
##
##             CAD
## AMI                  No        Yes
##   Definite   0.09302326 0.47663551
##   NotCertain 0.90697674 0.52336449
##
##   Parameters of node AngPec (multinomial distribution)
##
## Conditional probability table:
##   Atypical      None   Typical
## 0.1271186 0.3601695 0.5127119
##
##   Parameters of node CAD (multinomial distribution)
##
## Conditional probability table:
##
## , , Heartfail = No
##
##      AngPec
## CAD     Atypical       None    Typical
##   No  0.86363636 0.80645161 0.17204301
##   Yes 0.13636364 0.19354839 0.82795699
##
## , , Heartfail = Yes
##
##      AngPec
## CAD     Atypical       None    Typical
##   No  0.62500000 0.95652174 0.60714286
##   Yes 0.37500000 0.04347826 0.39285714
##
##
##   Parameters of node Heartfail (multinomial distribution)
##
## Conditional probability table:
##
## , , STcode = Nonusable
##
##          Hypertrophi
## Heartfail        No       Yes
##       No  0.7500000 0.3225806
##       Yes 0.2500000 0.6774194
##
## , , STcode = Usable
##
##          Hypertrophi
## Heartfail        No       Yes
```

```
##       No  0.9516129 0.3939394
##       Yes 0.0483871 0.6060606
##
##
##   Parameters of node Hyperchol (multinomial distribution)
##
## Conditional probability table:
##
##          CAD
## Hyperchol        No       Yes
##       No  0.6279070 0.2523364
##       Yes 0.3720930 0.7476636
##
##   Parameters of node Hypertrophi (multinomial distribution)
##
## Conditional probability table:
##
##            SuffHeartF
## Hypertrophi         No        Yes
##         No  0.65269461 0.91304348
##         Yes 0.34730539 0.08695652
##
##   Parameters of node Inherit (multinomial distribution)
##
## Conditional probability table:
##
##        CAD
## Inherit        No       Yes
##     No  0.8062016 0.5420561
##     Yes 0.1937984 0.4579439
##
##   Parameters of node QWave (multinomial distribution)
##
## Conditional probability table:
##
## , , CAD = No
##
##      AMI
## QWave  Definite NotCertain
##   No  0.4166667  0.8803419
##   Yes 0.5833333  0.1196581
##
## , , CAD = Yes
##
##      AMI
## QWave  Definite NotCertain
##   No  0.2941176  0.5357143
##   Yes 0.7058824  0.4642857
##
##
##   Parameters of node QWavecode (multinomial distribution)
##
```

```
## Conditional probability table:
##
## , , STcode = Nonusable
##
##            Hypertrophi
## QWavecode           No        Yes
##   Nonusable 0.22916667 0.00000000
##   Usable    0.77083333 1.00000000
##
## , , STcode = Usable
##
##            Hypertrophi
## QWavecode           No        Yes
##   Nonusable 0.01612903 0.00000000
##   Usable    0.98387097 1.00000000
##
##
##   Parameters of node Sex (multinomial distribution)
##
## Conditional probability table:
##
##         CAD
## Sex             No       Yes
##   Female 0.2635659 0.1214953
##   Male   0.7364341 0.8785047
##
##   Parameters of node Smoker (multinomial distribution)
##
## Conditional probability table:
##
##       CAD
## Smoker        No       Yes
##    No  0.3100775 0.1028037
##    Yes 0.6899225 0.8971963
##
##   Parameters of node STchange (multinomial distribution)
##
## Conditional probability table:
##
## , , Hypertrophi = No, STcode = Nonusable
##
##         CAD
## STchange         No        Yes
##      No  1.00000000 1.00000000
##      Yes 0.00000000 0.00000000
##
## , , Hypertrophi = Yes, STcode = Nonusable
##
##         CAD
## STchange         No        Yes
##      No  1.00000000 1.00000000
##      Yes 0.00000000 0.00000000
```

```
##
## , , Hypertrophi = No, STcode = Usable
##
##         CAD
## STchange         No        Yes
##      No  0.78723404 0.18181818
##      Yes 0.21276596 0.81818182
##
## , , Hypertrophi = Yes, STcode = Usable
##
##         CAD
## STchange         No        Yes
##      No  0.03571429 0.40000000
##      Yes 0.96428571 0.60000000
##
##
##   Parameters of node STcode (multinomial distribution)
##
## Conditional probability table:
##
##            AngPec
## STcode       Atypical      None   Typical
##   Nonusable 0.2000000 0.5529412 0.2148760
##   Usable    0.8000000 0.4470588 0.7851240
##
##   Parameters of node SuffHeartF (multinomial distribution)
##
## Conditional probability table:
##
##           STcode
## SuffHeartF  Nonusable     Usable
##        No  0.98734177 0.56687898
##        Yes 0.01265823 0.43312102
```

```
# identifying d-separations in the graph
dsep(bn, "STchange", "AngPec", "STcode")
```

```
## [1] FALSE
```

```
dsep(bn, "Hypertrophi", "STcode", "SuffHeartF")
```

```
## [1] TRUE
```

```
dsep(bn, "AngPec", "SuffHeartF", "STcode")
```

```
## [1] TRUE
```

```
dsep(bn, "SuffHeartF", "Heartfail", "Hypertrophi")
```

```
## [1] FALSE
```

```
dsep(bn, "Inherit", "Heartfail", "CAD")
```

```
## [1] TRUE
```

```
dsep(bn, "Sex", "Heartfail", "CAD")
```

```
## [1] TRUE
```

```
dsep(bn, "Hyperchol", "Heartfail", "CAD")
```

```
## [1] TRUE
```

```
dsep(bn, "AMI", "Heartfail", "CAD")
```

```
## [1] TRUE
```

```
dsep(bn, "QWave", "CAD", "AMI")
```

```
## [1] FALSE
```

```
dsep(bn, "SuffHeartF", "QWavecode", "Hypertrophi")
```

```
## [1] FALSE
```

c) Suppose it is known that a new observation is female with Hypercholesterolemia (high cholesterol). Absorb this evidence into the graph and revise the probabilities. How does the probability of heart-failure and coronary artery disease (CAD) change after this information is considered?

```
library(gRain)

# initial probability of the heartfail and CAD
cad_Y_prob <- sum(cad1_data$CAD == "Yes") / nrow(cad1_data)
cad_Y_prob
```

```
## [1] 0.4533898
```

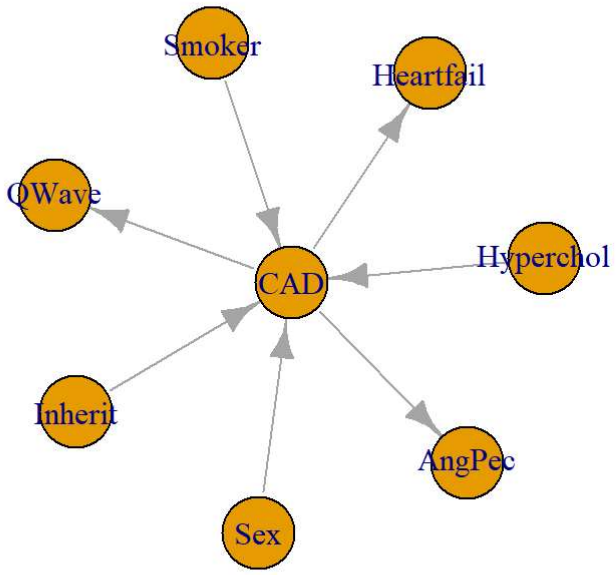
```
heartfail_Y_prob <- sum(cad1_data$Heartfail == "Yes") / nrow(cad1_data)
heartfail_Y_prob
```

```
## [1] 0.25
```

```
cpquery(bn_fit, (Heartfail == 'Yes'), (CAD == 'Yes'))
```

```
## [1] 0.1331089
```

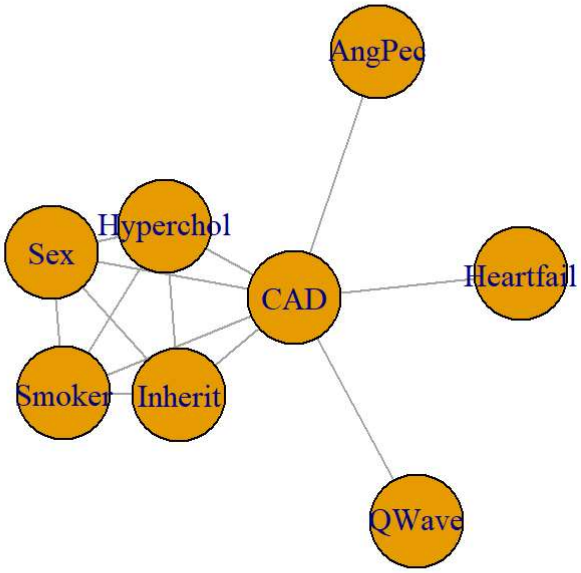
```
cad_dag <- dag(~ CAD:Sex:Smoker:Inherit:Hyperchol + AngPec:CAD + Heartfail:CAD + QWave:CAD)
plot(cad_dag, vertex.size = 30)
```




```
# grain creates graphical independence network
grain(cad_dag, data = cad1_data, smooth = 0.01)
```

```
## Independence network: Compiled: TRUE Propagated: FALSE Evidence: FALSE
```

```
cad1_bn <- compile(grain(cad_dag, data = cad1_data, smooth = 0.01))
plot(cad1_bn)
```




```
# Updating a new observation, is female with Hypercholesterolemia.
cad_find <- setFinding(cad1_bn, nodes = c("Sex", "Hyperchol"), states = c("Female","Yes"))
cad_find
```

```
## Independence network: Compiled: TRUE Propagated: TRUE Evidence: TRUE
```

```
getFinding(cad_find)
```

```
## $nodes
## [1] "Sex"       "Hyperchol"
##
## $is_hard
## [1] TRUE TRUE
##
## $hard_state
## [1] "Female" "Yes"
##
## $evi_weight
## $evi_weight[[1]]
## Sex
## Female   Male
##      1      0
##
## $evi_weight[[2]]
## Hyperchol
##  No Yes
##   0   1
##
##
## attr(,"class")
## [1] "grain_evidence" "list"
```

```
# With evidence (new observation)
querygrain(cad_find, nodes=c("CAD","Heartfail"), type = "marginal")
```

```
## $CAD
## CAD
##       No      Yes
## 0.503633 0.496367
##
## $Heartfail
## Heartfail
##        No       Yes
## 0.7585883 0.2414117
```

For CAD:

• The initial probability of CAD = ‘Yes’ was approximately 0.4533898

• The probability of CAD = ‘Yes’ has increased to approximately 0.496367 after incorporating the new evidence.

For Heartfail:

• The initial probability of Heartfail = ‘Yes’ was 0.25

• The probability of Heartfail = ‘Yes’ has slightly decreased to 0.2414117 after incorporating the new evidence.

Comparison

After incorporating the information of a female with Hypercholesterolemia into the Bayesian network, we observe changes in the estimated probabilities of CAD and Heart Failure (Heartfail). CAD's probability has increased, suggesting a higher likelihood of CAD given the new evidence, while the probability of Heart Failure (Heartfail) has slightly decreased. These shifts reflect how the new evidence influences the Bayesian network's inference about these health conditions.