Assignment - Bayesian Networks

Code

Loading the Data

Hide

```
data_assignment_3 <- read.csv(file = "C:/Users/Suma Marri/Documents/GitHub/COVID/COVID.csv",
                              colClasses = "character"
                         )
data_assignment_3
```

Fe...	Tiredness	Dry.Cough	Difficulty.in.Breathing	Sore.Throat	None_Symp...	Pa...	Nasa
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	0	1	1

1-10 of 316,800 rows | 1-8 of 27 columns   Previous **1** 2 3 4 5 6 ... 100 Next

This COVID-19 Symptoms Checker dataset is from the Kaggle Repository and is based on the guidelines given by the World Health Organization and the Ministry of Health and Family Welfare, India. The dataset helps identify whether any person has coronavirus based on pre-defined standard symptoms. For this assignment, I have chosen a more suitable dataset for Bayesian networks, that would pertain to the analysis of causation between symptoms, age, and other factors that would show if for example a person has a certain symptom, are they more susceptible to be old, or have another symptom, or the severity of the disease based on factors. None_Sympton and None_Experiencing points to a patient having no top 5 symptoms and not experiencing any other symptoms respectively. For this analysis, the symptom variables used would be Fever, Tiredness, Dry.Cough, Difficulty.in.Breathing, and Sore.Throat, which are the top 5 symptoms of COVID-19 as specified by the WHO. Other symptoms that are used would be the Pains, Nasal.Congestion, Runny.Nose, and Diarrhea. Variables for age help us identify if the virus is truly severe to those who are older. Three genders are used: Male, Female, and

Transgender. The severity for the virus is considered to be None, Mild, Moderate, and Severe. The contact tell us if the person has been in contact with someone with the virus. The country variable tells us which country the person visited, but for the purpose of this analysis, it will not be used.

Preparing the dataset for bnlearn package to have all factor variables

Removing Country and making sure we have factor variables for Bayesian networks

Hide

```
d <- subset(data_assignment_3, select = -c(Country))
summary(d)
```

```
    Fever            Tiredness          Dry.Cough         Difficulty.in.Breathing Sore.Throat
None_Sympton
 Length:316800      Length:316800      Length:316800      Length:316800           Length:316800
Length:316800
 Class :character   Class :character   Class :character   Class :character        Class :charact
er   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character        Mode  :charact
er   Mode  :character
    Pains           Nasal.Congestion    Runny.Nose          Diarrhea         None_Experiencing
Age_0.9
 Length:316800      Length:316800      Length:316800      Length:316800      Length:316800
Length:316800
 Class :character   Class :character   Class :character   Class :character   Class :character
Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
Mode  :character
  Age_10.19          Age_20.24          Age_25.59          Age_60.          Gender_Female
Gender_Male
 Length:316800      Length:316800      Length:316800      Length:316800      Length:316800
Length:316800
 Class :character   Class :character   Class :character   Class :character   Class :character
Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
Mode  :character
 Gender_Transgender Severity_Mild      Severity_Moderate  Severity_None      Severity_Severe
Contact_Dont.Know
 Length:316800      Length:316800      Length:316800      Length:316800      Length:316800
Length:316800
 Class :character   Class :character   Class :character   Class :character   Class :character
Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Mode  :character
Mode  :character
  Contact_No        Contact_Yes
 Length:316800      Length:316800
 Class :character   Class :character
 Mode  :character   Mode  :character
```

Hide

```
for(j in 1:ncol(d)){
  d[,j] <- factor(as.numeric(d[,j]))}
```

```
Registered S3 method overwritten by 'data.table':
  method           from
  print.data.table
```

Hide

```
summary(d)
```

```
 Fever      Tiredness  Dry.Cough  Difficulty.in.Breathing Sore.Throat None_Sympton Pains      Na
sal.Congestion Runny.Nose
 0:217800   0:158400   0:138600   0:158400                0:217800    0:297000     0:201600   0:
144000         0:144000
 1: 99000   1:158400   1:178200   1:158400                1: 99000    1: 19800     1:115200   1:
172800         1:172800
 Diarrhea   None_Experiencing Age_0.9    Age_10.19  Age_20.24  Age_25.59  Age_60.    Gender_Fema
le Gender_Male
 0:201600   0:288000          0:253440   0:253440   0:253440   0:253440   0:253440   0:211200
0:211200
 1:115200   1: 28800          1: 63360   1: 63360   1: 63360   1: 63360   1: 63360   1:105600
1:105600
 Gender_Transgender Severity_Mild Severity_Moderate Severity_None Severity_Severe Contact_Dont.K
now Contact_No Contact_Yes
 0:211200           0:237600      0:237600          0:237600      0:237600        0:211200
0:211200   0:211200
 1:105600           1: 79200      1: 79200          1: 79200      1: 79200        1:105600
1:105600   1:105600
```

Building the Bayesian Network Models

Constructing the Bayesian Models using a constraint-based, score-based, hybrid, and a local discovery algorithm. Using the the algorithms benchmarked by Dr. Smith with regards to computation time.

Hide

```
require(bnlearn)
```

```
Loading required package: bnlearn

Attaching package: ‘bnlearn’

The following object is masked from ‘package:NMF’:

    compare
```

Hide

```
d_algorithms <- c("iamb.fdr", "hc", "h2pc", "aracne")
list_bnlearn <- list()
for(j in d_algorithms) try({
  list_bnlearn[[j]] <- do.call(
    what = j,
    args = list(x = d)
    )
  }
  )
```

```
Warning: vstructure Fever -> Tiredness <- Dry.Cough is not applicable, because one or both arcs are oriented in the opposite direction.Warning: vstructure Tiredness -> Dry.Cough <- Difficulty.in.Breathing is not applicable, because one or both arcs are oriented in the opposite direction.Warning: vstructure Fever -> Difficulty.in.Breathing <- Dry.Cough is not applicable, because one or both arcs are oriented in the opposite direction.Warning: vstructure Tiredness -> Sore.Throat <- Difficulty.in.Breathing is not applicable, because one or both arcs are oriented in the opposite direction.
```

Hide

```
list_bnlearn
```

```
$iamb.fdr

  Bayesian network learned via Constraint-based methods

  model:
    [partially directed graph]
  nodes:                                26
  arcs:                                 44
    undirected arcs:                    34
    directed arcs:                      10
  average markov blanket size:          3.54
  average neighbourhood size:           3.38
  average branching factor:             0.38

  learning algorithm:                   IAMB-FDR
  conditional independence test:        Mutual Information (disc.)
  alpha threshold:                      0.05
  tests used in the learning procedure:  3968


$hc

  Bayesian network learned via Score-based methods

  model:
   [Dry.Cough][Pains][Age_0.9][Gender_Female][Severity_Mild][Contact_Dont.Know][None_Sympton|Dr
y.Cough][Runny.Nose|Pains]
   [Age_60.|Age_0.9][Gender_Transgender|Gender_Female][Severity_Severe|Severity_Mild][Contact_Ye
s|Contact_Dont.Know]
   [Fever|None_Sympton][Nasal.Congestion|Pains:Runny.Nose][Age_25.59|Age_0.9:Age_60.]
   [Gender_Male|Gender_Female:Gender_Transgender][Severity_None|Severity_Mild:Severity_Severe]
   [Contact_No|Contact_Dont.Know:Contact_Yes][Tiredness|Fever:Dry.Cough:None_Sympton]
   [Diarrhea|Pains:Nasal.Congestion:Runny.Nose][Age_20.24|Age_0.9:Age_25.59:Age_60.]
   [Severity_Moderate|Severity_Mild:Severity_None:Severity_Severe]
   [Difficulty.in.Breathing|Fever:Tiredness:Dry.Cough:None_Sympton]
   [None_Experiencing|Pains:Nasal.Congestion:Runny.Nose:Diarrhea][Age_10.19|Age_0.9:Age_20.24:Ag
e_25.59:Age_60.]
   [Sore.Throat|Fever:Tiredness:Dry.Cough:Difficulty.in.Breathing:None_Sympton]
  nodes:                                26
  arcs:                                 46
    undirected arcs:                    0
    directed arcs:                      46
  average markov blanket size:          3.62
  average neighbourhood size:           3.54
  average branching factor:             1.77

  learning algorithm:                   Hill-Climbing
  score:                                BIC (disc.)
  penalization coefficient:             6.333013
  tests used in the learning procedure:  1520
  optimized:                            TRUE
```

```
$h2pc

  Bayesian network learned via Hybrid methods

  model:
   [Dry.Cough][Pains][Age_0.9][Gender_Female][Severity_Mild][Contact_Dont.Know][None_Sympton|Dr
y.Cough][Runny.Nose|Pains]
   [Age_60.|Age_0.9][Gender_Transgender|Gender_Female][Severity_Severe|Severity_Mild][Contact_Ye
s|Contact_Dont.Know]
   [Fever|None_Sympton][Nasal.Congestion|Pains:Runny.Nose][Age_25.59|Age_0.9:Age_60.]
   [Gender_Male|Gender_Female:Gender_Transgender][Severity_None|Severity_Mild:Severity_Severe]
   [Contact_No|Contact_Dont.Know:Contact_Yes][Tiredness|Fever:Dry.Cough:None_Sympton]
   [Difficulty.in.Breathing|Fever:Dry.Cough:None_Sympton][Diarrhea|Pains:Nasal.Congestion:Runny.
Nose]
   [Age_20.24|Age_0.9:Age_25.59:Age_60.][Severity_Moderate|Severity_Mild:Severity_None:Severity_
Severe]
   [Sore.Throat|Fever:Tiredness:Difficulty.in.Breathing:None_Sympton]
   [None_Experiencing|Pains:Nasal.Congestion:Runny.Nose:Diarrhea][Age_10.19|Age_0.9:Age_20.24:Ag
e_25.59:Age_60.]
  nodes:                                26
  arcs:                                 44
    undirected arcs:                    0
    directed arcs:                      44
  average markov blanket size:          3.54
  average neighbourhood size:           3.38
  average branching factor:             1.69

  learning algorithm:                   Hybrid^2 Parent Children
  constraint-based method:              Hybrid Parents and Children
  conditional independence test:        Mutual Information (disc.)
  score-based method:                   Hill-Climbing
  score:                                BIC (disc.)
  alpha threshold:                      0.05
  penalization coefficient:             6.333013
  tests used in the learning procedure: 4104
  optimized:                            TRUE


$aracne

  Bayesian network learned via Pairwise Mutual Information methods

  model:
    [undirected graph]
  nodes:                                26
  arcs:                                 309
    undirected arcs:                    309
    directed arcs:                      0
  average markov blanket size:          23.77
  average neighbourhood size:           23.77
  average branching factor:             0.00
```

```
  learning algorithm:                    ARACNE
  mutual information estimator:          Maximum Likelihood (disc.)
  tests used in the learning procedure:  325
```

The score-based algorithm hc() and the hybrid algorithm h2pc() produce directed graphs. Lets see if we can produce directed graphs with any other constraint-based and local discovery algorithms

Hide

```
d_algorithms <- c("pc.stable", "gs", "iamb", "inter.iamb", "mmpc", "si.hiton.pc", "hpc", "chow.l
iu")
list2_bnlearn <- list()
for(j in d_algorithms) try({
  list2_bnlearn[[j]] <- do.call(
    what = j,
    args = list(x = d)
    )
  },silent = TRUE
  )
```

```
Warning: vstructure Fever -> Tiredness <- Dry.Cough is not applicable, because one or both arcs
are oriented in the opposite direction.Warning: vstructure Tiredness -> Dry.Cough <- Difficulty.
in.Breathing is not applicable, because one or both arcs are oriented in the opposite direction.
Warning: vstructure Fever -> Difficulty.in.Breathing <- Dry.Cough is not applicable, because one
or both arcs are oriented in the opposite direction.Warning: vstructure Tiredness -> Sore.Throat
<- Difficulty.in.Breathing is not applicable, because one or both arcs are oriented in the oppos
ite direction.Warning: vstructure Fever -> Tiredness <- Dry.Cough is not applicable, because one
or both arcs are oriented in the opposite direction.Warning: vstructure Tiredness -> Dry.Cough <
- Difficulty.in.Breathing is not applicable, because one or both arcs are oriented in the opposi
te direction.Warning: vstructure Fever -> Difficulty.in.Breathing <- Dry.Cough is not applicabl
e, because one or both arcs are oriented in the opposite direction.Warning: vstructure Tiredness
-> Sore.Throat <- Difficulty.in.Breathing is not applicable, because one or both arcs are orient
ed in the opposite direction.Warning: vstructure Fever -> Tiredness <- Dry.Cough is not applicab
le, because one or both arcs are oriented in the opposite direction.Warning: vstructure Tirednes
s -> Dry.Cough <- Difficulty.in.Breathing is not applicable, because one or both arcs are orient
ed in the opposite direction.Warning: vstructure Fever -> Difficulty.in.Breathing <- Dry.Cough i
s not applicable, because one or both arcs are oriented in the opposite direction.Warning: vstru
cture Tiredness -> Sore.Throat <- Difficulty.in.Breathing is not applicable, because one or both
arcs are oriented in the opposite direction.Warning: vstructure Fever -> Tiredness <- Dry.Cough
is not applicable, because one or both arcs are oriented in the opposite direction.Warning: vstr
ucture Tiredness -> Dry.Cough <- Difficulty.in.Breathing is not applicable, because one or both
arcs are oriented in the opposite direction.Warning: vstructure Fever -> Difficulty.in.Breathing
<- Dry.Cough is not applicable, because one or both arcs are oriented in the opposite direction.
Warning: vstructure Tiredness -> Sore.Throat <- Difficulty.in.Breathing is not applicable, becau
se one or both arcs are oriented in the opposite direction.
```

Hide

```
list2_bnlearn
```

```
$pc.stable

  Bayesian network learned via Constraint-based methods

  model:
    [partially directed graph]
  nodes:                                26
  arcs:                                 44
    undirected arcs:                    34
    directed arcs:                      10
  average markov blanket size:          3.54
  average neighbourhood size:           3.38
  average branching factor:             0.38

  learning algorithm:                   PC (Stable)
  conditional independence test:        Mutual Information (disc.)
  alpha threshold:                      0.05
  tests used in the learning procedure: 988


$gs

  Bayesian network learned via Constraint-based methods

  model:
    [partially directed graph]
  nodes:                                26
  arcs:                                 44
    undirected arcs:                    34
    directed arcs:                      10
  average markov blanket size:          3.54
  average neighbourhood size:           3.38
  average branching factor:             0.38

  learning algorithm:                   Grow-Shrink
  conditional independence test:        Mutual Information (disc.)
  alpha threshold:                      0.05
  tests used in the learning procedure: 2514


$iamb

  Bayesian network learned via Constraint-based methods

  model:
    [partially directed graph]
  nodes:                                26
  arcs:                                 44
    undirected arcs:                    34
    directed arcs:                      10
  average markov blanket size:          3.54
  average neighbourhood size:           3.38
```

```
  average branching factor:              0.38

  learning algorithm:                    IAMB
  conditional independence test:         Mutual Information (disc.)
  alpha threshold:                       0.05
  tests used in the learning procedure:  3804


$inter.iamb

  Bayesian network learned via Constraint-based methods

  model:
    [partially directed graph]
  nodes:                                 26
  arcs:                                  44
    undirected arcs:                     34
    directed arcs:                       10
  average markov blanket size:           3.54
  average neighbourhood size:            3.38
  average branching factor:              0.38

  learning algorithm:                    Inter-IAMB
  conditional independence test:         Mutual Information (disc.)
  alpha threshold:                       0.05
  tests used in the learning procedure:  3874


$mmpc

  Bayesian network learned via Constraint-based methods

  model:
    [undirected graph]
  nodes:                                 26
  arcs:                                  44
    undirected arcs:                     44
    directed arcs:                       0
  average markov blanket size:           3.38
  average neighbourhood size:            3.38
  average branching factor:              0.00

  learning algorithm:                    Max-Min Parent Children
  conditional independence test:         Mutual Information (disc.)
  alpha threshold:                       0.05
  tests used in the learning procedure:  2359


$si.hiton.pc

  Bayesian network learned via Constraint-based methods
```

```
  model:
    [undirected graph]
  nodes:                                26
  arcs:                                 44
    undirected arcs:                    44
    directed arcs:                      0
  average markov blanket size:          3.38
  average neighbourhood size:           3.38
  average branching factor:             0.00

  learning algorithm:                   Semi-Interleaved HITON-PC
  conditional independence test:        Mutual Information (disc.)
  alpha threshold:                      0.05
  tests used in the learning procedure:  1394


$hpc

  Bayesian network learned via Constraint-based methods

  model:
    [undirected graph]
  nodes:                                26
  arcs:                                 44
    undirected arcs:                    44
    directed arcs:                      0
  average markov blanket size:          3.38
  average neighbourhood size:           3.38
  average branching factor:             0.00

  learning algorithm:                   Hybrid Parents and Children
  conditional independence test:        Mutual Information (disc.)
  alpha threshold:                      0.05
  tests used in the learning procedure:  3894


$chow.liu

  Bayesian network learned via Pairwise Mutual Information methods

  model:
    [undirected graph]
  nodes:                                26
  arcs:                                 25
    undirected arcs:                    25
    directed arcs:                      0
  average markov blanket size:          1.92
  average neighbourhood size:           1.92
  average branching factor:             0.00

  learning algorithm:                   Chow-Liu
```

```
  mutual information estimator:          Maximum Likelihood (disc.)
  tests used in the learning procedure:  325
```

Since the other constraint-based algorithms and the local discovery algorithms do not produce a fully directed graph, lets proceed with the initial algorithms due to their benchmarked performance to save time: iamb.fdr(), hc(), h2pc(), and aracne().

Scoring the Models

Since we do not have a lot of success with the learning of constraint based and local discovery algorithm, we can go on with finding the best model depending on the scoring type for the score-based and hybrid algorithms. Below, we will still try to score the constraint-based and local discovery algorithm. The selection of the model will depend on the type of scoring and the score itself for the algorithm. Testing of the available scores for discrete Bayesian networks for categorical variables will be performed. The measures that will be tested are loglik - which measures the log likelihood, meaning the goodness of fit of the model to the sample of the data. This is imperative, as we do want model that is a good fit, bic - which is a criterion for model selection amongst a set of models (lowest bic model is preferred) and it is partly based on the likelihood function, aic - which is an estimator of the prediction and informs about the quality of the model compared to other models in a set of models, bde - which is the logarithm of the Bayesian Dirichlet equivalent (uniform) score, a score equivalent Dirichlet posterior density and bds - the logarithm of the Bayesian Dirichlet sparse score, which is a sparsity-inducing Dirichlet posterior density and not score equivalent.

bic

Hide

```
d_algorithms <- c("iamb.fdr", "hc", "h2pc", "aracne")

M_score_bic <- list()
for (j in d_algorithms) try({
  M_score_bic[j] <- score(
    x=list_bnlearn[[j]],
    data = d,
    type = "bic"
  )
}
)
```

```
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
```

Hide

```
M_score_bic <- data.frame(M_score_bic)
M_score_bic
```

hc <dbl>	h2pc <dbl>
-3284101	-3328486

1 row

aic

Hide

```
d_algorithms <- c("iamb.fdr", "hc", "h2pc", "aracne")

M_score_aic <- list()
for (j in d_algorithms) try({
  M_score_aic[j] <- score(
    x=list_bnlearn[[j]],
    data = d,
    type = "aic"
  )
}
)
```

```
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
```

Hide

```
M_score_aic <- data.frame(M_score_aic)
M_score_aic
```

hc <dbl>	h2pc <dbl>
-3283290	-3327803

1 row

loglik

Hide

```
d_algorithms <- c("iamb.fdr", "hc", "h2pc", "aracne")

M_score_loglik <- list()
for (j in d_algorithms) try({
  M_score_loglik[j] <- score(
    x=list_bnlearn[[j]],
    data = d,
    type = "loglik"
  )
}
)
```

```
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
```

Hide

```
M_score_loglik <- data.frame(M_score_loglik)
M_score_loglik
```

hc <dbl>	h2pc <dbl>
-3283138	-3327675

1 row

bde

Hide

```
d_algorithms <- c("iamb.fdr", "hc", "h2pc", "aracne")

M_score_bde <- list()
for (j in d_algorithms) try({
  M_score_bde[j] <- score(
    x=list_bnlearn[[j]],
    data = d,
    type = "bde"
  )
}
)
```

```
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
```

Hide

```
M_score_bde <- data.frame(M_score_bde)
M_score_bde
```

hc	h2pc
<dbl>	<dbl>
-3283483	-3328004

1 row

bds

Hide

```
d_algorithms <- c("iamb.fdr", "hc", "h2pc", "aracne")

M_score_bds <- list()
for (j in d_algorithms) try({
  M_score_bds[j] <- score(
    x=list_bnlearn[[j]],
    data = d,
    type = "bds"
  )
}
)
```

```
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
Error in network.score(x = x, data = data, type = type, ..., by.node = by.node,  :
  the graph is only partially directed.
```

Hide

```
M_score_bds <- data.frame(M_score_bds)
M_score_bds
```

hc	h2pc
<dbl>	<dbl>
-3283493	-3328016

1 row

Model Score Comparison

Hide

```
#Combining the scores in a table df
g <- rbind(M_score_bic,M_score_aic,M_score_loglik, M_score_bde, M_score_bds)
rownames(g) <- c("bic", "aic", "loglik", "bde", "bds")
g
```

	hc <dbl>	h2pc <dbl>
bic	-3284101	-3328486
aic	-3283290	-3327803
loglik	-3283138	-3327675
bde	-3283483	-3328004
bds	-3283493	-3328016

5 rows

Hide

```
#Formatting the table to see which algorithm performs better
h <- data.frame(t(g))
colnames(h) <- rownames(g)
h
```

	bic <dbl>	aic <dbl>	loglik <dbl>	bde <dbl>	bds <dbl>
hc	-3284101	-3283290	-3283138	-3283483	-3283493
h2pc	-3328486	-3327803	-3327675	-3328004	-3328016

2 rows

Hide

```
#Sorting largest to smallest in terms of performance of algorithms
sorted_h <- h[order(h$bic,h$aic,h$loglik,h$bde,h$bds),]
sorted_h
```

	bic <dbl>	aic <dbl>	loglik <dbl>	bde <dbl>	bds <dbl>
h2pc	-3328486	-3327803	-3327675	-3328004	-3328016
hc	-3284101	-3283290	-3283138	-3283483	-3283493

2 rows

Now that we have scores for the algorithms sorted, we can see that hc() performed better according to all scoring types. The best model according to the table is hc(). All the scoring types show the same results according to the table.

Hide

```
apply(sorted_h, 2, FUN=max)
```

```
     bic      aic   loglik      bde      bds 
-3284101 -3283290 -3283138 -3283483 -3283493 
```

Visualizing the model hc()

Setting the node and edge attributes

Hide

```
#Node Attributes
hc_covid <- hc(d)
v_nodes <- nodes(hc_covid)
names(v_nodes) <- v_nodes
strength_covid <- arc.strength(
  x = hc_covid,
  data = d
)
n_nodes <- nnodes(hc_covid)

v_fillcolor <- viridis::viridis(n_nodes)
names(v_fillcolor) <- v_nodes

v_shape <- c(
  rep("circle",floor(n_nodes/3)),
  rep("ellipse",floor(n_nodes/3)),
  rep("box",n_nodes - 2*floor(n_nodes/3))
  )
names(v_shape) <- v_nodes

#Edge Attributes
v_edges <- paste0(strength_covid[,"from"],"~",
                  strength_covid[,"to"])
names(v_edges) <- v_edges

v_edgecolor <- v_fillcolor[strength_covid[,"from"]]
names(v_edgecolor) <- v_edges
```

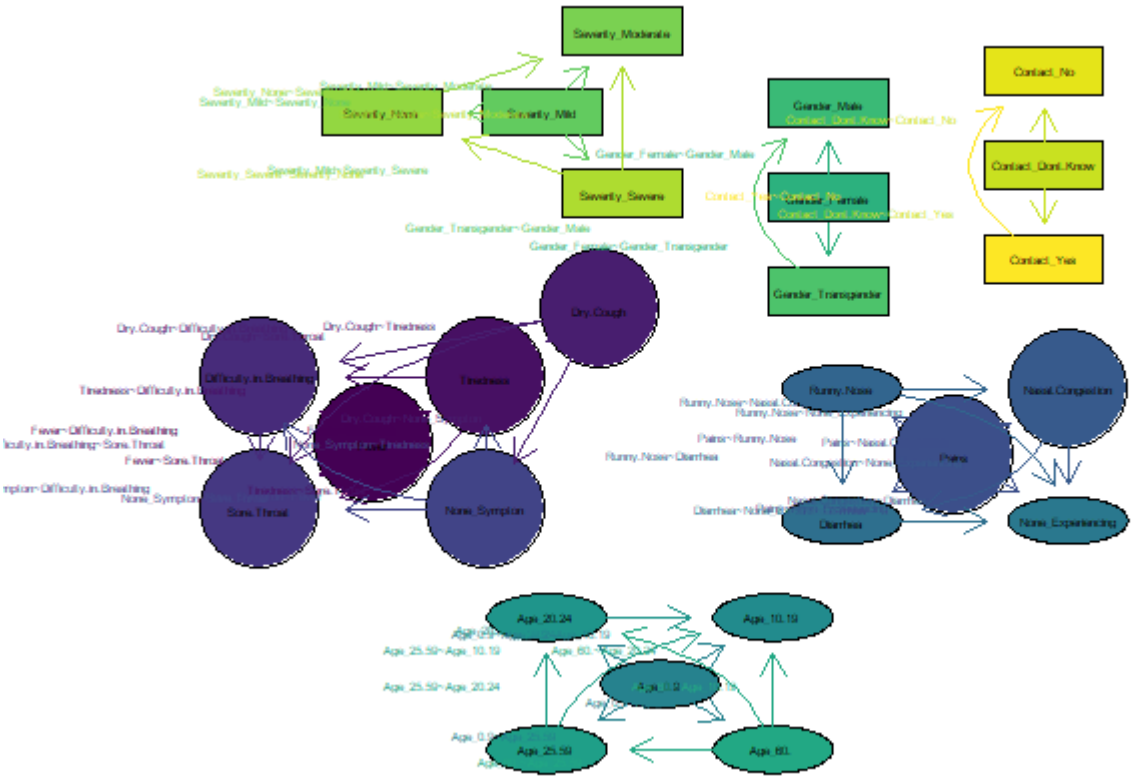
Plotting the model

Hide

```
#Converting the bnlearn model to a graphNEL model
# library(BiocManager)
# install.packages("BiocManager")
# BiocManager::install(c("graph", "RBGL", "Rgraphviz"))
graphNEL_covid <- as.graphNEL(hc_covid)
```

Hide

```
Rgraphviz::plot(
  x = graphNEL_covid,
  y = "twopi",  attrs = list(),
  nodeAttrs = list(
    fillcolor = v_fillcolor,
    shape = v_shape
    ),
  edgeAttrs = list(
    label = v_edges,
    weight = graph::edgeWeights(graphNEL_covid),
    color = v_edgecolor,
    fontcolor = v_edgecolor
    )
  )
```



Predict the target variable

Predicting the target

For this analysis, since the bayesian model treats all the nodes as predictor and target, lets assume we are predicting the presence of the Nasal Congestion in a patient. Let’s predict whether the patient will have nasal congestion or not depending on the symptoms and other features that the patient might have.

Hide

```
fit_model <- bn.fit(
  x = hc_covid,
  data = d
)
pred_table <- data.frame(pred = predict(fit_model, node = "Nasal.Congestion", data = d),actual =
d$Nasal.Congestion)
summary(pred_table)
```

```
 pred       actual
 0:172725   0:144000
 1:144075   1:172800
```

Evaluating Model Fit

Hide

```
#Calculating AUC
AUC_model <- Metrics::auc(pred_table$actual,pred_table$pred)
print(paste0("AUC:", AUC_model))
```

```
[1] "AUC:0.64183912037037"
```

Hide

```
#Calculating Model Accuracy
Accuracy_model <- Metrics::accuracy(pred_table$actual,pred_table$pred)
print(paste0("Accuracy: ", Accuracy_model))
```

```
[1] "Accuracy: 0.636556186868687"
```

Hide

```
# Constructing the confusion matrix
remove.packages("rlang")
```

```
Removing package from ‘C:/Users/Suma Marri/Documents/R/win-library/4.1’
(as ‘lib’ is unspecified)
```

Hide

```
install.packages("rlang")
```

```
Error in install.packages : Updating loaded packages
```

Hide

```
install.packages("vctrs")
```

```
Error in install.packages : Updating loaded packages
```

Hide

```
install.packages("pillar")
```

```
Error in install.packages : Updating loaded packages
```

Hide

```
confusion_matrix_model <- caret::confusionMatrix(pred_table$pred,pred_table$actual,"1")
```

```
Registered S3 methods overwritten by 'proxy':
  method               from
  print.registry_field registry
  print.registry_entry registry
```

Hide

```
cm <- data.frame(confusion_matrix_model$byClass)
cm
```

	confusion_matrix_model.byClass <dbl>
Sensitivity	0.5837269
Specificity	0.6999514
Pos Pred Value	0.7001076
Neg Pred Value	0.5835461
Precision	0.7001076
Recall	0.5837269
F1	0.6366422
Prevalence	0.5454545
Detection Rate	0.3183965

	confusion_matrix_model.byClass <dbl>
Detection Prevalence	0.4547822

If I was using this model at work, I would not use this for the analysis of this dataset. According to the evaluation scores above that show the model performance in prediction of nasal congestion, the scores are not that impressive. Let's assume if the dataset only had features concerning symptoms, and the age features as well as the contact features were taken out, the prediction could be more accurate since age does not contribute towards a person having nasal congestion. However, since we are dealing with a graphical model, lets run a cross-validation to further see whether the model performs well without specifying a target:

Hide

```
# Repeated 2-fold Cross-validation
cv_model <- bn.cv(
  data = d,
  bn = "hc",
  k = 2,
  runs = 2
)
cv_model
```

```
  k-fold cross-validation for Bayesian networks

  target learning algorithm:            Hill-Climbing
  number of folds:                      2
  loss function:                        Log-Likelihood Loss (disc.)
  number of runs:                       2
  average loss over the runs:           10.36366
  standard deviation of the loss:       5.638984e-05
```

The graphical model seems to perform well, however, I would not use this for prediction in a classification problem if we were trying to predict either a symptom, or severity. I would rather use a clustering technique or do a logistic regression for this kind of problem. This model would work well for a target prediction if we were only predicting presence of a disease looking at other diseases or maybe even age as features.