Graphing bnlearn models with Rgraphviz

Aaron Smith

6/25/2020

To get the most out our Bayesian networks, we need to creat visually appealing and informative visualizations of our graphs.

For models from the bnlearn package, the Rgraphviz package enables us to build compelling plots.

# Load the data

For this example, we will use the alarm data set in the bnlearn package. Then build a model, and visualize the model.

require(bnlearn)

## Loading required package: bnlearn

##   
## Attaching package: 'bnlearn'

## The following object is masked from 'package:stats':  
##   
## sigma

data(alarm)  
hc\_alarm <- hc(alarm)

# Node attributes

There is a list of node attributes that we can control in Rgraphviz plots. The ones that Dr. Smith finds the most useful are

* fillcolor
* shape

Node attributes are labeled as nodeAttrs in the plot function, are contained in named lists, and need to be named vectors with the node being the name.

v\_nodes <- nodes(hc\_alarm)  
names(v\_nodes) <- v\_nodes  
strength\_alarm <- arc.strength(  
 x = hc\_alarm,  
 data = alarm  
)  
n\_nodes <- nnodes(hc\_alarm)  
  
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

There is a list of edge attributes that we can control in Rgraphviz plots. The ones that Dr. Smith finds the most useful are

* label
* color
* fontcolor

Edge attributes are labeled as edgeAttrs in the plot function, are contained in named lists, and need to be named vectors with the edge being the name.

Edges are named with this convention: node\_1~node\_2

v\_edges <- paste0(strength\_alarm[,"from"],"~",strength\_alarm[,"to"])  
names(v\_edges) <- v\_edges  
  
v\_edgecolor <- v\_fillcolor[strength\_alarm[,"from"]]  
names(v\_edgecolor) <- v\_edges

# Converting the bnlearn model to a graphNEL model

R has a ton of specialized packages for graphical models. To use Rgraphviz, we need to convert our bnlearn model to a graphNEL model.

graphNEL\_alarm <- as.graphNEL(hc\_alarm)

## Loading required namespace: graph

# Plotting the model

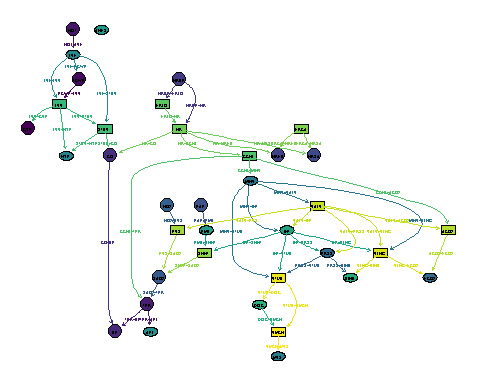
The layout of the model has a big impact on how the results look. The layout options are

* dot
* neato
* twopi
* circo
* fdp

The following plots all show the same model, just different layouts.

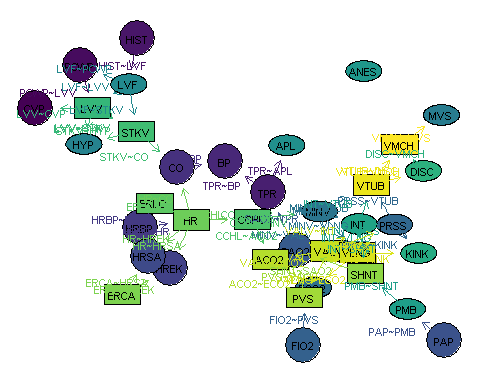
## dot

Rgraphviz::plot(  
 x = graphNEL\_alarm,  
 y = "dot",  
 attrs = list(),   
 nodeAttrs = list(  
 fillcolor = v\_fillcolor,  
 shape = v\_shape  
 ),   
 edgeAttrs = list(  
 label = v\_edges,  
 weight = graph::edgeWeights(graphNEL\_alarm),  
 color = v\_edgecolor,  
 fontcolor = v\_edgecolor  
 )  
)



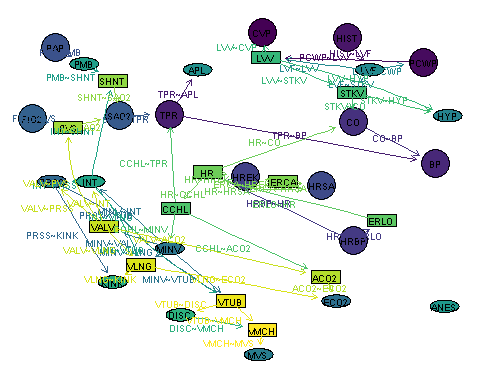
## neato

Rgraphviz::plot(  
 x = graphNEL\_alarm,  
 y = "neato",  
 attrs = list(),   
 nodeAttrs = list(  
 fillcolor = v\_fillcolor,  
 shape = v\_shape  
 ),   
 edgeAttrs = list(  
 label = v\_edges,  
 color = v\_edgecolor,  
 fontcolor = v\_edgecolor  
 )  
)



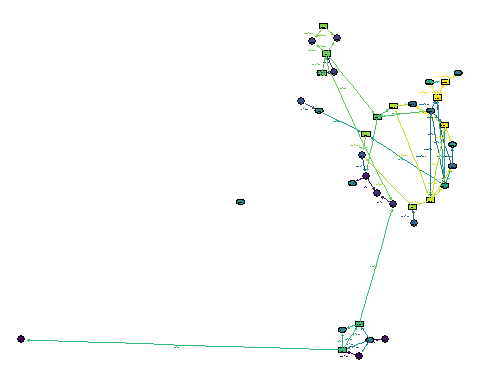
## twopi

Rgraphviz::plot(  
 x = graphNEL\_alarm,  
 y = "twopi",  
 attrs = list(),   
 nodeAttrs = list(  
 fillcolor = v\_fillcolor,  
 shape = v\_shape  
 ),   
 edgeAttrs = list(  
 label = v\_edges,  
 weight = graph::edgeWeights(graphNEL\_alarm),  
 color = v\_edgecolor,  
 fontcolor = v\_edgecolor  
 )  
)



## circo

Rgraphviz::plot(  
 x = graphNEL\_alarm,  
 y = "circo",  
 attrs = list(),   
 nodeAttrs = list(  
 fillcolor = v\_fillcolor,  
 shape = v\_shape  
 ),   
 edgeAttrs = list(  
 label = v\_edges,  
 weight = graph::edgeWeights(graphNEL\_alarm),  
 color = v\_edgecolor,  
 fontcolor = v\_edgecolor  
 )  
)



## fdp

Rgraphviz::plot(  
 x = graphNEL\_alarm,  
 y = "fdp",  
 attrs = list(),   
 nodeAttrs = list(  
 fillcolor = v\_fillcolor,  
 shape = v\_shape  
 ),   
 edgeAttrs = list(  
 label = v\_edges,  
 weight = graph::edgeWeights(graphNEL\_alarm),  
 color = v\_edgecolor,  
 fontcolor = v\_edgecolor  
 )  
)

