Exercises: Visualising Networks

2023-09-25

Networks

In this set of exercises, we will go through creating a network in igraph, visualising it using plot, setting node attributes and plotting some features of the network.

First, read in the network data - the edgelist is in the form of a .txt file, so use read.table instead of read.csv

This network comes from a school that was used for modelling the spread of COVID-19. The original publication as well as the data can be found here: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3009790/

After reading the data, drop the repeated edges from the edgelist using distinct in dplyr. Then make sure the ordering of the nodes from the graph vertices matches the edgelist using match.

Finally, assign the node attributes using the $ on V(school_graph).

```r
# Set working directory here!!
library(dagitty)
library(ggdag)
library(igraph)
library(dplyr)
library(ggplot2)

# Read in network data
nodes <- read.csv("school_nodelist.csv")
edges <- read.table("school_edgelist_copy.txt")

edges <- edges[, 1:2] %>% distinct(V1, V2)

school_graph <- graph_from_edgelist(as.matrix(edges))
reorder <- match(nodes$ID, V(school_graph))
nodes <- arrange(nodes, reorder)
V(school_graph)$role <- nodes$Role
```

Now look for some summary statistics about the data - how many individuals are there? How many roles, and which are the different roles? How many individuals of each role are there?

```r
# How many different kinds of roles are there in the network
unique(V(school_graph)$role)
```

```
## [1] "teacher" "student" "staff"   "other"
```

```r
# How many people are there in each role?
nrow(nodes %>% filter(Role == 'staff'))
```

```
## [1] 55
```

Currently, it's a bit hard to visualise the full graph, because there are two many nodes.

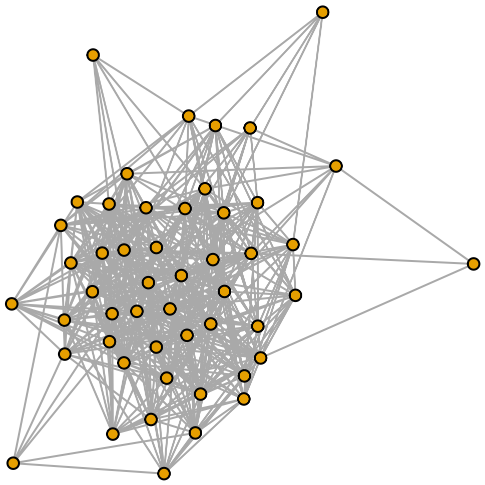
One thing we could try to do is sample the nodes using the sample function applied to V(graph) - try doing this with 50 nodes.

Then, use subgraph in igraph to generate a subgraph of the full network with these nodes.

If we have set the node attributes previously for our graph, these attributes will carry over to the subgraph!

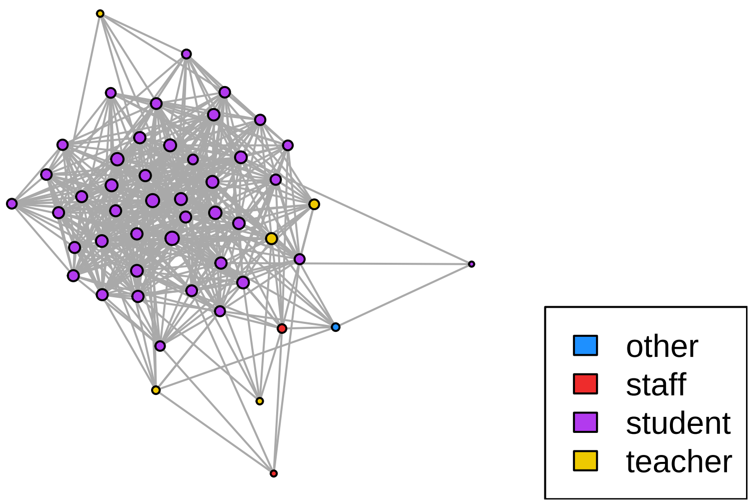
Plot the resulting subgraph. Colour the nodes according to the role that each of them plays (hint: use as.factor to convert the roles to factors, and then assign the colors as node attributes using V(graph)$color).

```r
# Randomly subsample the nodes from the full graph - set the seed to get the
# same nodes each time!
set.seed(123)
sample_size <- 50
subgraph_nodes <- sample(V(school_graph), sample_size)

# Create subgraph using these nodes
school_subgraph <- as.undirected(subgraph(school_graph, subgraph_nodes))
plot(school_subgraph, vertex.label = NA, vertex.size = 5, edge.size = 2)
```



```r
# Get the corresponding subgraph node attributes, i.e.
#subgraph_node_attrs <- nodes %>% filter(ID %in% subgraph_nodes) %>% select(ID, Role)

# Set the colours of the roles
colors <- c('dodgerblue', 'firebrick2', 'darkorchid2', 'gold2')
V(school_subgraph)$color <- colors[as.factor(V(school_subgraph)$role)]


labels <- c('other', 'staff', 'student', 'teacher')
# Customise layout
layout = layout_nicely(school_subgraph)
# Get degrees for plotting
degrees <- degree(school_subgraph)
# Plot!
plot(school_subgraph, layout = layout, vertex.label = NA, vertex.size = sqrt(degrees + 2))
# Set legend to show how colours correspond to roles.
legend("bottomright", legend = levels(as.factor(V(school_subgraph)$role)), fill = colors)
title(paste("School Network Subsample, n=", as.character(sample_size)))
```

School Network Subsample, n= 50

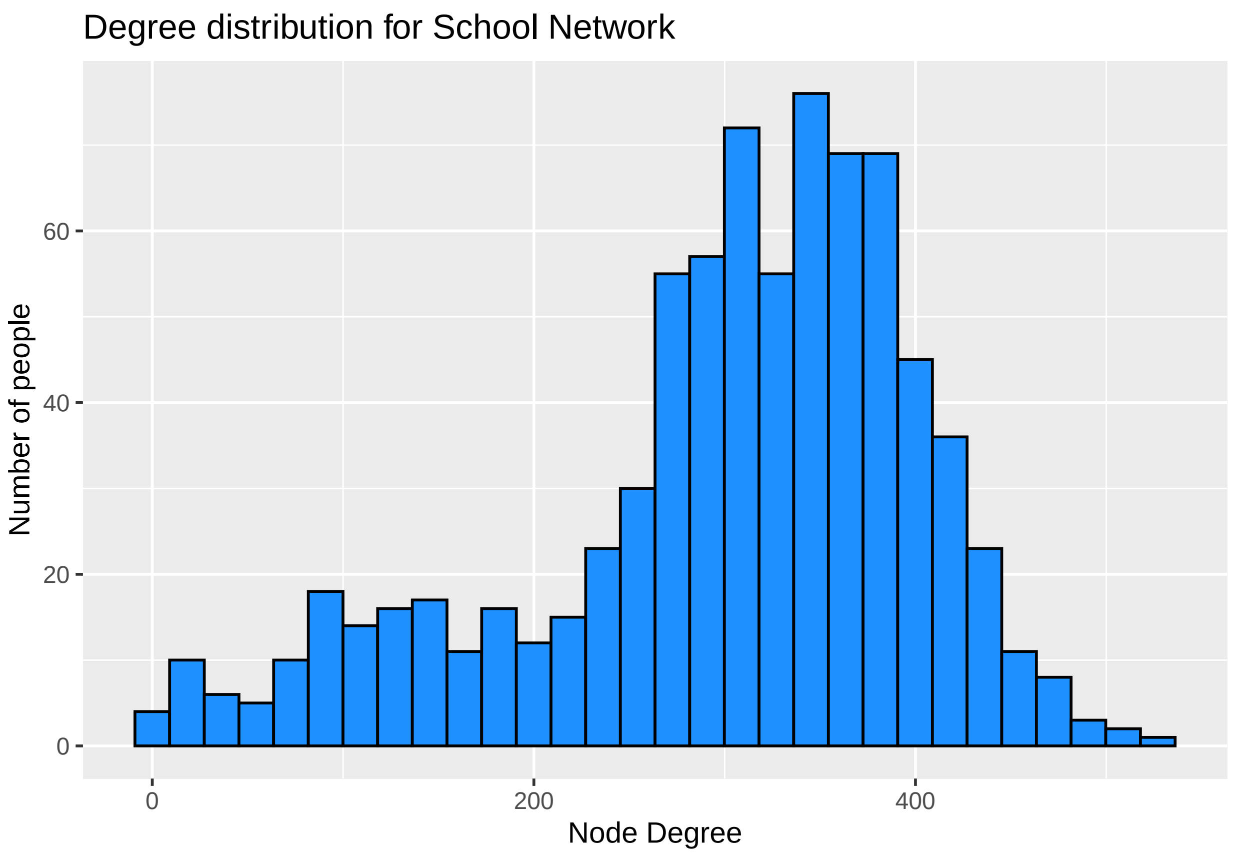


When we sample the vertices, we are excluding full data about the network.

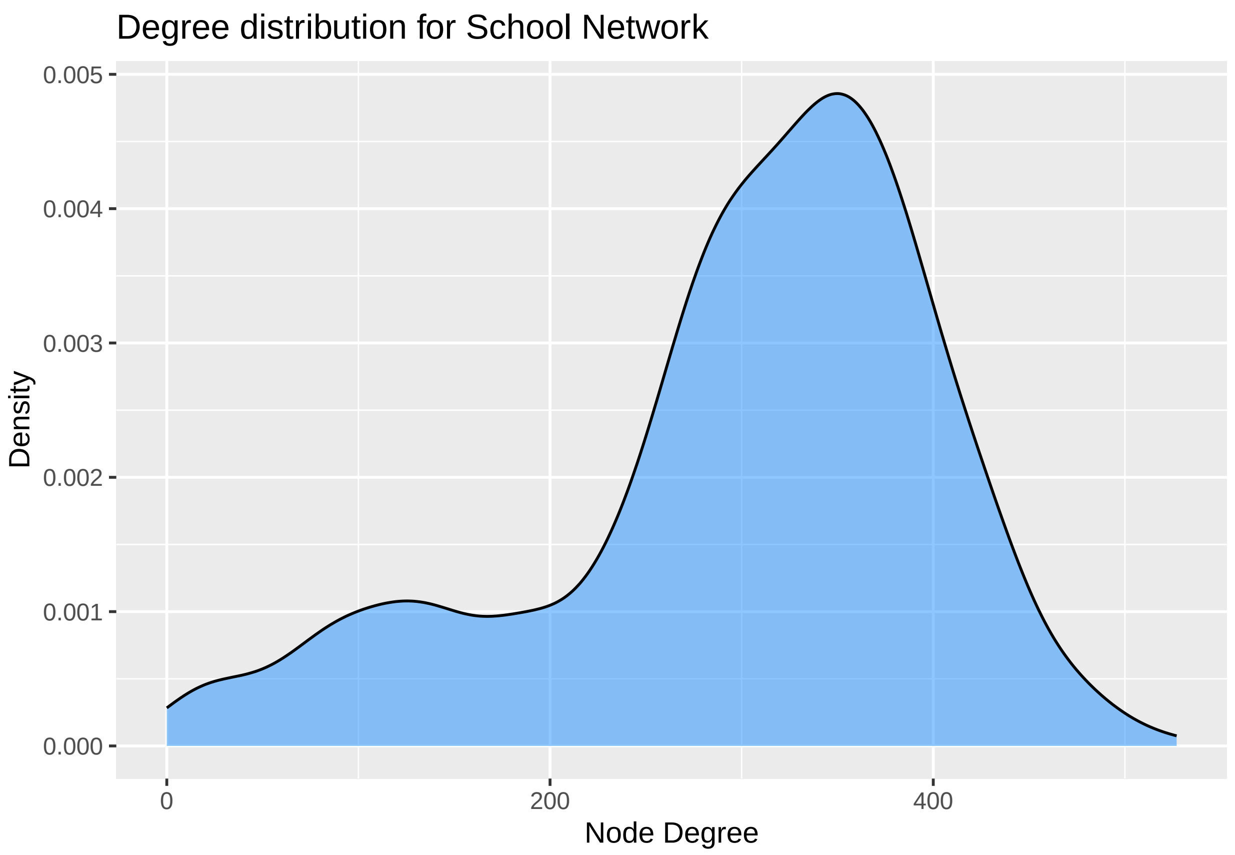
As we discussed in lectures, one way to visualise information about the whole graph is to show the degree distribution.

Plot the degree distribution for the whole graph. Do this once for every node, and then repeat with the distributions for each separate role. Do the degree distributions look significantly for the different roles?

```r
nodes$degree <- degree(school_graph)

degree_dist <- ggplot(nodes, aes(x = degree)) +
  geom_histogram(fill = 'dodgerblue', color = 'black') +
  labs(
    x = "Node Degree",
    y = "Number of people"
  ) + ggtitle("Degree distribution for School Network")
degree_dist
```

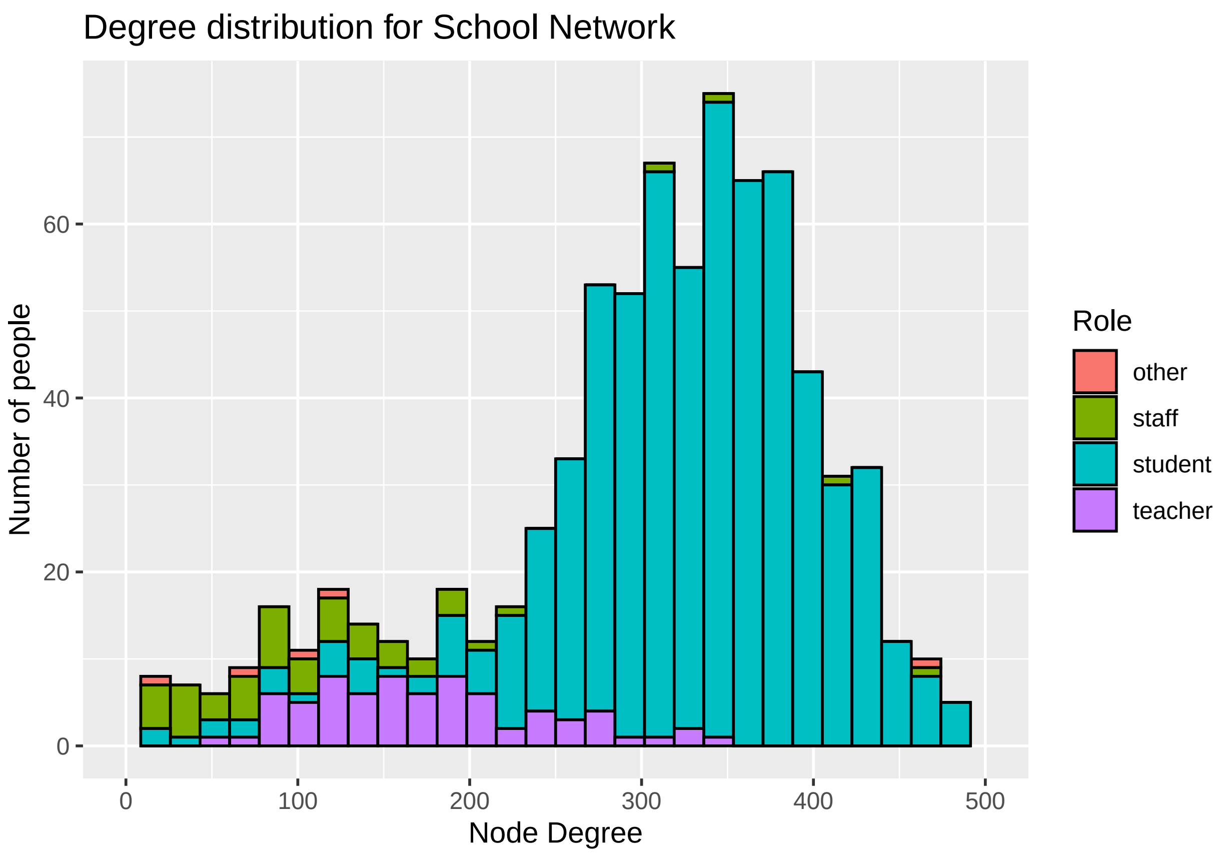


```
degree_density <- ggplot(nodes, aes(x = degree)) +
  geom_density(fill = 'dodgerblue', color = 'black', alpha = 0.5) +
  labs(
    x = "Node Degree",
    y = "Density"
  ) + ggtitle("Degree distribution for School Network")
degree_density
```

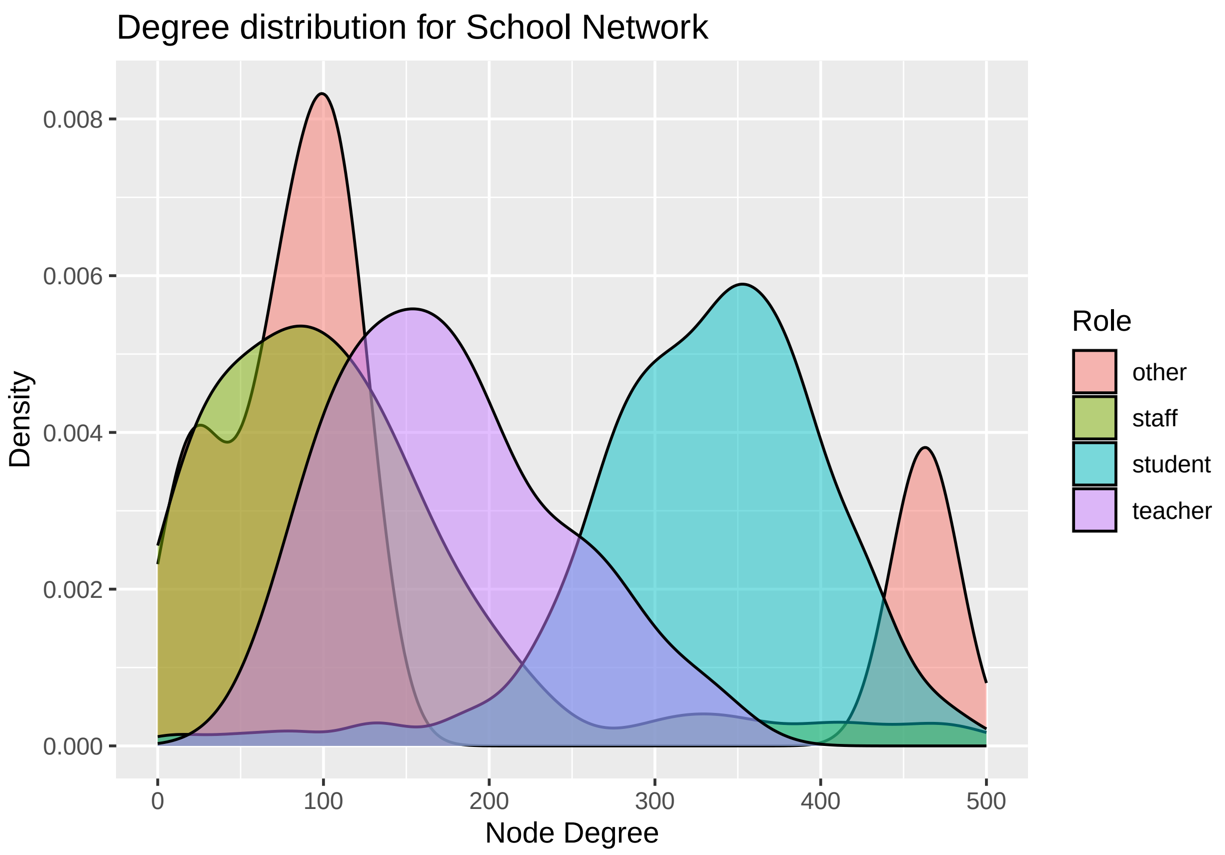


Now show the distributions separated by roles. Compare the histogram with the density plot - which do you prefer?

```
degree_dist_role <- ggplot(nodes, aes(x = degree, fill = Role)) +
  geom_histogram(color = 'black') +
  labs(
    x = "Node Degree",
    y = "Number of people"
  ) + ggtitle("Degree distribution for School Network") + xlim(0, 500)
show(degree_dist_role)
```



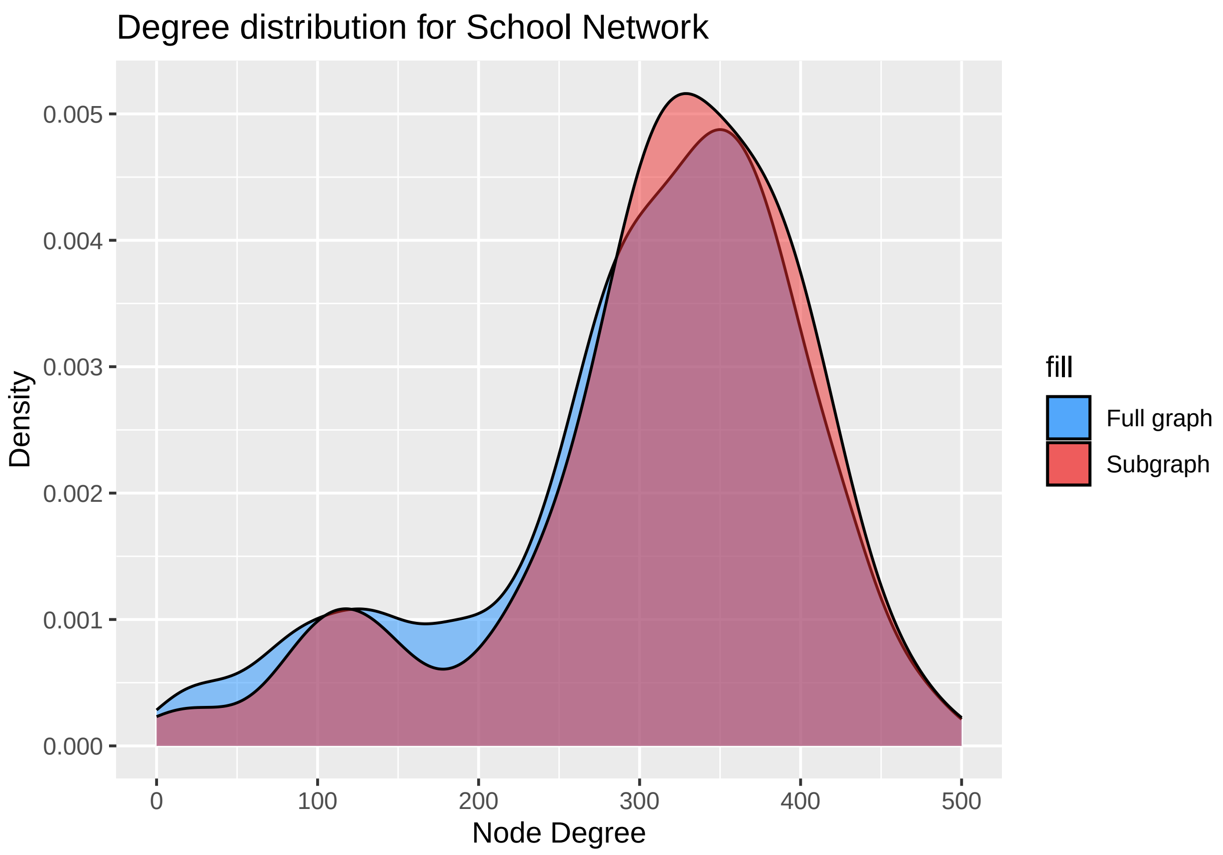
```
degree_density_role <- ggplot(nodes, aes(x = degree, fill = Role)) +
  geom_density(color = 'black', alpha = 0.5) +
  labs(
    x = "Node Degree",
    y = "Density"
  ) + ggtitle("Degree distribution for School Network") + xlim(0, 500)
show(degree_density_role)
```



Let's also check whether the subgraph is a good representation of the full graph in terms of degree distribution. Plot the degree distributions for the two graphs in a single plot.

Do they look similar?

```
subgraph_node_degrees <- nodes %>% filter(ID %in% subgraph_nodes) %>%
  select(ID, degree)

color_manual <- c("Full graph" = 'dodgerblue', "Subgraph" = 'firebrick2')

subgraph_degree_comparison <- ggplot(nodes, aes(x = degree, fill = 'Full graph')) +
  geom_density(alpha = 0.5, color = 'black') +
  geom_density(data = subgraph_node_degrees, aes(x = degree, fill = 'Subgraph'),
               color = 'black', alpha = 0.5) +
  labs(
    x = "Node Degree",
    y = "Density"
  ) + ggtitle("Degree distribution for School Network") + xlim(0, 500) +
  scale_fill_manual(values = color_manual)

subgraph_degree_comparison
```



DAGs

Let's look at creating a DAG in dagitty: Create a simple DAG with 5 variables, called A, B, C, D and E, and make up some associations between the variables.

If you prefer, you could make a DAG from your real research!

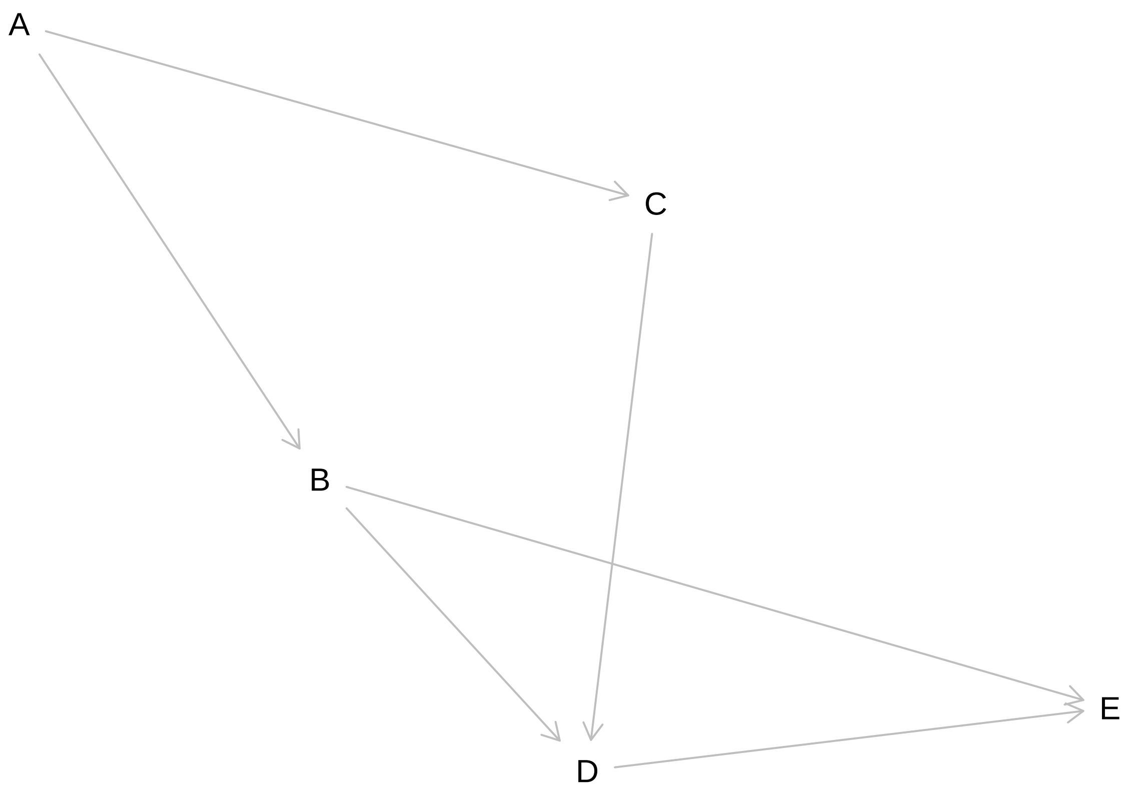
If you want to, you can also try playing around with the features in ggdag - this requires you to make a dagify() object instead. See this tutorial for some ideas of how to start https://cloud.r-project.org/web/packages/ggdag/vignettes/intro-to-ggdag.html and the kinds of functions available.

```r
# Create a DAG using dagitty
dag_example<- dagitty("dag{
                        A
                        B
                        C
                        D
                        E
                        A -> B
                        A -> C
                        B -> D
                        B -> E
                        C -> D
                        D -> E

                        }")

# Check that it is acyclic!! Then plot:
```

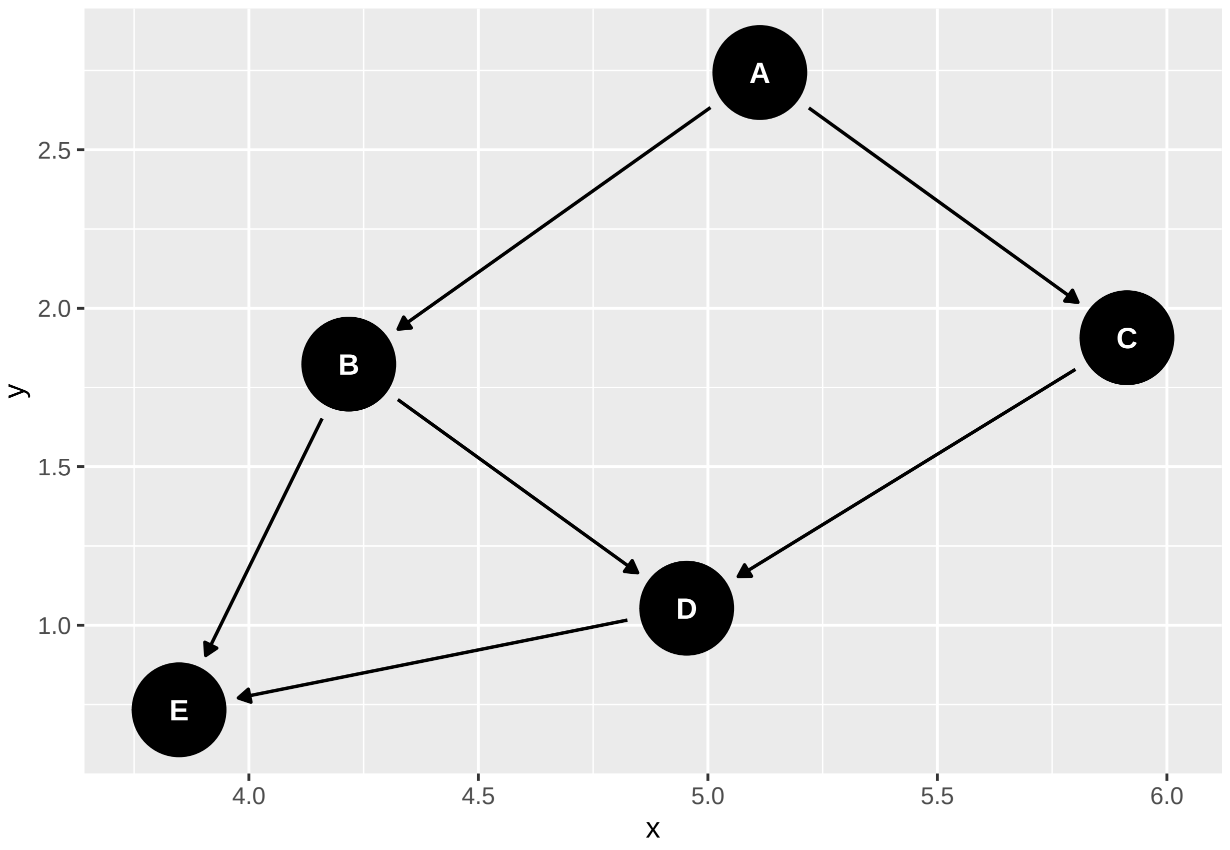
```r
print(isAcyclic(dag_example))
```

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

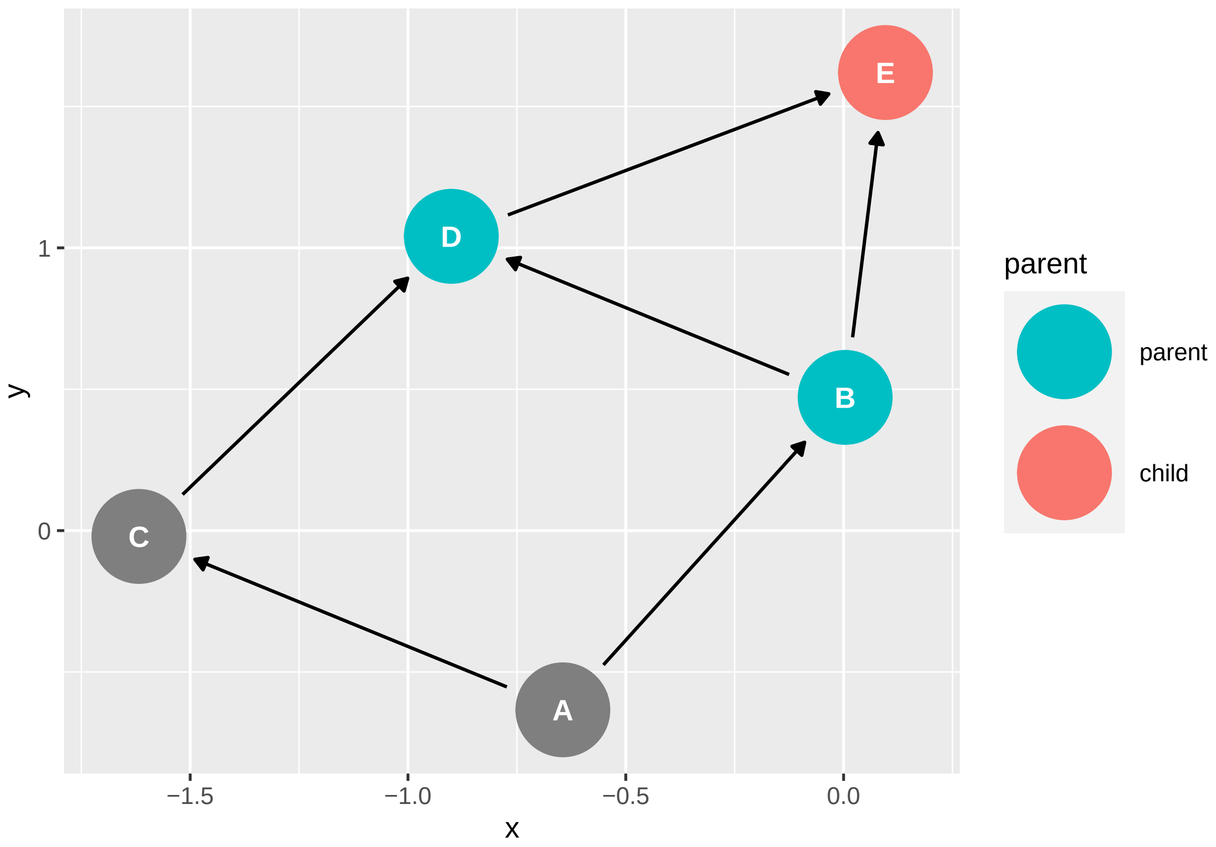
```r
plot(dag_example)
```



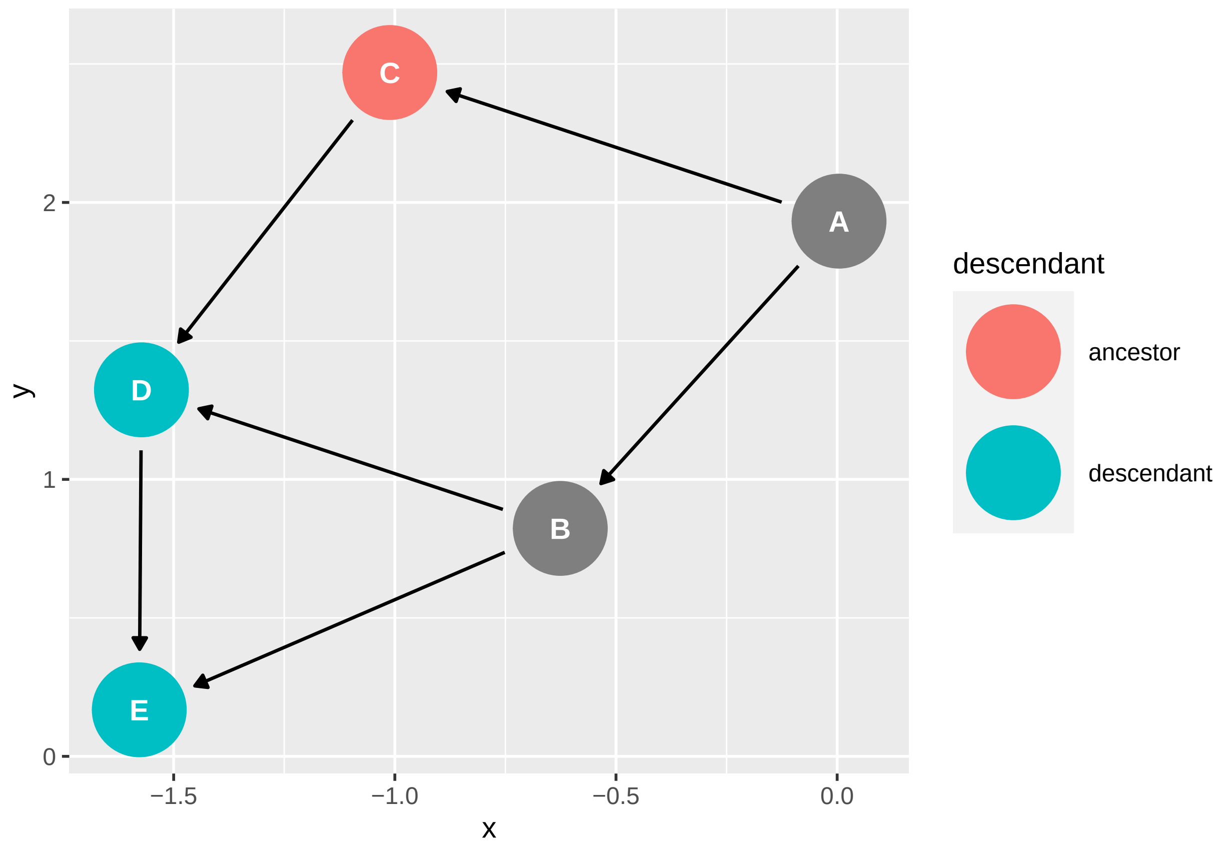
```r
# Create the same DAG using dagify

ggdag_example <- dagify(E ~ B+D,
                        D ~ B+C,
                        C ~ A,
                        B ~ A,
                       exposure = "A",
                       outcome = "E"
)

# Create some nice ggdag plots
show(ggdag(ggdag_example))
```



```
show(ggdag_parents(ggdag_example, "E"))
```



```
show(ggdag_descendants(ggdag_example, "C"))
```


C
A
D
B
E
2
1
0
y
−1.5
−1.0
−0.5
0.0
x
descendant
ancestor
descendant