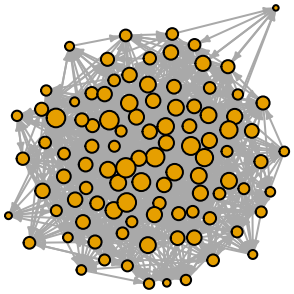
Joseph Risi - Assignment 1

```
# Load in necessary packages and set the seed.

set.seed(123)
library(tidyr)
library(dplyr)
library(igraph)
library(ggplot2)
```

As shown in the plots below, a random graph where edges are created with a constant probability does not particularly benefit from the Fruchterman-Reingold algorithm.

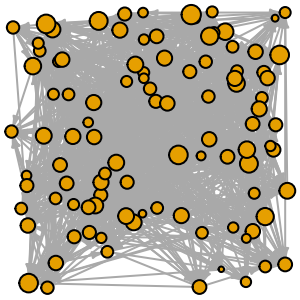
```
# Create a random graph where the probability = 0.1 that an edge will be created
graph <- sample_gnp(100, 0.1, directed = T)

# Set the size of nodes to be proportional to their degree
V(graph)$size <- degree(graph) / 2

# Shrink the arrow sizes
E(graph)$arrow.size <- 0.25

# Save the layouts for Fruchterman-Reingold layout and the random layout
fr_layout_gr <- layout_with_fr(graph)
random_layout_gr <- layout_randomly(graph)

# Plot the graph using the Fruchterman-Reingold layout
plot(graph,
     layout = fr_layout_gr,
     vertex.label = NA,
     main = paste0("Constant Probability-Edge Network displayed\n",
                   "using FR algorithm"))
```

Constant Probability−Edge Network displayed using FR algorithm



```
# Plot the graph using the random layout
plot(graph,
     layout = random_layout_gr,
     vertex.label = NA,
     main = paste0("Constant Probability-Edge Network displayed\n",
                   "randomly"))
```

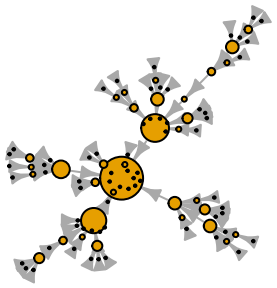
Constant Probability−Edge Network displayed randomly



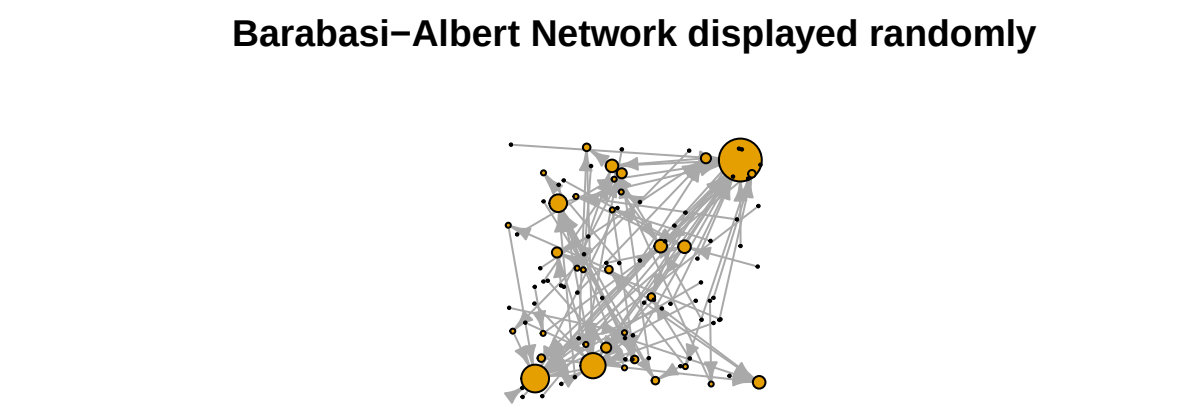
As shown in the plots below, a random graph created using the Barbasi-Albert model (which incorporates preferential attachment) **does** benefit from the Fruchterman-Reingold algorithm.

```
# Generate a graph using the Barabasi-Albert model which takes into account
# preferential attachment meaning popular nodes get more popular over time.
fr_graph <- sample_pa(100)

# Set the size of nodes to be equal to double their degree
V(fr_graph)$size <- degree(fr_graph) * 2

# Shrink the arrow size
E(fr_graph)$arrow.size <- 0.5

# Save the layouts for Fruchterman-Reingold layout and the random layout
fr_layout_fr <- layout_with_fr(fr_graph)
random_layout_fr <- layout_randomly(fr_graph)

# Plot the graph using the Fruchterman-Reingold layout
plot(fr_graph,
     layout = fr_layout_fr,
     vertex.label = NA,
     main = "Barabasi-Albert Network displayed using FR algorithm")
```

Barabasi−Albert Network displayed using FR algorithm



```
# Plot the graph using the random layout
plot(fr_graph,
     layout = random_layout_fr,
     vertex.label = NA,
     main = "Barabasi-Albert Network displayed randomly")
```

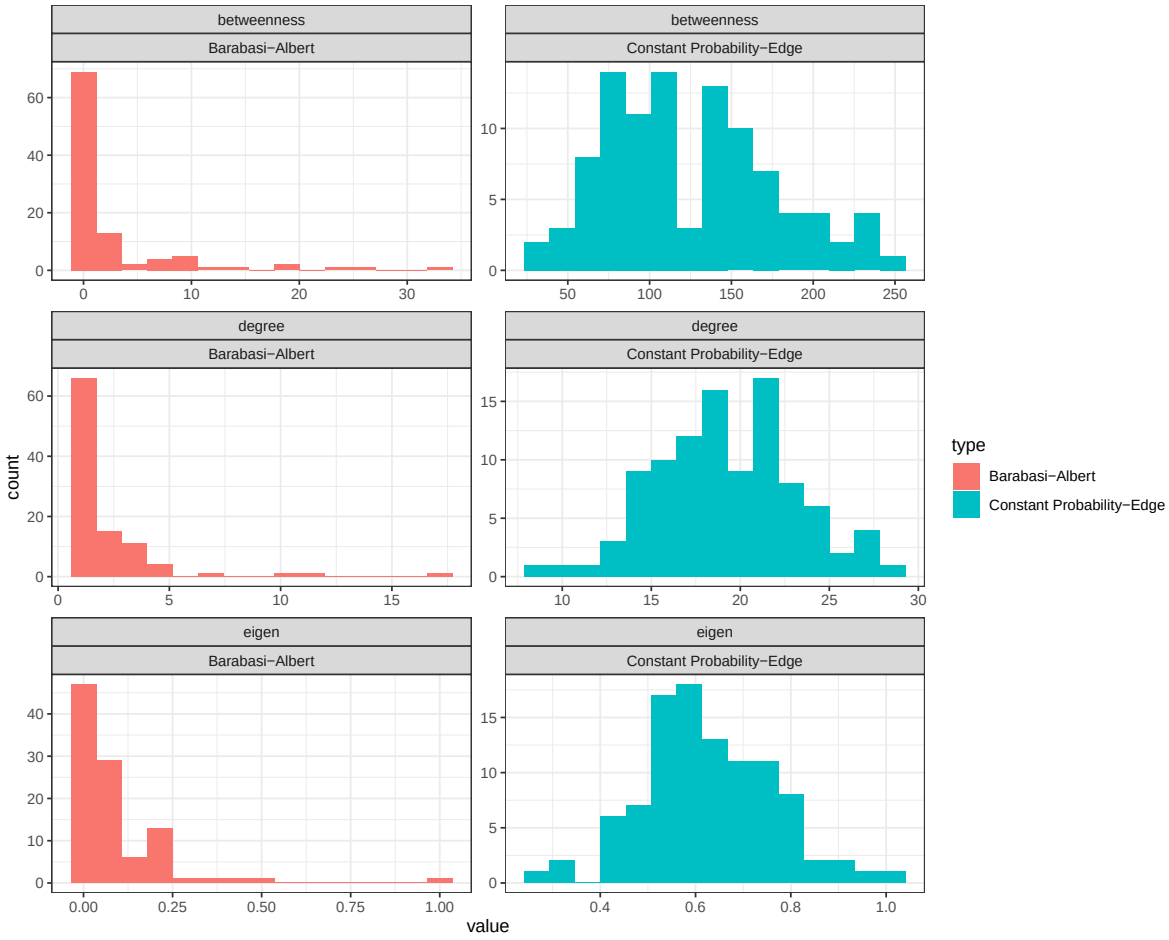


The question is why does the Fruchterman-Reingold algorithm make a difference for the Barabasi-Albert network but not the other network? To answer this question, we will look at the distribution of centrality scores for these two networks.

Notice how for each of the centrality measures (degree, betweenness, and eigen-centrality) the distributions for the constant probability-edge networks look more or less normal meaning most nodes are pretty well connected to each other. Meanwhile the Barabasi-Albert networks have extreme right skews meaning most nodes are not very central to the network while only a few are very central. This is by design and is meant to illustrate the strengths of the Fruchterman-Reingold algorithm. When you have very clustered data (in the sense that groups of nodes are clustered around each other or you have some very central nodes which other nodes cluster around), the Fruchterman-Reingold algorithm will shine and help humans perceive those clusters. When you do not have a high degree of clustering and/or there are no nodes acting as hubs, the Fruchterman-Reingold algorithm is not going to be as useful for humans in discerning patterns in the social network.

```
# calculate degree centrality, betweenness centrality, and eigen-centrality for
# the constant probability-edge graph
graph_stats <-
    tibble(degree = degree(graph),
           betweenness = betweenness(graph),
           eigen = eigen_centrality(graph)$vector,
```

```
            type = "Constant Probability-Edge")

# calculate degree centrality, betweenness centrality, and eigen-centrality for
# the constant probability-edge graph
fr_graph_stats <-
    tibble(degree = degree(fr_graph),
           betweenness = betweenness(fr_graph),
           eigen = eigen_centrality(fr_graph)$vector,
           type = "Barabasi-Albert")

# Bind the two data frames together and make it a long data frame
stats <-
    bind_rows(graph_stats, fr_graph_stats) %>%
    pivot_longer(cols = degree:eigen,
                 names_to = "measure",
                 values_to = "value")

# Compare each of the centrality measures for each of the graph types
ggplot(stats, aes(x = value)) +
    geom_histogram(bins = 15, aes(fill = type)) +
    facet_wrap(~measure*type, scales = "free", ncol = 2) +
    theme_bw()
```


betweenness
Barabasi−Albert
betweenness
Constant Probability−Edge
degree
Barabasi−Albert
degree
Constant Probability−Edge
eigen
Barabasi−Albert
eigen
Constant Probability−Edge
count
value
type
Barabasi−Albert
Constant Probability−Edge