Assignment 2 - Joseph Risi

```r
# load packages
library(dplyr)
library(purrr)
library(tidyr)
library(igraph)
library(ggplot2)
```

```r
# 1000 nodes
n <- 1000

# directed graphs
directed <- T

# 10 simulation runs
nr_runs <- 1:10

# number of edges to add in each time step for Barabási-Albert networks
m <- 20
```

Set up Barabási-Albert graphs

```r
# set random seed
set.seed(42)

# simulate 10 Barabási-Albert networks
ba_graphs <- map(nr_runs,
                 sample_pa, n = n, m = m, directed = directed)
names(ba_graphs) <- paste0("run", "_", nr_runs)
```

```
# Find the density of each of the networks
density_ba <- map_dbl(ba_graphs, edge_density)
```

Density value(s) for the Barabási-Albert graphs: 0.0198098‘

Set up Erdős–Rényi graphs

```
# Set random seed
set.seed(42)

# simulate 10 Erdős-Rényi graphs
er_graphs <- map(nr_runs,
                 sample_gnp, n = n, p = unique(density_ba), directed = directed)
names(er_graphs) <- paste0("run", "_", nr_runs)
```

Calculate descriptive statistics for each of the graphs

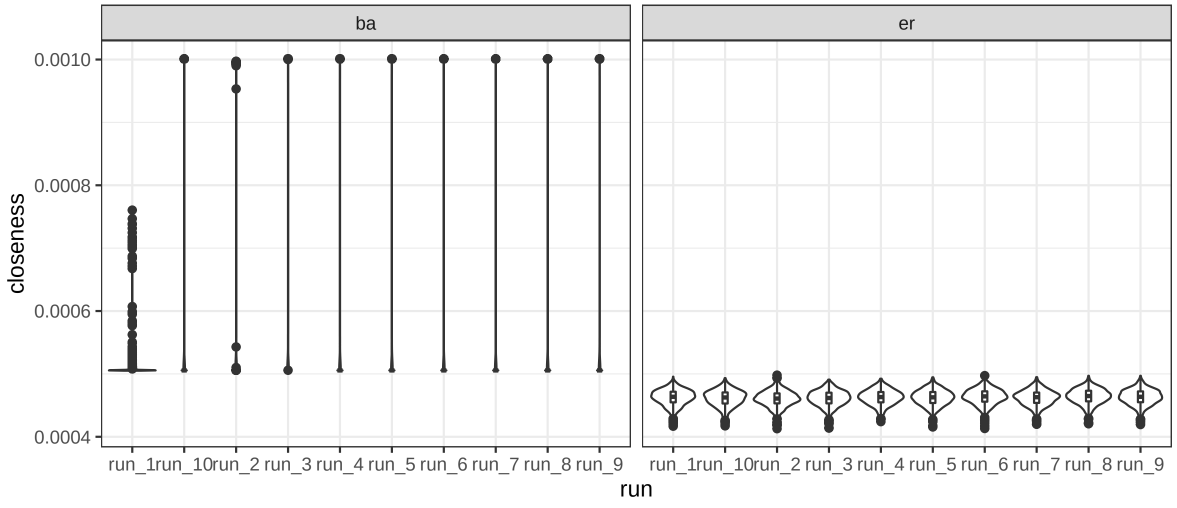
```
# Function used to make our data frames long and tidy, easier to graph in ggplot
Convert_To_Df <- function(df, statistic, graph) {

    df %>%
        pivot_longer(everything(),
                     names_to = "run",
                     values_to = statistic) %>%
        mutate(graph = graph)
    }
```

Closeness Centrality

In the Barabási-Albert graphs, we see nodes have higher average closeness. This makes sense because Barabási-Albert graphs form *hubs* forms which can be useful for connecting otherwise very distant groups of nodes. In Erdős–Rényi by contrast, each node forms a tie with another node with a constant probability. It is very unlikely for hubs to form as a result, and it can be hard for otherwise very distant ties to then be able to reach other.

```
# get rid of scientific notation
options(scipen = 5)
```

```r
# Calculate closeness centrality for each of the Barabási-Albert graphs
# And then make resulting data frame long and tidy
closeness_ba <-
    map_df(ba_graphs, igraph::closeness, mode = "total") %>%
    Convert_To_Df("closeness", "ba")

# Calculate closeness centrality for each of the Erdős-Rényi graphs
# And then make resulting data frame long and tidy
closeness_er <-
    map_df(er_graphs, igraph::closeness, mode = "total") %>%
    Convert_To_Df("closeness", "er")

closeness <- bind_rows(closeness_ba, closeness_er)

ggplot(closeness, aes(x = run, y = closeness)) +
    geom_violin() +
    geom_boxplot(width = 0.1) +
    facet_wrap(~graph) +
    theme_bw()
```



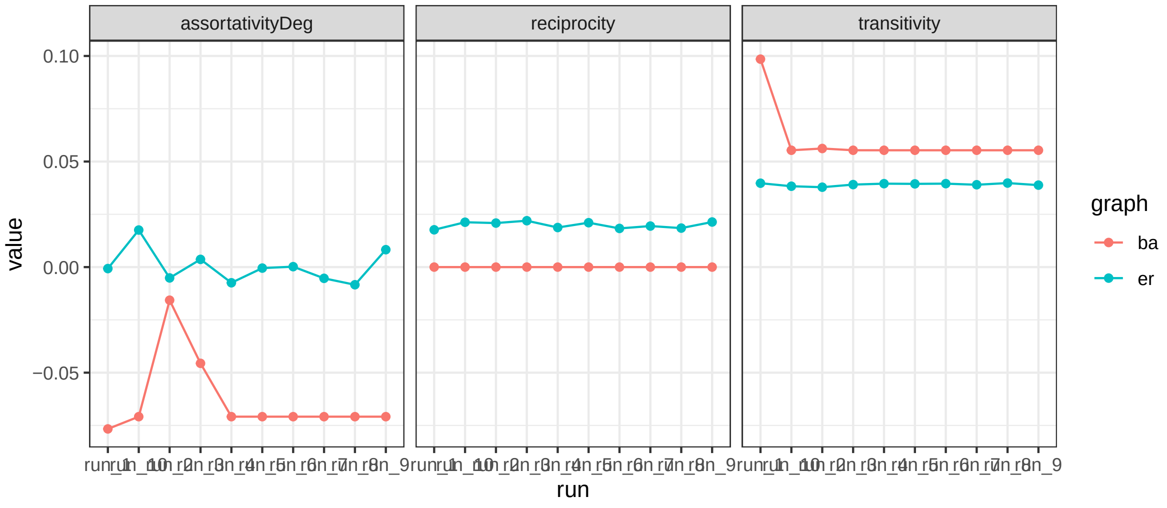
Reciprocity, Transitivity, Degree Assortativity

- **Assortativity**
  - We see for the Barabási-Albert graphs there is actually a slight negative correlation between degree size and the likelihood of forming a tie. This is likely because

Barabási-Albert graphs are created using preferential attachment i.e., the rich get richer. Thus, those nodes with the most connections are connected to many "unpopular" nodes with very low degree centrality. As a result, popular nodes are actually slightly more likely to have a tie form with an unpopular node than with another popular node.
  - We see for the Erdős–Rényi graphs there is almost no correlation between degree centrality and the likelihood of forming a tie. This makes sense because ties are created randomly with a constant probability. Attributes of the nodes (e.g. node degree or popularity) has no bearing on whether or not a tie actually forms.

- **Reciprocity**
  - We see there is no reciprocity in the Barabási-Albert graphs. This is because a popular node is never going to send a tie back to a less popular node. In comparison, Erdős–Rényi graphs have more reciprocity, but in an absolute sense it is still quite small. Again, this is because ties form randomly with a constant probability. The fact that node 1 sent a tie to node 2 does not affect the probability that node 2 will send a tie back to node 1. In fact, the values for reciprocity are nearly equivalent to the probability of forming a tie, generally.
- **Transitivity**
  - We see higher levels of transitivity in the Barabási-Albert graphs as compared to the Erdős–Rényi graphs. This is again likely capturing the dynamic that relatively unpopular nodes connect to the relatively more popular nodes. Thus because so many nodes have in common a connection to the hub, it becomes relatively easier to form a transitive triangle. Relative, of course, to the Erdős–Rényi graphs. In the Erdős–Rényi graphs, nodes form ties following a constant probability. Thus, it is relatively harder to have a transitive triangle emerge in such a situation when ties are formed independently of each other.

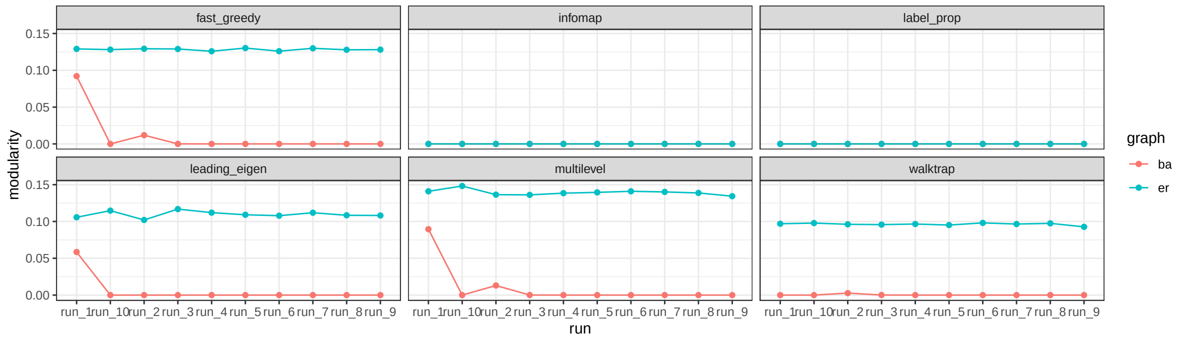
```r
# Calculate reciprocity, transitivity, and assortativity for each Barabási-Albert graph
reciprocity_ba <- map_df(ba_graphs, reciprocity)
transitivity_ba <- map_df(ba_graphs, transitivity)
assortativityDeg_ba <- map_df(ba_graphs, assortativity.degree)

# Calculate reciprocity, transitivity, and assortativity for each Erdős-Rényi graph
reciprocity_er <- map_df(er_graphs, reciprocity)
transitivity_er <- map_df(er_graphs, transitivity)
assortativityDeg_er <- map_df(er_graphs, assortativity.degree)

# Combine all observations into one, long, tiday data frame.
rta <-
    pmap(list(list(reciprocity_ba, transitivity_ba, assortativityDeg_ba,
                    reciprocity_er, transitivity_er, assortativityDeg_er),
               list("reciprocity", "transitivity", "assortativityDeg",
                    "reciprocity", "transitivity", "assortativityDeg"),
               list("ba", "ba", "ba", "er", "er", "er")),
             Convert_To_Df) %>%
    reduce(bind_rows) %>%
    pivot_longer(cols = c(reciprocity, transitivity, assortativityDeg),
                 names_to = "statistic",
                 values_to = "value") %>%
    filter(!is.na(value))

ggplot(rta, aes(x = run, y = value)) +
    geom_point(aes(color = graph)) +
    geom_line(aes(color = graph, group = graph)) +
    facet_wrap(~statistic) +
    theme_bw()
```



Community Detection

For every algorithm, each algorithm reports either a higher modularity score for Erdős–Rényi graphs or the same modularity score (i.e., a score of 0) for both Erdős–Rényi graphs and Barabási-Albert graphs. The modularity scores are quite low for both sets of graphs indicating there is likely not any communities which is what we would suspect. However, it is understandable that the Barabási-Albert graphs would have even lower modularity scores because the hubs are likely connected to many of the nodes and many of the nodes are likely connected to the hubs. Since community detection algorithms try to find nodes who are more likely to be tied to each other than to other nodes, the dynamics in the Barabási-Albert graphs make it hard for this dynamic to emerge.

```r
# Function for calculating the modularity for each community detection algorithm
CommunityComparison <- function(graph) {

    graph <- as.undirected(graph, mode = "collapse")
    fast_greedy <- cluster_fast_greedy(graph)
    infomap <- cluster_infomap(graph)
    leading_eigen <- cluster_leading_eigen(graph)
    label_prop <- cluster_label_prop(graph)
    multilevel <- cluster_louvain(graph)
    walktrap <- cluster_walktrap(graph)

    modularities <- map_df(list("fast_greedy" = fast_greedy,
                                "infomap" = infomap,
                                "leading_eigen" = leading_eigen,
                                "label_prop" = label_prop,
                                "multilevel" = multilevel,
                                "walktrap" = walktrap),
                           modularity)
}

community_ba <- map(ba_graphs, CommunityComparison)
community_ba <-
    pmap_dfr(list(community_ba, as.list(names(community_ba))),
             function(df, run) {df %>% mutate(run = run)}) %>%
    pivot_longer(-run, names_to = "communityAlgo", values_to = "modularity") %>%
    mutate(graph = "ba")

community_er <- map(er_graphs, CommunityComparison)
community_er <-
    pmap_dfr(list(community_er, as.list(names(community_er))),
             function(df, run) {df %>% mutate(run = run)}) %>%
    pivot_longer(-run, names_to = "communityAlgo", values_to = "modularity") %>%
    mutate(graph = "er")

community <- bind_rows(community_ba, community_er)

ggplot(community, aes(x = run, y = modularity)) +
    geom_point(aes(color = graph)) +
    geom_line(aes(color = graph, group = graph)) +
    facet_wrap(~communityAlgo) +
    theme_bw()
```


fast_greedy
infomap
label_prop
leading_eigen
multilevel
walktrap
modularity
run_1
run_10
run_2
run_3
run_4
run_5
run_6
run_7
run_8
run_9
run
graph
ba
er