Exploring Relationships Between Network Properties: A Regression Analysis

Lamichhane, Bishal
University of Nevada, Reno
blamichhane@unr.edu

December 19, 2023

Abstract

Networks model complex relationships in a system. Network statistics, such as centrality measures, characterize different aspects of these relationships, highlighting nodes that are important to the network. In this project, we explore the relationships between various node and network properties using regression. We conclude that the structure of the graph or network has the most significant impact on the effectiveness of establishing a regression relationship. Furthermore, we find that in ER graphs, a single centrality measure suffices to predict the response variable and explain all the variation in the model. However, in the case of real-world networks, even the use of all centrality measures does not fully account for the variation in the model.

1 Introduction

Networks represent various complex systems, from social connections to biological systems. In the interconnected world of networks, the robustness of systems to targeted attacks or random failures becomes significant. From understanding the spread of epidemics to ensuring the reliability of supply chains, the ability to predict and mitigate the impact of node removal is invaluable. Removal of nodes from a network changes the network structure. Depending on the position and "importance" of a node, the removal might have a different effect in the network. For instance while modeling a infectious disease, we want to figure out which nodes or set of nodes have the biggest role in spreading the disease and hence removing those nodes will delay the spread. Centrality measures provide the relative importance of the nodes in the network. This project aims to empirically investigate the relationship between various centrality measures and overall network robustness.

1.1 Objective

The primary objective is to use regression analysis to determine if node-level centrality and graph-level spectral measures can predict the robustness of a network following the removal of a node. This will involve:

- Analyzing a theoretical graph model with alterations in parameters like edge probability p.
- Investigating the effect of node removal on network robustness using a measure robust to disconnectedness.
- Comparing findings across multiple instances of graph types of the same size to maintain consistency

The main question that I want to answer from this project is : **Does the choice of graph model have a greater impact on the regression results compared to the local properties of the nodes, or is it the opposite?**

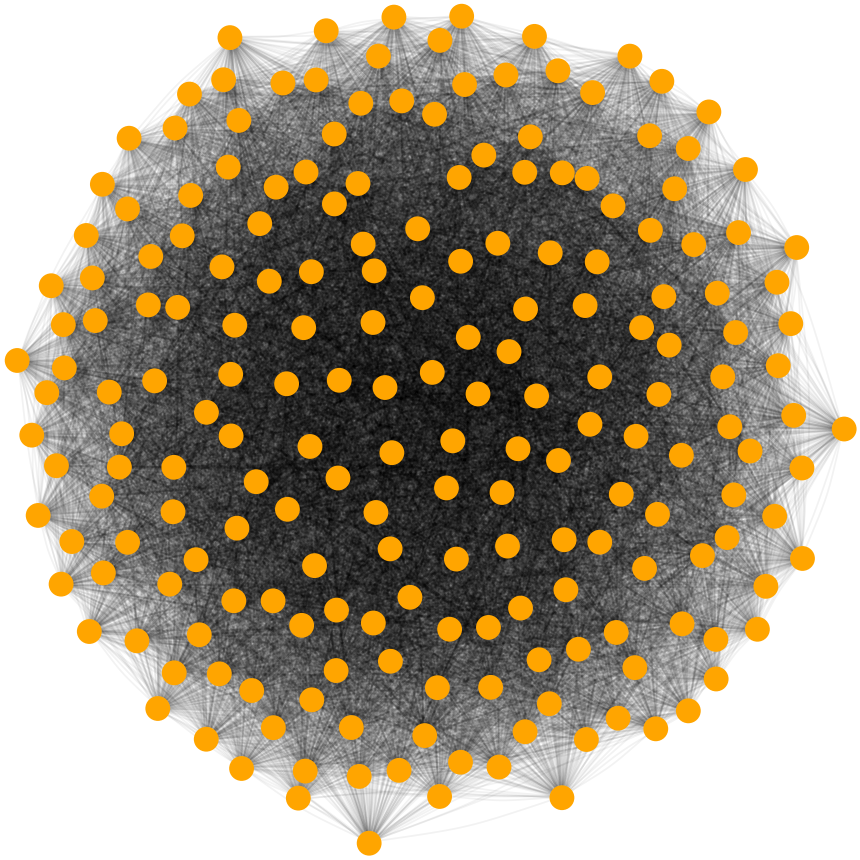
2 Methods

2.1 Network and graphs

Networks can be described as graphs $G = (V, E)$ where V represents a set of vertices (or nodes) and E is a set of edges (or links) connecting these vertices. The power of the network lies in the relationship that we can establish via the edges. While theoretical graphs differ structurally from most applied graphs, they have a rigorous theory that allow the study the empirical and theoretical relationship more clearly since we are able to control the parameters to see their effects in the network properties. In this project, we study Erdős–Rényi graph model. In the Erdős–Rényi model $G(n, p)$, a graph is formed by starting with a set of n

vertices and adding edges between each pair of vertices independently with probability p. Formally, for a fixed n and $p \in [0, 1]$, each of the $\binom{n}{2}$ possible edges between the vertices is included in the graph with probability p, independent of other edges.

The main goal of this project is to study how p impacts the relationship between network properties with respect to regression. The plot below shows one instance of Erdős–Rényi (ER) graph with $n = 100$ and $p = 0.5$.



2.2 Data

This project doesn't use any external data. We calculate different centrality measures and network properties from the graph created above. Below are the centrality measures and spectral properties under consideration:

1. **Closeness Centrality**: Node's average distance to all other nodes.
2. **Betweenness Centrality**: Frequency a node appears on shortest paths between other nodes.
3. **Eigenvector Centrality**: Node's connectedness to other well-connected nodes.
4. **Page Rank Centrality**: Assesses a node's importance based on the structure of incoming links.
5. **Harmonic Centrality**: Reciprocal of the shortest path distances.
6. **Local Clustering Coefficient**: Measures how close a node's neighbors are to form a clique.
7. **Subgraph Centrality**: Accounts for the number of closed walks centered at a node.
8. **Spectral Radius:** The largest absolute eigenvalue of the adjacency matrix , which has implications for the graph's robustness.

2.2.1 Data for Regression

Once the network and node level properties are calculated, we process it for regression. We consider two different network measures as our response variable.

Response Variables:

- Inverse difference of the spectral radius of the graph before and after removal of a node: Reflects the impact of node removal on network connectivity, a key aspect of network robustness.
- **Triangle Count**: Number of times a node is part of a triangle: Indicates the node's role in local network cohesion, which is crucial for understanding network stability and resilience.

Rest of the measures discussed above are used as predictor variables. ER graphs are homogeneous and hence removal of one node doesn't seem to affect the Spectral Radius value. Hence, we use Triangle Count as the response variable for ER graphs.

2.2.2 Data Creation

First we create two instances of ER graphs as follow.

```
set.seed(323423)
A <- erdos.renyi.game(n =200 , p = 0.75)

set.seed(223432)
B<- erdos.renyi.game(n=200, p = 0.75)
```

The following functions are used to generate the data frame.

```
# Function to calculate the robustness measure based on spectral radius
calculate_spectral_radius_robustness <- function(graph) {
  original_spectral_radius <- max(eigen(as_adjacency_matrix(graph),
                                        symmetric = TRUE)$values)
  robustness_measure <- numeric(vcount(graph))

  for (v in V(graph)) {
    # Remove node and calculate new spectral radius
    g_modified <- delete_vertices(graph, v)
    new_spectral_radius <- max(eigen(as_adjacency_matrix(g_modified),
                                     symmetric = TRUE)$values)

    # Calculate difference and add a small constant if necessary
    diff_spectral_radius <- original_spectral_radius - new_spectral_radius
    if (abs(diff_spectral_radius) < 1e-5) {
      diff_spectral_radius <- diff_spectral_radius + 1e-5
    }

    # Robustness measure is the inverse of this difference
    robustness_measure[v] <- 1 / diff_spectral_radius
  }

  return(robustness_measure)
}

# Function to calculate spectral properties of the graph
calculate_spectral_properties <- function(g) {
  laplacian_matrix <- laplacian_matrix(g)
  eigenvalues <- eigen(laplacian_matrix, symmetric = TRUE)$values
  sorted_eigenvalues <- sort(eigenvalues, decreasing = TRUE)

  spectral_radius <- max(sorted_eigenvalues) #Largest eigen value
  spectral_gap <- sorted_eigenvalues[1] - sorted_eigenvalues[2]
  algebraic_connectivity <- sorted_eigenvalues[length(sorted_eigenvalues) - 1]

  return(list(spectral_radius = spectral_radius, spectral_gap = spectral_gap,
              algebraic_connectivity = algebraic_connectivity))
}
```

After the spectral properties are calculated,we calculate node properties like centrality measures below. The function takes an igraph object and creates a dataframe that contains all the node properties and spectral properties.

```
calculate_centrality_and_robustness <- function(g,
                                                graph_type,
                                                existing_df = NULL) {
  # Calculate centrality measures
  closeness_centrality <- closeness(g)
  betweenness_centrality <- betweenness(g)
  eigenvector_centrality <- evcent(g)$vector
  pagerank_centrality <- page_rank(g)$vector
```

```
  # Additional node properties
  harmonic_centrality <- harmonic_centrality(g)
  local_clustering_coefficient <- transitivity(g, type = "local")
  subgraph_centrality <- subgraph_centrality(g)
  # Triangle count
  triangle_count <- count_triangles(g)

  # Calculate robustness measures
  robustness_measures <- calculate_spectral_radius_robustness(g)

  # Calculate spectral properties
  spectral_properties <- calculate_spectral_properties(g)

  # Assign node IDs based on graph type
  node_ids <- paste0(graph_type, 1:vcount(g))

  # Create new data frame with calculated values
  new_data <- data.frame(Node = node_ids,
                         ClosenessCentrality = closeness_centrality,
                         BetweennessCentrality = betweenness_centrality,
                         EigenvectorCentrality = eigenvector_centrality,
                         PageRankCentrality = pagerank_centrality,
                         TriangleCount = triangle_count,
                         HarmonicCentrality = harmonic_centrality,
                         LocalClusteringCoefficient = local_clustering_coefficient,
                         SubgraphCentrality = subgraph_centrality,
                         GraphType = substr(graph_type, 1, 1),
                         RobustnessMeasure = robustness_measures)

  # Append new data to existing dataframe or create a new one
  if (is.null(existing_df)) {
    return(new_data)
  } else {
    return(rbind(existing_df, new_data))
  }
}
```

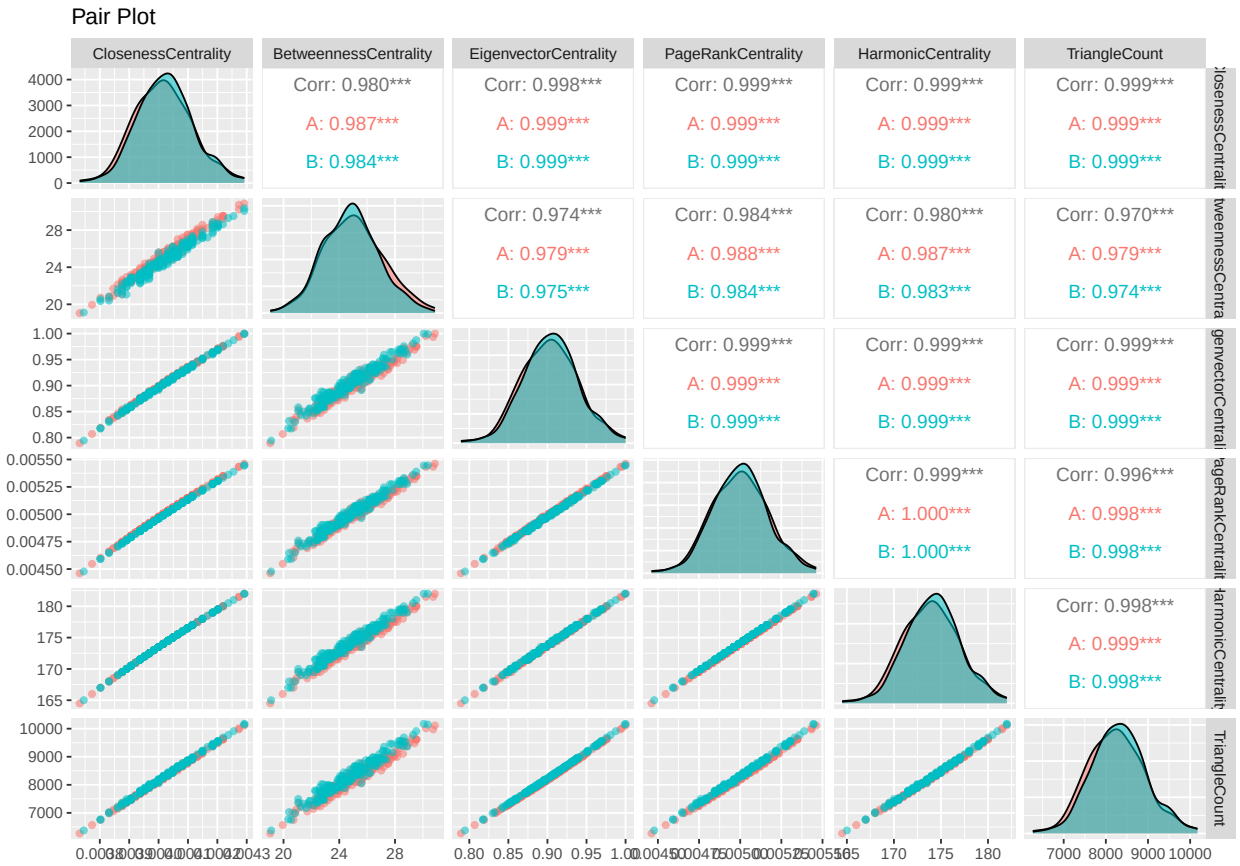
Now the code below generates the dataframe.

```
graph_type <- "A"
existing_df <- NULL
updated_df <- calculate_centrality_and_robustness (A, graph_type, existing_df)
existing_df <- updated_df


graph_type <- "B"
updated_df <- calculate_centrality_and_robustness (B, graph_type, existing_df)
existing_df <- updated_df
head(existing_df)


##   Node ClosenessCentrality BetweennessCentrality EigenvectorCentrality
## 1   A1         0.004098361              26.82408             0.9335742
```

```
## 2   A2         0.003952569              23.92749             0.8799925
## 3   A3         0.004016064              24.57734             0.9061807
## 4   A4         0.003906250              22.84056             0.8614104
## 5   A5         0.004149378              28.16148             0.9512972
## 6   A6         0.004081633              26.87178             0.9269162
##   PageRankCentrality TriangleCount HarmonicCentrality
## 1        0.005147445          8813              176.5
## 2        0.004889815          7787              172.0
## 3        0.005002619          8289              174.0
## 4        0.004804410          7479              170.5
## 5        0.005233438          9130              178.0
## 6        0.005119377          8659              176.0
##   LocalClusteringCoefficient SubgraphCentrality GraphType RobustnessMeasure
## 1                  0.7480689       3.005729e+62         A          1.252198
## 2                  0.7458812       2.670608e+62         A          1.410416
## 3                  0.7517685       2.831925e+62         A          1.329567
## 4                  0.7470782       2.559012e+62         A          1.472329
## 5                  0.7455496       3.120934e+62         A          1.205624
## 6                  0.7446680       2.963010e+62         A          1.270367
```

3 Results

First let us have glimpse of the data in the form of a pairplot.



From the pair plot, we can see that there is a clear relationship between the network properties. This is because ER graph only depends on the value of p when n is fixed, and that value of p dictates most if not all properties of the graph, which is also evident from the pair plot.

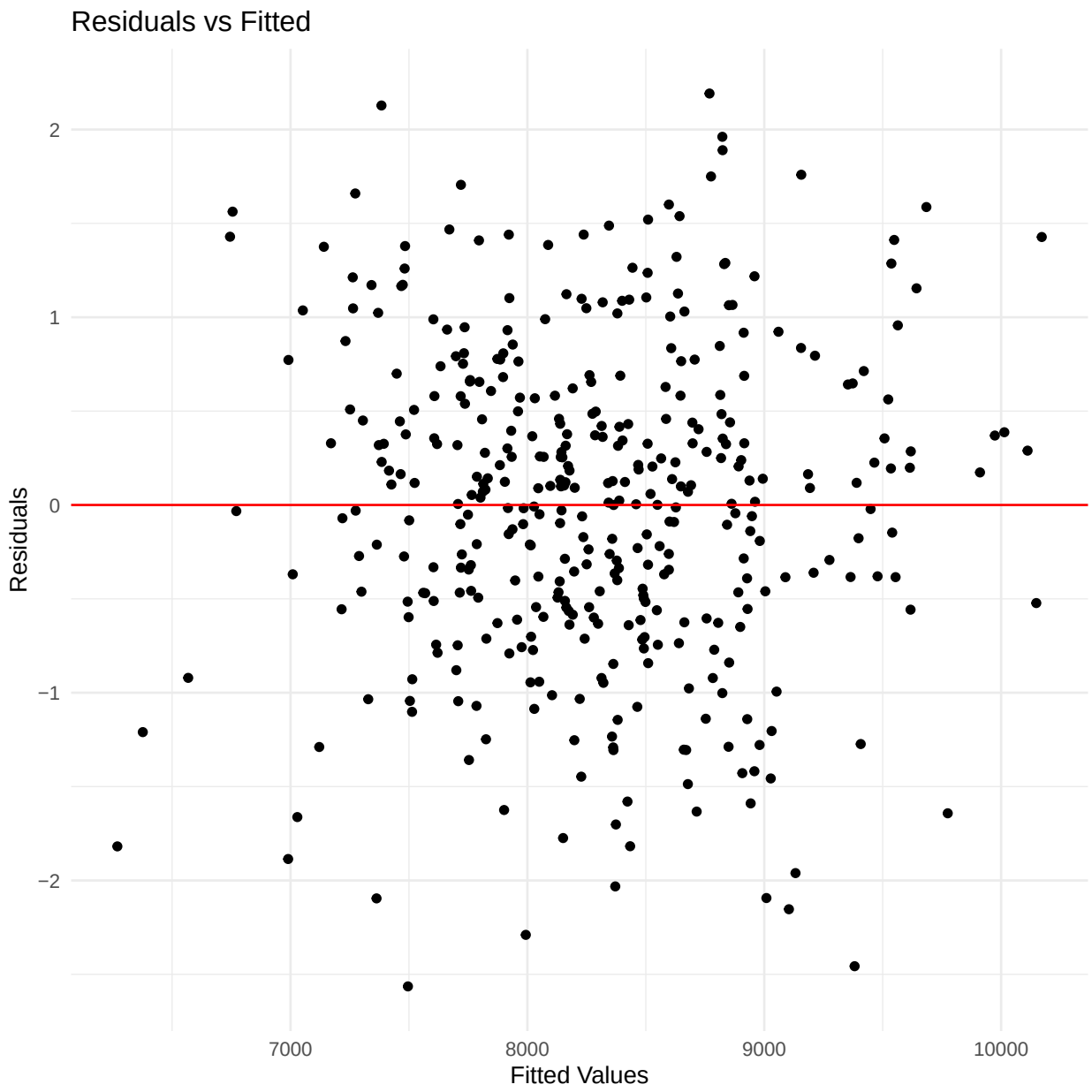
3.1 Regression

Let us first fit the full model as follows.

```
df<- existing_df[,-1]

fit <- lm(TriangleCount~ .,data = df)
summary(fit)


##
## Call:
## lm(formula = TriangleCount ~ ., data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -2.56433 -0.54408  0.07071  0.56921  2.19168
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 5.901e+04  5.060e+03   11.66   <2e-16 ***
## ClosenessCentrality         4.204e+06  1.066e+05   39.44   <2e-16 ***
## BetweennessCentrality      -5.674e+01  1.688e+00  -33.61   <2e-16 ***
## EigenvectorCentrality       1.112e+04  5.492e+02   20.25   <2e-16 ***
## PageRankCentrality          1.703e+07  8.724e+05   19.52   <2e-16 ***
## HarmonicCentrality         -9.583e+02  5.704e+01  -16.80   <2e-16 ***
## LocalClusteringCoefficient  5.208e+03  1.681e+02   30.99   <2e-16 ***
## SubgraphCentrality          8.049e-61  3.691e-62   21.81   <2e-16 ***
## GraphTypeB                  1.041e+02  6.769e+00   15.38   <2e-16 ***
## RobustnessMeasure           8.378e+02  3.414e+01   24.54   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8677 on 390 degrees of freedom
## Multiple R-squared:      1, Adjusted R-squared:      1
## F-statistic: 2.578e+07 on 9 and 390 DF,  p-value: < 2.2e-16
```

From the full model we can see that the Adjusted R-squared is 1, and all predictor variables seems to be significant. This is due to the dependence of the network properties on p. Let us do some diagnostics to check the assumptions and any other systematic problem in our model.



Now we run a Breusch-Pagan to test for constant variance on top of the residual plot above.

The null hypothesis (H_0) and the alternative hypothesis (H_a) for the Breusch-Pagan test are:

- H_0: The residuals are homoscedastic. This means there is a constant variance of the residuals across all levels of the independent variables.

- H_a: The residuals are heteroscedastic. This implies that the variance of the residuals is not constant and varies with the independent variables.

```
##
##  studentized Breusch-Pagan test
##
## data:  fit
## BP = 24.731, df = 9, p-value = 0.003284
```

Since the p-value is much less than 0.05, we reject the null hypothesis that there is a constant variance. In process of fixing this I tried the following things:

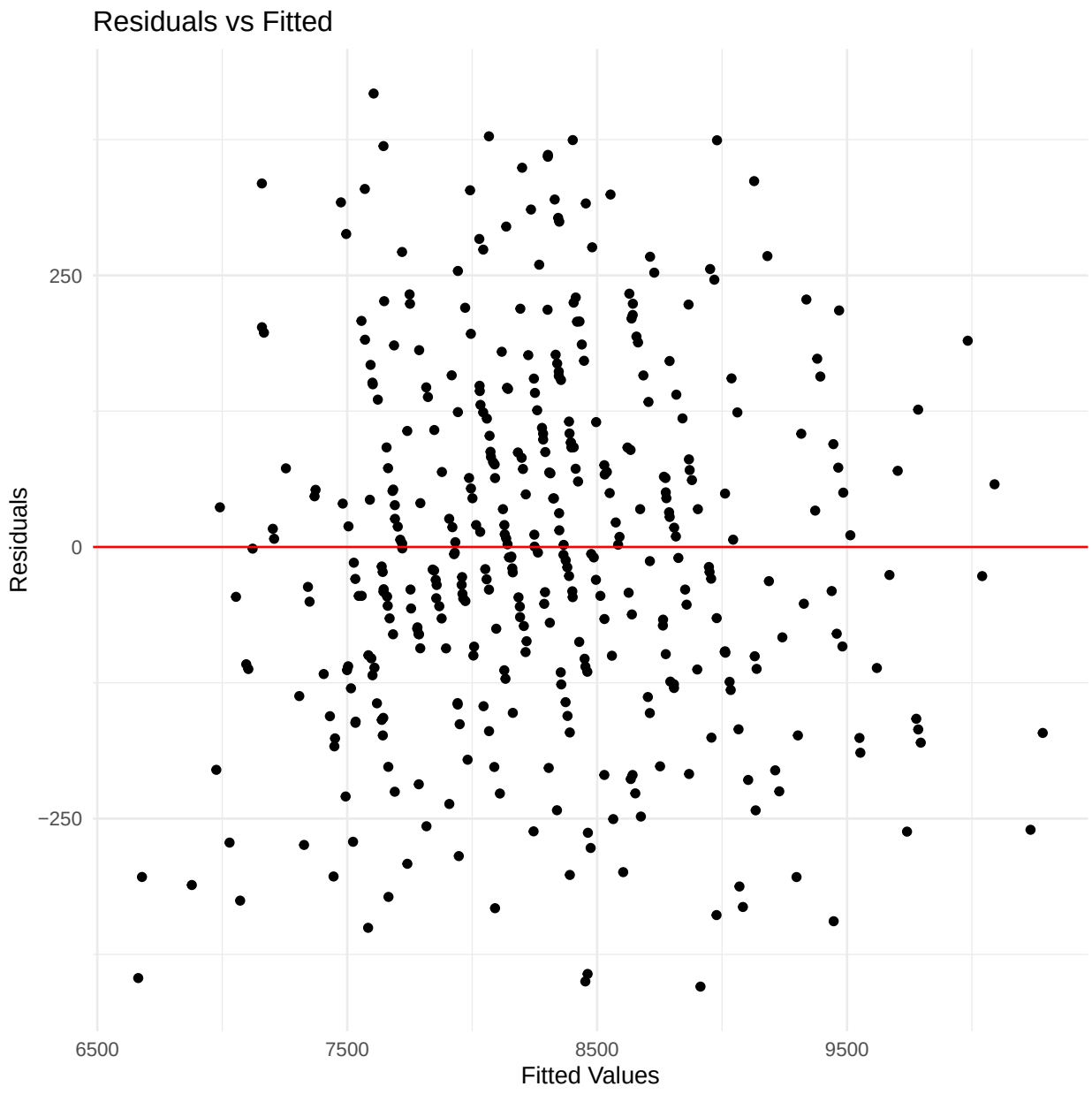
- Transformation of response variable using Box-Cox [Assumptions failed]
- GLMs:Poisson, Generalized Poisson, Negative Binomial and Gaussian with log link. Based on AIC and BIC MLR fit had the lowest values.
- VIF values revealed major auto correlation in our predictor variables(apparent from the pair plot), suggesting Betweeness Centrality as our predictor variable.

Based on all the diagnostics and multiple fits, I found the following model to be the best fit.

```
best_fit <- lm(TriangleCount ~ I(BetweennessCentrality^2)   , data = df)
summary(best_fit)
```

```
##
## Call:
## lm(formula = TriangleCount ~ I(BetweennessCentrality^2), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -404.62 -111.17   -6.52  107.00  417.33
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 4.437e+03  5.076e+01   87.42   <2e-16 ***
## I(BetweennessCentrality^2) 6.136e+00  7.987e-02   76.83   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 166.5 on 398 degrees of freedom
## Multiple R-squared:  0.9368, Adjusted R-squared:  0.9367
## F-statistic:  5903 on 1 and 398 DF,  p-value: < 2.2e-16
```

First let us check the residuals.




```
##
##  studentized Breusch-Pagan test
##
## data:  best_fit
## BP = 0.22662, df = 1, p-value = 0.634
```

The p-value from Breusch-Pagan test is much greater than 0.05. Hence we fail to reject the null hypothesis that the residuals are homoscedastic at 0.05 significance level.

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(best_fit)
## W = 0.99485, p-value = 0.2027
```

From the Shapiro-Wilk test the normality assumption is also met since the p-value is greater than 0.05. Hence, we take best_fit to be the best model.

3.1.1 Prediction

One of the goals of the project is to study how well can we predict one node property using other node/network properties. First let us randomly split our data in 70% for training and 30% for in data testing.

```
set.seed(123)

# Calculate the size of 70% of the data
train_size <- round(nrow(df) * 0.7)

# Randomly sample row indices for the training data
train_indices <- sample(1:nrow(df), train_size)

# Split the data into training and testing sets
train_data <- df[train_indices, ]
test_data <- df[-train_indices, ]
```

Fit the model and make predictions.

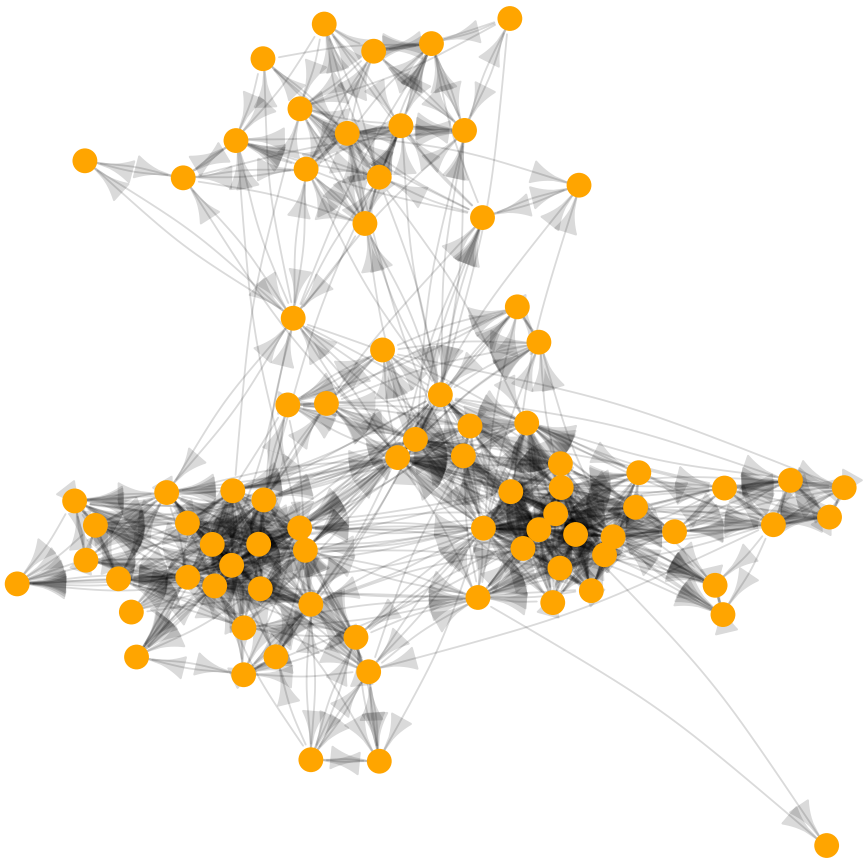
```
fit_predict<- lm(TriangleCount ~ I(BetweennessCentrality^2) , data=train_data)
# Predicting on the test data
predictions <- predict(fit_predict, newdata = test_data)

mape <- mean(abs((test_data$TriangleCount - predictions) /
                   test_data$TriangleCount)) * 100
print(paste("Mean Absolute Percentage Error:", mape))


## [1] "Mean Absolute Percentage Error: 1.52877593870687"
```

From the prediction, MAPE(Mean Absolute Percentage Error) is 1.53. This means on average the model's predictions are off by 1.53%.

In ER graphs there seems to be a perfect if we use all the centrality measures and almost a perfect fit even when we use just one. From the analysis above, we conclude that for ER graphs one centrality measure is enough for prediction since most of the variation in our model can be explained by just one centrality measure. This is always not the case in real world graphs. Below, we look at an applied graph and see how the regression relationships differ in those graphs.

3.2 Real World Graphs

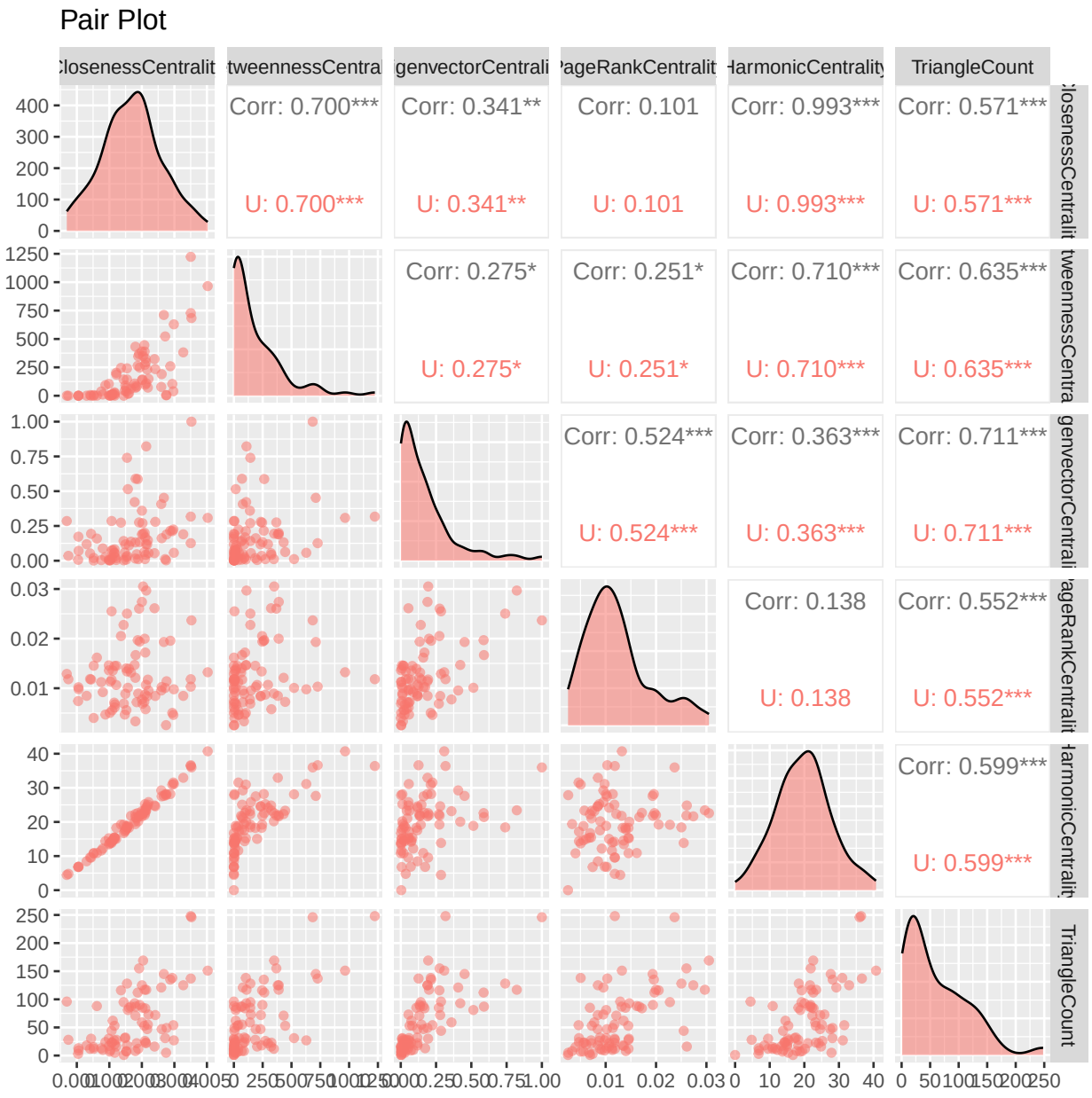
We look at a personal friendship network of a faculty of a UK university consisting of 81 vertices(individuals) and 817 directed and weighted connections.



Now get the data for regression.

```
##   Node ClosenessCentrality BetweennessCentrality EigenvectorCentrality
## 1   U1         0.002890173              347.1190          0.011365774
## 2   U2         0.003676471              711.2849          0.452156852
## 3   U3         0.001517451                5.5000          0.001156645
## 4   U4         0.002217295              185.9787          0.018501103
```

```
## 5   U5         0.004504505              727.4570          0.126093091
## 6   U6         0.003125000              296.0071          0.118838076
##   PageRankCentrality TriangleCount HarmonicCentrality
## 1        0.008937130            12           20.19643
## 2        0.019316075           145           27.63333
## 3        0.004012504            15           10.80999
## 4        0.014529515            28           18.24921
## 5        0.010326041           137           36.63333
## 6        0.010684184            22           24.16905
##   LocalClusteringCoefficient SubgraphCentrality GraphType RobustnessMeasure
## 1                  0.3333333      205.001899+0i         U         42.894979
## 2                  0.5253623    15152.193958+0i         U          1.682093
## 3                  0.7142857        4.793661+0i         U       1449.194253
## 4                  0.3589744       31.112896+0i         U         69.403861
## 5                  0.3624339    13854.247336+0i         U          8.775275
## 6                  0.3333333     2045.275844+0i         U         23.613103
```



From the pair plot we can see that the relationship are not as pronounced as in ER graphs.

```
df<- existing_dfU[,-1]
fitU <- lm( TriangleCount ~ ClosenessCentrality+
              BetweennessCentrality +EigenvectorCentrality+
              PageRankCentrality+ HarmonicCentrality+
              LocalClusteringCoefficient+ RobustnessMeasure , data =df)
summary(fitU)


##
## Call:
## lm(formula = TriangleCount ~ ClosenessCentrality + BetweennessCentrality +
```

```
##     EigenvectorCentrality + PageRankCentrality + HarmonicCentrality +
##     LocalClusteringCoefficient + RobustnessMeasure, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -70.825 -14.866  -2.763  11.264  73.762
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -3.167e+01  2.374e+01  -1.334 0.186444
## ClosenessCentrality        -5.184e+04  3.017e+04  -1.719 0.089980 .
## BetweennessCentrality       8.050e-02  2.155e-02   3.735 0.000373 ***
## EigenvectorCentrality       1.119e+02  2.116e+01   5.288 1.27e-06 ***
## PageRankCentrality          2.202e+03  6.471e+02   3.403 0.001093 **
## HarmonicCentrality          7.673e+00  3.740e+00   2.052 0.043836 *
## LocalClusteringCoefficient  3.213e+01  2.534e+01   1.268 0.208893
## RobustnessMeasure          -8.259e-04  2.818e-03  -0.293 0.770311
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.95 on 72 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.7626, Adjusted R-squared:  0.7395
## F-statistic: 33.03 on 7 and 72 DF,  p-value: < 2.2e-16
```

We do home some significant predictors. From the pair-plot we could see evidence of multi-collinearity. Below are the VIF values

```
## Loading required package: carData
```

```
##        ClosenessCentrality      BetweennessCentrality
##                  77.291357                   2.595647
##      EigenvectorCentrality         PageRankCentrality
##                   1.736581                   1.776434
##         HarmonicCentrality LocalClusteringCoefficient
##                  83.910681                   1.762869
##          RobustnessMeasure
##                   1.229742
```

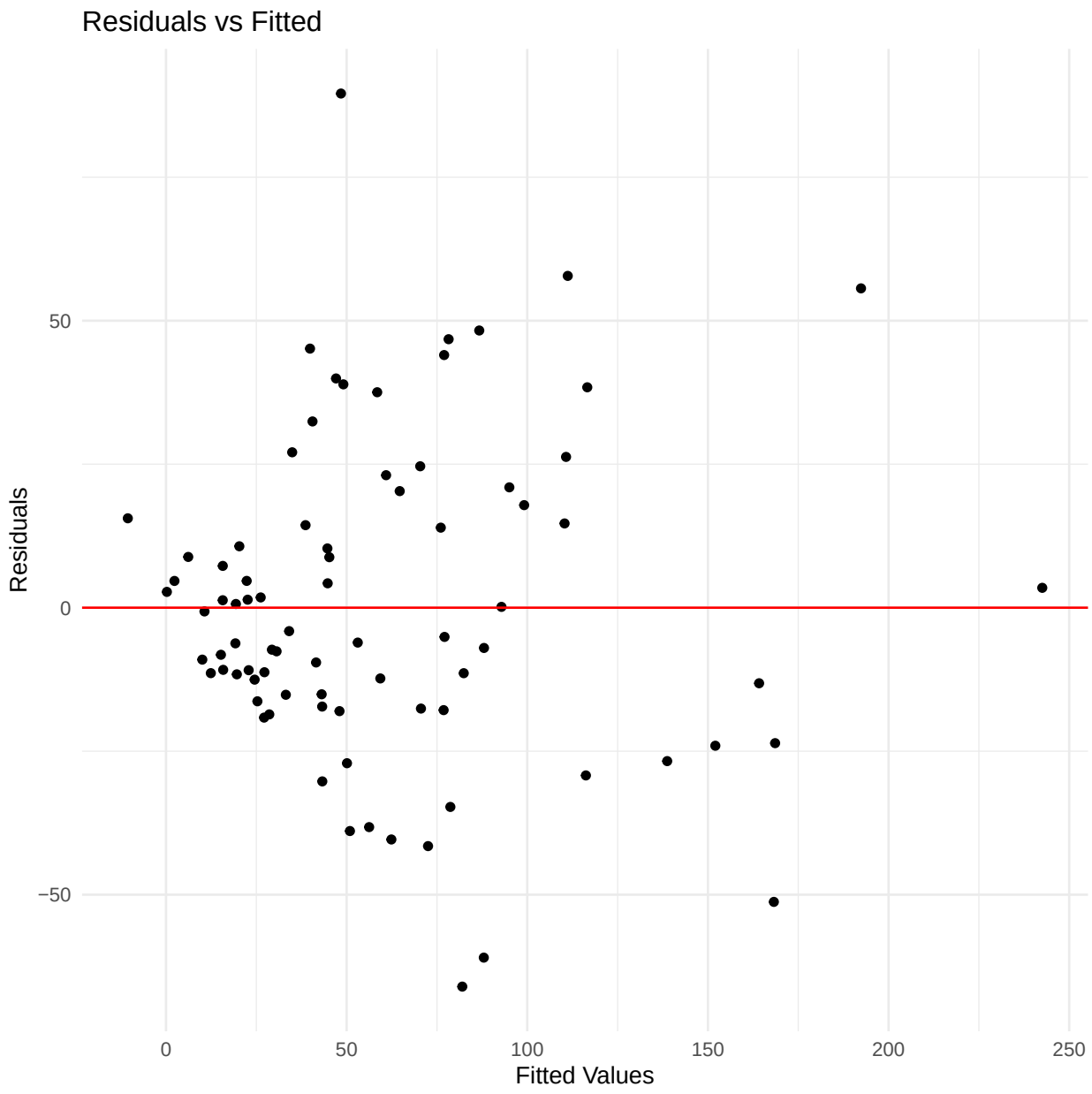
The following is the best-fit for UKFaculty(UF) graph.

```
fitU1 <- lm( TriangleCount ~  BetweennessCentrality
             +EigenvectorCentrality+  PageRankCentrality+
               LocalClusteringCoefficient+ RobustnessMeasure , data =df)
summary(fitU1)
```

```
##
## Call:
## lm(formula = TriangleCount ~ BetweennessCentrality + EigenvectorCentrality +
##     PageRankCentrality + LocalClusteringCoefficient + RobustnessMeasure,
```

```
##     data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -66.011 -16.305  -5.092  14.676  89.569
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -7.030e+00  1.937e+01  -0.363   0.7177
## BetweennessCentrality       1.102e-01  1.765e-02   6.243 2.35e-08 ***
## EigenvectorCentrality       1.334e+02  2.013e+01   6.628 4.59e-09 ***
## PageRankCentrality          1.555e+03  6.168e+02   2.522   0.0138 *
## LocalClusteringCoefficient  1.315e+01  2.443e+01   0.538   0.5920
## RobustnessMeasure          -2.212e-03  2.825e-03  -0.783   0.4361
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.78 on 75 degrees of freedom
## Multiple R-squared:  0.7416, Adjusted R-squared:  0.7244
## F-statistic: 43.05 on 5 and 75 DF,  p-value: < 2.2e-16
```

Some basic diagnostics.




```
##
##  studentized Breusch-Pagan test
##
## data:  fitU1
## BP = 9.054, df = 5, p-value = 0.1069
```

The p-value from Breusch-Pagan test is much greater than 0.05. Hence we fail to reject the null hypothesis that the residuals are homoscedastic at 0.05 significance level.

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(fitU1)
## W = 0.97925, p-value = 0.2126
```

We see that the normality assumption is also met. One thing to notice here is the difference between the number of predictor compared to ER graphs. Theoretical graphs have nice relationship, and especially in graphs like ER where the entire graphs is characterized by p, using one centrality measures explains most of the variation in the model. But in case of UF graph, all the predictor variables included together wasn't enough to explain all the variation. This suggests that the structure of the graph plays the biggest role rather than the local properties of the node.

4 Discussion

In this section, we further study ER graphs and how the value of p impacts the regression relationship. This discussion will further explain the perfect fit in the ER graphs and how the underlying structure of the graph impacts the regression relationship. First we define 7 different ER graphs with varying p but fixed n.

```
set.seed(323423)
C<- erdos.renyi.game(n =100 , p = 0.3)
D <- erdos.renyi.game(n=100, p = 0.4)
E <- erdos.renyi.game(n =100 , p = 0.5)
F<- erdos.renyi.game(n=100, p = 0.6)
G<- erdos.renyi.game(n =100 , p = 0.7)
H <- erdos.renyi.game(n=100, p = 0.8)
I <- erdos.renyi.game(n=100, p = 0.9)
```

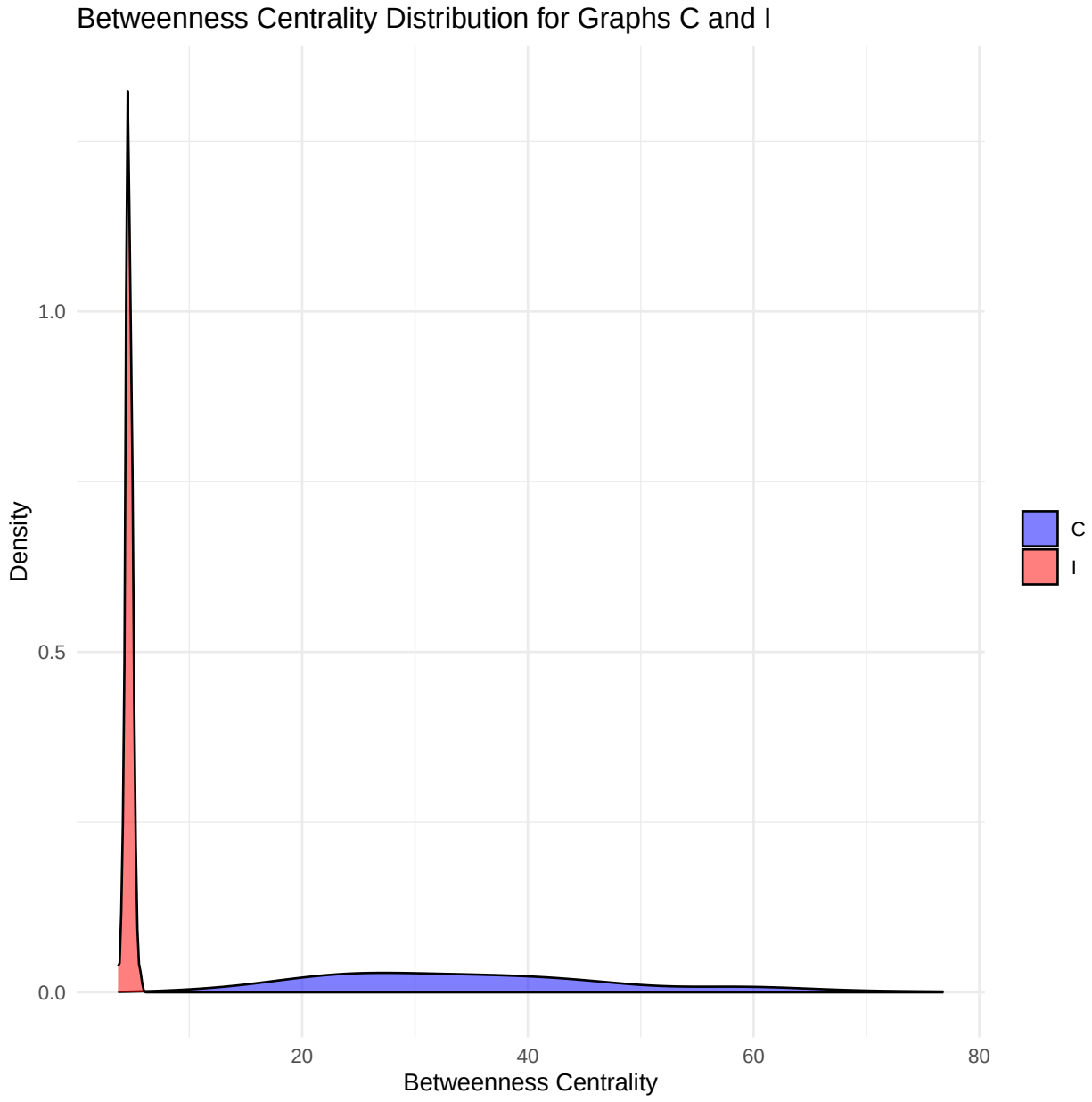
Now we create a dataframe for regression.

```
##   Node ClosenessCentrality BetweennessCentrality EigenvectorCentrality
## 1   C1         0.006024096              44.79958             0.7547503
## 2   C2         0.005952381              37.25815             0.7096966
## 3   C3         0.005780347              27.11325             0.5824377
## 4   C4         0.006097561              47.15017             0.8206283
## 5   C5         0.005780347              23.77155             0.6122627
## 6   C6         0.005681818              16.31880             0.5541385
##   PageRankCentrality TriangleCount HarmonicCentrality
## 1        0.011056703           135               65.5
## 2        0.010473933           130               64.5
## 3        0.008982340            87               62.0
## 4        0.011625587           172               66.5
## 5        0.008934409            96               62.0
## 6        0.007994965            79               60.5
##   LocalClusteringCoefficient SubgraphCentrality GraphType RobustnessMeasure
```

```
## 1  0.2721774  66738519745  C  2.849350
## 2  0.2988506  59008624880  C  3.227052
## 3  0.2900000  39743758982  C  4.795225
## 4  0.3065954  78897463879  C  2.407166
## 5  0.3200000  43918307568  C  4.332822
## 6  0.3419913  35975481767  C  5.285743
```

From the pair plot it is evident that the value of p completely impacts the distribution of the centrality measures. As p increases, the graph becomes denser and the paths between the nodes become shorter, leading to higher closeness centrality. Denser graph also implies higher probabilities of triangle formation and hence increase the probability of triangle count too. The distributions for betweenness centrality has opposite trend compared to other variables. Betweenness centrality measures the extent to which a node acts as a bridge along the shortest path between two other nodes. Nodes with higher betweenness centrality facilitate interaction within the network. As p increase, the probability of edge formation also increases, leading to larger number of alternative shortest paths. This reduces the betweenness centrality of individual nodes since there are many alternative routes between any two nodes. This effect can also be seen from the coefficient of the full model for the ER graph. The coefficient for Closeness and EigenVector centrality are positive, but the coefficient for betweenness centrality are negative.

Below let us examine the distribution plots of graphs C(p=0.3) and I(p=0.9) to see the difference between them more closely.



From the plot above we can see that there is a peak at lower values of betweenness centrality for graph I. The variance is much smaller compared to graph C. The peak in graph I is due to a dense graph making most nodes equally important from betweenness centrality's viewpoint. But graph C has a wider variation in the betweenness centrality, highlighting only those nodes as important, that actual act as bridges for two nodes in the graph.

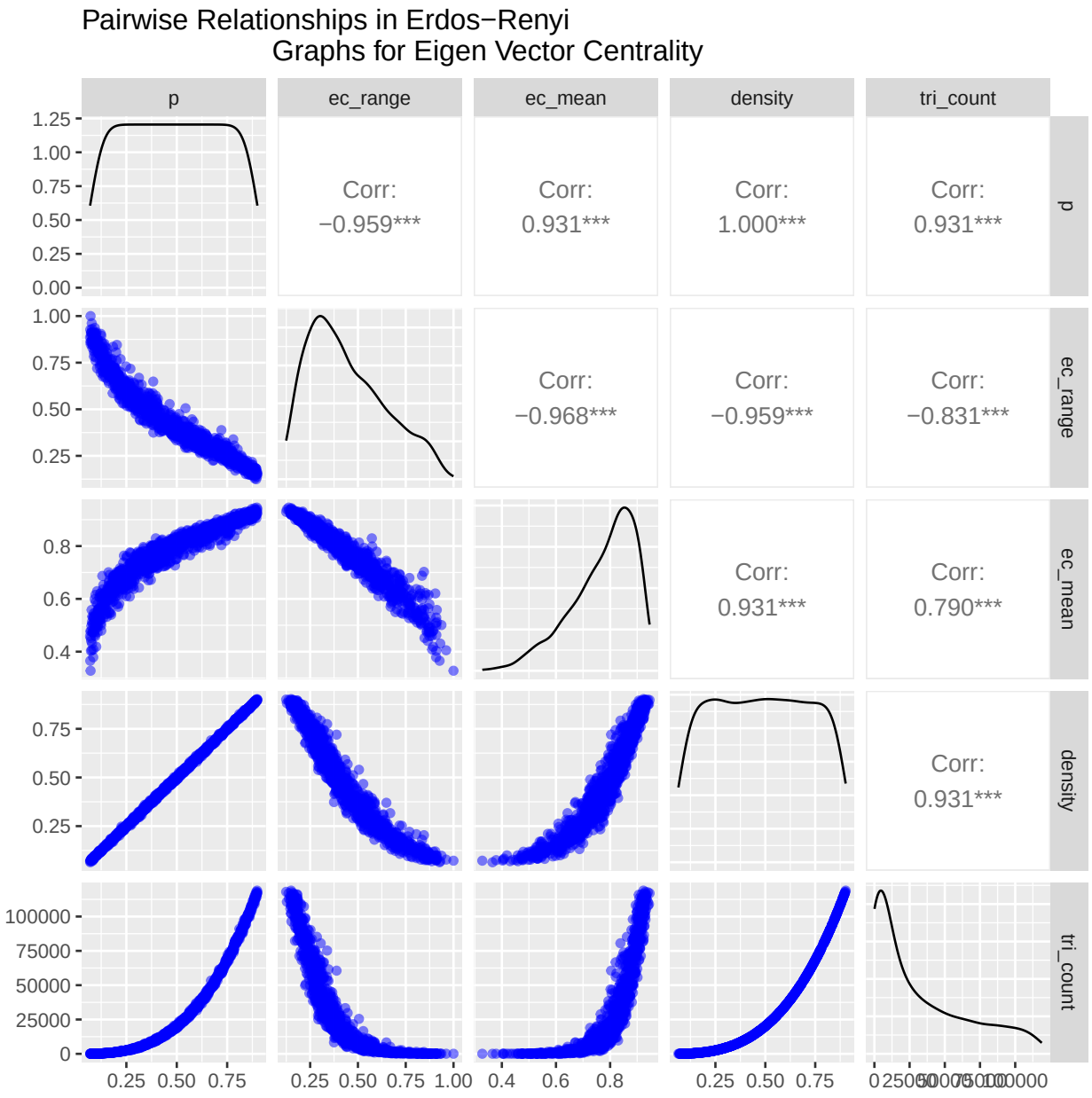
The reason we get perfect fit for Triangle count in Erdos-Renyi is because of the simple structure of the graph that is only dependent of the edge probability, p. And as we discussed

above we already have a theoretical reason for why this is true. The perfect fit in ER graphs can be attributed to the inherent structure of ER graphs and the role of edge probability.

From the above discussions we saw that in ER graphs the perfect fit is due to the underlying theoretical relationships. Now let us investigate the relationship between structural properties of ER graphs such as centrality measures and triangle counts with respect to edge probability p to understand how well these properties can predict the underlying value of p that generates the graph.

Below we simulate 1000 different ER graphs with varying values for p but fixed n. We will aggregate the centrality measures and also use network level properties.

```
##            p     cc_mean     bc_mean   ec_mean        cc_sd        bc_sd
## 1 0.07000000 0.003942937 0.003960412 0.4593928 0.0002845631 0.0002845631
## 2 0.07083083 0.003814926 0.003831418 0.3654362 0.0002987876 0.0002987876
## 3 0.07166166 0.004066221 0.004081633 0.4774963 0.0002990083 0.0002990083
## 4 0.07249249 0.004063861 0.004081633 0.3276858 0.0003370206 0.0003370206
## 5 0.07332332 0.004087470 0.004106793 0.4518604 0.0003011676 0.0003011676
## 6 0.07415415 0.004045141 0.004056812 0.4028655 0.0003238147 0.0003238147
##       ec_sd    cc_range bc_range  ec_range    density tri_count
## 1 0.1831454 0.001350284 208.3127 0.8857413 0.06969697        45
## 2 0.1877827 0.001439864 276.5449 0.9272671 0.06444444        49
## 3 0.1879486 0.001438293 226.4730 0.8612154 0.07494949        68
## 4 0.1622331 0.001911585 350.4110 1.0000000 0.07252525        62
## 5 0.1795420 0.001521036 342.5204 0.8483952 0.07575758        77
## 6 0.1787151 0.001562944 289.2149 0.9097874 0.07313131        81
```



Let us fit a regression with respect to Eigen Vector Centrality.

```
fit1<- lm(p~ec_range+ec_mean+density+tri_count, data= results)
summary(fit1)


##
## Call:
## lm(formula = p ~ ec_range + ec_mean + density + tri_count, data = results)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0225926 -0.0039070  0.0000227  0.0038685  0.0195190
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.249e-03  7.153e-03   0.454    0.650
## ec_range    -4.025e-03  5.155e-03  -0.781    0.435
## ec_mean      1.220e-03  6.658e-03   0.183    0.855
## density      9.952e-01  5.478e-03 181.694   <2e-16 ***
## tri_count    1.300e-09  2.001e-08   0.065    0.948
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.006022 on 995 degrees of freedom
## Multiple R-squared:  0.9994, Adjusted R-squared:  0.9994
## F-statistic: 3.963e+05 on 4 and 995 DF,  p-value: < 2.2e-16
```

We can see that when we use density as one of our predictor variables, the fit is perfect.

The density of an Erdős–Rényi graph is defined as the ratio of the number of edges in the graph to the number of possible edges. For an undirected graph without self-loops, the number of possible edges is given by $\binom{n}{2}$, where n is the number of nodes in the graph.

In an Erdős–Rényi graph, each edge is included in the graph independently with probability p. Therefore, the expected number of edges E in the graph is $p \times \binom{n}{2}$.

The density d of the graph can be calculated as:

$$d = \frac{E}{\binom{n}{2}} = \frac{p \times \binom{n}{2}}{\binom{n}{2}} = p$$

Thus, in an Erdős–Rényi graph, the probability p directly equals the density of the graph. As p increases, the graph becomes denser with more edges; as p decreases, the graph becomes sparser. Therefore we see a perfect fit and using density as the only predictor variable of p is sufficient.

In the full model including Density, all other predictors are non-significant due to this relationship between density and p. If we remove density as our predictor variables then all other variables become significant.

4.1 Conclusion

Centrality measures highlight "important" nodes in a network. Different centrality measures focus on a particular aspect of importance. In some networks, being a central node is important, while in other networks, connecting to other important nodes might be more valuable. In a homogeneous graph like a Erdős–Rényi, these centrality measures often overlap, where a set of nodes are highlighted by most centrality measures. However, in real world networks, this is not guaranteed. We conclude that the network structure plays a pivotal role defining

the relationships between the node and network properties. Exploring these relationships help us further understand the network, and provides a prior for advanced network analysis like link prediction and fault detection in a network.