R Notebook

Sherman ALINE

Introduction

Our dataset (http://konect.cc/networks/edit-zhwiki/) comes from the Sinophone (Chinese languages and dialects) version of Wikipedia. Specifically it is the dataset of edits of articles. Source nodes are all users, and destination nodes are all articles getting edited. Thus, the structure of our network is naturally bipartite. However we can build a one-mode graph from this. In this report we will do this and analyze a subgraph corresponding to a one-year window of edit histories.

metadata

```
Node meaning        User, article
Edge meaning        Edit
Network format      Bipartite, undirected
Edge type       Unweighted, multiple edges
Temporal data       Edges are annotated with timestamps

Files:
    meta.edit-zhwiki -- Metadata about the network
    out.edit-zhwiki -- The adjacency matrix of the network in whitespace-separated values format, with o
      The meaning of the columns in out.edit-zhwiki are:
        First column: ID of from node
        Second column: ID of to node
        Third column: edge weight (=1)
        Fourth column: timestamp of the edge Unix time
```

We must have a theory about why sharing artifacts tells us something about relationship between actors: In the edits of a wikipedia page, each edit is consecutive. Sometimes different users may disagree about proper edit of a page. Different editors may be more comfortable with different types of edits: some users may add information, other users may correct grammar or check that claims are cited and follow proper guidelines, etc. But at it's core, each page of wikipedia will contain edits building on one another, so althought our graph is connected only between users and the pages, their are implied interactions between users as they build and modify eachother's work.

We are mainly interested in interactions between users.. Looking for a *community structure* as there likely to be social ties, or ties through common interests and knowledge.

In order to decrease the computational intensity by shrinking the dataset, we look at a year-long (52 week) subset of the data. This slice is from Sun, 27 Oct 2002 11:38:36 GMT to Sun, 26 Oct 2003 11:38:36 GMT.

Feature Creation / Transformation

To make our graph more intuitive to the nature of Wikipedia, we modify our features.

One-Mode Graph

First, we are easily able to convert our bipartite graph into a one-mode graph due to the consecutive nature of Wikipedia edits. In a given article, each edit is building off of the edit before it, which has it's own user who made that edit. So for each article we simply re-assign the destination node to be the source node from the previous edit.

Weights

We choose not to use the existing tnet library to incorporate temporal data into our network. According to the author of the package, Tore Opsahl, the purpose of tnet is to run analysis on windows of data, for the purpose of drawing causal connections.

We are not interesting in causality, but rather the temporal distance between to actions. If many edits have been made in a small period of time, they are more likely to be related than an edit that occurs after months of inactivity.

For this reason, for each edge we use as weight the normalized inverse of the time difference between the two edits.

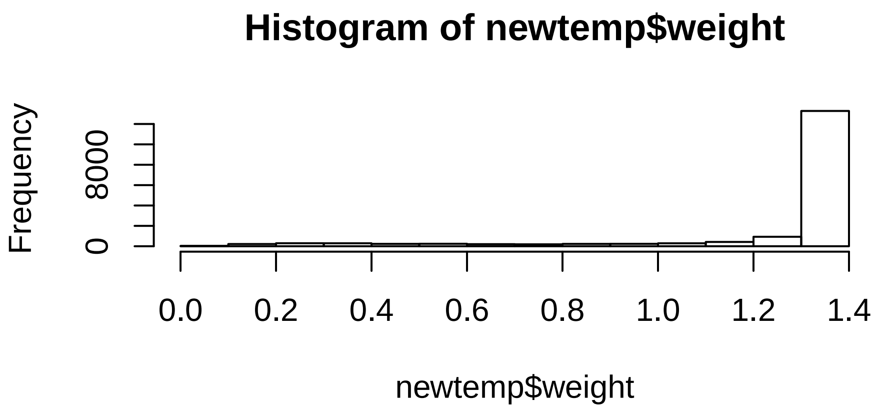
After these two steps, we build our graph and combine duplicate nodes, adding their weights together.

Basic Statistics & Graph Transformation

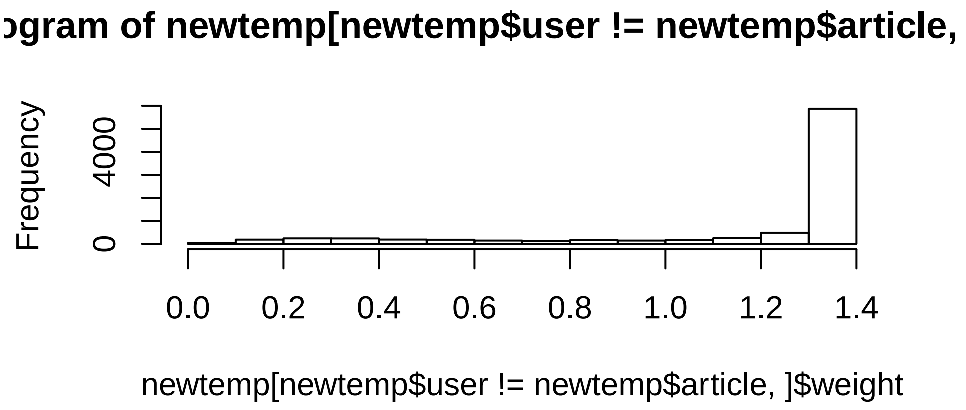
In this section we calculate some basic summary statistics (degree distribution, shortest-path distribution, transitivity, diameter, radius, betweenness, . . .) and visualizations. Motivated by these statistics we apply additional transformations to our graph and re-run the statistics.

```
## # A tibble: 17,100 x 3
##    user  article weight[,1]
##    <chr> <chr>        <dbl>
##  1 u 10  3             1.36
##  2 u 13  3104790       1.36
##  3 u 13  u 13          1.36
##  4 u 13  4             1.36
##  5 u 13  u 13          1.36
##  6 u 13  u 13          1.36
##  7 u 13  6             1.36
##  8 u 13  u 13          1.36
##  9 u 13  u 13          1.36
## 10 u 13  u 13          1.36
## # ... with 17,090 more rows
```

Edge weights



Edge weights for non-self loops



We can see that many of the high-frequency edits (which will have large weights due to short time differences) made by users editing their own edit (self-loop). Based on the graph about 6000 highest-weight edges are self-loops. However even with these removed the distribution of edge weights still seems to follow a power distribution, i.e. the distribution of time length between consecutive edits.

After loading an igraph object, we simplify it to remove self-loops.

Vertices and Edges

```
## [1] 3360
```

```
## [1] 3840
```

Here we observe that graph contains 3360 vertices and 3840 edges.

Is it connected? Is it bipartite?

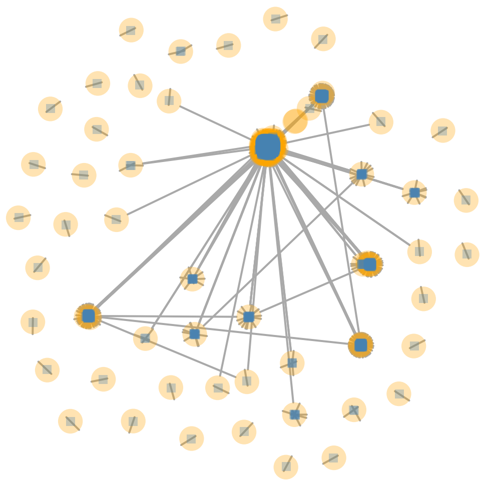
```
## [1] FALSE
```

```
## [1] FALSE
```

We see that the graph is not bipartite which is as desired. It is also not connected, we will fix this later by taking a subgraph!

Visualization

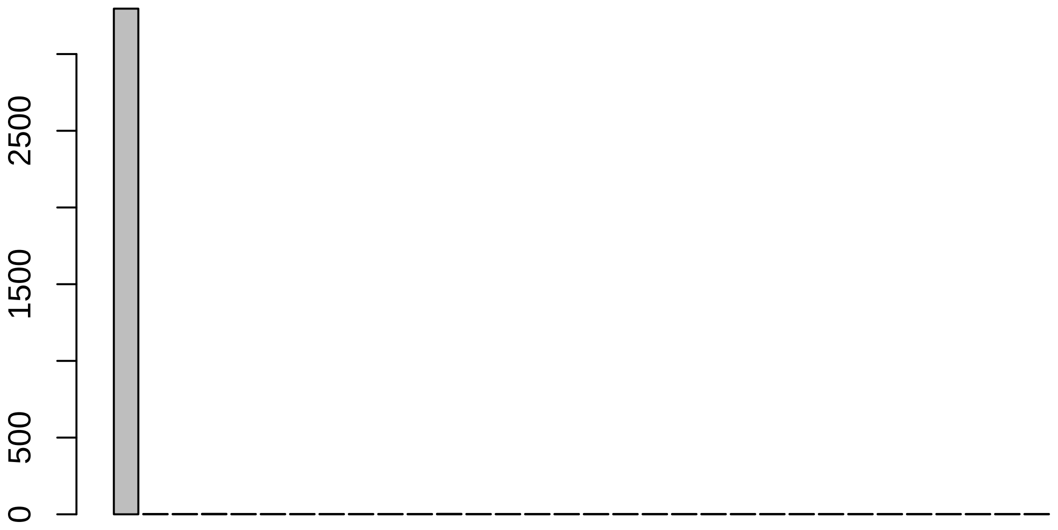
To better understand these information, we plot below all the vertices and nodes forming this graph



Orange dots are users, blue squares are articles. We see that every document has at least one user close to it, this is the first user in our dataset who has submitted an edit to the article. We can also clearly see many user-document groups which are disconnected from the rest of the graph.

Induced Subgraph

As we have seen, our graph is not connected. To get a connected graph we induce a subgraph. Lets look at the connected components correponding to the above graph:



Is it connected?

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

Now the graph is connected! We'll look at the distribution of our final graph, an induced subgraph of a simpified graph, and plot the network it once more.

Vertices and Edges

```
## [1] 3296
```

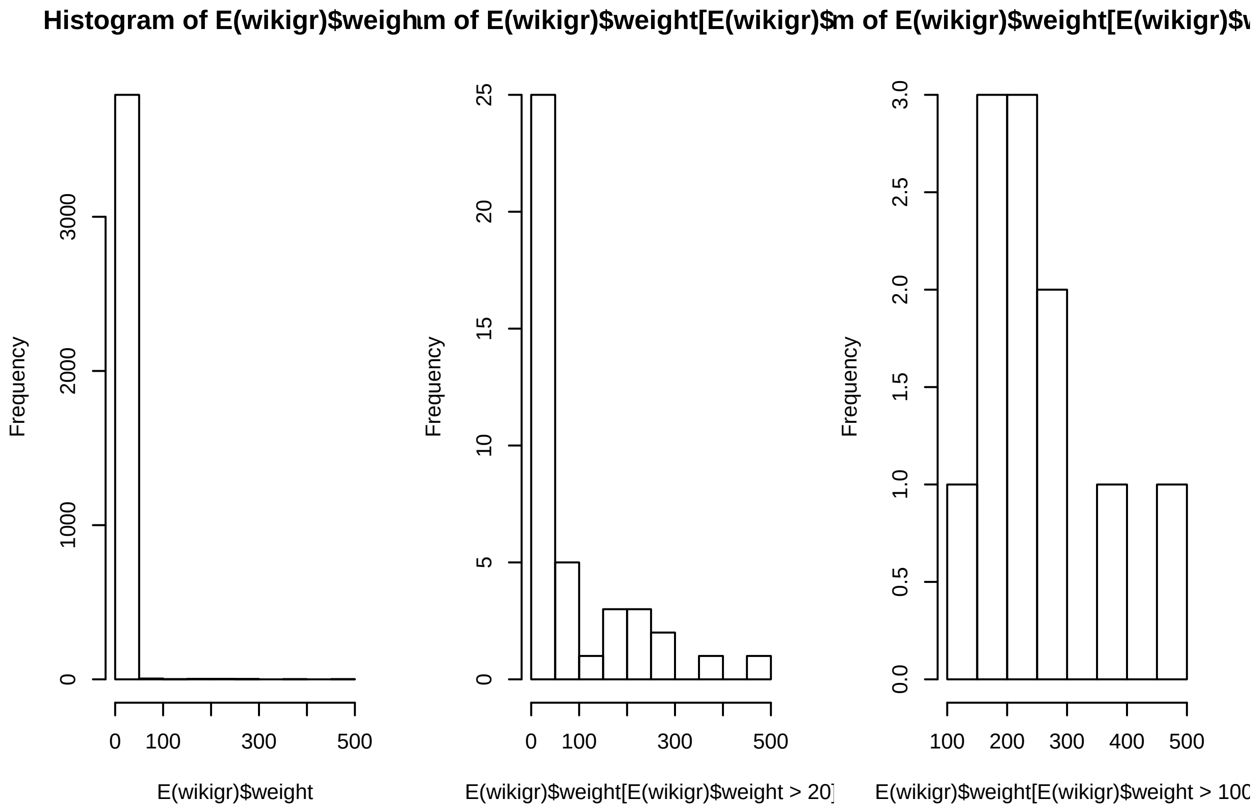
```
## [1] 3807
```

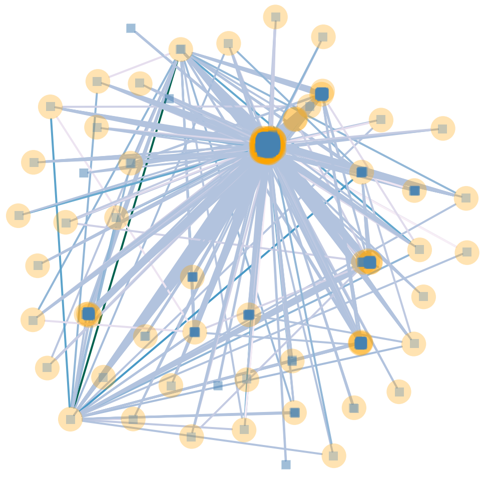
We see that our connected subgraph has 3296 vertices and 2807 edges.

Edge weights

Below we show the quantiles and histograms for all the data, weights larger than 20 and weights larger than 100. This is in order to better view the data despite it's extreme skewness .

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.0691   1.3649   1.3649   2.5736   1.3649 459.4885
```





Edge weights are represented by color and darkness. Blue edges are close to the mean, light color pinkish edges are very low (the majority). Dark blue and green edges are very high weights (rare). For reference we are using the Brewer pallete "PuBuGn" and we transform our data to better utilize all the colors. Pink edges may be harder to find due to being covered by darker edges.

We can see that the nature of our graph is hard to observe, because vertices in the center are much closer than the ones we see far away. We have already seen that a large proportion of edits we removes were self-loop edits. It is possible that in these clusters users go back and forth editing each other which leads to this much denser cluster in the center and the outer region which has fewer articles and fewer users.

Statistical summaries and global characteristics

```
##      Density Transitivity     Diameter       Radius        Girth     Cohesion
## 0.0007010843 0.0072164159 6.0000000000 3.0000000000 3.0000000000 1.0000000000
```

The estimated **density** of the network tells us that 0.07% of the total possible numbers of edges are actually observed (very low number). Another way to think about density, is as given the probability that, if we were to choose two random nodes in the network, this random dyad will have probability p=0.0007 of being connected.

The **transitivity** can be defined as the overall probability for the network to have adjacent nodes interconnected, here p=0.007 which is quite small. Thus revealing the quasi-inexistence of tightly connected communities (clusters).

Here the **diameter** is equal to 6, meaning that the longest shortest-path between two vertices in this graph contains 6 edges.

The **radius** is the minimum among all the maximum distances between a vertex to all other vertices. In our case it is 3.

Here **girth** = 3,which means thatthe number of edges in the shortest cycleof this graph is equalas well to 3.

Since **cohesion** = 1, only one vertex would need to be removed in order to render the graph disconnected.

Characteristics of the Network

Now we want to take a closer look at the different local characteristics of the network's vertices. In the context of our application this is the characteristic of relationships between users. The results are as follows:

```
##      Degree          Strength           Betweenness       Eccentricity
##  Min.   :  1.00   Min.   :   0.3012   Min.   :      0   Min.   :3.000
##  1st Qu.:  1.00   1st Qu.:   1.3649   1st Qu.:      0   1st Qu.:5.000
##  Median :  1.00   Median :   1.3649   Median :      0   Median :5.000
##  Mean   :  2.31   Mean   :   5.9451   Mean   :   6042   Mean   :4.992
##  3rd Qu.:  1.00   3rd Qu.:   1.3649   3rd Qu.:      0   3rd Qu.:5.000
##  Max.   :760.00   Max.   :2901.6999   Max.   :2154291   Max.   :6.000
##    Closeness
##  Min.   :3.290e-05
##  1st Qu.:7.564e-05
##  Median :8.039e-05
##  Mean   :7.866e-05
##  3rd Qu.:8.080e-05
##  Max.   :1.315e-04
```

The **degree** of a vertex is the number of its attached links,the higher its value, the more the vertex is important in a graphas many links converge to it. In this graph, the mean of the differentvertices degrees is equal to 2, which means that in average, each user editsanother user's edition twice.But we can see that there is some users withextreme value of edits (760).

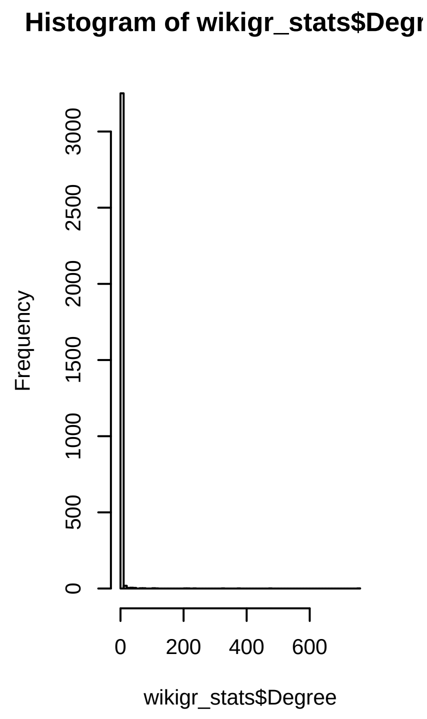
This graph is weighted, thus we can calculate the **strength** which is defined as the weighted degree. Here, taking intoaccount the normalized inverse of time difference between twoedits, we can see that in average, each user is editing roughly 6 times another user's edition. As with degree, there exists someusers with very high weighted edits (2901)

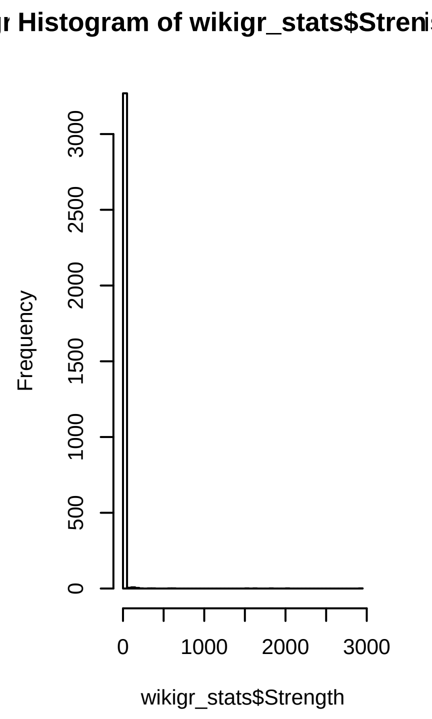
A vertex **betweenness** is defined as the number of shortest path between all pairs of vertices that pass through the vertex. In our case,we can see that the mean betweenness is 6042 indicating that mostly all the users are important to the network.

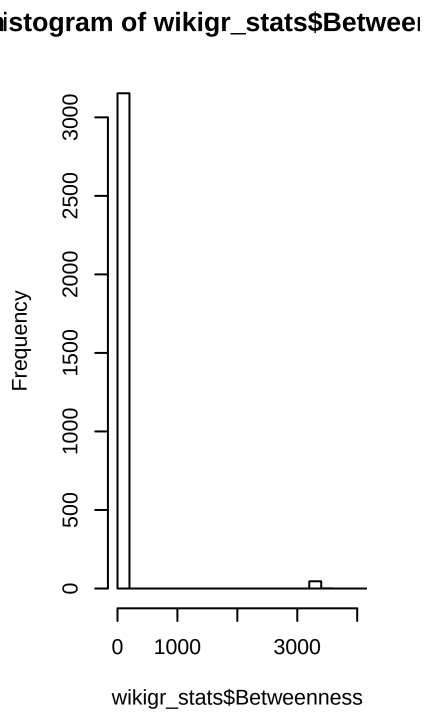
The **eccentricity** of a vertex is the shortest-path length from the farthest other vertex in the graph. In our case, this means that in average, the smallest edition frequence from a given user to a farthest one in the network is roughly 5.

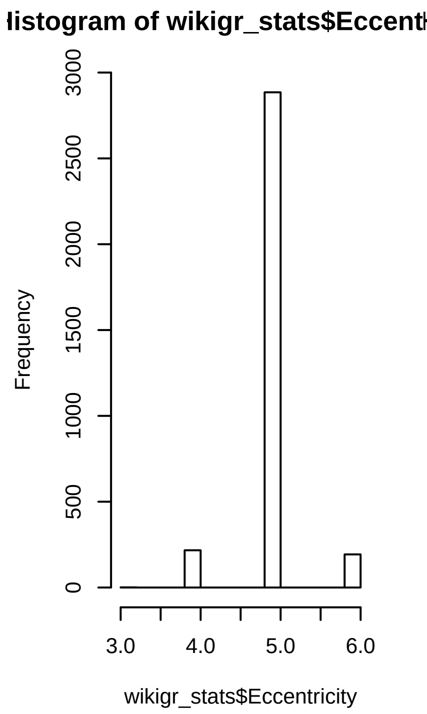
The **closeness** is the inverse of the average of the shortest paths from one vertex to all other edges in a graph. The mean for our vertices is 7.866e-05, implying mean of the average of the shortest paths is quite large at 12712. The vertex with the smallest average of all shortest paths is the inverse of the maximum, $1/1.315e-04 = 7604$

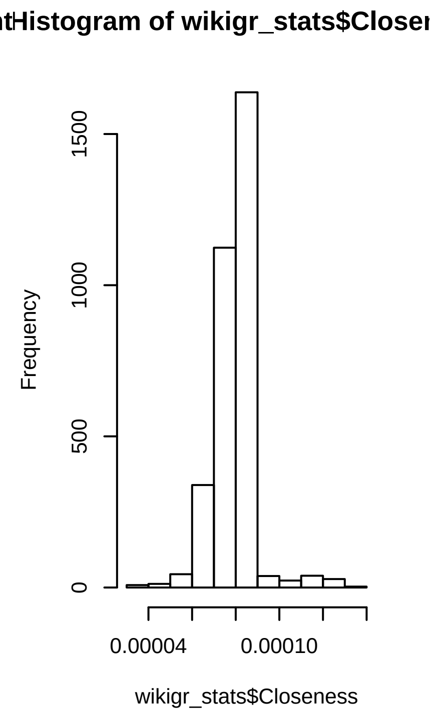
Distributions of Characteristics









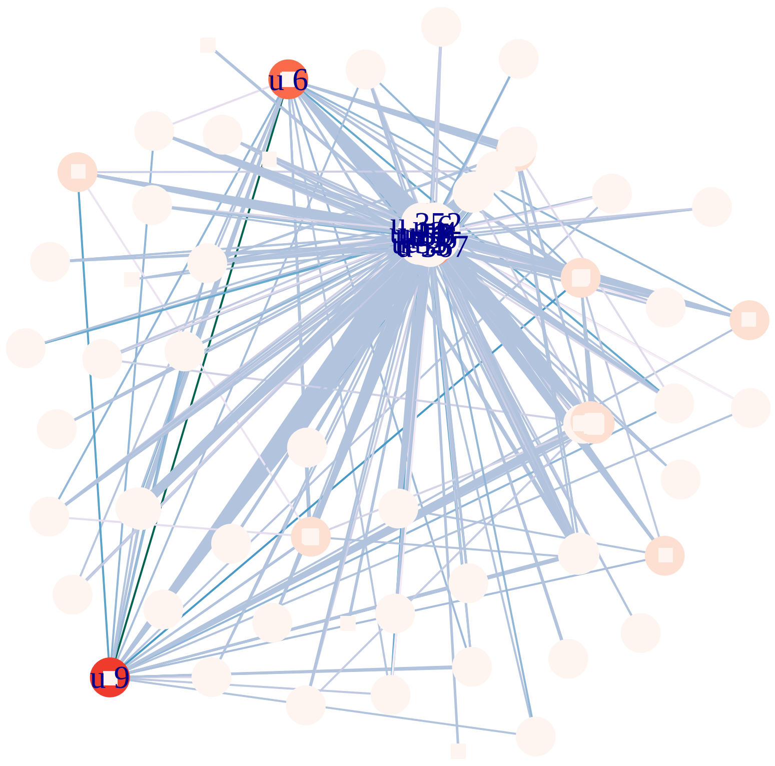


Degree, Strength and Betweenness are all extremely skewed to the left. Their means are 2.3, 5.9 and 6042. Eccentricity and Closeness more closely approximate Gaussian distributions. Closeness stil has signifcant left skew.

Visualize Characteristics on the Graph

Here we color vertices according to their attributes and show labels for more relevant vertices. Bounds are chosen such that some vertex names are legible.

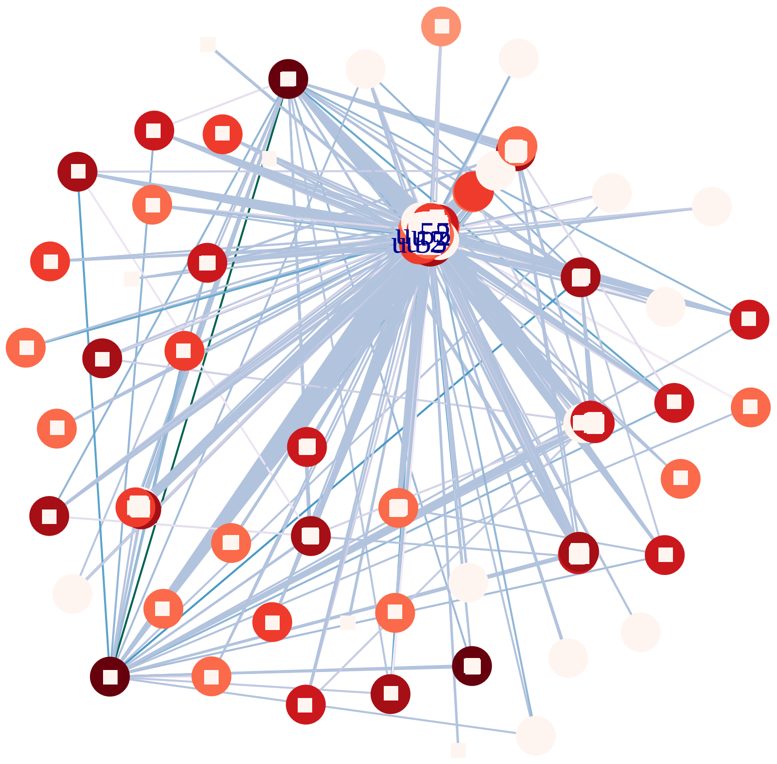
Degree



Labels are for vertices with degree higher than 100.

From the graph we can see there are a small number of users who have a lot edits but do not interact as much with users in the main cluster in the center. The users in ther center interact the most with each other.

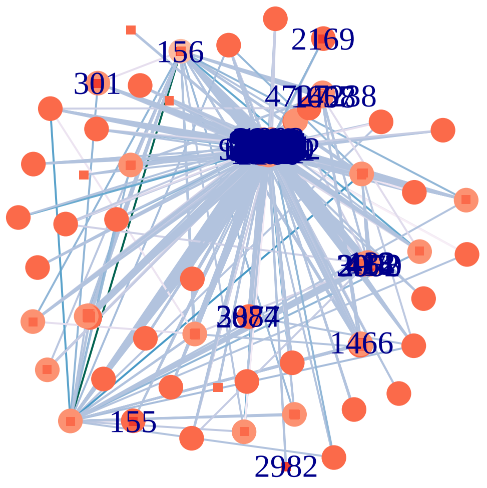
Betweenness



Labels are for vertices with betweenness higher than 400.

We can see that the labeled users again are clustered in the center, this makes sense as users with high betweenness have many edges to plot and will be clustered at the center. We can also observe that some darker (higher weight) edges come out from here.

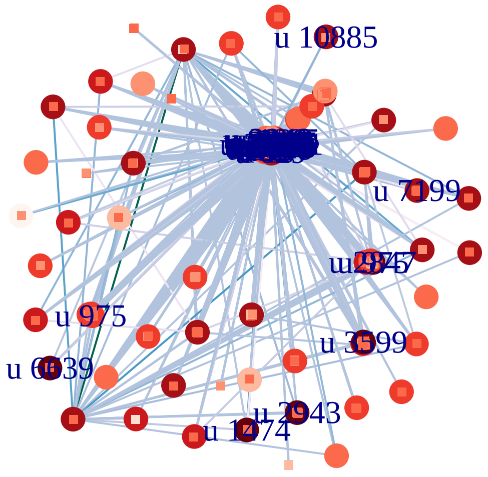
Eccentricity



Labels are for vertices with maximum eccentricity of 6.

We can see that all our high-eccentricity labels do not have a u on them. This means they are articles, and based on the construction of our graph articles all will only have on edge, connecting to the first edit they received. Thus it is quite intuitive that articles will have high eccentricity due to being distant from edits as editors interact with one-another.

Closeness

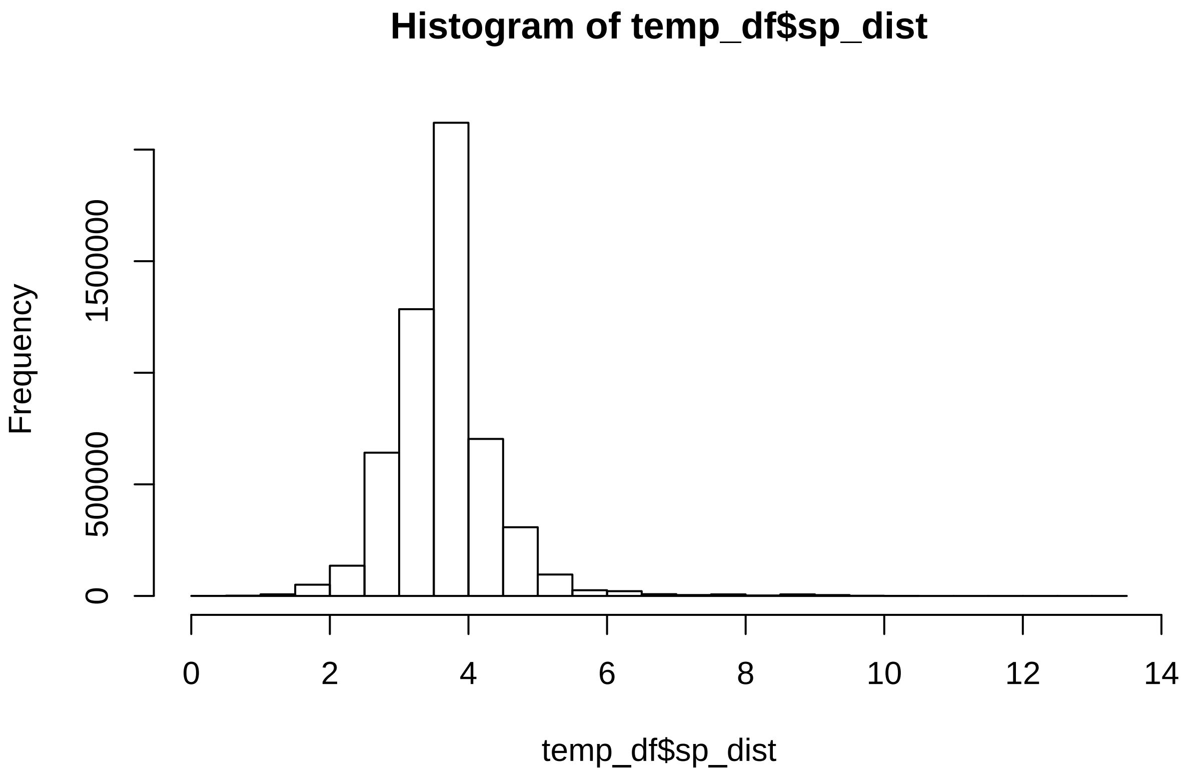


Labels are for vertices with closeness greater than 1.2e-04.

As expected, nodes with high closeness are clustered near the center.

Shortest Path Distribution

The distribution of the shortest-paths distances



Comparing with Null Models

Compare the network with random ones (ER and BA models).

Erdos-Renyi (ER)

Now we generate an Erdos-Renyi mode with the same number of vertices and edges as our graph.

```
## [1] 0
```

Connected Graphs

```
## [1] 0
```

Due to the ratio of edges to vertices, the random graphs are not connected! We attempted very large values of B (up to 50000) but zero graphs are connected. Thus, we conclude that ER method is not suitable for our dataset and do not try to compare the distribution, as all graph generated are disconnected.

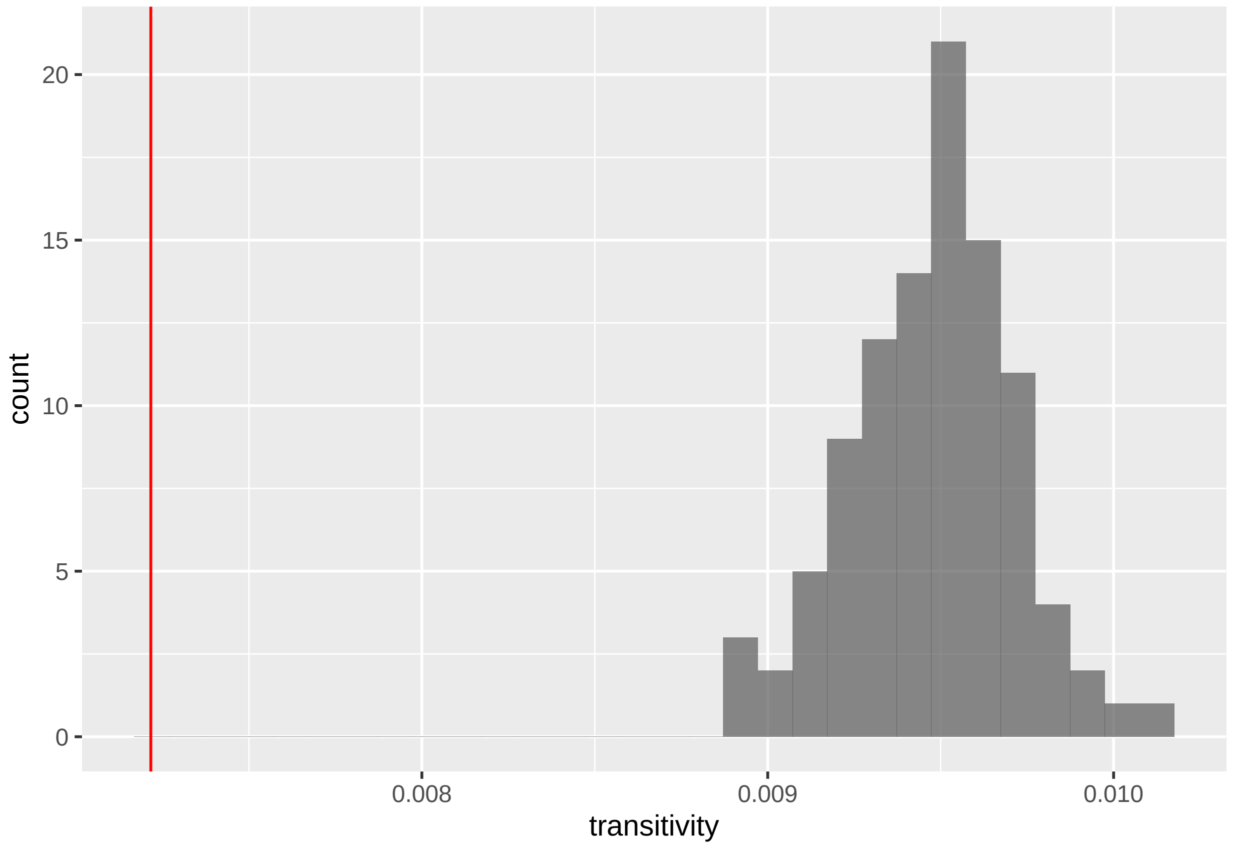
Barabasi-Albert (BA)

Connected Graphs

```
## [1] 100
```

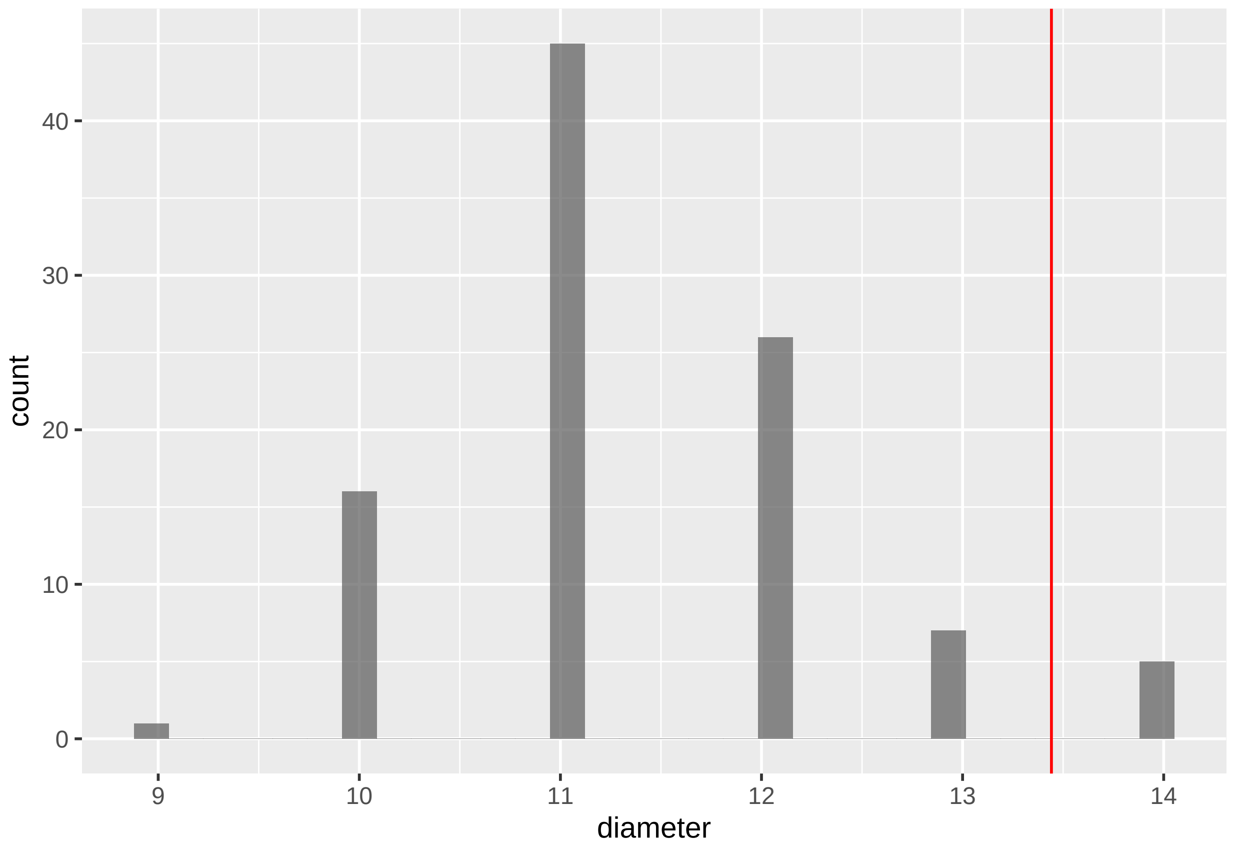
With the BA method we get much better rate of connectedness!

Compare Transitivity



Transitivity is much higher in our randomly generated graphs.

Compare Diameter



The mean and median diameter are higher in the random graphs but there is some overlap with our mean and the distribution at least.

Conlusion

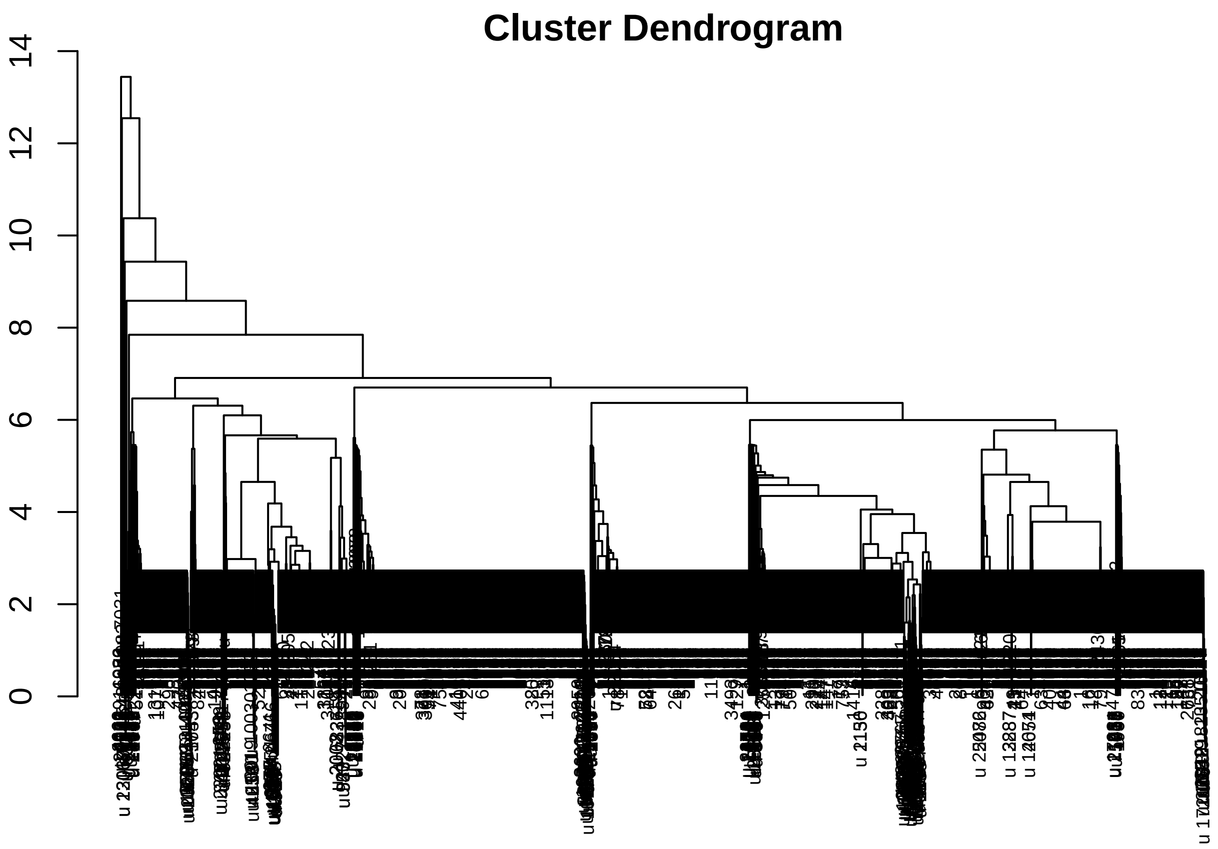
Overall it seems extremely unlikely for our graphs to be modeled by random graphs. Permutation tests would be more fit to testing our graph but their computational intensity renders them outside the scope of our project.

Clustering

For our dataset we are interested in modularity within the network, whcih would signal that some sort of community forms around common topics ad common users.

Check if clustering allows for better insights of the network. (by mkaing graphs with the clusters duh)

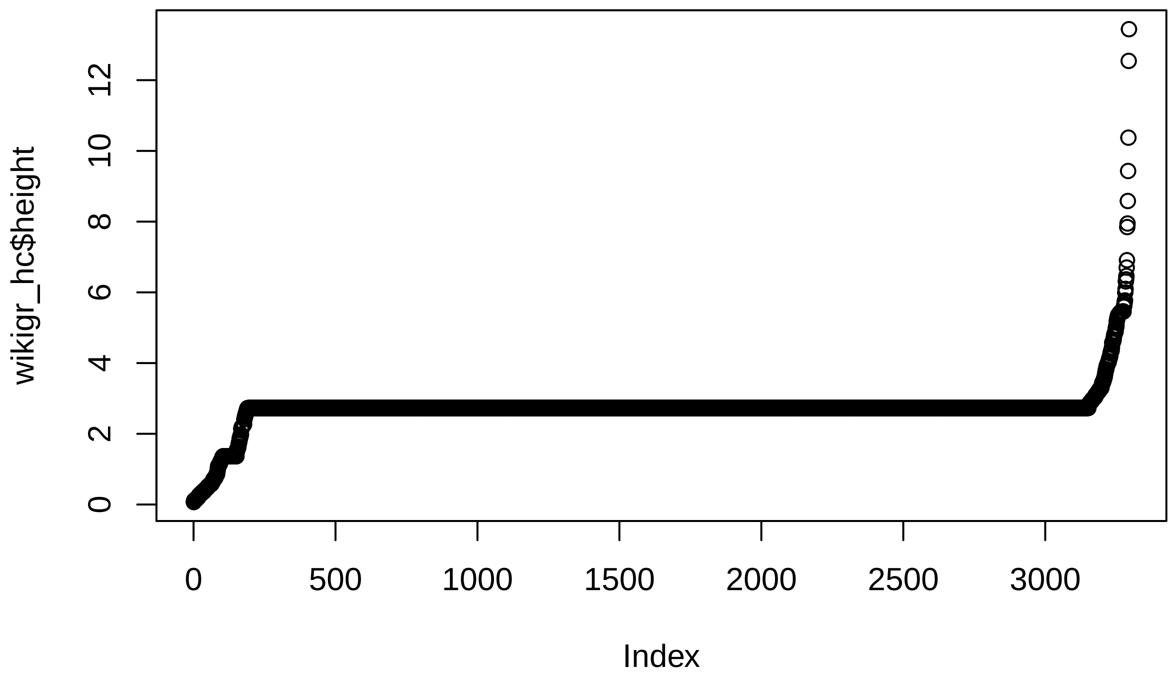
Shortest Paths - Complete



Here we want to measure the distance (or dissimilarity) between the users. In other words, we want to look for the users with similar articles edits.

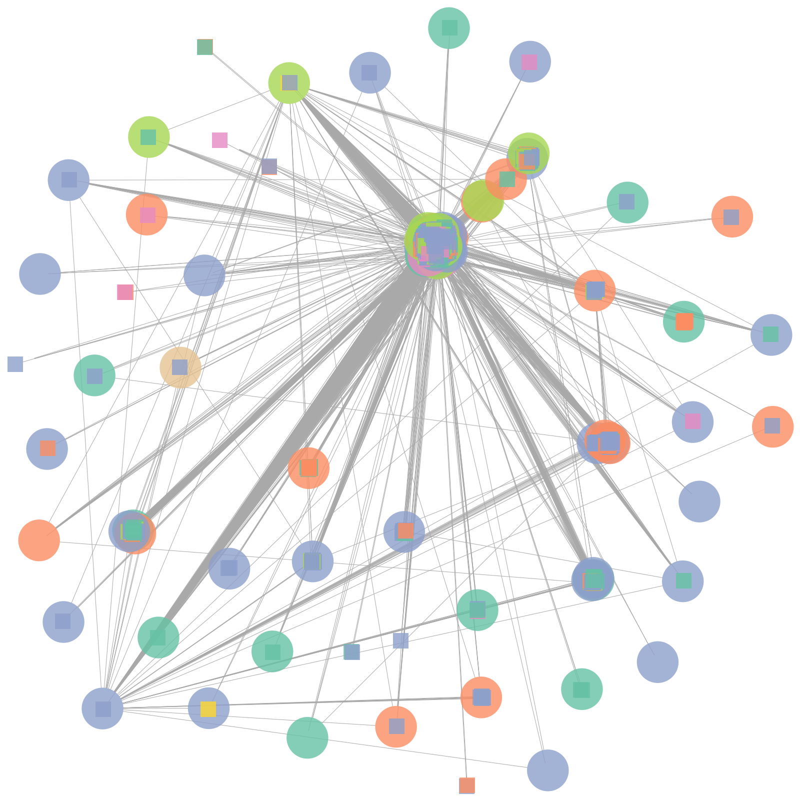
Here, we have a lot of users, which makes the interpretation ofthe Cluster Dendrogram difficult (the more objects there are to cluster, the more complex becomes the result). However, we know that the horizontal axis represents the clustersand the vertical scale (height) represents the distance or dissimilarity.

The position of a label has a little meaning too,the higher the position the later the object links with others,and hence more like it is an outlier or a stray one (here height=14)



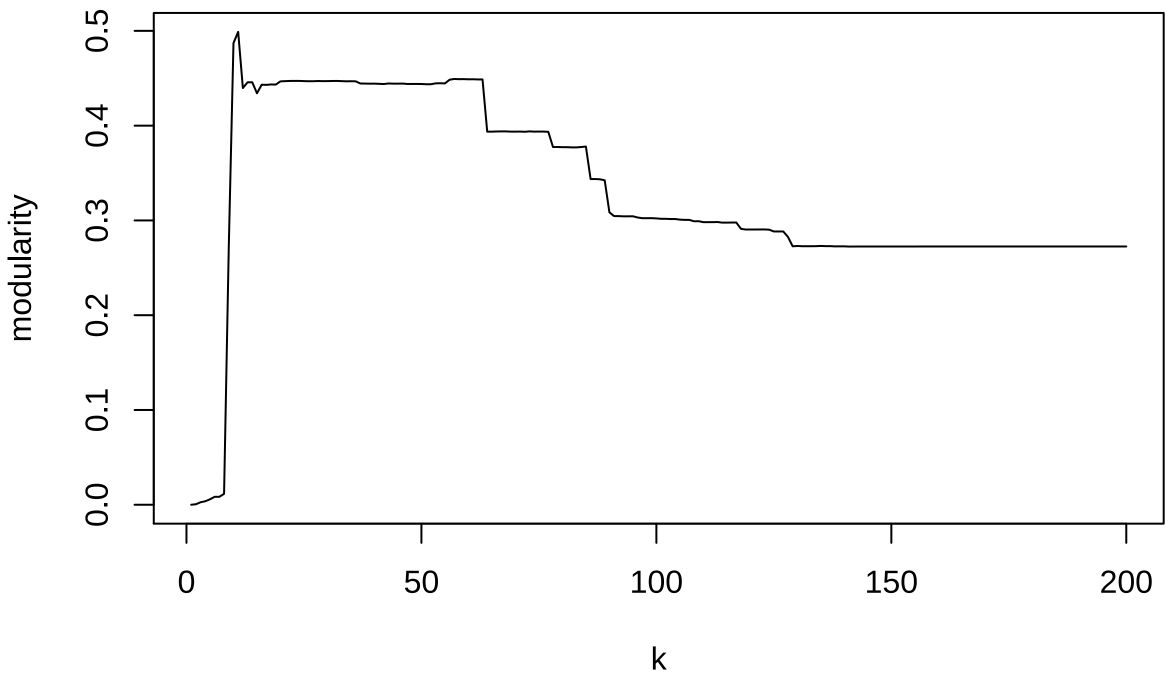
View the graph with clusters by color

With a height of 6, 14 clusters are chosen. We graph them below.



Modularity

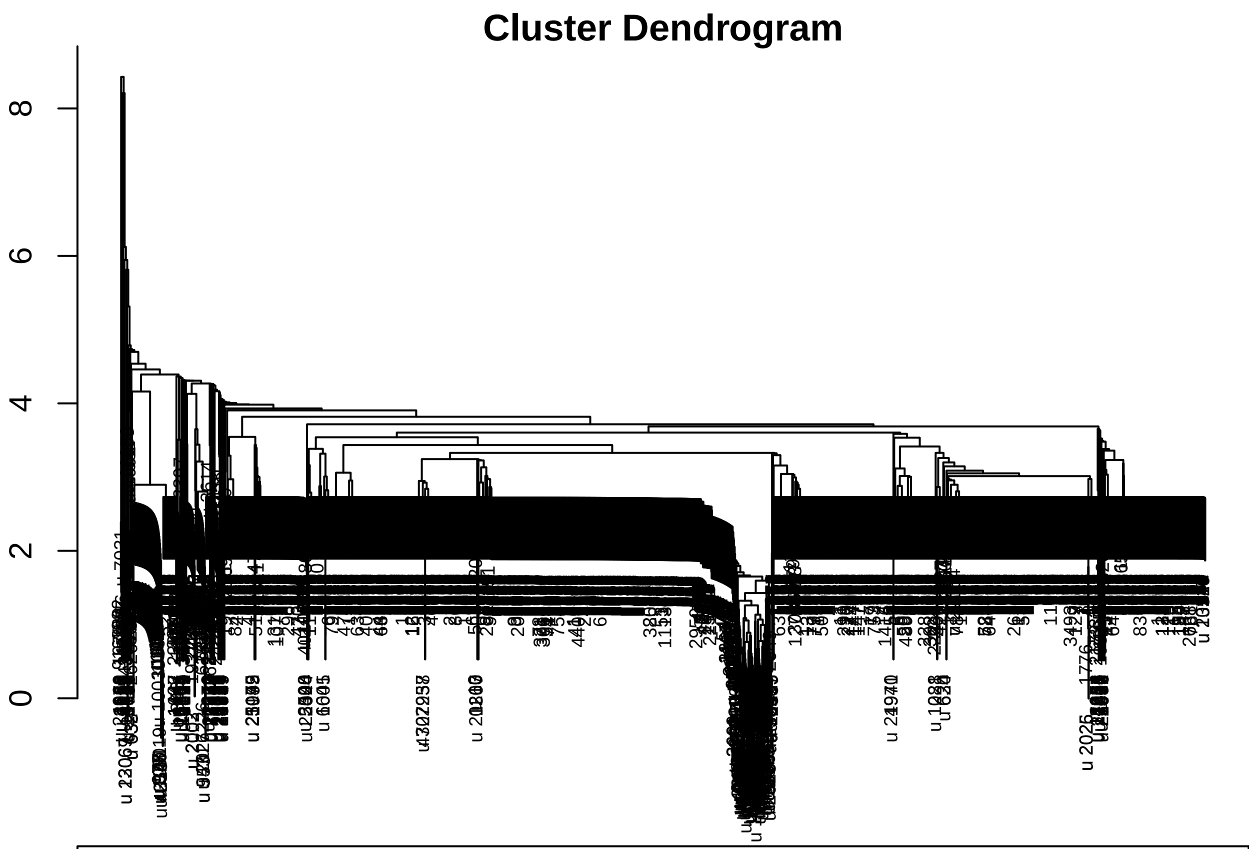
```
## [1] 0.4457943
```

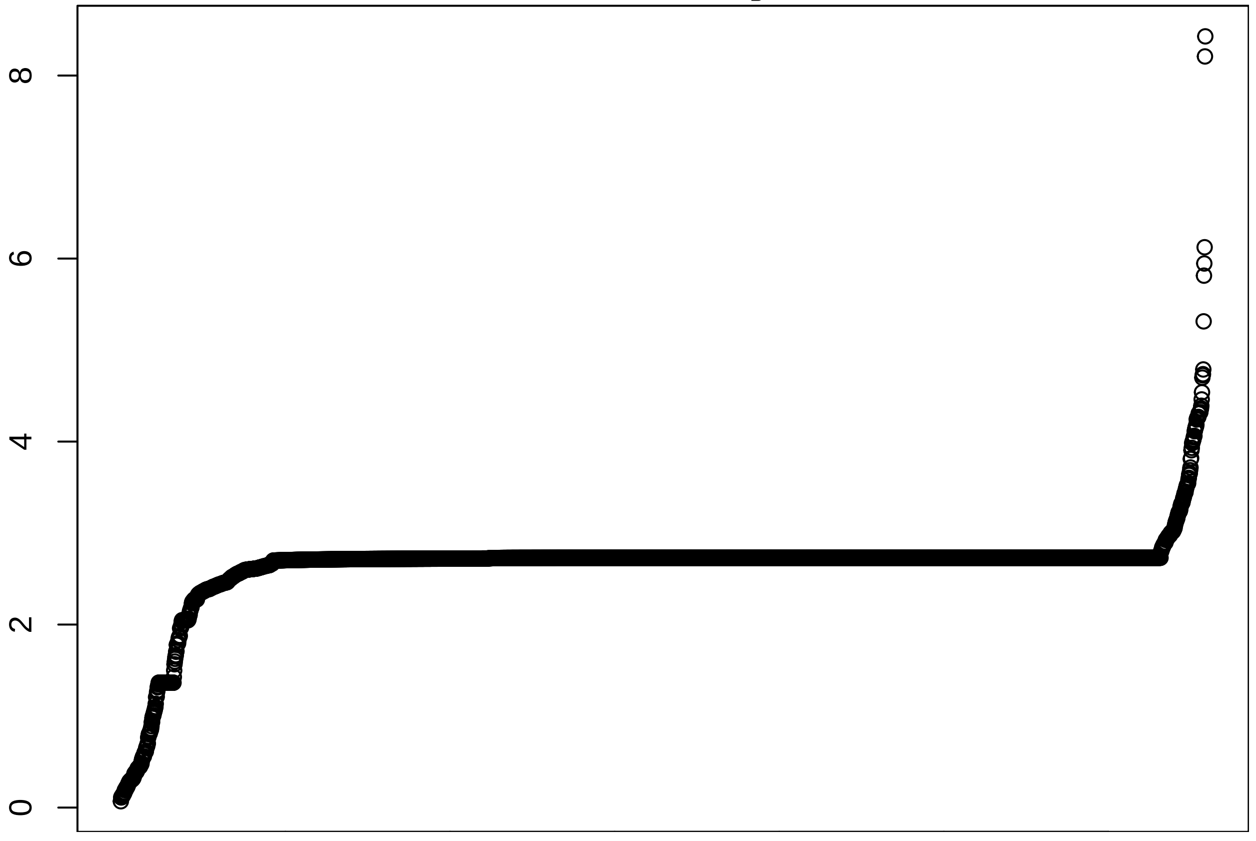


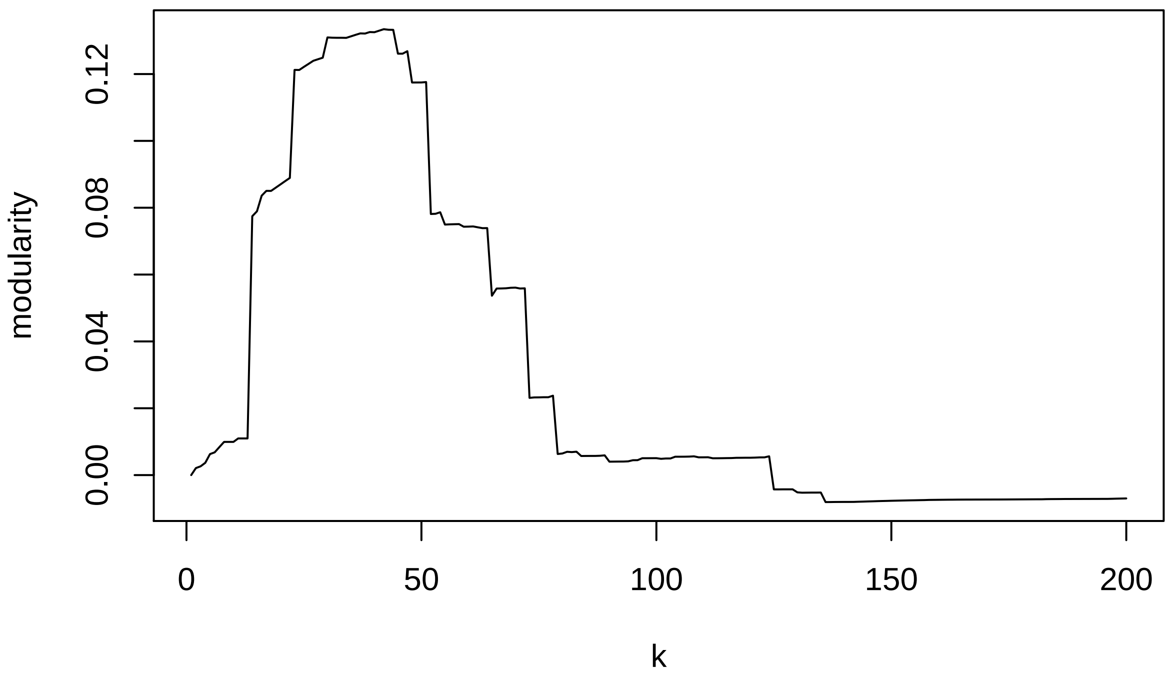
Modularity is a measure of the structureof networks, which is designed to measure the strength of division of a network into clusters. Networks with high modularity have dense connections between the vertices within clusters but sparse connections between vertices in different clusters. Here we got a positive value of modularity equals to 0.44 indicating a presence of community structures.

From the graph we can see that we can get higher modularity by choosing slightly fewer clusters. Around 10 gets modularity of 0.5.

Average Cluster method







We can see clearly that for all cluster choices, this method performs much worse at maximizing modularity.

Modularity Optimization Clustering Algorithms

Here we try three other methods: 1. **hierarchical cluster**, 2. **multilevel clustering**, and 3. **annealing clustering**.

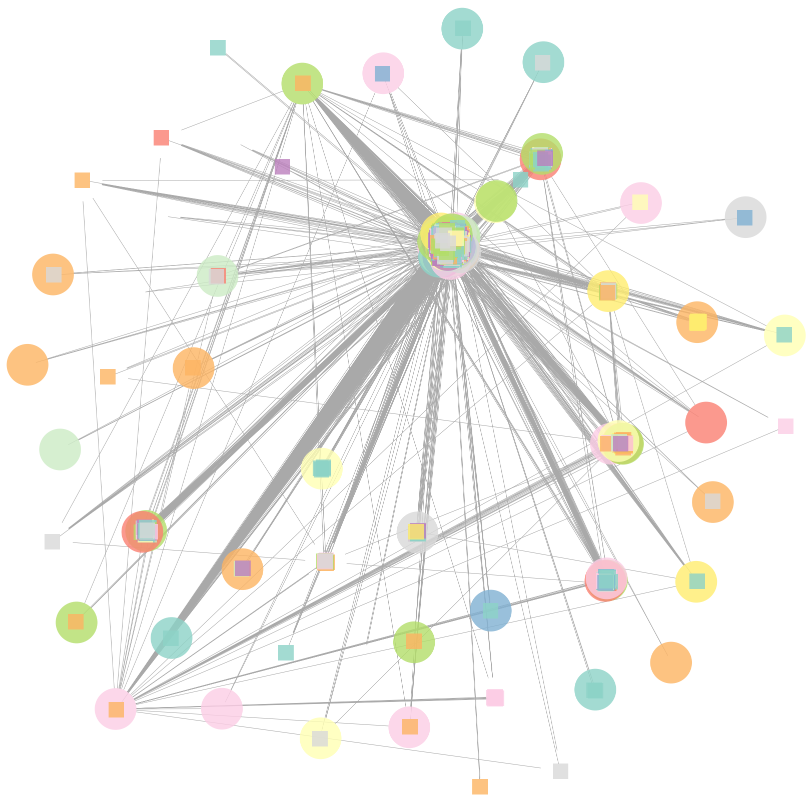
```
##   Modularity cluster_size
## 1  0.7785379           24
## 2  0.3763209           37
## 3  0.3034343           24
```

From the table we can see that Hierarchical clustering achieves the best modularity of 0.78. This suggest that in the social world of volunteer Wikipedia editors, there is a hierarchical-community type structure. Perhaps different users have different preferences on the types of edits they are comfortable making. For example we can imagine the following workflow:

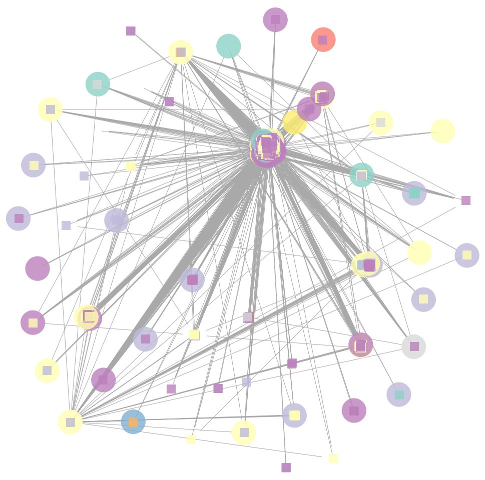
- user 1 is an expert in Graph Theory and sees that a certain article on Wikipedia does not have the most up to date science, so he decided to add the information
- user 2 sees this edit has grammar errors, and corrects them
- user 3 sees that there are no citations for the new information and adds a [citation needed]
- user 4 enjoys researching topics online, and so finds a citation to support the claim
- user 5 adds more information
  - users 2, 3, 4 all make more edits
- user 6 sees this topic is taking up a lot of space and creates it's own sub-section

Although we did not consider that Wikipedia would be very likely to be hierarchical, there is some logic too it if we can verify that different users perform different editing tasks which tend to be done in a consecutive manner. However, validating this hypothesis is outside of the scope of this project.

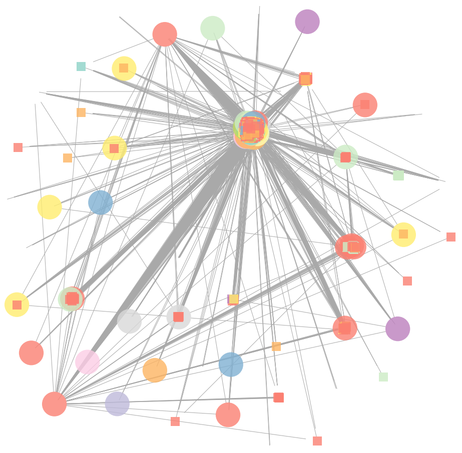
Hierarchical Clusters



Louvain Clusters



Annealing Clusters

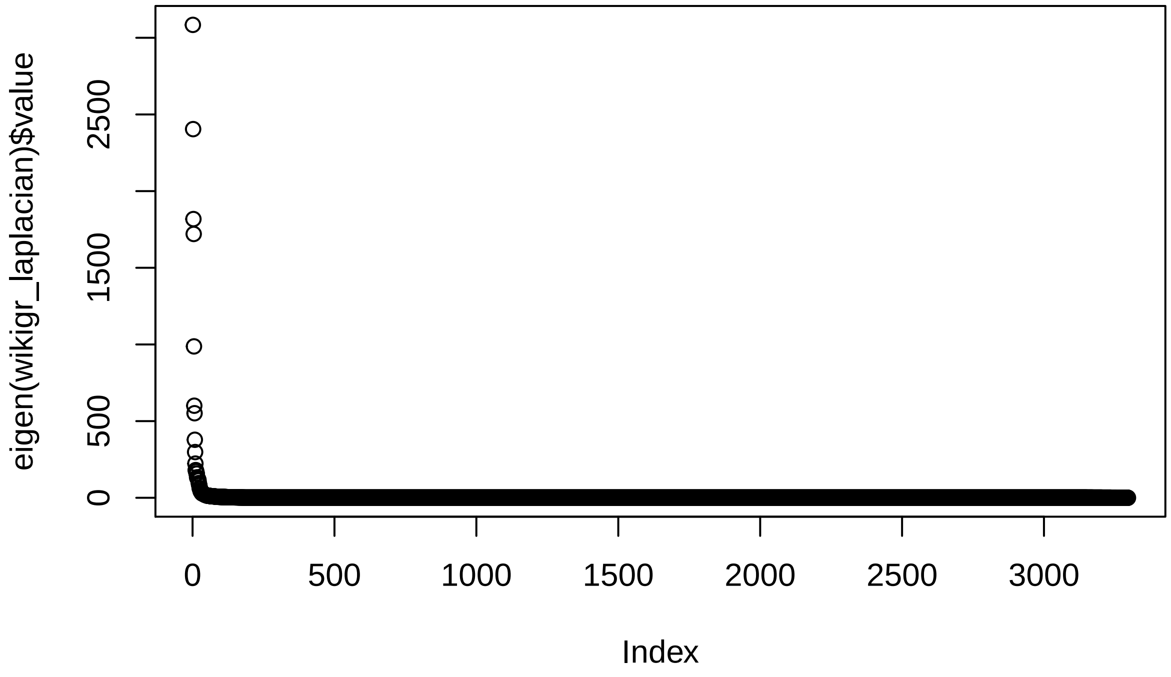


Due to the complexity of our graphs, the view of these clusters does not give much useful information. One interesting note is that the articles themselves (squares) often are in different clusters than the users who are plotted next to those articles. This indicates that the user who first edited that article in our dataset is more often editing articles related to other topics. When a article-user pair is the same color, they are in the same cluster because that user's first edit in our dataset is in similar to his other edits in the dataset.

Spectral Clustering

We use k-means algorith to perform spectral clustering. We use 24 clusters as that is what gave us the best modularity in out of all previous algorithms.

```
wikigr_laplacian <- laplacian_matrix(wikigr, weights = NULL)
plot(eigen(wikigr_laplacian)$value)
```



```r
wiki_eigs <- eigen(wikigr_laplacian)$vectors[ ,vcount(wikigr)-(1:24)]
wiki_spectral <- LICORS::kmeanspp(wiki_eigs, k=24, nstart=10)
modularity(wikigr, wiki_spectral$cluster)
```

```
## [1] 0.01871499
```

We observe that the modularity is much poorer than the previous algorithms. Additionally, with 24 clusters it is difficuly to plot with enough colors to distinguish so we will not plot. We also tried using smaller numbers of clusters such as 5 and 10 but the results were even poorer! To save space in the report and computation time, we omit these results.

References

- https://toreopsahl.com/tnet/two-mode-networks/clustering/
- https://toreopsahl.com/tnet/longitudinal-networks/
- http://konect.cc/networks/edit-zhwiki/