WORK GROUP

Francesco Esposito & Paolo Mantovani

2023-06-06

## Libraries Needed

set.seed(78987)  
library(igraph)  
library(tidyverse)  
library(ggraph)  
library(networkDynamic)  
library(tidygraph)  
library(graphics)  
library(dplyr)  
library(readxl)

#edges <- read.csv("C:/Users/PAOLO/OneDrive - Universita Cattolica Sacro Cuore - ICATT/Desktop/Git/Assignment12june/network.csv/edges.csv", header=T, as.is=T)  
#nodes <- read.csv("C:/Users/PAOLO/OneDrive - Universita Cattolica Sacro Cuore - ICATT/Desktop/Git/Assignment12june/network.csv/nodes.csv", header=T, as.is=T)  
edges <- read.csv("C:/Users/kecco/OneDrive/CATTOLICA/NETWORK ANALYSIS/edges.csv",header=T, as.is=T)  
nodes <- read.csv("C:/Users/kecco/OneDrive/CATTOLICA/NETWORK ANALYSIS/nodes.csv",header=T, as.is=T)

# The Dataset

This Dataset contains data of the network of . E-road network is a numbering system for roads in Europe developed by the United Nations Economic Commission for Europe (UNECE), also including Central Asian countries. It in quite important in order to facilitate transportation, connectivity and cooperation among the European nations.

The network is undirected and unweighted. The nodes represent cities while edge between two nodes denotes that they are connected by an E-road. We can start from creating the the network using the two files, one that contains the nodes of the network (the file nodes.csv) and the other which contains the edges (the file edges.csv). From the nodes file we will keep only the columns which give us information about the name of the city and the “id” of the city, which are the id’s of the file edges that indicate connected cities.

Let’s change the name of the variables (columns) of the dataset. We’ll have the two column of dataset “nodes” named (ID, name) and the two columns of the dataset “edges” named (from, to).

nodes <- nodes[, c(1, 2)]  
# Rename the columns  
  
colnames(nodes) <- c("ID", "name")  
colnames(edges) <- c("from", "to")

NODES resume:

glimpse(nodes)

## Rows: 1,174  
## Columns: 2  
## $ ID <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,…  
## $ name <chr> "Greenock", "Glasgow", "Preston", "Birmingham", "Southampton", "L…

We have 1174 cities, which will be the number of nodes of the network

EDGES resume:

glimpse(edges)

## Rows: 1,417  
## Columns: 2  
## $ from <int> 0, 1, 1, 2, 3, 3, 5, 5, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 8, 8, 8,…  
## $ to <int> 1, 2, 16, 3, 4, 854, 6, 879, 7, 21, 22, 410, 452, 453, 888, 8, 47…

As we can see, the edges in the network will be 1417, which is not an high number of edges with respect to the number of nodes.

Create the network

net <- graph\_from\_data\_frame(d=edges, vertices=nodes, directed=F)

Now we can see the main features of our network:

paste("The Size of the network:", gsize(net))

## [1] "The Size of the network: 1417"

paste("The Order of the network:", gorder(net))

## [1] "The Order of the network: 1174"

First of all we would like to specify that during the analysis we realized that when calculating betweenness, the most important summit turned out to be the city of Brest.

# Calculate betweenness centrality  
betweenness\_values <- betweenness(net)  
betweenness\_sorted <- sort(betweenness\_values, decreasing = TRUE)  
head(betweenness\_sorted)

## Brest Moscow Saint Petersburg Le Mans   
## 147644.9 146078.3 134358.4 120368.0   
## Rennes Minsk   
## 119855.2 119155.7

After looking on the internet why this city could be so important we saw that in Europe there are two cities named Brest: one is in France and one is in Belarus. So there could be some discrepancy between the edges that involve the city named Breast. First of all we would like to specify that during the analysis we realized that when calculating betweenness, the most important summit turned out to be the city of Brest. Thus we decided to delete it:

# Deleting Brest  
net = delete\_vertices(net, V(net)$name == "Brest")

We’ll see that this elimination implies significant variations. We’ll study them speeaking about the betweenness later.

After removing brest from the network we will have new size and new order:

# Size of our network  
paste("size of the network", gsize(net))

## [1] "size of the network 1413"

# order of the network  
paste("order of the network", gorder(net))

## [1] "order of the network 1173"

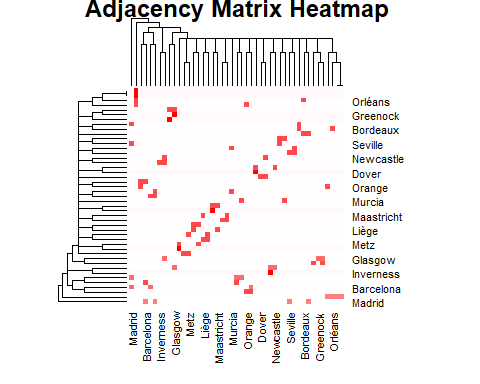
With the size not being that much larger than the order we can already conclude that the network must be somewhat sparse. This makes sense as we are dealing with a road network only involving major roads across the entirety of Europe and parts of Asia.

Let’s plot now an adjacency matrix which is a square matrix used in order to represent the relationships between nodes in a network. In our specific case, in the matrix which you can see here below each row and column in the matrix corresponds to a city, and the value corresponds to 1 if there is a connection (road) between the two cities, 0 otherwise.

adj\_mat <- net[c(1:45), c(1:45)]  
adj\_mat <- as.matrix(adj\_mat)

For example, looking at the first row and column for “Greenock,” we can see that there is a road connecting Greenock to “Glasgow” (indicated by a value of 1 in the corresponding cell). Similarly, in the second row and column for “Glasgow,” we can see that there are roads connecting Glasgow to “Greenock” and “Preston.”

# Create a color palette for the heatmap  
color\_palette <- colorRampPalette(c("white", "red"))(n = 130)  
# Plot the heatmap  
heatmap(adj\_mat, col = color\_palette, main = "Adjacency Matrix Heatmap")



OVERVIEW OF THE GRAPH

Let’s see now, how the newtork is characterized about the components, which are groups or subsets of node that are connected between each other. By looking at component we can see if in our graph there are subgroups or isolated subgraphs.

comp <- components(net)  
comp$no

## [1] 26

The amount of components is relatively large (26). It is reasonable to think that the number of components in high because of the presence of islands or somehow isolated city which have only one connection.

We can also visualize the size (number of nodes) of each component.

comp$csize

## [1] 39 1038 3 4 8 5 2 15 3 4 5 4 4 2 2  
## [16] 2 2 7 2 2 2 10 2 2 2 2

Most nodes belong to one large component, the reason is that they belong to continental europe. All the other components are rather small and so the assumption we made before (about islands and isolated zones of europe). This aspect will be confirmed later, providing a graphic of the distribution.

We can also take a look at different measurements like the diamater. The diameter measures the maximum distance or number of steps required to travel from one node to another, considering the shortest path.

diameter(net, directed = FALSE)

## [1] 62

This means that the longest shortest path between any two nodes in the graph is 62 edges long. In other words, there is at least one pair of nodes in the graph that are 62 edges apart in terms of the shortest path between them.

Another interesting property is the edge density, sometimes also referred to as cost. This value describes the proportion of present edges compared to all possible edges in the graph (assuming single edges between nodes). In fact, edge density is calculated as the number ov edges over the number of possible edges. We got a first hint at the sparsity of the network just from looking at the amount of nodes and edges presented in the graph, by calculating the edge density we confirm the idea we had at the beginning of the sparsity of the graph.

edge\_density(net)

## [1] 0.002055638

We see that the edge density have a very low value indicating that only about 0.2% of possible edges are present, thus the network is definitely sparse. However, if we contextualize this result to the network we are analyzing, it seems to be logical. It is normal that distant nodes (cities) are not connected to each other to go from one city to another you may pass from other cities, or, at any rate even between non-distant nodes there is no direct edge.

Clustering coefficient:

transitivity(net)

## [1] 0.03409091

The value indicates that the network has a relatively low level of clustering. This means that the nodes in the network are not strongly interconnected and do not form tightly-knit clusters. The value of 0.03409091 suggests that, on average, only a small fraction of the neighbors of a node are connected to each other. Our nodes have a very low tendency to form cluster or groups.

Finally, we are likely interested in how long it takes on average to get from A to B in our graph. This is exactly what the average shortest path lengths tells us.

mean\_distance(net, directed = FALSE, unconnected = TRUE)

## [1] 19.49411

Given some knowledge about the network we are dealing with, the results make a lot of sense. The diameter is large compared to the average shortest paths, as our network includes for example connections from Portugal to Greece (which is a very big distance). While there is no distance information in our network, it appears straightforward that such a connection would travel through many nodes and edges. However, in contrast especially for nodes in the central part of Europe it seems likely that there are: a) many nodes and b) many connections between the nodes, making (graph) distances short. Finally, the relative sparsity of the network is easy to explain, as it seems unlikely that there are direct connections between nodes that are thousands of kilometres apart.

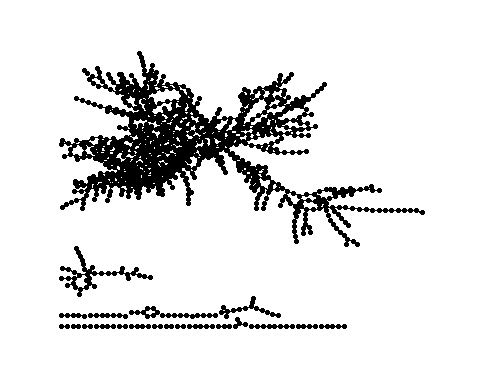
VISUALIZATION

Since our nodes are geographic positions could be useful, in order to understand better how our network behaves, to use the coordinates (longitude and latitude) of our nodes to have a more precise plot, because as we can see in the next plot without using coordinates the plot would be more confused, as we can see below here:

ggraph(net, layout = "auto") +  
 geom\_edge\_link() +  
 geom\_node\_point() +  
 theme\_graph()

## Using "stress" as default layout

## Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` in the `default\_aes` field and elsewhere instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



Firstly we have to import our coordinates dataset and do some codes to adjust our tables to plot the network correctly

coordinates <- read\_xlsx("C:/Users/kecco/OneDrive/CATTOLICA/NETWORK ANALYSIS/coordinates.xlsx")

## New names:  
## • `` -> `...4`  
## • `` -> `...5`

coordinates <- coordinates[-402,]  
coordinates <- coordinates[,c(1,2,3)]  
ID <- 0:1172  
coordinates <- cbind(coordinates, ID)  
coordinates <- mutate(coordinates, lon = as.numeric(lon),  
 lat = as.numeric(lat),  
 ID = as.numeric(ID))  
  
nodes <- nodes[-402,]  
nodes <- cbind(nodes, ID)  
nodes <- nodes[,c(2,3)]  
net\_df <- as.data.frame(get.data.frame(net))  
joined\_coord <- nodes %>% left\_join(coordinates, by = "ID")  
attach(joined\_coord)

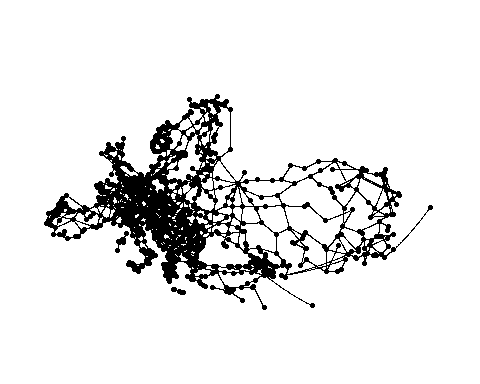
## Il seguente oggetto è mascherato \_da\_ .GlobalEnv:  
##   
## ID

We can see in the plot below the clear shape of Europe and we can do some Comment. For example in the left side of the graph (Western Europe), nodes are more dense, while in the right side (eastern europe) are more sparse. Because of this difference of sparsity is easier to see nodes that have more edges than in the western europe, in whihch in some areas is more difficult to identify them.

ggraph(net, x = lon, y = lat) +  
 geom\_edge\_link() +  
 geom\_node\_point() +  
 theme\_graph() +  
 coord\_map(projection = "albers", lat0 = 40, lat1 = 65) +  
 theme(plot.margin = margin(0, 0, 0, 0, "cm"))

## Warning: Removed 9 rows containing non-finite values (`stat\_edge\_link()`).

## Warning: Removed 2 rows containing missing values (`geom\_point()`).



CENTRALITY MEASURES

Centrality measures in network analysis quantify the importance or influence of individual nodes within a network. By looking at these measures we want to find which nodes play significant roles or have central position.

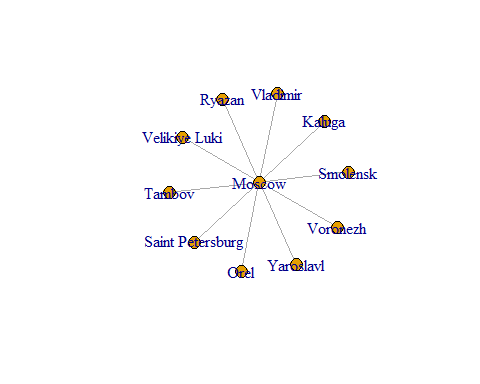
DEGREE

We will start from the degree. Let’s compute the values degree, sort them and reach the main actors.

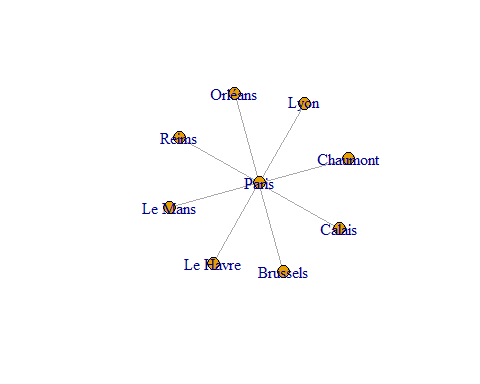
d <- degree(net)  
sorted <- sort(d, decreasing = TRUE)  
head(sorted)

## Moscow Paris Liège Munich Berlin Budapest   
## 10 8 8 8 8 8

ord\_deg <- order(d, decreasing = TRUE)[1]   
ord\_deg2 <- order(d, decreasing = TRUE)[2]  
g\_moscow <- subgraph.edges(net, E(net)[.inc(ord\_deg)])  
g\_paris <- subgraph.edges(net, E(net)[.inc(ord\_deg2)])  
  
plot(g\_moscow)



plot(g\_paris)

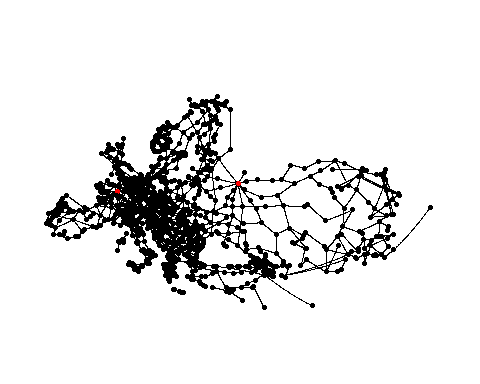


Let’s show now the two cities with the highest degree value in the European plot.

ggraph(net, x = lon, y = lat) +  
 geom\_edge\_link() +  
 geom\_node\_point(color = ifelse(Name == "Moscow" | Name == "Paris", "red", "black")) +  
 theme\_graph() +  
 coord\_map(projection = "albers", lat0 = 40, lat1 = 65) +  
 theme(plot.margin = margin(0, 0, 0, 0, "cm"))

## Warning: Removed 9 rows containing non-finite values (`stat\_edge\_link()`).

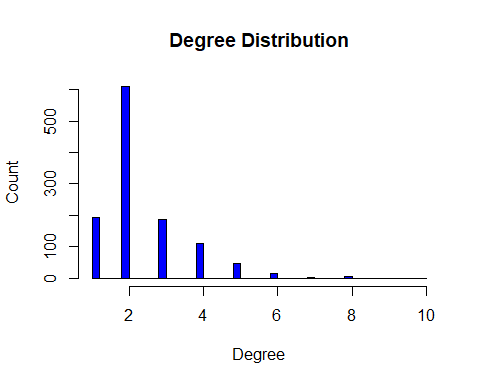
## Warning: Removed 2 rows containing missing values (`geom\_point()`).



We can see that the cities that have highest connections are Moscow (10 connections) and Paris (8 connections), it means that this two nodes are important in our network (not surprising, in fact those are two of the most important city in europe).

Histogram of the degree:

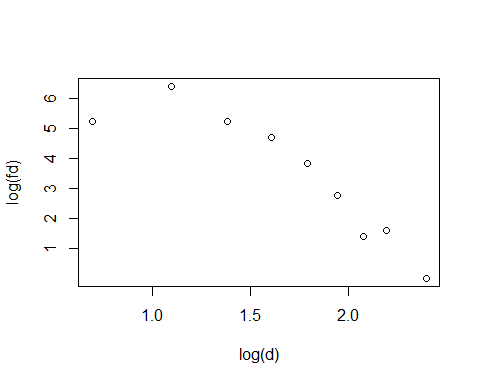
# Plot the degree distribution  
hist(d, breaks = "FD", col = "blue", xlab = "Degree", ylab = "Count", main = "Degree Distribution")



From the plot we can see that the majority of nodes have a degree lower or equal than 2.

We can now find the degree distribution, which is the probability distribution that describes the frequencies or proportions of nodes with different degrees:

degree\_dist <- function (g) {  
 fd <- table(degree(g))  
 d <- as.numeric(names(fd)) + 1 # degree + 1  
 list(d = d, fd = fd)  
}  
  
dd <- degree\_dist(net)  
with(dd, plot(log(d), log(fd)))



Now let us summarize the degree distribution by computing its exponent. Recall that a power law with exponent stipulates that and so . We can now fit either a simple linear regression or a Poisson log-linear regression.

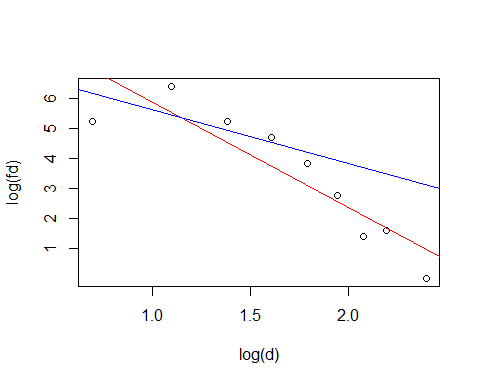
(m0 <- lm(log(fd) ~ log(d), data = dd))

##   
## Call:  
## lm(formula = log(fd) ~ log(d), data = dd)  
##   
## Coefficients:  
## (Intercept) log(d)   
## 9.388 -3.504

(m1 <- glm(fd ~ log(d), family = poisson, data = dd))

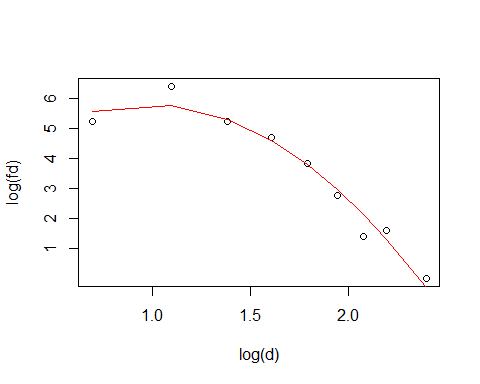
##   
## Call: glm(formula = fd ~ log(d), family = poisson, data = dd)  
##   
## Coefficients:  
## (Intercept) log(d)   
## 7.423 -1.786   
##   
## Degrees of Freedom: 8 Total (i.e. Null); 7 Residual  
## Null Deviance: 1898   
## Residual Deviance: 831.6 AIC: 883.6

with(dd, plot(log(d), log(fd)))  
abline(a = m0$coef[1], b = m0$coef[2], col = "red")   
abline(a = m1$coef[1], b = m1$coef[2], col = "blue")

 The number of observation is quite small to obtain adequate fittibility of the model. From the plot seems that the linear model is more suitable.

Seeing the behavior of the observations, its quite clear that the behavior may be better represented by a polynomial model. Thanks to this observation, we decided to predict and try to fit a polynomial model of degree = 2:

# Fit the model  
model <- lm(log(fd) ~ poly(log(d), degree = 2), data = dd)  
  
# Create the plot  
plot(log(dd$d), log(dd$fd), xlab = "log(d)", ylab = "log(fd)")  
  
# Generate predicted values  
pred <- predict(model, newdata = dd)  
sorted\_index <- order(dd$d)  
lines(log(dd$d)[sorted\_index], pred[sorted\_index], col = "red")



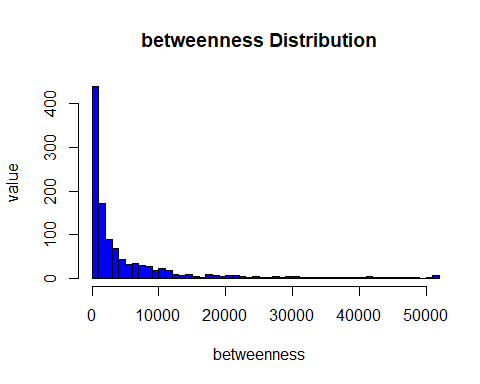
This seems to fit better the data, even if it may overfit them. However, in the following steps we’ll use the traditional linear model in order to compare the original distribution with the samples network, because it is quite better than the polynomial regression.

BETWEENNESS

Betweenness centrality measures the ability of a node to be a “bridge” controlling the flow of informations between different parts of the network. This measure is calculated by determining the fraction of shortest paths in the network that pass through a node. In fact, if a node has a betweenness of 0, it means that the node is not involved in any shorter paths between pairs of other nodes in the graph. In other words, the node is not located along the shortest paths between any pair of nodes in the graph. So that node will not be very important in the network. On the other hand if a node has high betweenness it means that the node is involved in many shortest paths, so it will be important in the network for the flow of informations.

As we have done in the previous case we check the distribution of the betweeness.

# betweenes  
b <- betweenness(net)  
hist(b, breaks = "FD", col = "blue", xlab = "betweenness", ylab = "value", main = "betweenness Distribution", xlim = c(0, 50000))

 If a node has a betweenness of 0, it means that the node is not involved in any shorter paths between pairs of other nodes in the graph. In other words, the node is not located along the shortest paths between any pair of nodes in the graph.

Also in this case we try to fit both the linear model and the Poisson model.

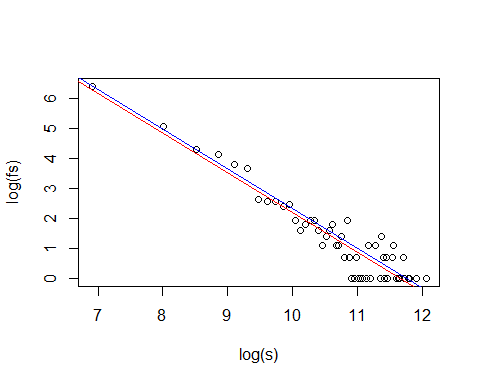
netstat\_dist <- function (s, ...) {  
 h <- hist(s, plot = FALSE, ...)  
 bin\_breaks <- as.vector(stats::filter(h$breaks, c(1, 1) / 2))  
 bin\_breaks <- bin\_breaks[-length(h$breaks)]  
 list(s = bin\_breaks[h$counts > 0], fs = h$counts[h$counts > 0])  
}  
  
net\_betw <- netstat\_dist(betweenness(net), breaks = 100)  
(m0 <- lm(log(fs) ~ log(s), data = net\_betw))

##   
## Call:  
## lm(formula = log(fs) ~ log(s), data = net\_betw)  
##   
## Coefficients:  
## (Intercept) log(s)   
## 15.51 -1.33

(m1 <- glm(fs ~ log(s), family = "poisson", data = net\_betw))

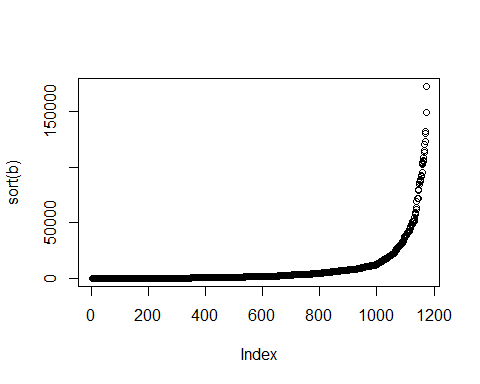
##   
## Call: glm(formula = fs ~ log(s), family = "poisson", data = net\_betw)  
##   
## Coefficients:  
## (Intercept) log(s)   
## 15.620 -1.328   
##   
## Degrees of Freedom: 53 Total (i.e. Null); 52 Residual  
## Null Deviance: 4652   
## Residual Deviance: 46.69 AIC: 226.1

with(net\_betw, plot(log(s), log(fs)))  
abline(a = m0$coef[1], b = m0$coef[2], col = "red")   
abline(a = m1$coef[1], b = m1$coef[2], col = "blue")

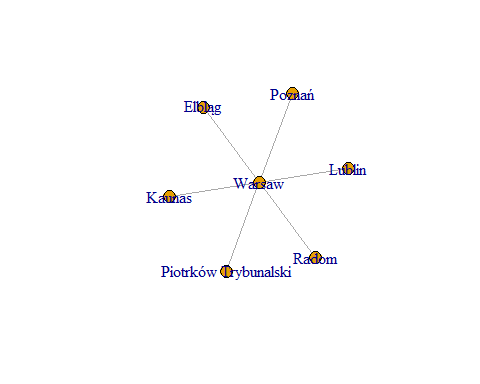
 We can see that the both the linear model and the Poisson are very similar and seems to fit the data well.

We can also check the importance of the nodes recognizing the important actors in our network analyzing the betweenness as we did for degree.

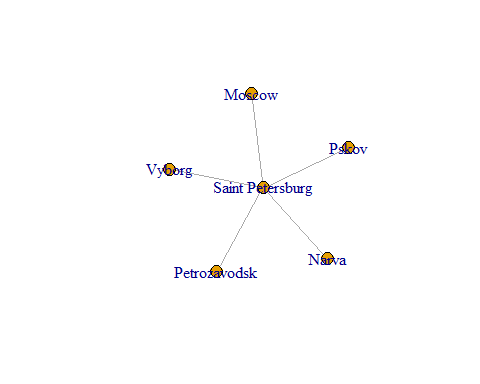
# Plot sorted betweenness centrality  
plot(sort(b))



imp\_act1 <- order(b, decreasing = TRUE)[1] #highest is at position (idx) 610  
imp\_act2 <- order(b, decreasing = TRUE)[2]  
  
g1 <- subgraph.edges(net, E(net)[.inc(imp\_act1)])  
g2 <- subgraph.edges(net, E(net)[.inc(imp\_act2)])  
plot(g1)



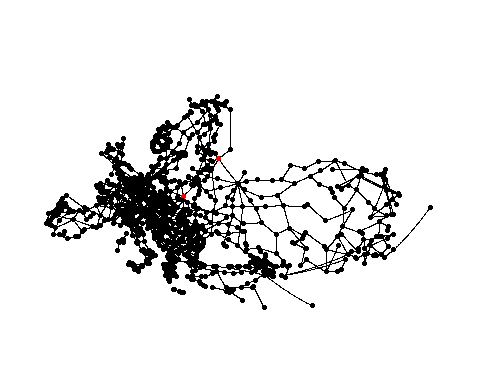
plot(g2)



ggraph(net, x = lon, y = lat) +  
 geom\_edge\_link() +  
 geom\_node\_point(color = ifelse(Name == "Warsaw" | Name == "Saint Petersburg", "red", "black")) +  
 theme\_graph() +  
 coord\_map(projection = "albers", lat0 = 40, lat1 = 65) +  
 theme(plot.margin = margin(0, 0, 0, 0, "cm"))

## Warning: Removed 9 rows containing non-finite values (`stat\_edge\_link()`).

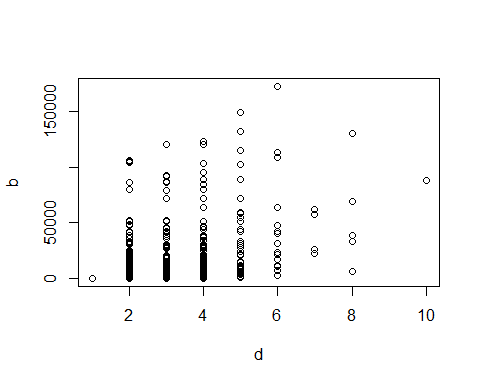
## Warning: Removed 2 rows containing missing values (`geom\_point()`).



As we can see, the variation in the betweenness values after removing the vertex “Brest” is quite important. It can be attributed to the structural changes in the network. When a vertex is removed, the paths and connections between the remaining vertices might change, resulting in different betweenness values. When a vertex is removed, the paths that previously went through that vertex may now be rerouted through different paths and vertices. This can lead to changes in the betweenness values of the remaining vertices.

RELATION B AND D + CORRELATION

plot(d, b)



cor(d, b)

## [1] 0.4872393

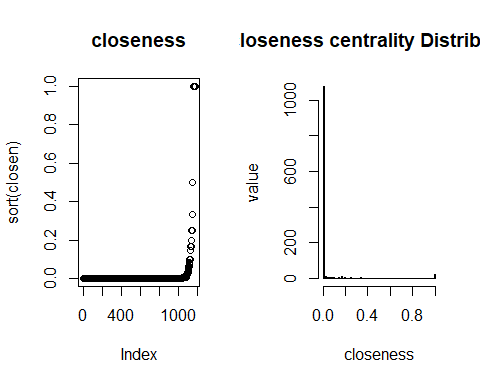
The positive correlation of 0.48 suggests that there is a general tendency for nodes with high degree (having a large number of direct connections) to also have high betweenness (being often found along the shortest paths between other nodes). However, it should be noted that correlation does not imply a cause-and-effect relationship between between betweenness and degree. There may be an indirect relationship or influences of other factors in the network that contribute to this moderate positive correlation.

CLOSENESS Closeness centrality measures how many steps are required to access every other node from a given node. In other words, important nodes have easy access to other nodes given multiple connections. The closeness centrality is defined as the reciprocal of the sum of the shortest path distances from a node to all other nodes in the network. Mathematically: A higher closeness centrality score indicates that a node is closer to other nodes in the network and can reach them more quickly.

closen <- closeness(net)  
top\_24 <- head(sort(closen, decreasing = TRUE), 50)  
top\_24

## Å Svolvær Usharal   
## 1.0000000 1.0000000 1.0000000   
## Druzhba Trosna Glukhkov   
## 1.0000000 1.0000000 1.0000000   
## Courtney Troyes Memmingen   
## 1.0000000 1.0000000 1.0000000   
## Füssen Krasnodar Djoubga   
## 1.0000000 1.0000000 1.0000000   
## Digoin Chalon-sur-Saône Lugoj   
## 1.0000000 1.0000000 1.0000000   
## Ilia Spezzano Albanese Sybaris   
## 1.0000000 1.0000000 1.0000000   
## Sant'Eufemia Lamezia Catanzaro Alcamo   
## 1.0000000 1.0000000 1.0000000   
## Trapani Eleusina Thebes   
## 1.0000000 1.0000000 1.0000000   
## Porto-Vecchio Førde Bastia   
## 0.5000000 0.5000000 0.3333333   
## Bonifacio Sassari Sandane   
## 0.3333333 0.3333333 0.3333333   
## Lavik Sveio Aksdal   
## 0.3333333 0.2500000 0.2500000   
## Florina Vévi Kırıkhan   
## 0.2500000 0.2500000 0.2500000   
## Reyhanlı Porto Torres Cagliari   
## 0.2500000 0.2000000 0.2000000   
## Olbia Heraklion Stord   
## 0.2000000 0.1666667 0.1666667   
## Bokn Antakya Kristalopigi   
## 0.1666667 0.1666667 0.1666667   
## Géfira Topboğazi Cilvegözü   
## 0.1666667 0.1666667 0.1666667   
## Chania Agios Nikolaos   
## 0.1428571 0.1428571

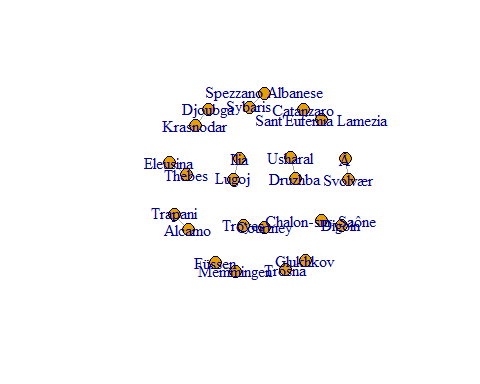
par(mfrow=c(1,2))  
plot(sort(closen), main = "closeness") ;hist(closen, breaks = 100, col = "blue", xlab = "closeness", ylab = "value", main = "closeness centrality Distribution")



We can see from the closeness plot that there is a big group of nodes with the highest closeness, we can see below what are these nodes.

To plot the network involving only the 24 vertices with highest closeness:

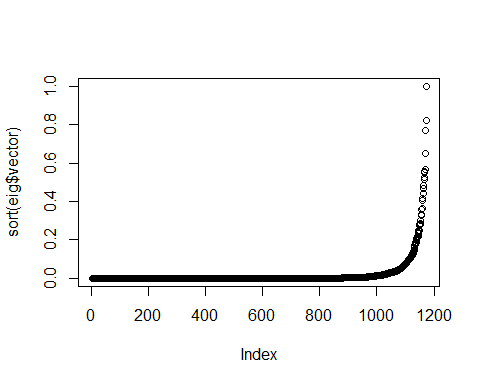
# Create the subgraph with the top 24 vertices  
subgraph <- induced\_subgraph(net, V(net)[order(closen, decreasing = TRUE)[1:24]])  
# Plot the subgraph  
plot(subgraph)



High closeness for these nodes means that are closer, in average, to other nodes. As we can see from the plot the points with higher closeness centrality are small cities or not much important cities. This because some of this small cities are, in some cases, connected with only one city which is very near.

Eigenvector Centrality Measures the importance of a node taking into account his connection with other important nodes. A node with a high eigenvector centrality means that is connected with other nodes that are important for the network.

eig <- eigen\_centrality(net)  
plot(sort(eig$vector))



imp\_eig <- order(eig$vector, decreasing = TRUE)[1] #highest is at position (idx) 610  
imp\_eig2 <- order(eig$vector, decreasing = TRUE)[2] #highest is at position (idx) 610  
imp\_eig3 <- order(eig$vector, decreasing = TRUE)[3] #highest is at position (idx) 610  
imp\_eig4 <- order(eig$vector, decreasing = TRUE)[4] #highest is at position (idx) 610  
#highest is at position (idx) 610  
V(net)$name[imp\_eig]

## [1] "Paris"

V(net)$name[imp\_eig2]

## [1] "Metz"

V(net)$name[imp\_eig3]

## [1] "Reims"

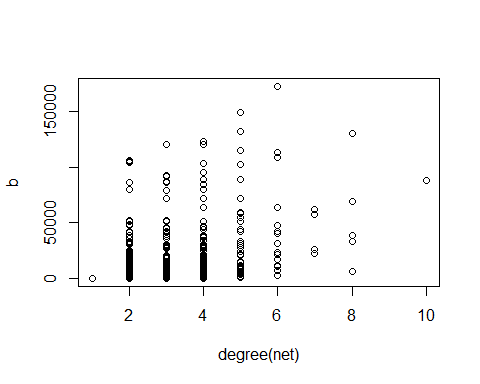
V(net)$name[imp\_eig4]

## [1] "Brussels"

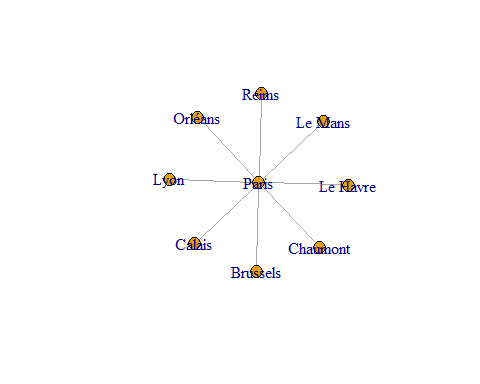
head(sort(eig$vector, decreasing = TRUE), 50)

## Paris Metz Reims   
## 1.0000000 0.8249955 0.7702978   
## Brussels Liège Le Mans   
## 0.6485099 0.5667592 0.5542082   
## Orléans Luxembourg Charleville-Mézières   
## 0.5521757 0.5239585 0.5148709   
## Lyon Rouen Geneva   
## 0.4828394 0.4674290 0.4404500   
## Calais Lausanne Tours   
## 0.4151245 0.4080763 0.3666606   
## St. Vith Bordeaux Toulouse   
## 0.3652844 0.3606514 0.3346710   
## Leuven Le Havre Chaumont   
## 0.3264429 0.3001180 0.2860588   
## Saarbrücken Charleroi Chambéry   
## 0.2850544 0.2810846 0.2783326   
## Eupen Angers Clermont-Ferrand   
## 0.2748604 0.2496858 0.2483760   
## Amsterdam Saint-Avold Antwerp   
## 0.2377822 0.2313342 0.2202159   
## Trier Namur Ghent   
## 0.2186227 0.2114676 0.2108713   
## San Sebastián Amiens Bruges   
## 0.2067173 0.2032730 0.1969431   
## Bern Bastogne Martigny   
## 0.1875570 0.1854946 0.1831112   
## Arlon Maastricht Orange   
## 0.1769504 0.1736894 0.1603085   
## Montargis Mulhouse Rennes   
## 0.1518730 0.1469048 0.1382296   
## Caen Mont Blanc Eindhoven   
## 0.1344477 0.1320418 0.1296193   
## Grenoble Mâcon   
## 0.1284177 0.1224641

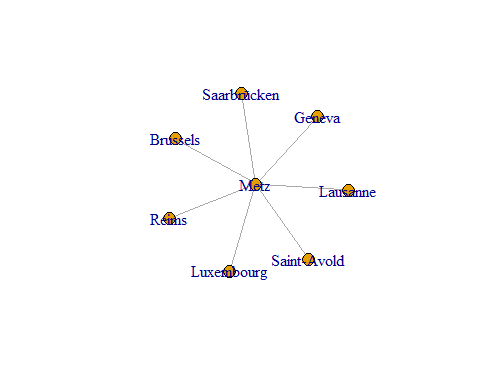
plot(degree(net), b)



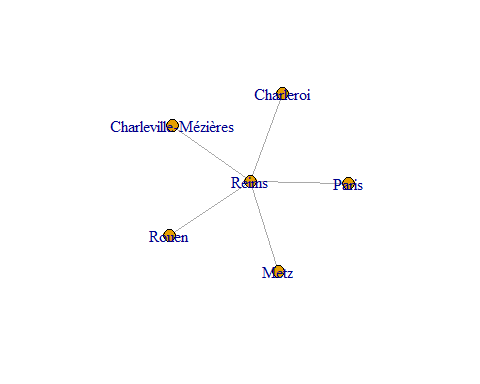
# ego network  
g1 <- subgraph.edges(net, E(net)[.inc(imp\_eig)])  
g2 <- subgraph.edges(net, E(net)[.inc(imp\_eig2)])  
g3 <- subgraph.edges(net, E(net)[.inc(imp\_eig3)])  
g4 <- subgraph.edges(net, E(net)[.inc(imp\_eig4)])  
plot(g1)



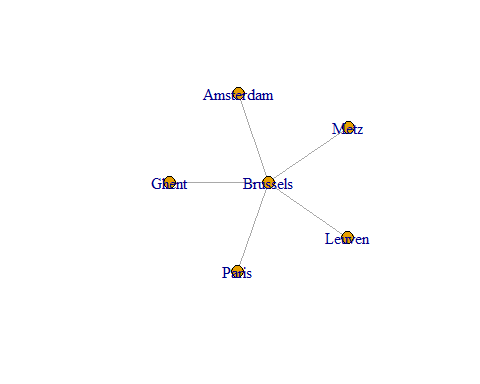
plot(g2)



plot(g3)



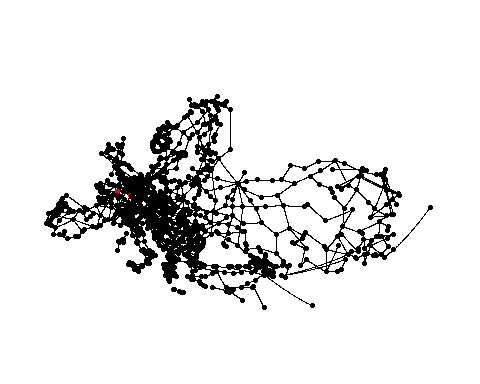
plot(g4)

 We can see from this plots that the node with higher eigenvector centrality is Paris, and the other whith highest eigen vector centrality is Metz which, beside having a lot of edges, is connected with some very important capitals: Luxembourg, Geneva and Brussels.

ggraph(net, x = lon, y = lat) +  
 geom\_edge\_link() +  
 geom\_node\_point(color = ifelse(Name == "Paris" | Name == "Metz" | Name == "Brussels" , "red", "black")) +  
 theme\_graph() +  
 coord\_map(projection = "albers", lat0 = 40, lat1 = 65) +  
 theme(plot.margin = margin(0, 0, 0, 0, "cm"))

## Warning: Removed 9 rows containing non-finite values (`stat\_edge\_link()`).

## Warning: Removed 2 rows containing missing values (`geom\_point()`).



COMMUNITY DETECTION he Louvain clustering algorithm is an iterative and hierarchical approach that aims to optimize the modularity of the network, which measures the strength of the community structure.

The algorithm starts by assigning each node in the network to its own individual community. Then, it iteratively merges communities in a way that maximizes the increase in modularity. This process involves two phases: the local optimization phase and the global optimization phase.

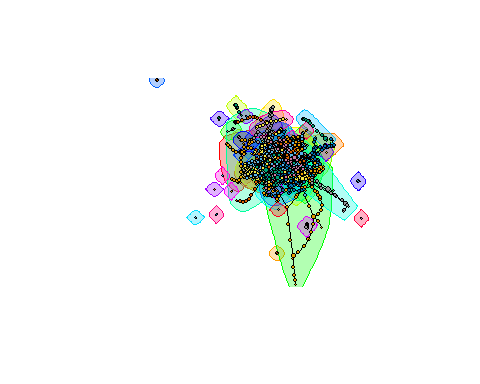
In the local optimization phase, the algorithm iterates over each node and evaluates the gain in modularity that would result from moving the node to its neighboring communities. It selects the community that yields the highest modularity gain and moves the node accordingly.

In the global optimization phase, the algorithm treats each community from the previous phase as a single node and constructs a new network where the nodes are the communities. The edges between these new nodes represent the total weights of the edges between the original communities.

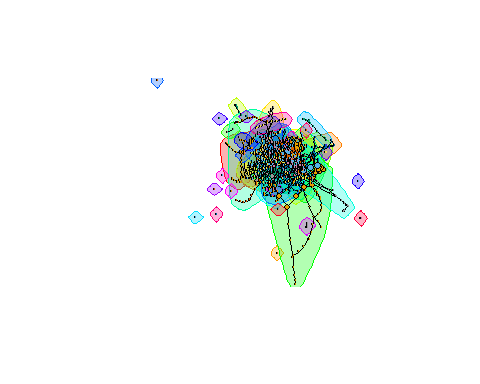
lc <- cluster\_louvain(net)

Louvain clustering with degree

set.seed(1001)  
# Define and assign a color palette  
pal <- c("red", "blue", "green", "yellow") # Customize the colors as needed  
  
# Plot with the updated code  
plot(lc, net, edge.color = 'black', vertex.label = '',  
 vertex.color = pal[as.numeric(as.factor(vertex\_attr(net, "city")))],  
 vertex.size = sqrt(d) \* 2.5, edge.width = sqrt(E(net)$weight / 800),  
 layout = layout.fruchterman.reingold, xlim = c(-0.7, 0.7), ylim = c(-0.9, 0.9))

 Louvain clustering with betweenness

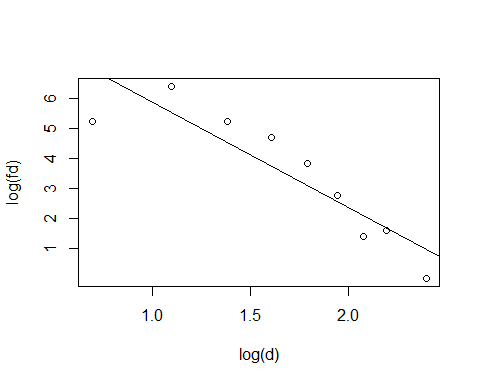
set.seed(1001)  
pal <- c("red", "blue", "green", "yellow") # Customize the colors as needed  
  
plot(lc, net, edge.color = 'black',vertex.label='',  
 vertex.color=pal[as.numeric(as.factor(vertex\_attr(net, "city")))],  
 vertex.size = sqrt(b)/50, edge.width=sqrt(E(net)$weight/800),  
 layout = layout.fruchterman.reingold, xlim = c(-0.5, 0.5), ylim = c(-0.9, 0.9))



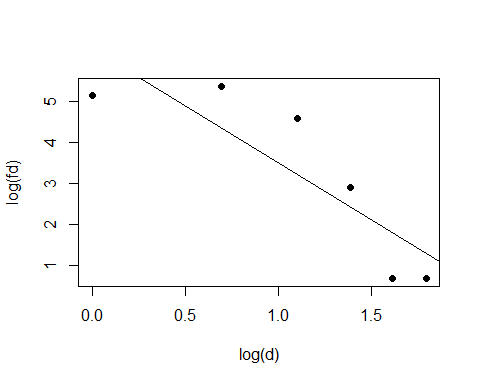
SAMPLING

As said before, we use the linear model because it seems to be more suitable in order to compare the sampling distribution with the original one, rather than the polynomial model which was better for the original, but is bad for the fittibility of the sample.

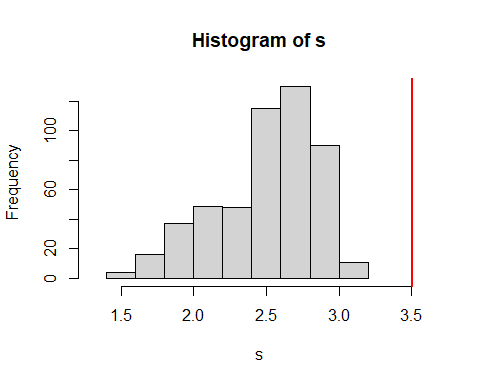
n <- 500  
gs <- induced\_subgraph(net, sample(V(net), n))  
  
dd0 <- degree\_dist(net)  
m0 <- lm(log(fd) ~ log(d), data = dd0)  
with(dd0, plot(log(d), log(fd))); abline(a=m0$coef[1], b = m0$coef[2])



dd1 <- degree\_dist(gs)  
m1 <- lm(log(fd) ~ log(d), data = dd1)  
with(dd1, plot(log(d), log(fd), pch = 19)); abline(a=m1$coef[1], b = m1$coef[2])

 Finally, let us study the variability in the sampling process. Sample ns = 500 induced subgraphs and plot the distribution of exponent estimates.

ns <- 500  
s <- replicate(ns, {  
 dd <- degree\_dist(induced\_subgraph(net, sample(V(net), n)))  
 -lm(log(fd) ~ log(d), data = dd)$coef[2]  
})  
alpha\_hat <- -m0$coef[2]  
hist(s, xlim = c(1.3, 3.6))  
abline(v = alpha\_hat, col = "red", lwd = 2)

 It seems to be quite different.