Social Network Analysis Home Assignment 1

{Mirzaaghayev Ramil}

due date - 08.05.2017 23:59

Table of Contents

Power law. Descriptive network analysis

Please send your reports to hse.ntwks@gmail.com with the subject of of the following structure: *[MAGOLEGO SNA 2017] {Mirzaaghayev} {Ramil} HA{1}*

Late submission policy: -1 point per day

Use this file as a template for your report.
Support your computations with figures and comments. Send ONLY .Rmd versions of your report.

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

Problem 1

Recall from the lecture that probability density function (PDF) for power law distributed variable is:

$$p(x) = Cx^{-\alpha}$$

Take logarithm of both sides:

$$\log p(x) = \log C - \alpha \log x$$

Now you can use standard R functions like `lm()` to calculate α coefficient via linear regression. However you might find that it is a bad idea.

Alternatively, you can compute cumulative density function (CDF)

$$f(x) = Pr(x < X)$$

of power law distribution. Good things about CDF of the power law are:

- It still has form of power law
- On log-log plot it looks more like a line

1. Derive the formula for CDF function of power law $F(x)$. It means you should calculate cumulative distribution function by integration of PDF function.

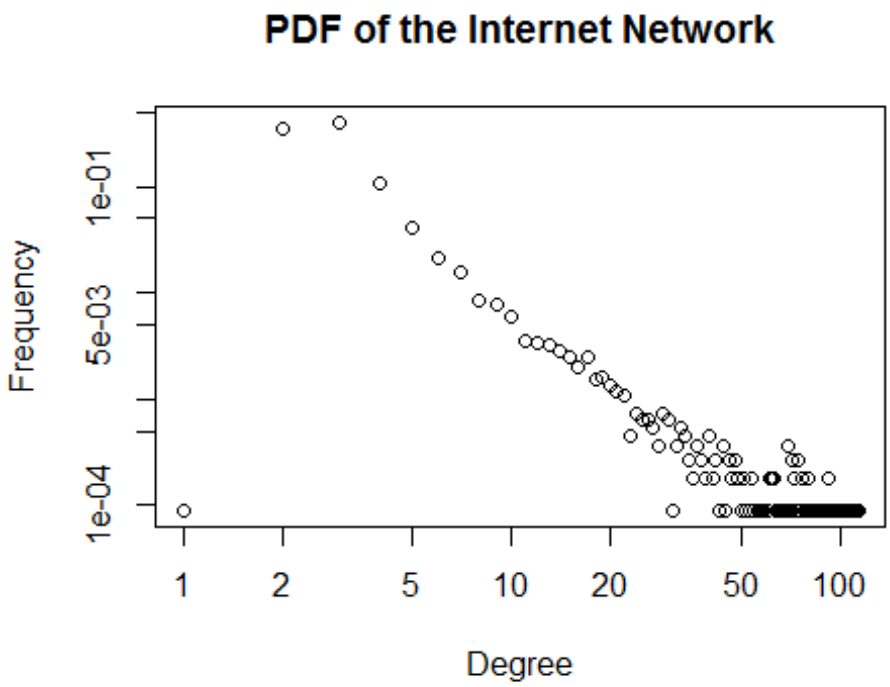
$$F(+\infty) = 1 \xrightarrow{\alpha>0} \int_{x_{min}}^{+\infty} p(x)dx = C \int_{x_{min}}^{+\infty} x^{-\alpha}dx = \frac{C}{1-\alpha}[(+\infty)^{1-\alpha} - {x_{min}}^{1-\alpha}] = \frac{C}{\alpha-1}{x_{min}}^{1-\alpha}$$

$$\frac{C}{\alpha-1}{x_{min}}^{1-\alpha} = 1 \xrightarrow{yields} C = (\alpha-1){x_{min}}^{\alpha-1}$$

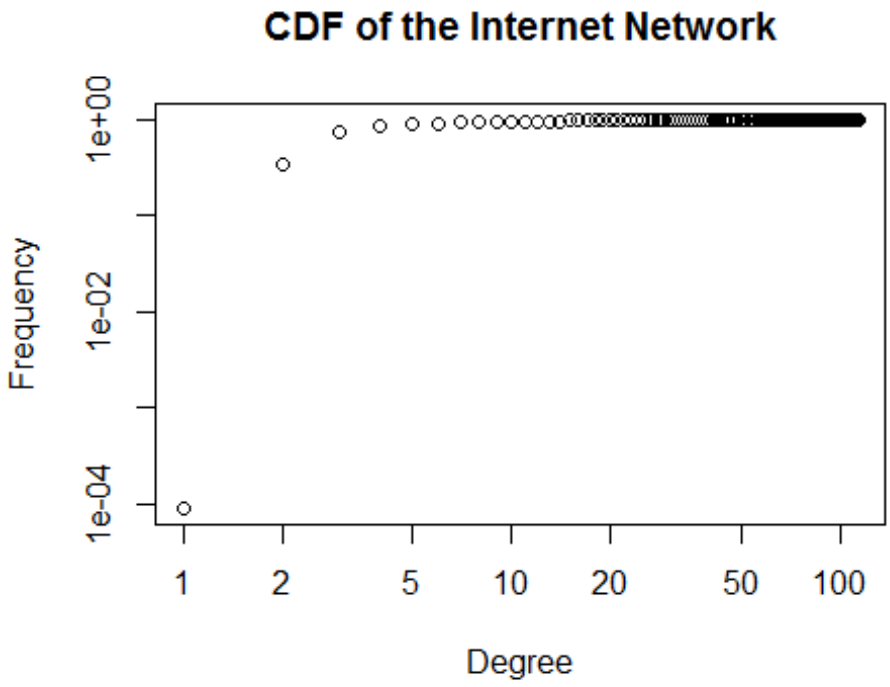
$$F(x_{min} < x) = \int_{x_{min}}^{x} p(x)dx = \frac{(\alpha-1){x_{min}}^{\alpha-1}}{1-\alpha}[x^{1-\alpha} - {x_{min}}^{1-\alpha}] = 1 - \left(\frac{x_{min}}{x}\right)^{\alpha-1}$$

2. Download Internet Network and plot PDF and CDF of the degree distribution in log-log scale

```
g = read.graph("Internet_AS.dat", directed = FALSE)

g.pdf = degree_distribution(g, cumulative = FALSE)

g.pdf = g.pdf[which(g.pdf>0)]

g.cdf = cumsum(g.pdf)


plot(g.pdf, log="xy", main="PDF of the Internet Network", xlab="Degree",
ylab="Frequency")
```

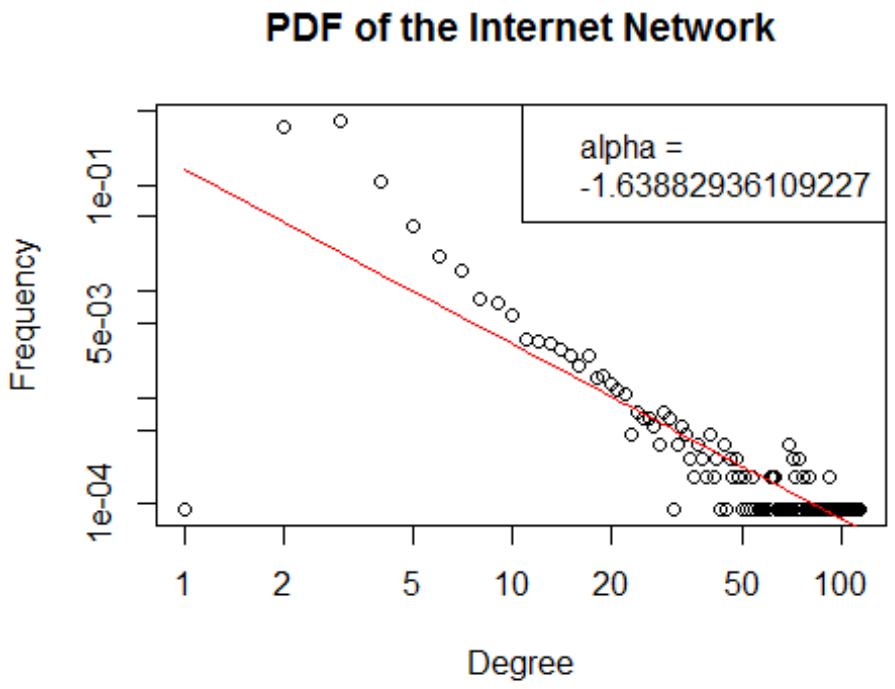



```
plot(g.cdf, log="xy", main="CDF of the Internet Network", xlab="Degree",
ylab="Frequency")
```

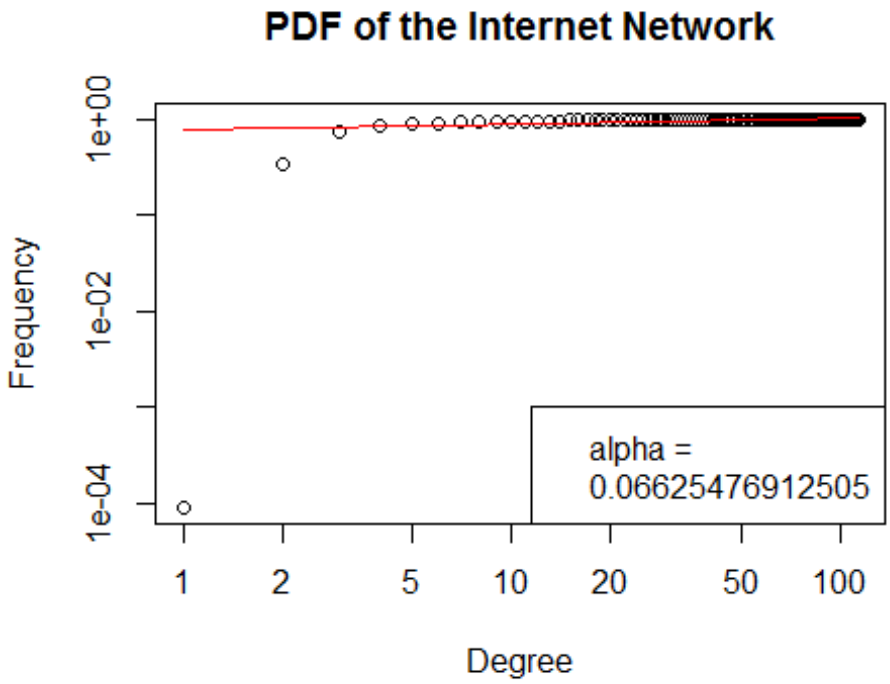


3. Fit linear regression model to PDF and CDF to estimate α. Plot fitted models along with data

```
# Fit PDF
x = as.numeric(1:max(degree(g)))[which(g.pdf>0)]
y = g.pdf[x]

g.fit = lm(formula=log(y)~log(x))
fit.coefficients = coef(g.fit)
fit.func = exp(fit.coefficients[[1]]+fit.coefficients[[2]]*log(x))
plot(g.pdf, log="xy", main="PDF of the Internet Network", xlab="Degree",
ylab="Frequency")
lines(fit.func, col="red")
legend("topright", c("alpha = ", fit.coefficients[[2]]))
```




```
# Fit CDF
x = as.numeric(1:max(degree(g)))[which(g.pdf>0)]+1
y = g.cdf[x]

g.fit = lm(formula=log(y)~log(x))
fit.coefficients = coef(g.fit)
fit.func = exp(fit.coefficients[[1]]+fit.coefficients[[2]]*log(x))
plot(g.cdf, log="xy", main="PDF of the Internet Network", xlab="Degree",
ylab="Frequency")
lines(fit.func, col="red")
legend("bottomright", c("alpha = ", fit.coefficients[[2]]))
```

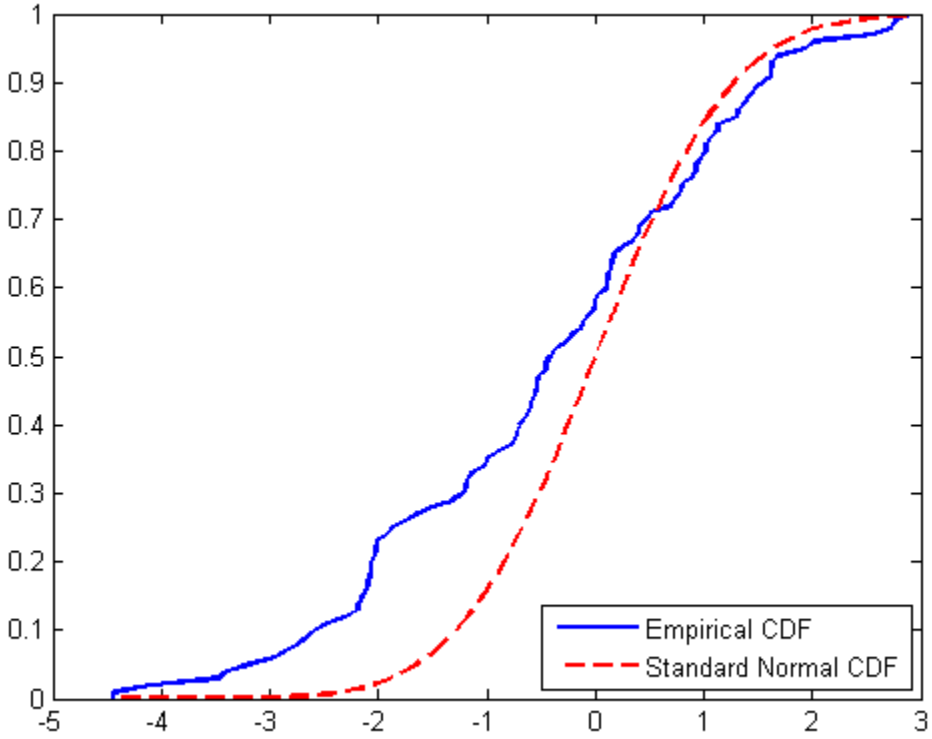


Problem 2

Kolmogorov-Smirnov test describes how similar are two distributions. In our case, when we have fitted model and original data, we can calculate their CDFs and Kolmogorov-Smirnov test shows us how well model approximates original data. In other words, it shows us the goodness-of-fit of our model.

$$D = \max_x \| f(x|\alpha, x_{min}) - f_{emp}(x) \|,$$

where $f(x|\alpha, x_{min})$ and $f_{emp}(x)$ are theoretical and empirical CDFs respectively.



To estimate x_{min} of the fitted power-law model we can use KS test:

- Pick some x_{min} value
- Fit power-law distribution to data (that is estimation of α) – now we have $f(x|\alpha, x_{min})$
- Perform KS test – compute D statistic
- Finnaly, choose x^*_{min} that provides minimal value of D statistic among all KS tests run above.

In R all this stuff can be done in one line of code.

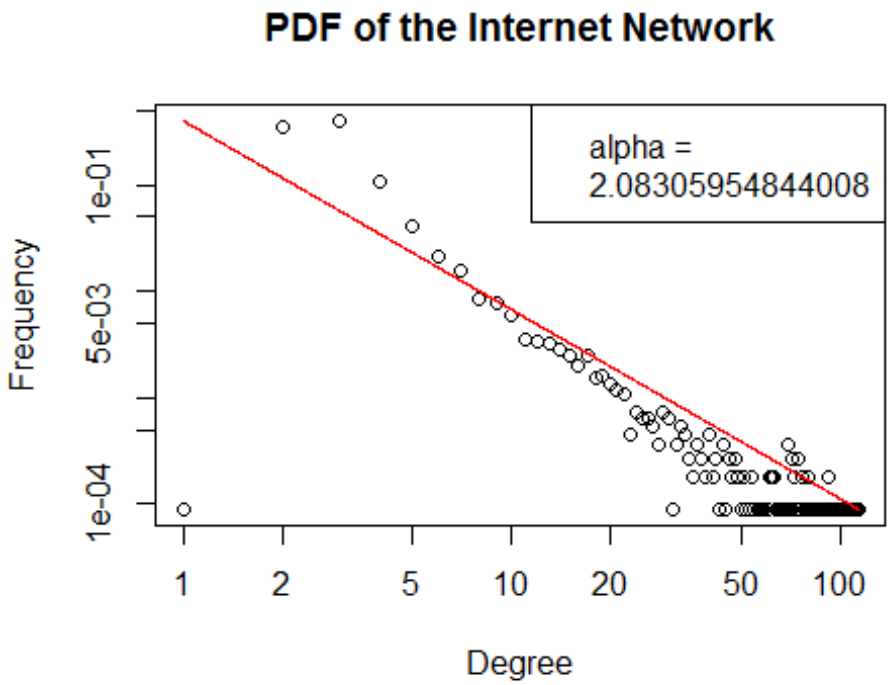
Again, use Internet Network
Properly load it into R and do following tasks:

1. Using `power.law.fit` find `xmin` value and corresponding `alpha`

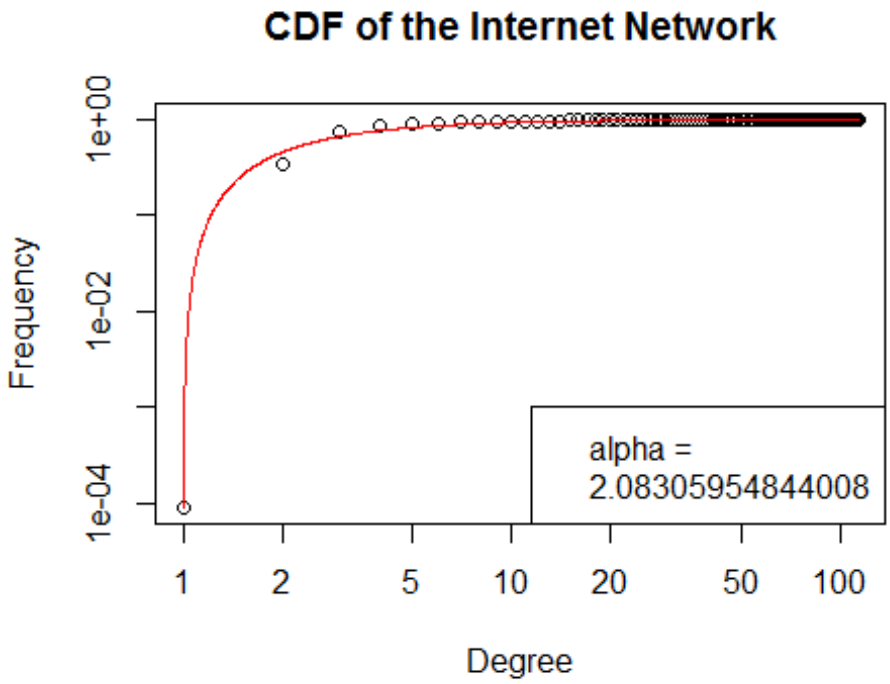
```
## Put your code here
g.fit = power.law.fit(degree(g), implementation = c("plfit"))
x_min = g.fit$xmin
alpha = g.fit$alpha

alpha

## [1] 2.08306
```

2. Put fitted model along with empirical PDF (CDF)

```
## PDF
x = as.numeric(1:max(degree(g)))
func_PDF = function(x) return((alpha-1)* x_min^(alpha-1) * x^(-alpha))
plot(g.pdf, log="xy", main="PDF of the Internet Network", xlab="Degree",
ylab="Frequency")
par(new=TRUE)
curve(func_PDF, from=x_min, to = max(x), n=length(x),log="xy", col="red",
add=FALSE, xlab = "", ylab = "", axes=FALSE)
legend("topright", c("alpha = ", alpha))
```




```
## CDF
x = as.numeric(1:max(degree(g)))+1
func_CDF = function(x) return(1-(x_min^(alpha-1))*x^(1-alpha))
plot(g.cdf, log="xy", main="CDF of the Internet Network", xlab="Degree",
ylab="Frequency")
par(new=TRUE)
curve(func_CDF, from=x_min, to = max(x), n=length(x),log="xy", col="red",
add=FALSE, xlab = "", ylab = "", axes=FALSE)

## Warning in xy.coords(x, y, xlabel, ylabel, log): 1 y value <= 0 omitted
## from logarithmic plot

legend("bottomright", c("alpha = ", alpha))
```



Problem 3.

For Wikipedia vote network (clear up comments in the begging of the file) derive the following characteristics:

1. The number of vertices and edges
2. The number of loops (edges that start and end at the same vertex)
3. The number of symmetrical edges
4. Degree distribution (without considering the direction)
5. The number of nodes with a degree greater than 1 and with a degree greater than 15
6. Find strongly connected components and their sizes.
7. Take subgraph of the original graph, which consists of the first 80 vertices and set color into red for those nodes in which the number of incoming edges is greater than the number of outgoing edges.Otherwise, set color in blue. For nodes with the same number of incoming and outgoing edges set color into green. Besides that, increase the size of vertices with a maximum value of transitivity (for example, you may set size into 10 for these nodes and 1 for others).

8.Take subgraph from the previous task and find maximal connected component. For this component highlight any way that corresponds to the diameter of the subgraph. How many such paths are in this graph?

8. Make average neighbor degree vs node degree scatter plot (one point on the plot per node) and aggregated plot, averaging over all nodes with the same degree (aggregated average vs degree, one value per degree). Explain your observations.

9. Make local clustering coefficient vs node degree scatter plot (one point on the plot per node) and aggregated, averaging over all nodes with the same degree (aggregated average vs degree, one value per degree). Explain your observations.

1. The number of vertices and edges.

```
## Put your code here
z = read_graph("Wiki-Vote.txt", directed = TRUE)
e = ecount(z)
v = vcount(z)
paste("Number of vertices", v)

## [1] "Number of vertices 8298"

paste("Number of edges", e)

## [1] "Number of edges 103689"
```

2. The number of loops (edges that start and end at the same vertex)

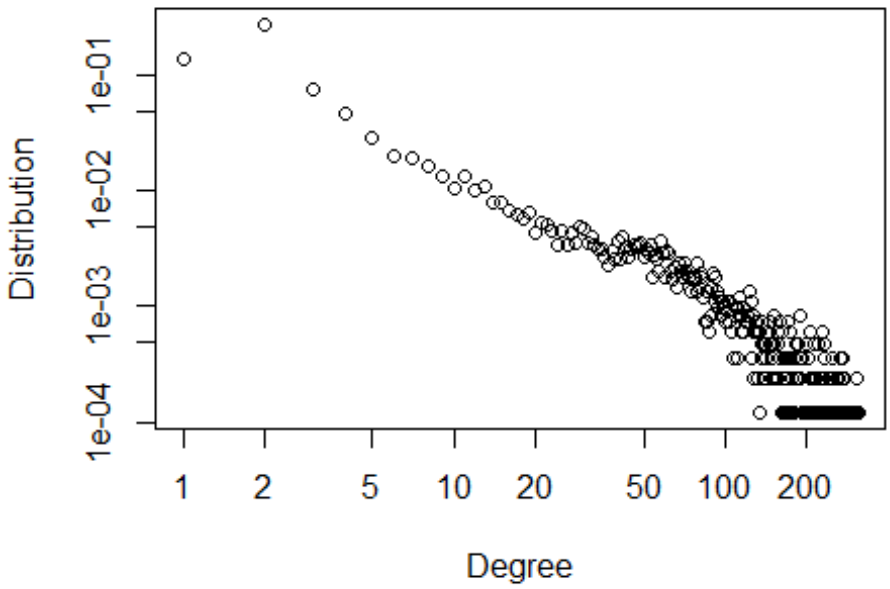
```
## Put your code here
paste("Number of loops in the given graph is", sum(which_loop(z)))

## [1] "Number of loops in the given graph is 0"
```

3. The number of symmetrical edges

```
## Put your code here
symm_ed = reciprocity(z)*e/2
paste("Number of symmetrical edges is",symm_ed)

## [1] "Number of symmetrical edges is 2927"
```

4. Degree distribution

```
## Put your code here
deg_dist = degree.distribution(z,cumulative = FALSE)
deg_dist = deg_dist[which(deg_dist>0)]
plot(deg_dist, log = "xy", xlab="Degree", ylab="Distribution")
```



5. The number of nodes with a degree greater than 1 and with a degree greater than 15

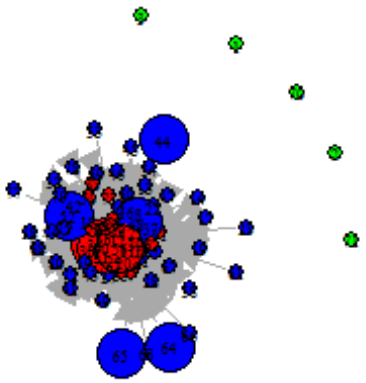
```
## Put your code here
paste("Number of nodes with a degree greater than 1 equals to",
sum(degree(z)>1))

## [1] "Number of nodes with a degree greater than 1 equals to 4800"

paste("Number of nodes with a degree greater than 15 equals to",
sum(degree(z)>15))

## [1] "Number of nodes with a degree greater than 15 equals to 2389"
```

6. Find strongly connected components and thier sizes.

```
## Put your code here
c = components(z, mode = "strong")

size = c$csize
components = c$membership
```

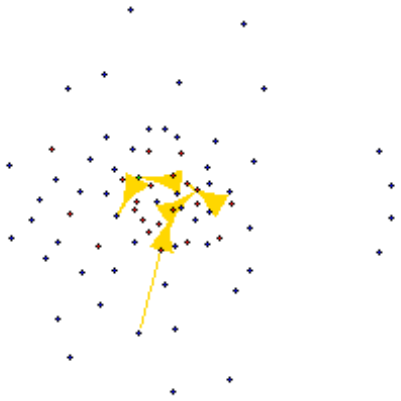
7. Take subgraph of the original graph, which consists of the first 80 vertices and set color into red for those nodes in which the number of incoming edges is greater than the number of outgoing edges.Otherwise, set color in blue. For nodes with the same number of incoming and outgoing edges set color into green. Besides that, increase the size of vertices with a maximum value of transitivity (for example, you may set size into 10 for these nodes and 1 for others).

```
## Put your code here
sub.z = subgraph(z, V(z)[1:80])

## Warning in subgraph(z, V(z)[1:80]): At structural_properties.c:
## 1984 :igraph_subgraph is deprecated from igraph 0.6, use
## igraph_induced_subgraph instead

trs = na.omit(transitivity(sub.z, type ="local"))
max.trs = max(trs)



for (node in V(sub.z)){
  incoming = incident(sub.z, node, mode="in")
  outgoing = incident(sub.z, node, mode="out")
  if(length(incoming) > length(outgoing)){
    V(sub.z)[node]$color = "red"
  } else if(length(incoming) < length(outgoing)){
    V(sub.z)[node]$color = "blue"
  } else {
    V(sub.z)[node]$color = "green"
  }
}

V(sub.z)$label.color = "black"
V(sub.z)$label.cex = 0.5

sub.z.node_size = rep(10, times=vcount(sub.z))

for (node in V(sub.z)[which(!is.na(trs))]){
  if(trs[node] == max(trs)){
    sub.z.node_size[node] = 30
  }
}

plot(sub.z, vertex.size = setNames(sub.z.node_size, V(sub.z)))
```



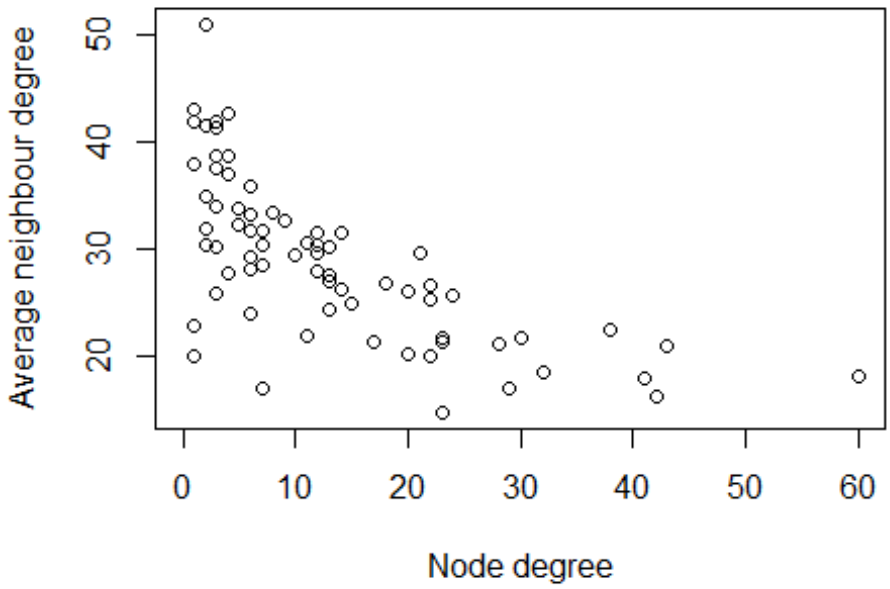
8.Take subgraph from the previous task and find maximal connected component. For this component highlight any way that corresponds to the diameter of the subgraph. HHow many such paths are in this graph?

```
## Put your code here
componenti = components(sub.z)
V(sub.z)$membership = componenti$membership
subsub = subgraph(sub.z,
which(componenti$membership==which.max(componenti$csize)))

## Warning in subgraph(sub.z, which(componenti$membership ==
## which.max(componenti$csize))): At structural_properties.c:
## 1984 :igraph_subgraph is deprecated from igraph 0.6, use
## igraph_induced_subgraph instead

diam = get.diameter(subsub)
# highlight diameter
E(subsub, path = diam)$color = "Gold"
plot(subsub, vertex.size = 3,vertex.label=NA )
```



9. Make average neighbour degree vs node degree scatter plot (one point on the plot per node) and aggregated plot, averaging over all nodes with the same degree (aggregated average vs degree, one value per degree). Explain your observations.

```
## Put your code here
avg_neigh = knn(sub.z)

node_degree = degree(sub.z)

plot(node_degree, avg_neigh$knn, ylab = "Average neighbour degree",
xlab="Node degree")
```




```
length(seq_along(unique(node_degree)))
```

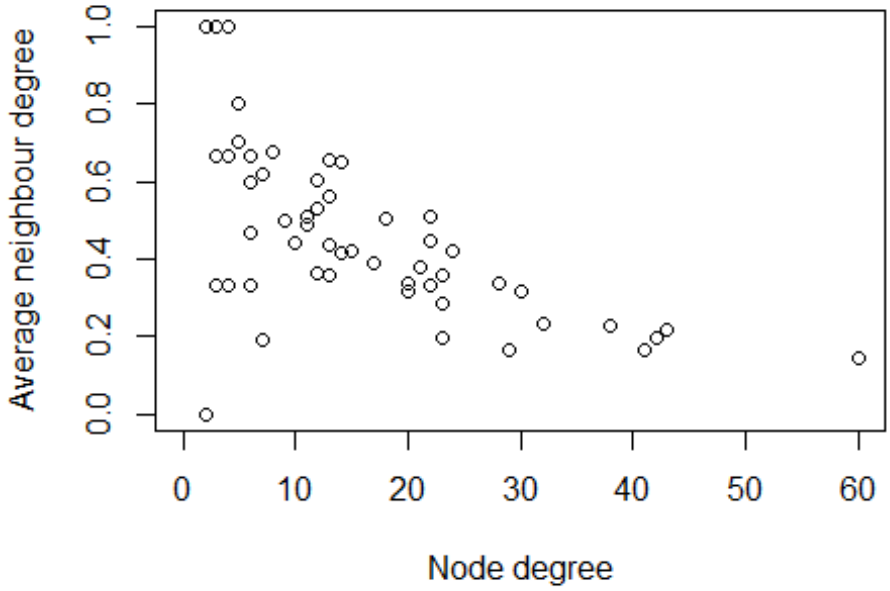
```
## [1] 32
```

```
length(unique(node_degree))
```

```
## [1] 32
```

10. Make local clustering coeff vs node degree scatter plot (one point on the plot per node) and aggregated, averaging over allnodes with the same degree (aggregated average vs degree, one value per degree). Explain your observations.

```
## Put your code here
trs= transitivity(sub.z, type = "local")
plot(node_degree, trs, ylab = "Average neighbour degree", xlab="Node degree")
```


Average neighbour degree
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
60
Node degree