Homework 1

Introduction to Network Analysis

Aleksey Shadrunov (Group 2, asshadrunov@edu.hse.ru)

2022-11-07

Contents

prepare packages and load data

install packages (uncomment if needed):

```
# on arch you might need to install gcc-fortran (necessary for compiling igraph)
# on arch you might need to install glpk (necessary for compiling igraph)

# install.packages("network")
# install.packages("sna")
# install.packages("igraph")
# remove.packages("igraph")
# # if there are some problems with the package above:
# remotes::install_github("igraph/rigraph@master")
# install.packages("intergraph")
# install.packages("knitr")
# install.packages("CINNA")
# install.packages("RColorBrewer")
```

load data `trade.Rdata`:

```
suppressPackageStartupMessages(library(network))
suppressPackageStartupMessages(library(sna))
suppressPackageStartupMessages(library(intergraph))
suppressPackageStartupMessages(library(knitr))
suppressPackageStartupMessages(library(CINNA))
suppressPackageStartupMessages(library(RColorBrewer))

load("trade.Rdata")
```

check what have been loaded:

```
ls()
```

```
## [1] "crude"       "diplomacy"   "food"        "manufacture" "minerals"
## [6] "trade.all"   "trade.att"
```

first examination of data

for further analysis I've selected network **"manufacture"**. let's convert it to matrix and check its dimensions.

```
manufacture = as.matrix(manufacture)
dim(manufacture)
```

```
## [1] 24 24
```

in addition, we have loaded attributes of the vertices in `trade.att` object. its dimensions are:

```
dim(trade.att)
```

```
## [1] 24  4
```

we may output several cells of each matrix to check:

```
manufacture[1:6,1:6]
```

```
##                ALGERIA ARGENTINA BRAZIL CHINA CZECHOSLOVAKIA ECUADOR
## ALGERIA              0         0      0     1               1       0
## ARGENTINA            1         0      1     1               0       1
## BRAZIL               1         1      0     1               1       1
## CHINA                1         1      1     0               1       0
## CZECHOSLOVAKIA       1         1      1     1               0       1
## ECUADOR              0         0      1     0               0       0
```

```
head(trade.att)
```

```
##                POP_GROWTH GNP SCHOOLS ENERGY
## ALGERIA               3.3 3.0      33    814
## ARGENTINA             1.6 0.3      56   2161
## BRAZIL                2.1 5.3      32   1101
## CHINA                 1.5  NA      43    618
## CZECHOSLOVAKIA        0.7  NA      44   6847
## ECUADOR               3.4 4.7      40    692
```

```
all(colnames(manufacture) == colnames(trade.all))
```

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

as we see, both matrices have the same column names, and `trade.att` contains `POP_GROWTH`, `GNP`, `SCHOOLS` and `ENERGY` columns. so, we can safely apply attributes to the network.

basic network statistics

first step is to create a network:

```
manufacture.net <- as.network(
                        x = manufacture,
                        directed = FALSE,
                        matrix.type = "adjacency"
                    )
```

here `directed` parameter equals `FALSE`, however it is not correct, since the network is directed. to prove that, we should compare adjacency matrix with its transposition. in our case, these matrices differ, so network is directed. but for simplicity we ignore that.

```
all(t(manufacture) == manufacture)  # is network undirected?
```

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

let's look at the network parameters:

```
manufacture.net
```

```
##  Network attributes:
##   vertices = 24
##   directed = FALSE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 195
##     missing edges= 0
##     non-missing edges= 195
##
##  Vertex attribute names:
##     vertex.names
##
## No edge attributes
```

as we observe, the network has 24 vertices and 195 edges, no loops and duplicating edges.
now let's count other statistics.

dyads

```
# number of dyads:
network.dyadcount(manufacture.net)
```

```
## [1] 276
```

```
# types of dyads
dyad.census(manufacture.net)
```

```
##      Mut Asym Null
## [1,] 195    0   81
```

number of dyads is just the number of all possible pairs of the vertices (potential conntection) and could be in simple case calculated as

$$\frac{n \cdot (n-1)}{2}$$

however, `census` is more informative: there are 195 mutual dyads and 81 null ones, so 81 pairs (about 30%) of vertices are not connected directly.

density

```
# number of ties we already know from above, but this is another way to get it
network.edgecount(manufacture.net)
```

```
## [1] 195
```

```
# density of the network
network.density(manufacture.net)
```

```
## [1] 0.7065217
```

network density describes the portion of the potential connections in a network that are actual connections. our network is highly dense, which means that there are many connections in the network and international trade of manufactured goods is very active.

triads

```
# types of triads
triad.census(manufacture.net)
```

```
##      003 012 102 021D 021U 021C 111D 111U 030T 030C 201 120D 120U 120C 210 300
## [1,] 174   0 275    0    0    0    0    0    0    0 710    0    0    0   0 865
```

in our undirected network we have only few types of triads, specifically 003, 102, 201, 300. as last two types are in total 78% of all triads, we may conclude again that connectivity is high.

case of directed network

now we should repeat the procedure with directed network, which our network in fact is.

```
manufacture.directed <- as.network(manufacture, directed = TRUE)
manufacture.directed
```

```
##  Network attributes:
##   vertices = 24
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 310
##     missing edges= 0
##     non-missing edges= 310
##
```

```
##  Vertex attribute names:
##     vertex.names
##
## No edge attributes
```

```
# number of dyads:
network.dyadcount(manufacture.directed)
```

```
## [1] 552
```

```
# types of dyads
dyad.census(manufacture.directed)
```

```
##      Mut Asym Null
## [1,] 115   80   81
```

```
# density of the network
network.density(manufacture.directed)
```

```
## [1] 0.5615942
```

```
# types of triads
triad.census(manufacture.directed)
```

```
##      003 012 102 021D 021U 021C 111D 111U 030T 030C 201 120D 120U 120C 210 300
## [1,] 174 164 111  171    7   19   17  381   18    0 115   12  230   20 226 359
```

in case of directed network amount of possible connections is $n(n-1)$. there are 310 total edges, so density is around 0.56, which is also a significant value.
number of mutual connections is higher than asymmetrical, so more countries exchange goods rather than trade in *one direction*.
considering triads, there are only 18 transitive ones from 2024, and most frequent type is still 300 (all three countries are mutually connected). also there are many 021D triads, which represent the distribution from one country to two others.

transitivity

the transitivity is an important property of a network, which describes whether $a \to b \to c \Rightarrow a \to c$. usually in social networks this parameter is high, and in other networks not so. for random network transitivity coefficient is $m/\frac{n(n-1)}{2}$

```
gtrans(manufacture.net, measure="weak") # transitivity
```

```
## [1] 0.785174
```

our network's transitivity coefficient is 0.78, which is a high value, as it should be in social networks. for comparison, in a random network with $m = 195$ edges $T = \frac{195}{12 \cdot 23} = 0.7$, so expectations are quite correct.

reciprocity

the reciprocity of a directed network reflects the proportion of edges that are symmetrical. that is, the proportion of outgoing edges that also have an incoming edge. it is commonly used to determine how inter-connected directed networks are.

now we may calculate different types of reciprocity.

```
grecip(manufacture.directed, measure = "dyadic")
```

```
##       Mut
## 0.7101449
```

```
grecip(manufacture.directed, measure = "dyadic.nonnull")
```

```
##       Mut
## 0.5897436
```

```
grecip(manufacture.directed, measure = "edgewise")
```

```
##       Mut
## 0.7419355
```

```
grecip(manufacture.directed, measure = "edgewise.lrr")
```

```
##       Mut
## 0.2784828
```

```
grecip(manufacture.directed, measure = "correlation")
```

```
## [1] 0.411357
```

the first one calculates number of mutual diads divided by total number of null or asymmetrical diads. the second one is the same without asymmetrical diads. edgewise reciprocity is the proportion of ties that are mutual. last two are not comprehensive.
to interpret, number of mutual diads is 0.7 of asymmetrical + null and 0.6 of mutual + null. 0.74 of edges are mutual.
personally, I may see informative the following equation: proportion of mutual diads to all diads. the value is:

```
dyad.census(manufacture.directed)[1,"Mut"] / sum(dyad.census(manufacture.directed))
```

```
##       Mut
## 0.4166667
```

so, only 0.4 of all potential connections are mutual, which is, in comparison, less than a half.

geodesic distances

the geodesic distance is a shortest path between two nodes in a graph, i. e. a path with the minimum number of edges. obviously, it is defined for each pair of vertices, so the output is a matrix $n \times n$.

```
gd = geodist(manufacture.net)
sort(unique(c(gd$gdist)))
```

```
## [1] 0 1 2
```

```
sort(unique(c(gd$counts)))
```

```
##  [1]  1  5  6  7  8  9 10 11 12 13 14 15
```

so, in our network the longest geodesic distance is 2, and the shortest is 1 (0 is between the same vertices). it means that any point could be reached in not more than 2 steps.
numbers of geodesic distances varies from 1 to 15, which is not so very interesting and again denotes the network is well-connected.

we can calculate an average distance in each column (of all geodesic distances to certain vertex) — also between 1 and 2.

```
colMeans(gd$gdist)
```

```
##  [1] 1.3750000 1.2916667 1.0416667 1.0416667 1.0416667 1.5416667 1.3750000
##  [8] 1.4583333 1.0416667 1.5416667 1.2500000 1.4166667 0.9583333 1.5416667
## [15] 1.6666667 1.2500000 1.2500000 0.9583333 0.9583333 1.4166667 1.2500000
## [22] 1.0000000 0.9583333 1.1250000
```

```
mean(gd$gdist)
```

```
## [1] 1.239583
```

in case of directed network the result is slightly different:

```
gd = geodist(manufacture.directed)
sort(unique(c(gd$gdist)))
```

```
## [1]   0   1   2   3 Inf
```

```
sort(unique(c(gd$counts)))
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11
```

```
gd = geodist(manufacture.directed, inf.replace = NA)
colMeans(gd$gdist, na.rm = TRUE)
```

```
##  [1] 1.318182 1.500000 1.409091 1.227273 1.318182 1.500000 1.363636 1.454545
##  [9] 1.227273 1.500000 1.272727 1.454545 1.136364 1.521739 1.636364 1.272727
## [17] 1.272727 1.136364 1.227273 1.391304 1.227273 1.136364 1.090909 1.227273
```

```
mean(gd$gdist, na.rm = TRUE)
```

```
## [1] 1.326415
```

naturally, average distance and maximum distance are higher in case of directed network, because not every path is used in both directions.

let's perform some more evaluations of the geodesic distance with `igraph` package:

```
detach(package:sna)
suppressPackageStartupMessages(library(igraph))

manufacture.graph = graph_from_adjacency_matrix(manufacture, mode = "undirected")
manufacture.dgraph = graph_from_adjacency_matrix(manufacture, mode = "directed")

mean_distance(manufacture.graph, directed = FALSE, unconnected = TRUE)
```

```
## [1] 1.293478
```

```
mean_distance(manufacture.dgraph, directed = TRUE, unconnected = TRUE)
```

```
## [1] 1.389328
```

mean values are quite the same as above.

centrality

degree centrality

this type of centrality is basically the number of ties linked with certain node — which is the degree.

```
indegree = degree(manufacture.dgraph, mode="in")
outdegree = degree(manufacture.dgraph, mode="out")
total = degree(manufacture.dgraph, mode="total")  # This is also called Freeman's degree
```

betweenness, closeness, eigenvector

betweenness centrality measures the number of shortest paths going through a specific vertex:

```
between = betweenness(manufacture.dgraph)
```

closeness is the mean geodesic distance between a given node and all other nodes. in-closeness centrality is the average number of steps one would have to go through to get to a given node from all other reachable nodes, out-closeness centrality — vice versa.

```
inclose = closeness(manufacture.dgraph, mode="in")
outclose = closeness(manufacture.dgraph, mode="out")
```

eigenvector centrality gives greater weight to a node the more it is connected to other highly connected nodes. it is often interpreted as measuring the network importance of the node.

```
ecentV = evcent(manufacture.graph)
eigen = ecentV$vector
```

now put everything in one table.

```
node = network.vertex.names(manufacture.net)
table = data.frame(indegree, outdegree, total, between, inclose, outclose, eigen)
names(table) = c(
    "Indegree",
    "Outdegree",
    "Total",
    "Betweenness",
    "Incloseness",
    "Outcloseness",
    "Eigenvector"
    )
kable(table, digits=3, caption = "Manufacture Centralities")
```

Table 1: Manufacture Centralities

	Indegree	Outdegree	Total	Betweenness	Incloseness	Outcloseness	Eigenvector
ALGERIA	13	4	17	0.291	0.034	0.024	0.683
ARGENTINA	10	13	23	1.215	0.030	0.030	0.733
BRAZIL	11	21	32	11.793	0.032	0.040	0.951
CHINA	15	21	36	11.824	0.037	0.040	0.941
CZECHOSLOVAKIA	13	21	34	9.247	0.034	0.040	0.949
ECUADOR	9	2	11	0.000	0.030	0.023	0.487
EGYPT	12	9	21	0.778	0.033	0.027	0.673
ETHIOPIA	10	2	12	0.000	0.031	0.023	0.569
FINLAND	15	21	36	7.513	0.037	0.040	0.956
HONDURAS	9	1	10	0.000	0.030	0.022	0.498
INDONESIA	14	14	28	1.361	0.036	0.031	0.805
ISRAEL	10	11	21	1.076	0.031	0.029	0.607
JAPAN	17	23	40	17.967	0.040	0.043	1.000
LIBERIA	9	0	9	0.000	0.029	NaN	0.489
MADAGASCAR	6	1	7	0.000	0.028	0.022	0.336
NEW_ZEALAND	14	11	25	1.009	0.036	0.029	0.798
PAKISTAN	14	13	27	1.795	0.036	0.030	0.793
SPAIN	17	22	39	22.305	0.040	0.042	1.000
SWITZERLAND	15	23	38	10.276	0.037	0.043	1.000
SYRIA	12	0	12	0.000	0.031	NaN	0.640
THAILAND	15	14	29	2.564	0.037	0.031	0.799
UNITED_KINGDOM	17	22	39	42.034	0.040	0.042	0.961
UNITED_STATES	18	23	41	45.717	0.042	0.043	1.000
YUGOSLAVIA	15	18	33	8.234	0.037	0.036	0.903

correlating centrality measures

now we can observe how the centralities correlate with each other.

```
centralities = table[,c(1:7)]
centralities[is.na(centralities)] = 0
res = cor(centralities)
kable(res, digits=2, caption="Correlations of  Centralities")
```

Table 2: Correlations of Centralities

	Indegree	Outdegree	Total	Betweenness	Incloseness	Outcloseness	Eigenvector
Indegree	1.00	0.79	0.88	0.69	0.98	0.66	0.89
Outdegree	0.79	1.00	0.99	0.67	0.81	0.91	0.95
Total	0.88	0.99	1.00	0.71	0.90	0.88	0.97
Betweenness	0.69	0.67	0.71	1.00	0.74	0.60	0.65
Incloseness	0.98	0.81	0.90	0.74	1.00	0.74	0.87
Outcloseness	0.66	0.91	0.88	0.60	0.74	1.00	0.81
Eigenvector	0.89	0.95	0.97	0.65	0.87	0.81	1.00

we see, that all values are more or less correlated, for example, Indegree and Incloseness, Outdegree and Outcloseness, Total and Eigenvector. so, if a country trades actively, its centralities are high as well.

additional centralities

```
pr_cent = proper_centralities(manufacture.graph)
```

```
##  [1] "subgraph centrality scores"
##  [2] "Topological Coefficient"
##  [3] "Average Distance"
##  [4] "Barycenter Centrality"
##  [5] "BottleNeck Centrality"
##  [6] "Centroid value"
##  [7] "Closeness Centrality (Freeman)"
##  [8] "ClusterRank"
##  [9] "Decay Centrality"
## [10] "Degree Centrality"
## [11] "Diffusion Degree"
## [12] "DMNC - Density of Maximum Neighborhood Component"
## [13] "Eccentricity Centrality"
## [14] "Harary Centrality"
## [15] "eigenvector centralities"
## [16] "K-core Decomposition"
## [17] "Geodesic K-Path Centrality"
## [18] "Katz Centrality (Katz Status Index)"
## [19] "Kleinberg's authority centrality scores"
## [20] "Kleinberg's hub centrality scores"
## [21] "clustering coefficient"
## [22] "Lin Centrality"
## [23] "Lobby Index (Centrality)"
## [24] "Markov Centrality"
## [25] "Radiality Centrality"
## [26] "Shortest-Paths Betweenness Centrality"
## [27] "Current-Flow Closeness Centrality"
## [28] "Closeness centrality (Latora)"
## [29] "Communicability Betweenness Centrality"
## [30] "Community Centrality"
## [31] "Cross-Clique Connectivity"
## [32] "Entropy Centrality"
## [33] "EPC - Edge Percolated Component"
## [34] "Laplacian Centrality"
## [35] "Leverage Centrality"
## [36] "MNC - Maximum Neighborhood Component"
## [37] "Hubbell Index"
## [38] "Semi Local Centrality"
## [39] "Closeness Vitality"
## [40] "Residual Closeness Centrality"
## [41] "Stress Centrality"
## [42] "Load Centrality"
## [43] "Flow Betweenness Centrality"
## [44] "Information Centrality"
## [45] "Dangalchev Closeness Centrality"
## [46] "Group Centrality"
## [47] "Harmonic Centrality"
## [48] "Local Bridging Centrality"
## [49] "Wiener Index Centrality"
```

```
NewCent = calculate_centralities(manufacture.graph, include = c(pr_cent[2], pr_cent[24]))
NewCent = as.data.frame(NewCent)
names(NewCent) = c("Topological", "Markov")
kable(NewCent, digits=3, caption = "Additional Network Centralities")
```

Table 3: Additional Network Centralities

	Topological	Markov
ALGERIA	0.883	0.035
ARGENTINA	0.803	0.041
BRAZIL	0.729	0.059
CHINA	0.720	0.059
CZECHOSLOVAKIA	0.727	0.059
ECUADOR	0.928	0.024
EGYPT	0.870	0.035
ETHIOPIA	0.874	0.030
FINLAND	0.733	0.059
HONDURAS	0.957	0.024
INDONESIA	0.832	0.044
ISRAEL	0.844	0.032
JAPAN	0.694	0.065
LIBERIA	0.932	0.024
MADAGASCAR	0.978	0.016
NEW_ZEALAND	0.823	0.044
PAKISTAN	0.818	0.044
SPAIN	0.694	0.065
SWITZERLAND	0.694	0.065
SYRIA	0.902	0.032
THAILAND	0.826	0.044
UNITED_KINGDOM	0.698	0.062
UNITED_STATES	0.694	0.065
YUGOSLAVIA	0.773	0.053

```
leaders = data.frame(
                Topological = node[head(order(NewCent$Topological, decreasing = TRUE), 6)],
                Markov = node[head(order(NewCent$Markov, decreasing = TRUE), 6)]
              )
kable(leaders, caption = "Additional centralities leaders")
```

Table 4: Additional centralities leaders

Topological	Markov
MADAGASCAR	SPAIN
HONDURAS	JAPAN
LIBERIA	SWITZERLAND
ECUADOR	UNITED_STATES
SYRIA	UNITED_KINGDOM
ALGERIA	CZECHOSLOVAKIA

the topological coefficient describes if a node shares neighbours with other nodes. Markov centrality (random walk closeness centrality) is a measure of centrality, which describes the average speed with which

randomly walking processes reach a node from other nodes of the network.
surprisingly, the first centrality has shown the countries with zero betweenness centrality. so, these countries share neighbours with other. the second centrality picked up more or less popular countries.

influential network indices

now using the dataframe we may get the leaders in import and export — influential countries.

```
leaders = data.frame(
                Import = node[head(order(table$Indegree, decreasing = TRUE), 6)],
                Export = node[head(order(table$Outdegree, decreasing = TRUE), 6)],
                Total = node[head(order(table$Total, decreasing = TRUE), 6)],
                Betweenness = node[head(order(table$Betweenness, decreasing = TRUE), 6)]
              )
kable(leaders, caption = "Trade leaders")
```

Table 5: Trade leaders

Import	Export	Total	Betweenness
UNITED_STATES	JAPAN	UNITED_STATES	UNITED_STATES
JAPAN	SWITZERLAND	JAPAN	UNITED_KINGDOM
SPAIN	UNITED_STATES	SPAIN	SPAIN
UNITED_KINGDOM	SPAIN	UNITED_KINGDOM	JAPAN
CHINA	UNITED_KINGDOM	SWITZERLAND	CHINA
FINLAND	BRAZIL	CHINA	BRAZIL

what we might observe is that there are several countries that are leading in each category (the USA, the UK, Japan, Spain, China). these are the most developed countries with the rich international trading. they have a lot of connections, which is demonstrated by **Degree centrality** statistics. also, as it was stated previously, top countries as a rule have all centralities at a high level, and the correlation is could be seen in the table.

besides frequent countries, Finland imports from many countries and Brazil exports to many countries, as their **Indegree and Outdegree centralities** are high, respectively.

```
leaders = data.frame(
                Country = node[head(order(table$Eigenvector, decreasing = TRUE), 6)],
                Eigenvector = table[head(order(table$Eigenvector, decreasing = TRUE), 6), 7]
              )
kable(leaders, digits = 3, caption = "Eigenvector centrality leaders")
```

Table 6: Eigenvector centrality leaders

Country	Eigenvector
SPAIN	1.000
JAPAN	1.000
SWITZERLAND	1.000
UNITED_STATES	1.000
UNITED_KINGDOM	0.961
FINLAND	0.956

what concerns **Eigenvector centrality**, it measures the transitive influence of nodes and thus represents the network importance of the node. connections with high-scoring nodes contribute more to the score of a node than connections from low-scoring nodes.

as expected, Spain, Japan, UK and US have high value of **Eigenvector centrality**. in addition, in this case Switzerland has also very significant score. we cannot state that it has the highest value, as several countries actually have the same value (1.0 in this case). so, they all could be called transitive influential (they have more connections with top-scoring countries).

visualisation

now we are ready to visualise our network. to begin, load `network` package

```
detach(package:igraph)
suppressPackageStartupMessages(library(network))
```

we will use nice colour palettes to plot the network.

```
pastel2 = brewer.pal(8, "Pastel2")
accent = brewer.pal(5, "Accent")
```

at first, let's draw a `circle` plot:

```
par(mar=c(1,1,1,1))
plot(
        manufacture.net,
        mode = "circle",
        displaylabels = TRUE,
        label.cex = 0.6,
        vertex.col = accent[1],
        edge.col = pastel2[8],
        label.col = accent[5]
    )
```

this is not a very informative presentation, the only possible conclusion is that there are many connections in the network and some nodes have more ties than others.

now we can plot this network again using `fruchtermanreingold` mode. at this time we may alter the size of vertices according to its centrality (for example, total degree centrality).

```
plot(
        manufacture.net,
        mode = "fruchtermanreingold",
        displaylabels = TRUE,
        label.cex = 0.6,
        vertex.cex = 2 * total / 41,
        edge.lwd = 0.1,
        layout.par = list(niter = 2000),
        vertex.col = accent[2],
        edge.col = pastel2[8],
        label.col = accent[5]
    )
```

now let's make the colour indicate if a node has more outcoming than incoming edges.

```
inout = ifelse(indegree > outdegree, accent[3], pastel2[3])
plot(
        manufacture.directed,
        mode = "fruchtermanreingold",
        displaylabels = TRUE,
        label.cex = 0.6,
        vertex.cex = 2 * total / 41,
        edge.lwd = 0.1,
        layout.par = list(niter = 2000),
```

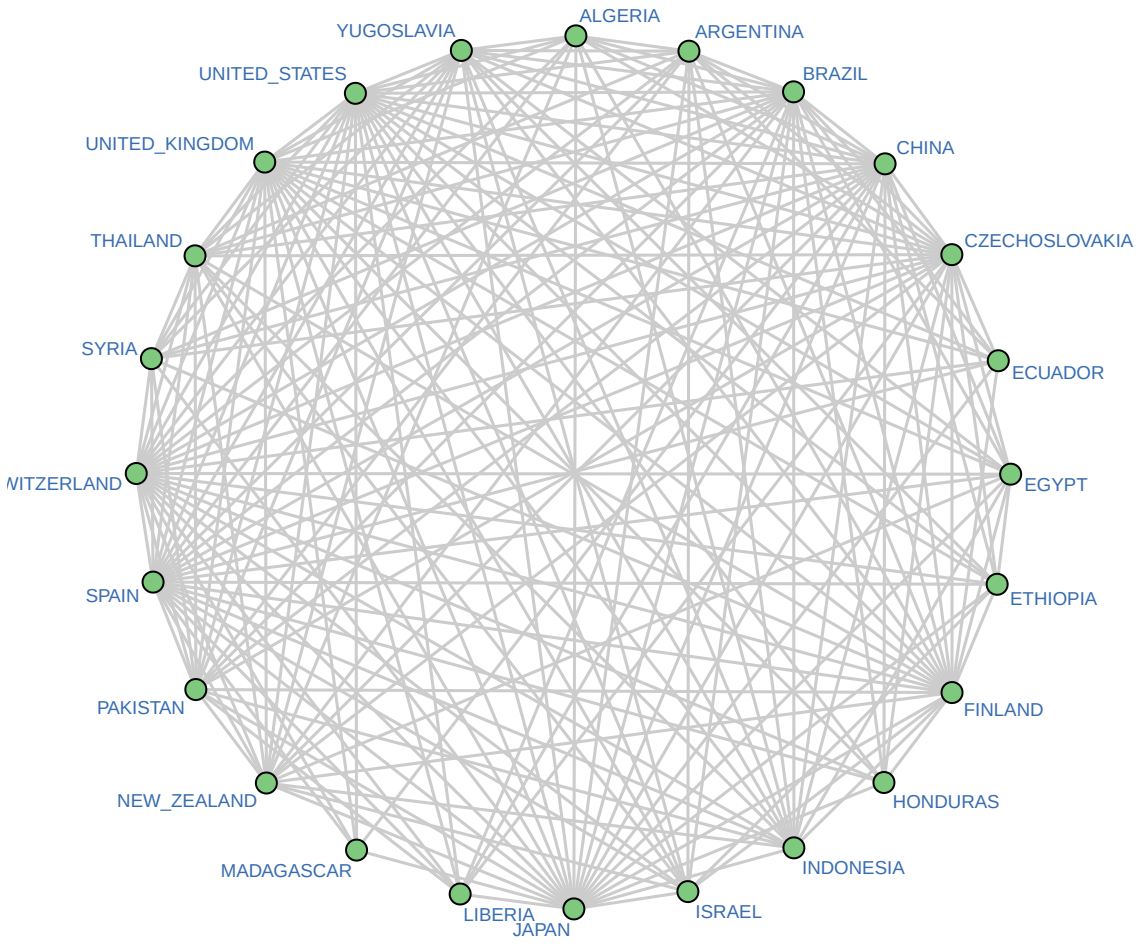

Figure 1: Circle plot

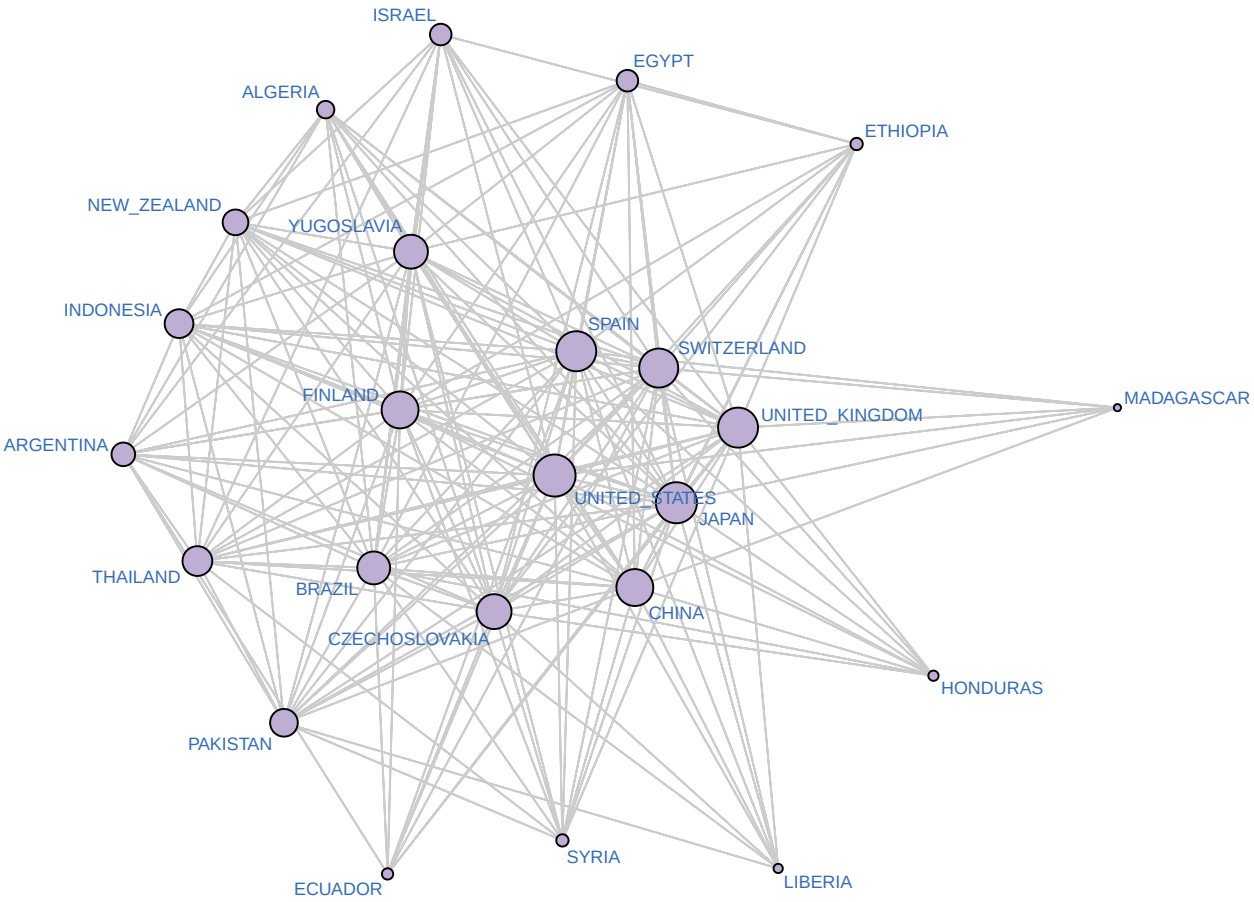

Figure 2: Size based on Total Degree centrality

```
        vertex.col = inout,
        edge.col = pastel2[8],
        label.col = accent[5]
    )
```

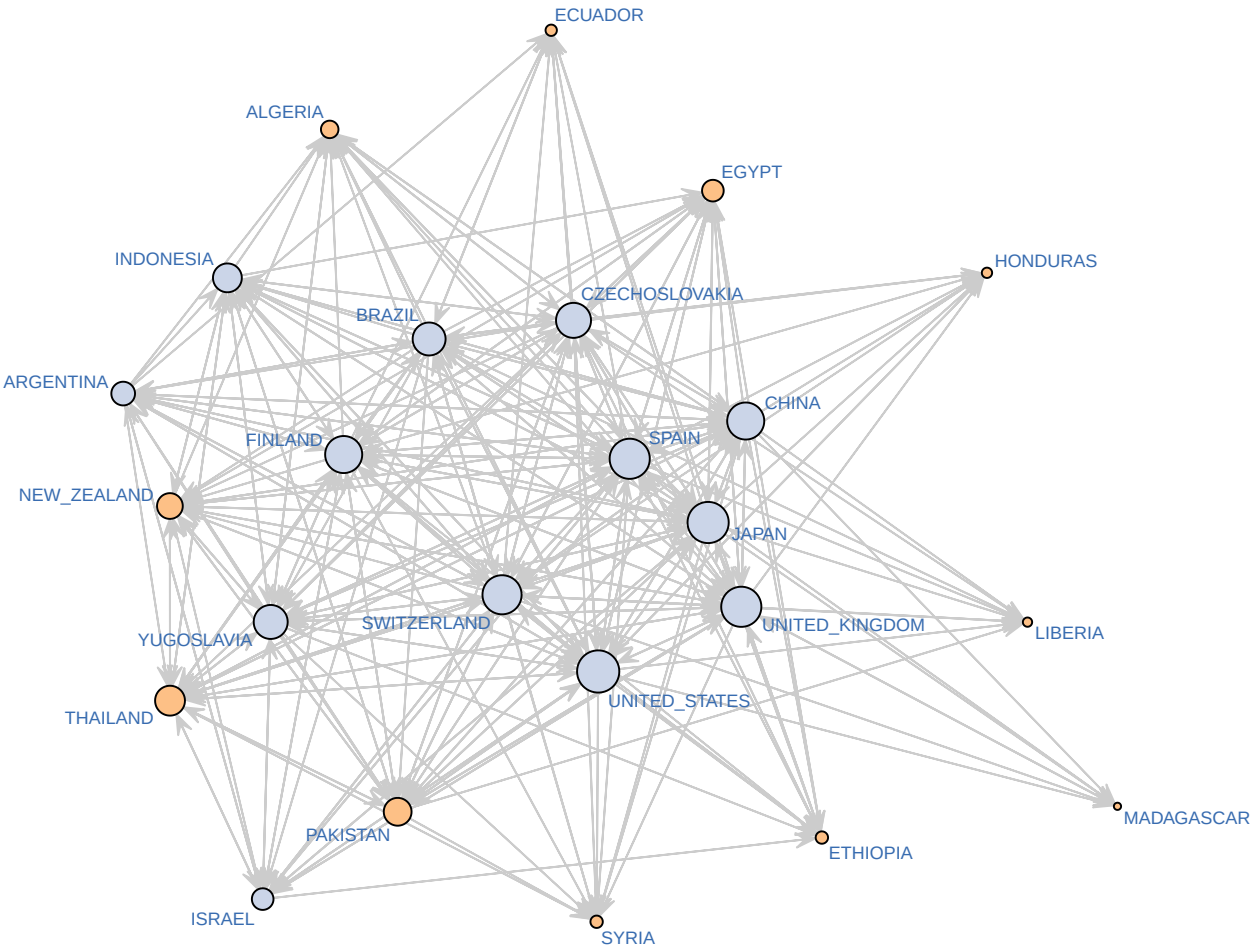

Figure 3: Orange if more incoming, blue if more outcoming edges

we observe that countries on periphery of the graph (such as Madagascar, Honduras, Syria) have mostly incoming edges, so they import from more countries than export to. this could probably be explained with the necessity to import a lot of different manufactured good from a wide range of international partners.

attributes

finally, we should utilise network attributes from the given data.

```
head(trade.att)
```

```
##                POP_GROWTH GNP SCHOOLS ENERGY
## ALGERIA               3.3 3.0      33    814
## ARGENTINA             1.6 0.3      56   2161
## BRAZIL                2.1 5.3      32   1101
## CHINA                 1.5  NA      43    618
```

```
## CZECHOSLOVAKIA         0.7  NA      44   6847
## ECUADOR                3.4 4.7      40    692
```

```
summary(trade.att)
```

```
##    POP_GROWTH          GNP             SCHOOLS          ENERGY
##  Min.   :0.100   Min.   :-1.900   Min.   : 1.00   Min.   :   24
##  1st Qu.:1.075   1st Qu.: 0.700   1st Qu.:29.75   1st Qu.:  469
##  Median :2.050   Median : 2.200   Median :45.00   Median : 1032
##  Mean   :1.954   Mean   : 2.473   Mean   :50.79   Mean   : 2572
##  3rd Qu.:2.700   3rd Qu.: 4.575   3rd Qu.:81.25   3rd Qu.: 4691
##  Max.   :3.700   Max.   : 5.900   Max.   :97.00   Max.   :11626
##                  NA's   :2
```

as we see, `trade.att` contains `POP_GROWTH`, `GNP`, `SCHOOLS` and `ENERGY` columns. as we know from description, these measures represent "average population growth between 1970 and 1981, average GNP growth per capita over the same period, secondary school enrollment ratio in 1981, and energy consumption in 1981 (in kilo coal equivalents per capita)".

`POP_GROWTH` varies from 0.1 to 3.7, `GNP` — from -1.9 to 5.9, `SCHOOLS` — from 1 to 97, and the possible maximum value is 100, as for the ration, and `ENERGY` lies between 24 and 11626. however, with last one, mean, median and even 3rd quartile are significantly lower than maximum value, which is a sign for some not numerous high values.

now we may apply attributes to the vertices:

```
set.vertex.attribute(manufacture.net, attrname="POP_GROWTH", value=trade.att[,1])
set.vertex.attribute(manufacture.net, attrname="GNP", value=trade.att[,2])
set.vertex.attribute(manufacture.net, attrname="SCHOOLS", value=trade.att[,3])
set.vertex.attribute(manufacture.net, attrname="ENERGY", value=trade.att[,4])
```

we can illustrate all of them with colour gradient.

POP_GROWTH

```
col_palette = brewer.pal(9, "Oranges")
colour_codes=as.numeric(cut(get.vertex.attribute(manufacture.net, "POP_GROWTH"), breaks=9))

plot(
        manufacture.net,
        mode = "fruchtermanreingold",
        displaylabels = TRUE,
        label.cex = 0.6,
        vertex.cex = 2 * total / 41,
        edge.lwd = 0.1,
        layout.par = list(niter = 2000),
        vertex.col = col_palette[colour_codes],
        edge.col = pastel2[8],
        label.col = accent[5]
    )
```

according to colours, leading countries in terms of volume of population are in the periphery of the graph, namely Syria, Liberia, Ecuador, Honduras, Algeria. lack of growth could be observed in central countries such as the UK, Switzerland, Finland, the USA. comparing with the size of nodes, fast-growing populations have lesser trade connections (and smaller size on the plot), while slow-growing countries have big or moderate number of links.

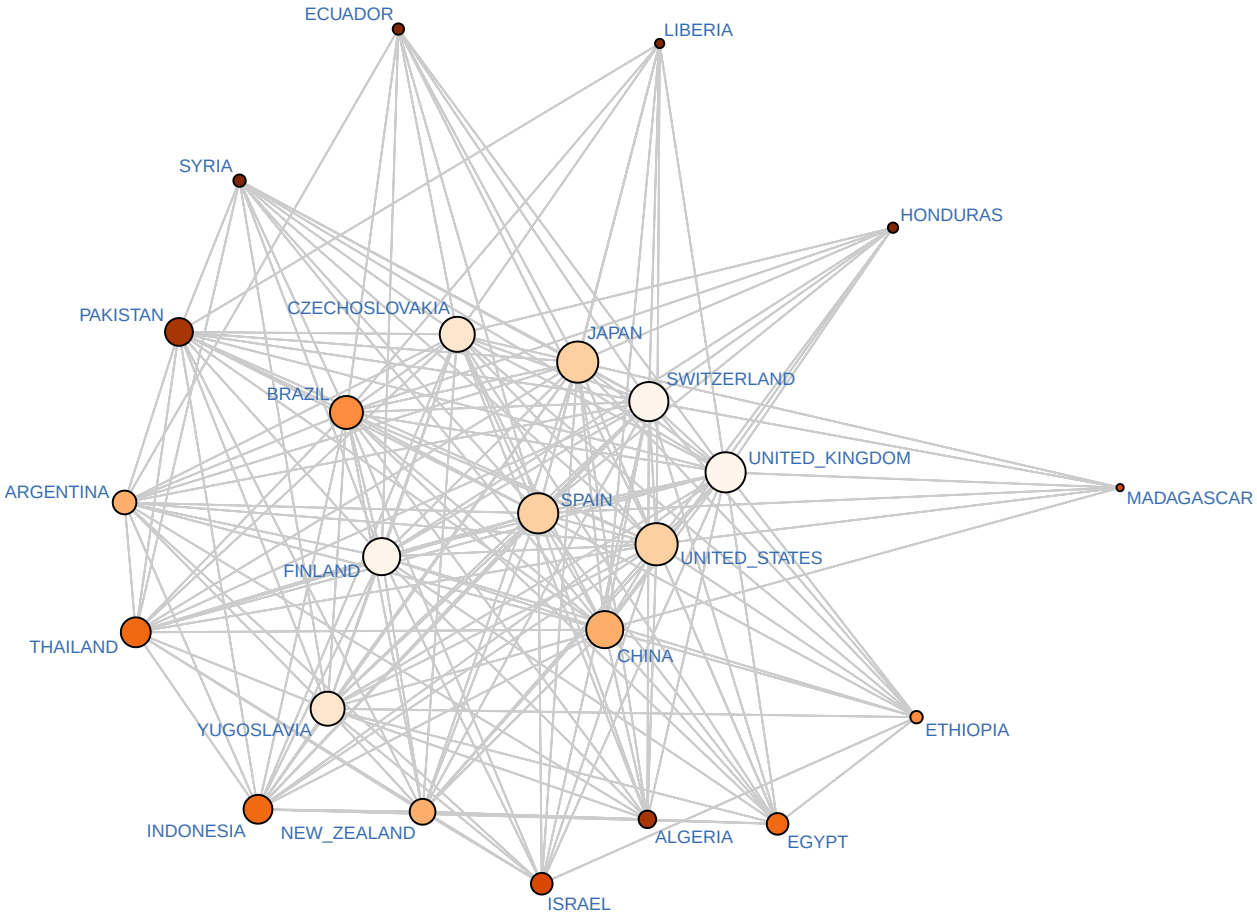

Figure 4: More intense colour means higher `POP_GROWTH`

GNP

```
col_palette = brewer.pal(9, "BuPu")
colour_codes = as.numeric(cut(get.vertex.attribute(manufacture.net, "GNP"), breaks = 9))

plot(
        manufacture.net,
        mode = "fruchtermanreingold",
        displaylabels = TRUE,
        label.cex = 0.6,
        vertex.cex = 2 * total / 41,
        edge.lwd = 0.1,
        layout.par = list(niter = 2000),
        vertex.col = col_palette[colour_codes],
        edge.col = pastel2[8],
        label.col = accent[5]
    )
```

speaking of GNP, leading countries here are Indonesia, Egypt, Brazil, Ecuador, Yugoslavia, Algeria. on the contrary, Liberia, Honduras, Ethiopia, Madagascar demonstrate worse economical perfomance. what is significant, the top countries are not the biggest on the plot — they have moderate number of links and middle size, which, probably, have a good potential for increase.

underperforming countries, as no surprise, have very minimal sizes on the plot.

there are two white circles on the graph — data from China and Czechoslovakia is not available.

SCHOOLS

```
col_palette = brewer.pal(9, "Greens")
colour_codes = as.numeric(cut(get.vertex.attribute(manufacture.net, "SCHOOLS"), breaks=9))

plot(
        manufacture.net,
        mode = "fruchtermanreingold",
        displaylabels = TRUE,
        label.cex = 0.6,
        vertex.cex = 2 * total / 41,
        edge.lwd = 0.1,
        layout.par = list(niter = 2000),
        vertex.col = col_palette[colour_codes],
        edge.col = pastel2[8],
        label.col = accent[5]
    )
```

in terms of school enrollment ratio, results are quite predicatble as well. in the developed countries with numerous edges the shade is darker, as school enrollment is at its peak. in countries with less amount of trade activity, observed educational score is not impressive as well.

ENERGY

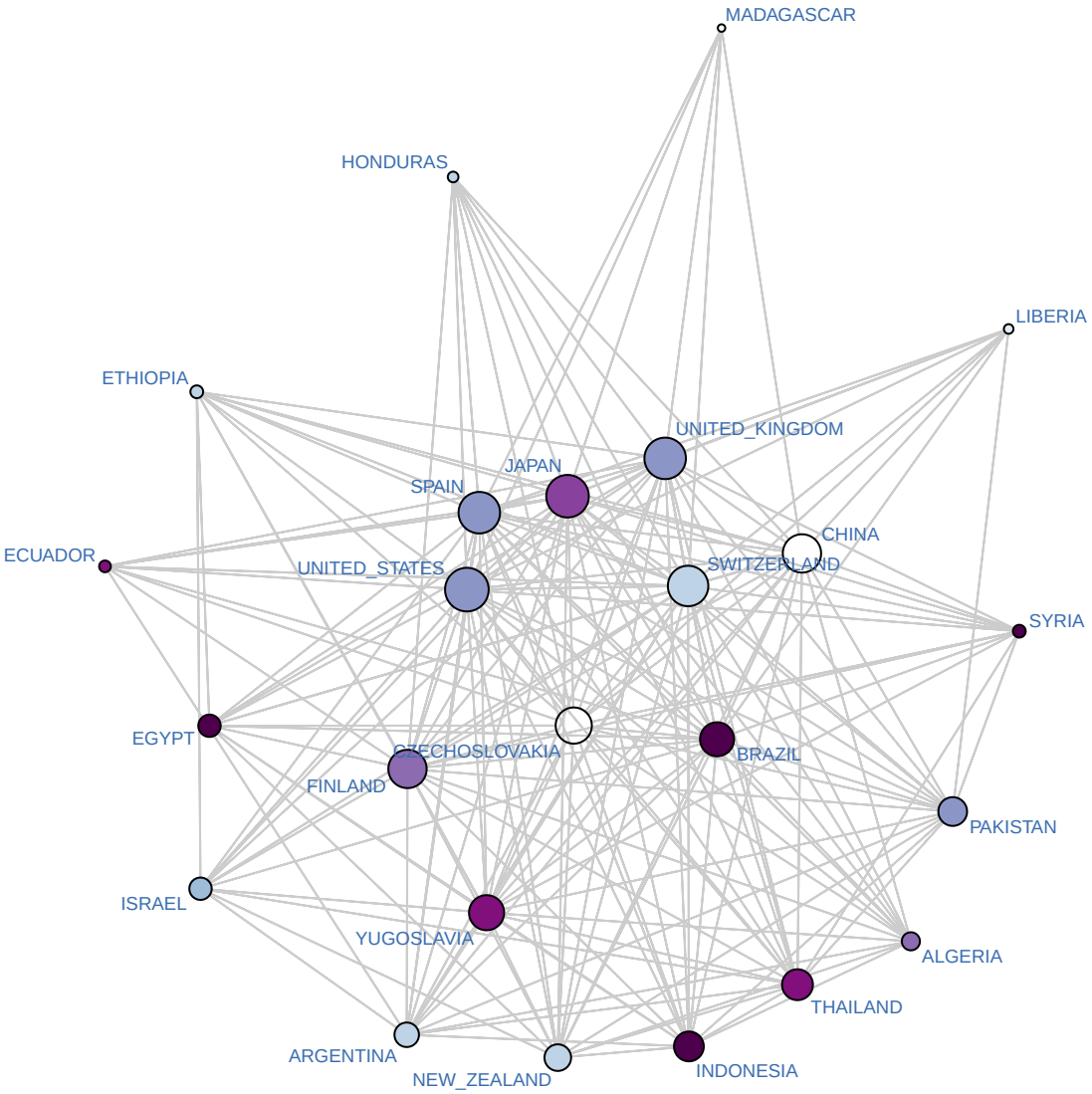

Figure 5: More intense colour means higher `GNP`

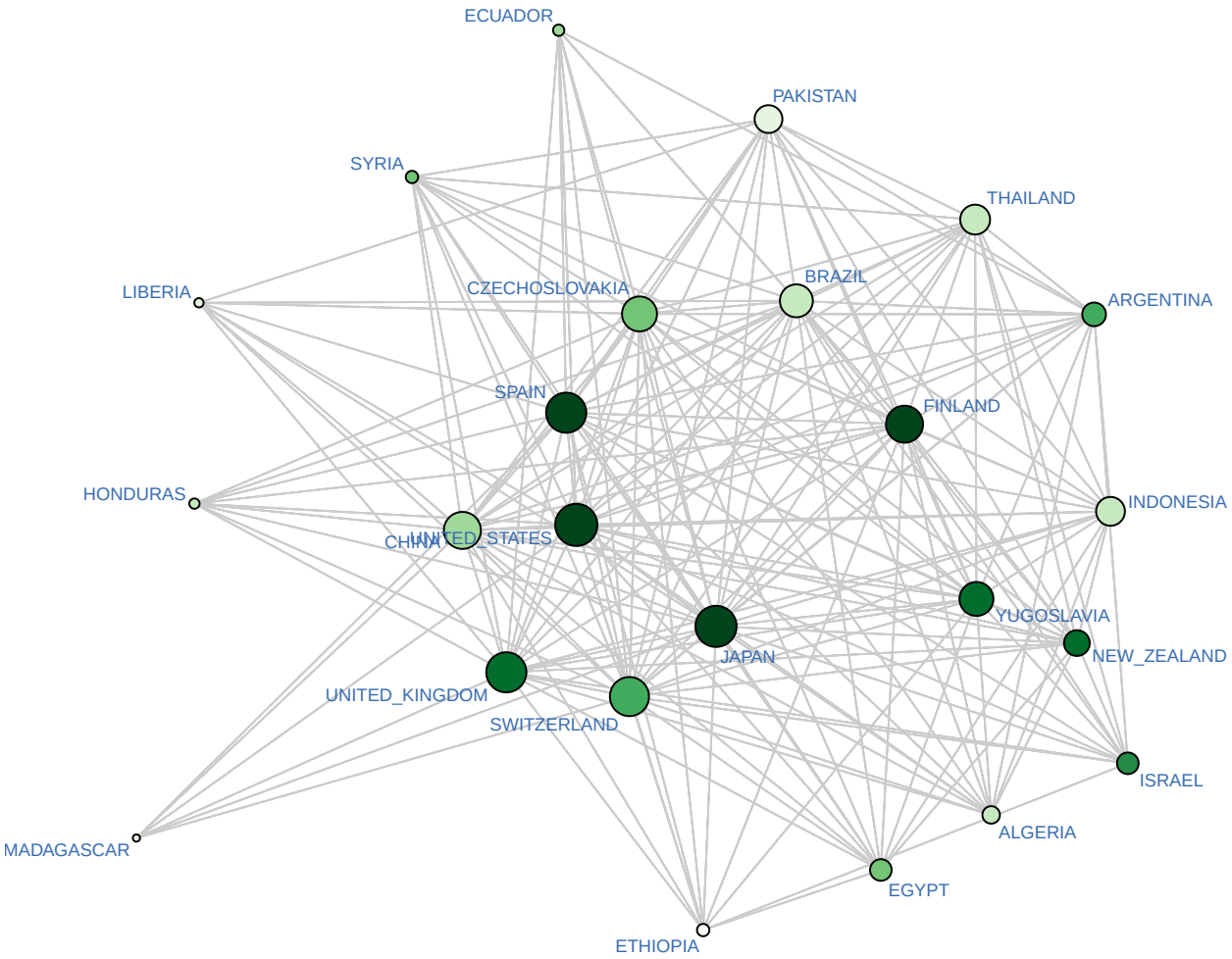

Figure 6: More intense colour means higher `SCHOOLS`

```
col_palette = brewer.pal(9, "Reds")
colour_codes = as.numeric(cut(get.vertex.attribute(manufacture.net, "ENERGY"), breaks=9))

plot(
        manufacture.net,
        mode = "fruchtermanreingold",
        displaylabels = TRUE,
        label.cex = 0.6,
        vertex.cex = 2 * total / 41,
        edge.lwd = 0.1,
        layout.par = list(niter = 2000),
        vertex.col = col_palette[colour_codes],
        edge.col = pastel2[8],
        label.col = accent[5]
    )
```

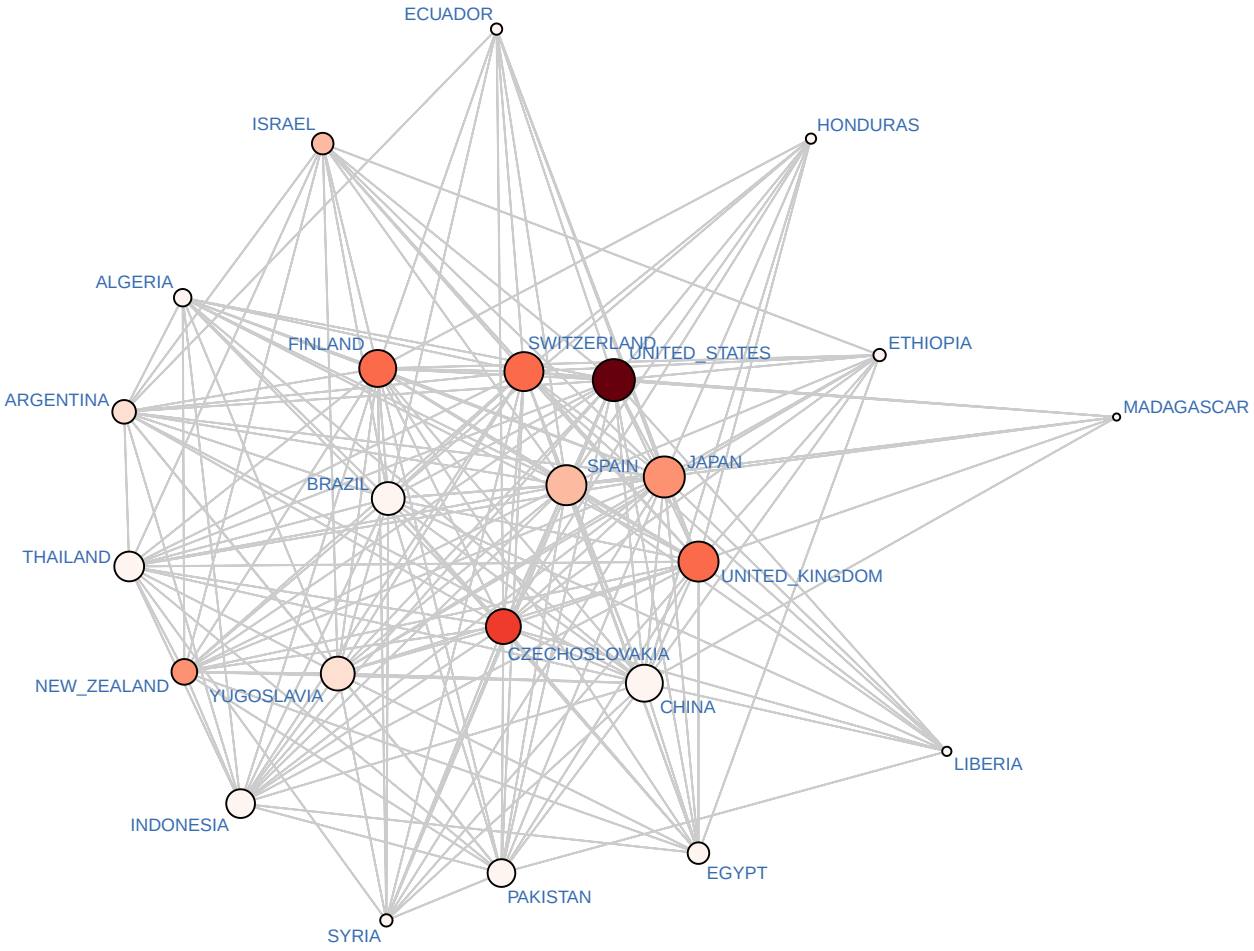

Figure 7: More intense colour means higher `ENERGY`

speaking of energy, the biggest consumer of it is the United States, after it there are some more countries (e. g. the UK, Finland, Switzerland, Japan, Czechoslovakia). these states have lots of edges, which proves them actively participating in the international trade with wide ranges of other countries. on the other hand,

countries with limited energy consumption (most of the countries, probably) are less in size on the graph, as they are less active in trading with other countries.

igraph

finally, we can switch to the `igraph` package.

```
detach(package:network)
suppressPackageStartupMessages(library(igraph))
```

now let's plot the network with size corresponding to Betweenness, and its colour corresponding to Eigenvector

```
col_palette = brewer.pal(9, "OrRd")
colour_codes = as.numeric(cut(eigen, breaks = 9))
plot(
    manufacture.graph,
    vertex.size = 5 + 0.35 * between,
    edge.arrow.size = 0.3,
    edge.color=pastel2[8],
    vertex.label.cex = 0.9,
    vertex.color = col_palette[colour_codes],
    vertex.shape = "circle",
    vertex.frame.color = pastel2[7],
    vertex.label.dist = 1.5,
    vertex.label.color = accent[5]
)
```

here it is clearly seen that United States and United Kingdom have the highest betweenness, which means that the number of shortest paths going through these vertices is higher than in others, so these node are important for the network. also these nodes have a very strong colour, as their transitive influence is significant (Eigenvector). the situation is reversed on the periphery.

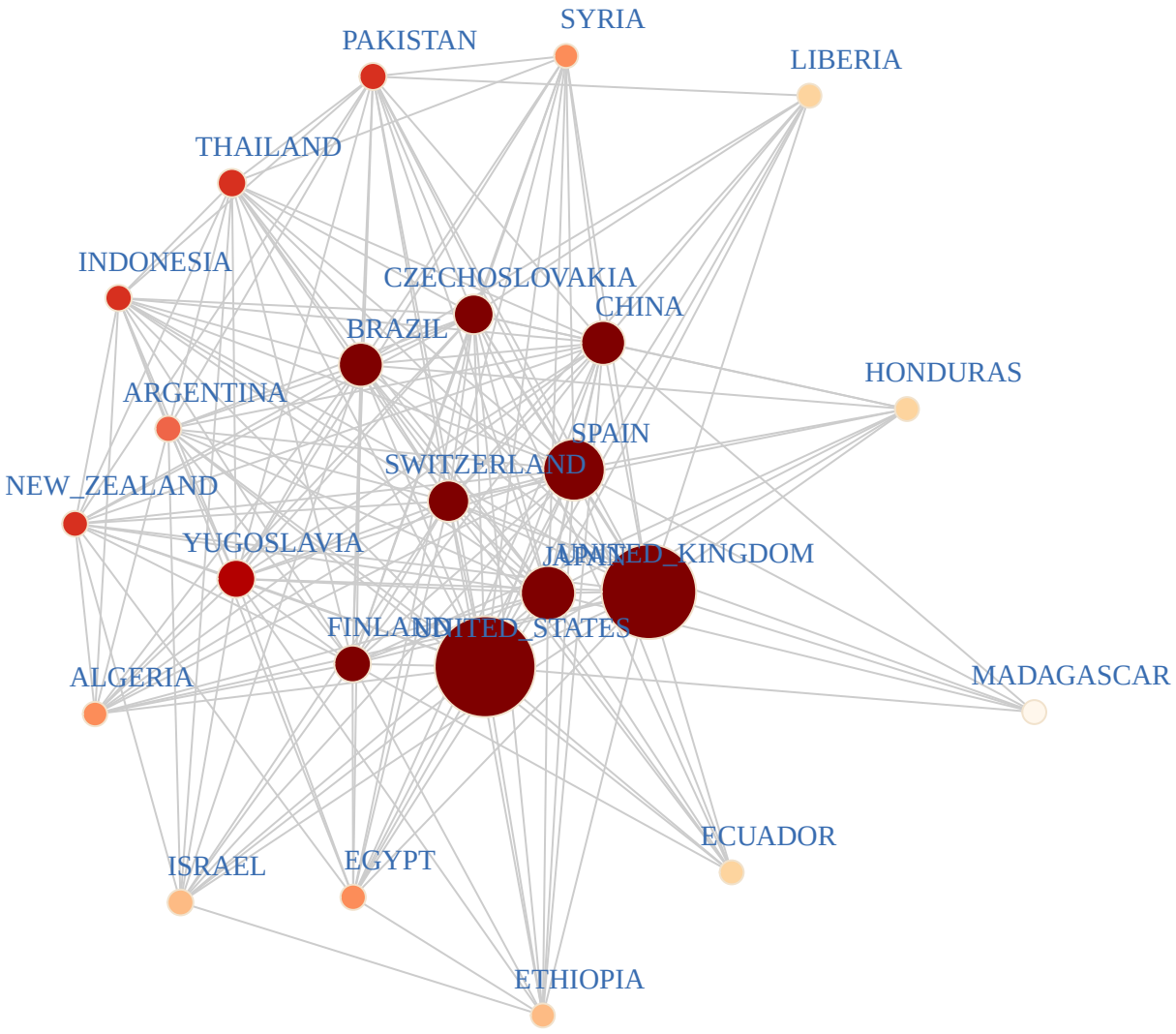

Figure 8: More intense colour means higher Eigenvector, and bigger size means higher Betweenness

community detection

community detection is an assignment of vertices to communities. the procedure is based on the modularity. the modularity is a measure of the network which describes the strength of division of a network into modules (communities). networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules.

let's perform community detection using three different methods.

Multilevel community detection

in this method nodes are moved between communities such that each node maximizes its own contribution to the modularity score. when all nodes stays in their communities, all the communities are collapsed into single nodes and the process continues (multilevel).

Multilevel method detects from two to four communities in our graph (Figure 9). so, probably, this method is quite unreliable in case of our network. we can print countries in a table next to modularity value, which is about 0.03 (Table 7).

```
eb <- multilevel.community(manufacture.graph)
plot(
        eb, manufacture.graph,
        vertex.size = 6,
        edge.color = "snow2",
        vertex.label.cex = 0.7,
        vertex.frame.color = "snow2",
        vertex.label.dist = 1.5,
        vertex.label.color = accent[5]
    )
```

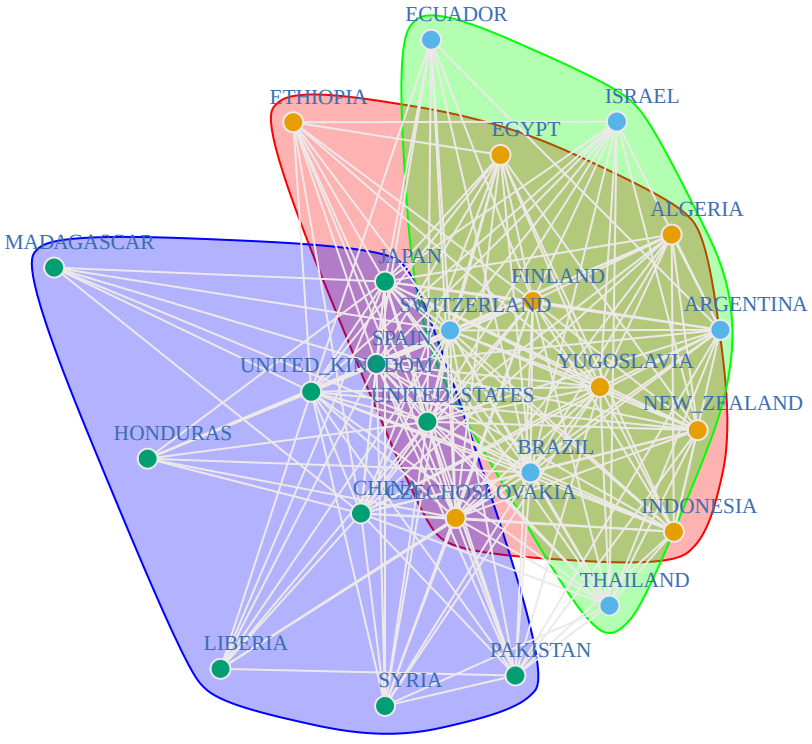

Figure 9: Multilevel community detection

Table 7: Multilevel communities and Modularity

ALGERIA	ARGENTINA	CHINA	0.031
CZECHOSLOVAKIA	BRAZIL	HONDURAS	
EGYPT	ECUADOR	JAPAN	
ETHIOPIA	ISRAEL	LIBERIA	
FINLAND	SWITZERLAND	MADAGASCAR	
INDONESIA	THAILAND	PAKISTAN	
NEW_ZEALAND		SPAIN	
YUGOSLAVIA		SYRIA	
		UNITED_KINGDOM	
		UNITED_STATES	

```
kable(
    c(eb[], modularity(eb)),
    col.names = NULL,
    digits = 4,
    caption = "Multilevel communities and Modularity"
    )
```

Fastgreedy community detection

fastgreedy community detection is a hierarchical approach. it tries to optimize modularity in a greedy manner. initially, every vertex belongs to a separate community, and communities are merged iteratively such that each step yields the largest increase in the current value of modularity. the algorithm stops when it is not possible to increase the modularity any more.

in case of Fastgreedy community detection we repeatedly get two communities (Figure 10). the first forms around the USA, and the second one — around United Kingdom (Table 8). Modularity is about 0.03 as well.

```
eb <- fastgreedy.community(manufacture.graph)
plot(
        eb,
        manufacture.graph,
        vertex.size = 6,
        edge.color = "snow2",
        vertex.label.cex = 0.7,
        vertex.shape = "circle",
        vertex.frame.color = "snow2",
        vertex.label.dist = 1.5,
        vertex.label.color = accent[5]
    )
```

```
kable(
    c(eb[], modularity(eb)),
    col.names = NULL,
    digits = 4,
    caption = "Fastgreedy communities and Modularity"
    )
```

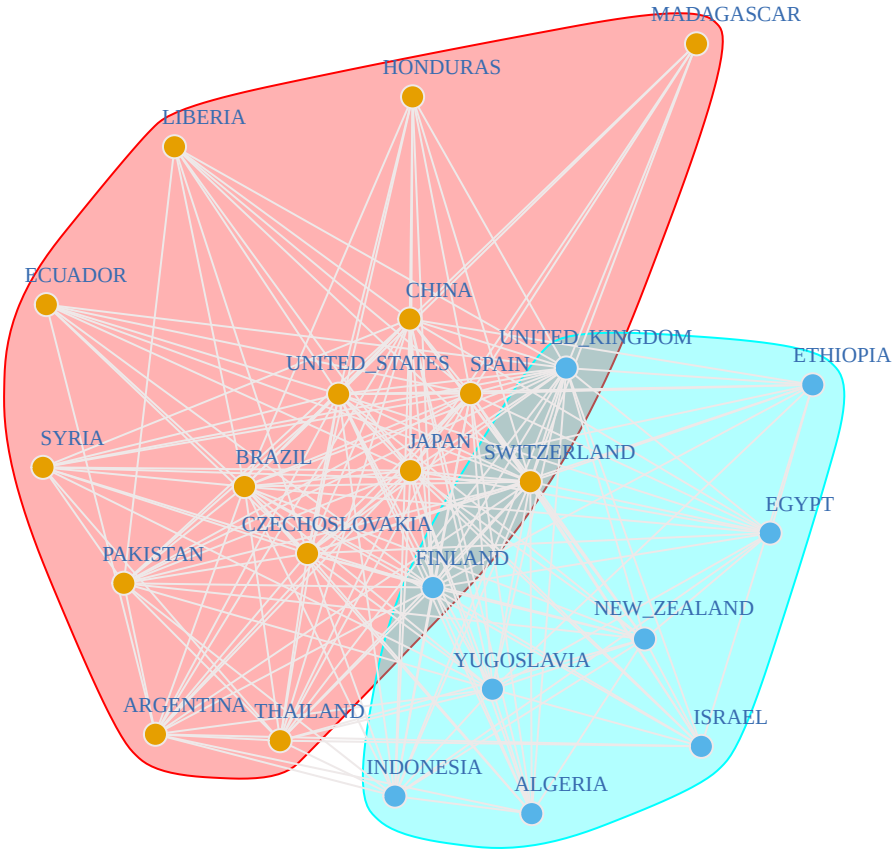

Figure 10: Fastgreedy community detection

Table 8: Fastgreedy communities and Modularity

ARGENTINA	ALGERIA	0.0285
BRAZIL	EGYPT	
CHINA	ETHIOPIA	
CZECHOSLOVAKIA	FINLAND	
ECUADOR	INDONESIA	
HONDURAS	ISRAEL	
JAPAN	NEW_ZEALAND	
LIBERIA	UNITED_KINGDOM	
MADAGASCAR	YUGOSLAVIA	
PAKISTAN		
SPAIN		
SWITZERLAND		
SYRIA		
THAILAND		
UNITED_STATES		

Spinglass community detection

Spinglass community detection is an approach from statistical physics, and it works similarly to particles that have spin. the process includes some simulation, after which the classes are formed. the number of possible classes is no grater than spins parameter.

in case of Spinglass method we again observe two communities (Figure 11, Table 9). the modularity is 0.04 — higher than in the previous attempts.

```
eb <- spinglass.community(manufacture.graph)
plot(
        eb,
        manufacture.graph,
        vertex.size = 6,
        edge.color = "snow2",
        vertex.label.cex = 0.7,
        vertex.shape = "circle",
        vertex.frame.color = "snow2",
        vertex.label.dist = 1.5,
        vertex.label.color = accent[5]
    )
```

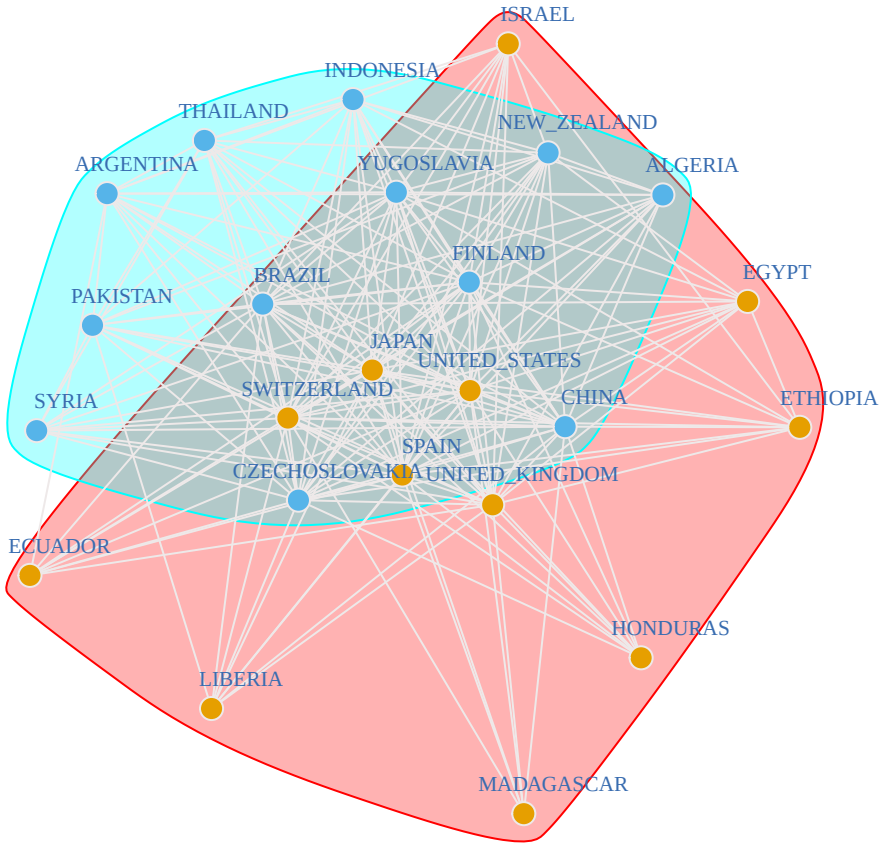

Figure 11: Spinglass community detection

```
kable(
    c(eb[], modularity(eb)),
```

Table 9: Spinglass communities and Modularity

ECUADOR	ALGERIA	0.0417
EGYPT	ARGENTINA	
ETHIOPIA	BRAZIL	
HONDURAS	CHINA	
ISRAEL	CZECHOSLOVAKIA	
JAPAN	FINLAND	
LIBERIA	INDONESIA	
MADAGASCAR	NEW_ZEALAND	
SPAIN	PAKISTAN	
SWITZERLAND	SYRIA	
UNITED_KINGDOM	THAILAND	
UNITED_STATES	YUGOSLAVIA	

```
    col.names = NULL,
    digits = 4,
    caption = "Spinglass communities and Modularity"
    )
```

conclusions

Finally, after all manipulations performed upon the network `manufacture`, we may summarise all the evidence and draw the conclusions based on the previous observations.

The given network describes business relations between 24 countries in terms of trading of manufactured goods in the end of XX century. The countries included in this dataset are: `ALGERIA, ARGENTINA, BRAZIL, CHINA, CZECHOSLOVAKIA, ECUADOR, EGYPT, ETHIOPIA, FINLAND, HONDURAS, INDONESIA, ISRAEL, JAPAN, LIBERIA, MADAGASCAR, NEW_ZEALAND, PAKISTAN, SPAIN, SWITZERLAND, SYRIA, THAILAND, UNITED_KINGDOM, UNITED_STATES, YUGOSLAVIA`. Besides the adjacency matrix (containing information about connectons between countries), the vertices attributes were provided, namely `POP_GROWTH, GNP, SCHOOLS, ENERGY`. All data combined, here is the description of the network.

General information

As was proven, the given network is directed, which means that countries exchanged manufactured goods not necessarily in a reciprocal way: for example, there is a link from Czechoslovakia to Argentina and no link in reverse.

Speaking of links, there are 195 of them in undirected mode and 310 considering directions. We may comapre those numbers with the number of Potential Connections:

$$PC = \frac{n(n-1)}{2}$$

or two times more in case of directed network. Hence, after calculation, we will obtain what is called Network Density — 0.71 and 0.56 respectively. Those are high values, they indicate that 70% or 56% of all possible links are in fact present.

Analysing diads and triads, we may conclude that only 30% of the vertices are not connected directly (at least asymmetrically). It again proves high cohesiveness of the network. The high value of the transitivity in observed network is natural for social networks. The geodesic distances in our network are rather low, since many vertices are connected directly. In fact, for undirected network maximum geodesic distance is 2, and is slightly higher for directed network — 3 — remembering that some node can be disconnected in directed networks. Mean distances are 1.2 and 1.3, respectively. All these measures indicate the face that our network is indeed coherent and well-connected — as many trading interactions exist between nearly all countries.

In order to evaluate the impact of one countries to others, centrality metrics were calculated (Table 1). Although each of the selected centralities shows a bit different result, in general they provide more or less similar estimation of whether a country is active or influential on the international market. This could be observed from the correlation matrix (Table 2). According to the table, basic centralities — Degree, Betweenness, Closeness, Eigenvector — are quite correlated (correlation coefficients are greater than 0.67, and some are close to 1.0).

Some curious observations could be made with additional centralities, specifically Topological Coefficient. In contrast with the others, Topological metric was higher for countries with few edges, such as `Madagascar, Honduras, Liberia, Ecuador`. This coefficient is calculated as the number of shared neighbours with all the neighbours of the node, divided by the number of neighbours. Possibly, listed countries have lower number of neighbours, hence the fraction is high.

To sum up, explored network has a lot of ties between countries, most of nodes are interconnected, which draws a conclusion about high activity of international trading during the observed years.

Influential nodes

Continuing our network research, the selection of significant indicies in the network was performed. On this purpose, we selected top six countries according to four basic centralities: Indegree, Outdegree, Total degree and Betweenness. The results of filtering are presented in Table 5. Firstly, there are countries which are leading in each category. Those are: `United States, United Kingdom, Japan, Spain`. All of them have the highest values of Degree centrality and Betweenness, which characterises them as big importers and exporters, and also being mediators for other countries. However, our conclusions are not based on amount of

trading but the number of links, or number of partners in the network. Still we may state these five countries have the largest number of partners on the market.

Besides, leading in Indegree centrality are `China` and `Finland`, so apparently these countries do import from a wide range of exporters. And the leaders amongst exporters are also `Switzerland` and `Brazil`.

Another important property of influential nodes must be Eigenvector centrality, as it is the bigger the more influential neighbours the node possess. We may see in Table 6 that `Spain, Japan, Switzerland` and `the USA` have centrality equal 1.0. Therefore, these countries are considered transitive influential (they have more connections with top-scoring countries than other).

Conclusions from visualisation

Our network has different properties, some of them we have discussed above. In addition, all vertices are followed with some attributes. It is convenient to compare these metrics on a plot — where they influence some geometrical properties and hence could be visually taken into account. In this work we performed such a comparison and here are the results.

Omitting trivial visualisation, we should look at Figure 3. There we assigned orange colour for nodes with Indegree prevailing over Outdegree. Turned out countries with lesser Total degrees have more incoming connections than outcoming. It could possibly be explained with the fact that such countries do not have enough products to export.

Next we performed visualisation of network attributes: `POP_GROWTH, GNP, SCHOOLS, ENERGY`. At first, we painted vertices in such way that more intense colour corresponds to the higher Population growth, and the size of the vertices still indicates Total degree centrality. At Figure 4 we may notice that leading countries in terms the growth of population are in the periphery of the graph, namely `Syria, Liberia, Ecuador, Honduras, Algeria`. Lack of growth could be observed in central countries such as `UK, Switzerland, Finland, USA`. Comparing with the size of nodes, fast-growing populations have lesser trade connections (and smaller size on the plot), while slow-growing countries have big or moderate number of links. Further observation are more curious. The plot illustrates that the top countries with respect to GNP growth are not the biggest on the plot, but have moderate number of connections (Figure 5). Apparently, this means that those countries are in their developing stage with enough room for advancement of their trade interactions.

Figures 6 abd 7 demonstrate School enrollment rate and Energy consumption. Here the results are quite expected: in the countries with vast number of connections both metrics are peaking, while less developed countries have limited educational engagement and energy consumption scores.

Finally, using `igraph` package we showed that Eigenvector correlate with Betweenness, as the countries with high metrics — `UK, USA, Spain, Japan, China` — are coloured in the darker shade and are bigger in size.

Concluding, visualisation is a very powerful method of exploring relations between various parameters of a network. As described above, it is possible to compare whether network statistics influence given network attribute, i. e. network configuration is locked with success or other external characteristics of nodes in reality.

Community detection

The last approach in network exploration was the community detection. Such methods are aimed to split the network on some modules (or communities). We emloyed three different methods in this work: Multilevel, Fastgreedy and Spinglass methods. Now let's analyse the outcomes from these manipulations.

The first method was not very reliable, as it divides the network in a variable number of groups (from 2 to 4). I would suggest the reason for such a behaviour is that the observed network is tightly coupled and there are not easily separable parts. The next method (Fastgreedy) forms two groups. One of them always contains `USA`, another — `UK`. The modularity in this case is about 0.03. The last one also creates two communities, however, no obvious explanation cannot be provided for them. The modularity is higher (0.04).

I believe these network is too linked to be splitted to communities, since three methods were unable to do it in any satisfying manner.

References

While working on this study, I used several publically available sources. I will attempt to list them here.

- Seminar tutorials
- stackoverflow.com
- 9 Lesson 4: YAML Headers | R Markdown Crash Course: https://zsmith27.github.io/rmarkdown_crash-course/lesson-4-yaml-headers.html
- Case Studies: Network Analysis in R: https://rstudio-pubs-static.s3.amazonaws.com/708966_c12587b9536146ca91cc0c805ef288dd.html
- R Markdown: The Definitive Guide: https://bookdown.org/yihui/rmarkdown/html-document.html
- R Markdown Cookbook: https://bookdown.org/yihui/rmarkdown-cookbook/figure-placement.html
- Summary of community detection algorithms in igraph 0.6 | R-bloggers: https://www.r-bloggers.com/2012/06/summary-of-community-detection-algorithms-in-igraph-0-6/
- community.rstudio.com
- www.geeksforgeeks.org
- igraph: Network Analysis and Visualization: https://cran.r-project.org/web/packages/igraph/igraph.pdf
- plot.network: Two-Dimensional Visualization for Network Objects in network: Classes for Relational Data: https://rdrr.io/cran/network/man/plot.network.html
- R Color Brewer's palettes – the R Graph Gallery: https://r-graph-gallery.com/38-rcolorbrewers-palettes.html
- RPubs - Creating a Network Plot in R: https://rpubs.com/brandonkopp/creating-a-network-plot-in-R
- en.wikipedia.org
- igraph R manual pages: https://igraph.org/r/doc/distances.html
- Arch Linux - gcc-fortran 12.2.0-1 (x86_64): https://archlinux.org/packages/core/x86_64/gcc-fortran/
- geodist function - RDocumentation: https://www.rdocumentation.org/packages/gmt/versions/1.2-0/topics/geodist
- The unique() function in R programming | DigitalOcean: https://www.digitalocean.com/community/tutorials/unique-function-r-programming