

# Human Disease Network Analysis

*Giorgia Adorni 806787  
Lorenzo Mammana 807391*

*2019-06-14*



# Contents

<b>1</b>	<b>Introduzione</b>	<b>5</b>
<b>2</b>	<b>Network analysis</b>	<b>7</b>
2.1	Operazioni preliminari . . . . .	7
2.2	Node similarity . . . . .	8
2.3	Centrality Analytics . . . . .	9
2.4	Centralità di grado . . . . .	9
2.5	Betweenness . . . . .	12
2.6	Closeness . . . . .	14
2.7	Eigenvector . . . . .	16
2.8	Pagerank . . . . .	18
2.9	Coefficiente di clustering . . . . .	20
2.10	Grafo utilizzato per il clustering . . . . .	21
2.11	Centrality Analytics . . . . .	21
<b>3</b>	<b>Clustering</b>	<b>31</b>
3.1	Operazioni preliminari . . . . .	31
3.2	Groundtruth . . . . .	32
3.3	Algoritmi di clustering . . . . .	37
3.4	Girvan–Newman . . . . .	38
3.5	Fastgreedy . . . . .	39
3.6	Louvain . . . . .	39
3.7	Spinglass . . . . .	41
3.8	Markov Cluster Algorithm . . . . .	41
3.9	Leiden Algorithm . . . . .	43
3.10	Label propagation . . . . .	44
3.11	Leading eigenvector . . . . .	45
3.12	Valutazione degli algoritmi . . . . .	46
3.13	Distribuzione di centralità . . . . .	50
<b>4</b>	<b>Conclusioni</b>	<b>53</b>
<b>5</b>	<b>Bibliografia</b>	<b>55</b>
<b>A</b>	<b>Appendice</b>	<b>57</b>
A.1	Analisi sulle malattie contrassegnate come Unclassified: . . . . .	57
A.2	Analisi sulle malattie contrassegnate come Multiple: . . . . .	59



# Introduzione

Nel presente lavoro viene proposta la valutazione in termini di accuratezza di svariati algoritmi di clustering, effettuati sulla rete conosciuta come “Human Disease Network” (HDN (2007)).

La rete in questione contiene nodi che rappresentano malattie e nodi di espressione genica, che sono così collegati:

- Malattia-Gene: se la mutazione del gene causa la malattia
- Malattia-Malattia: se vi è almeno un gene condiviso

L'obiettivo di studio su questa rete è quello di verificare la presenza di distinti moduli funzionali di espressione genica, che vanno ad influenzare un certo tipo di malattia. I ricercatori hanno dimostrato che alcuni geni tendono a raggrupparsi in moduli ben distinti, ma anche che molti di essi non lo fanno e tendono invece a isolarsi in zone periferiche della rete.

I ricercatori hanno manualmente etichettato ogni nodo, scegliendo tra 22 cluster che rappresentano il sistema fisiologico colpito dalla malattia, permettendo quindi di avere una groundtruth su cui effettuare delle valutazioni.

Basandosi sulle ipotesi effettuate in precedenza, sono riusciti a dimostrare come alcuni di questi cluster si trovino effettivamente vicini e ben connessi all'interno della rete (Es. i nodi relativi al cancro condividono un enorme numero di geni), mentre altri cluster sono sparsi per tutta la rete e quindi difficilmente classificabili utilizzando approcci automatici di partizionamento.

L'obiettivo del lavoro è stato quindi quello di valutare le performance di vari algoritmi di *graph partitioning* e *community detection*, nel rilevare i cluster all'interno della rete, utilizzando principalmente approcci non supervisionati e semi-supervisionati.



# Network analysis

## 2.1 Operazioni preliminari

Cominciamo caricando le librerie necessarie ed il grafo.

```
# set working directory
# setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
# source a set of functions
source("util.R")

# import libraries
libraries_list <- c("ggraph", "igraph", "dplyr", "readr", "DiagrammeR", "tidyverse",
                     "Cairo", "networkD3", "CINNA", "scales", "pander")
import_libraries(libraries_list)

# import graph
edges <- read.csv(
  "../dataset/diseasome [Edges].csv",
  head = TRUE
)

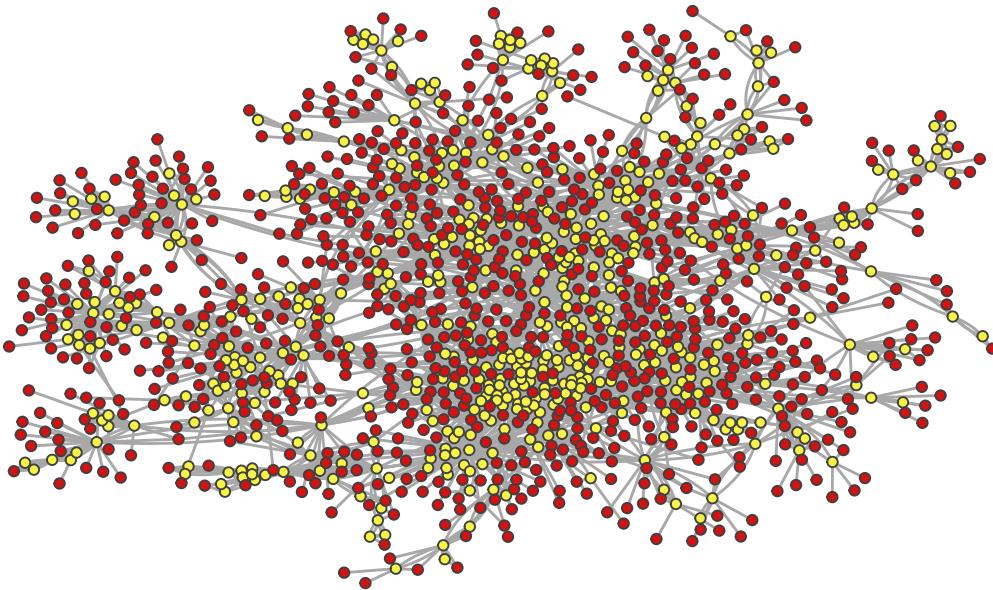
nodes <- read.csv(
  "../dataset/diseasome [Nodes].csv",
  head = TRUE
)

nodes <- nodes %>% select(-timeset)
edges <- edges %>% select(-timeset, -label)

graph <- graph.data.frame(edges, directed = TRUE, vertices = nodes)
# print(graph, e=TRUE, v=TRUE) ## IGRAPH 9af20f4 DNW- 1419 3926

ggraph(graph, layout="graphopt") +
  geom_edge_fan(colour = "gray66", show.legend = FALSE) +
  geom_node_point(fill= ifelse(nodes$X0 == "gene", "#D61111", "#F8F445"),
                  shape=21, col="grey25", show.legend = FALSE) +
  scale_size_continuous(range=c(1, 10)) +
  theme_graph(base_size = 11, base_family = "sans") +
  ggtitle("Human Disease Network")
```

## Human Disease Network



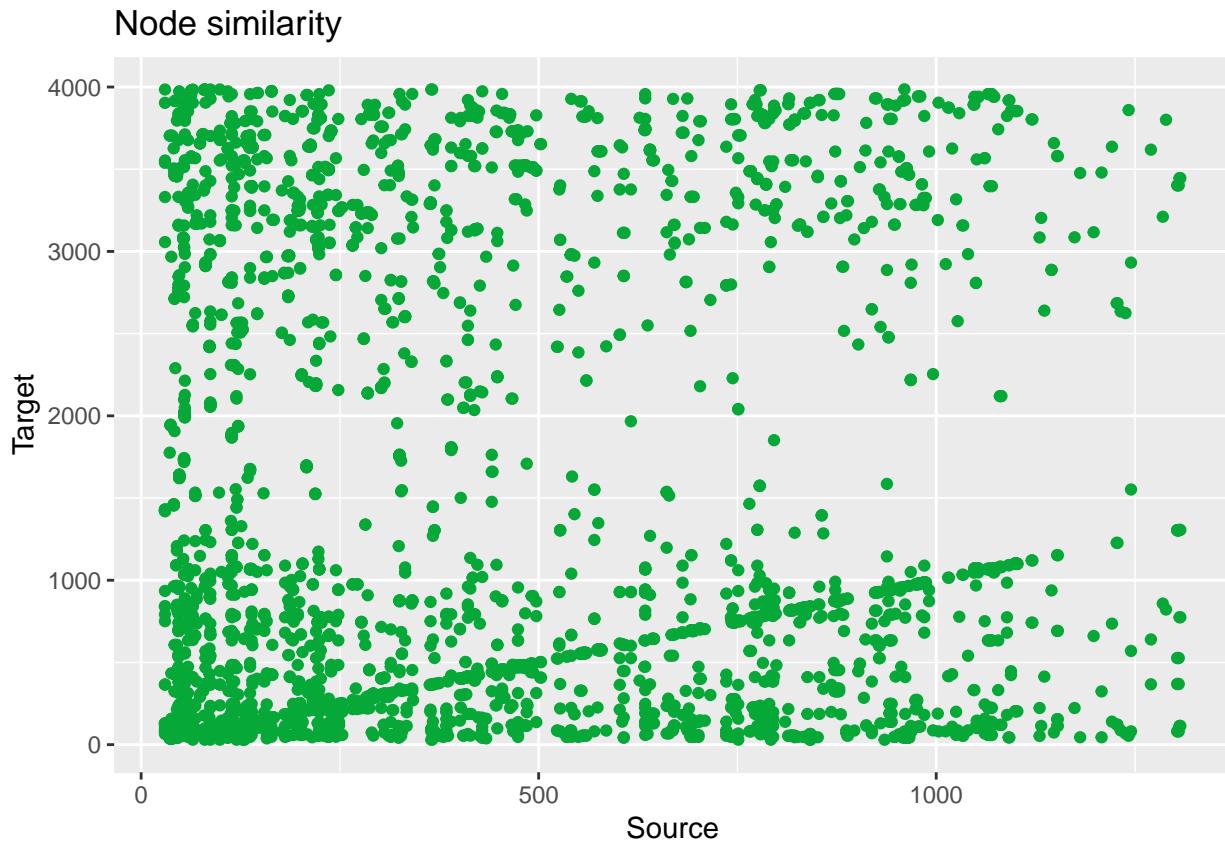
In rosso sono rappresentati i geni, mentre in giallo sono rappresentate le malattie.

```
print(paste("N° di nodi: ", vcount(graph)))
## [1] "N° di nodi: 1419"
print(paste("N° di geni: ", sum(V(graph)$X1 == "gene")))
## [1] "N° di geni: 903"
print(paste("N° di archi: ",ecount(graph)))
## [1] "N° di archi: 3926"
```

## 2.2 Node similarity

Plottiamo un grafo che, per ogni arco della rete, ha sulle ascisse il nodo sorgente, e sulle ordinate il nodo destinazione. Le colonne del grafo simili rappresentano nodi che sono simili all'interno della rete.

```
ggplot(edges, aes(x = edges$Source, y = edges$Target)) +
  geom_point(color="#06a938") +
  labs(x = "Source") +
  labs(y = "Target") +
  ggtitle("Node similarity")
```



Vediamo che sull'asse delle x abbiamo la metà dei nodi rispetto all'asse dell'y, questo è dovuto al fatto che i nodi genici hanno solamente archi entranti e sono etichettati con un valore superiore a 1300.

La parte inferiore del grafo rappresenta quindi tutti i collegamenti tra le malattie, mentre quella superiore quelli tra malattie e geni.

## 2.3 Centrality Analytics

Le reti complesse sono difficili da visualizzare, quindi si ricorre a misure e statistiche descrittive, quali le misure di centralità, per ricavare informazioni sulla struttura della rete. Esse risultano utili per rilevare quali sono e dove si trovano i nodi importanti all'interno della rete.

## 2.4 Centralità di grado

Il grado di un nodo è il numero di archi entranti o uscenti. Formalmente, l'outdegree di un nodo  $i$  è dato da  $\sum_{j=1}^n A_{ij}$  mentre l'indegree di un nodo  $j$  è dato da  $\sum_{i=1}^n A_{ij}$ . Questa quantità permetterà di calcolare la distribuzione di grado, essenziale per capire il livello di connessione del grafo: se è bassa, la maggior parte dei nodi avranno pochi link, mentre qualcuno ne avrà molti.

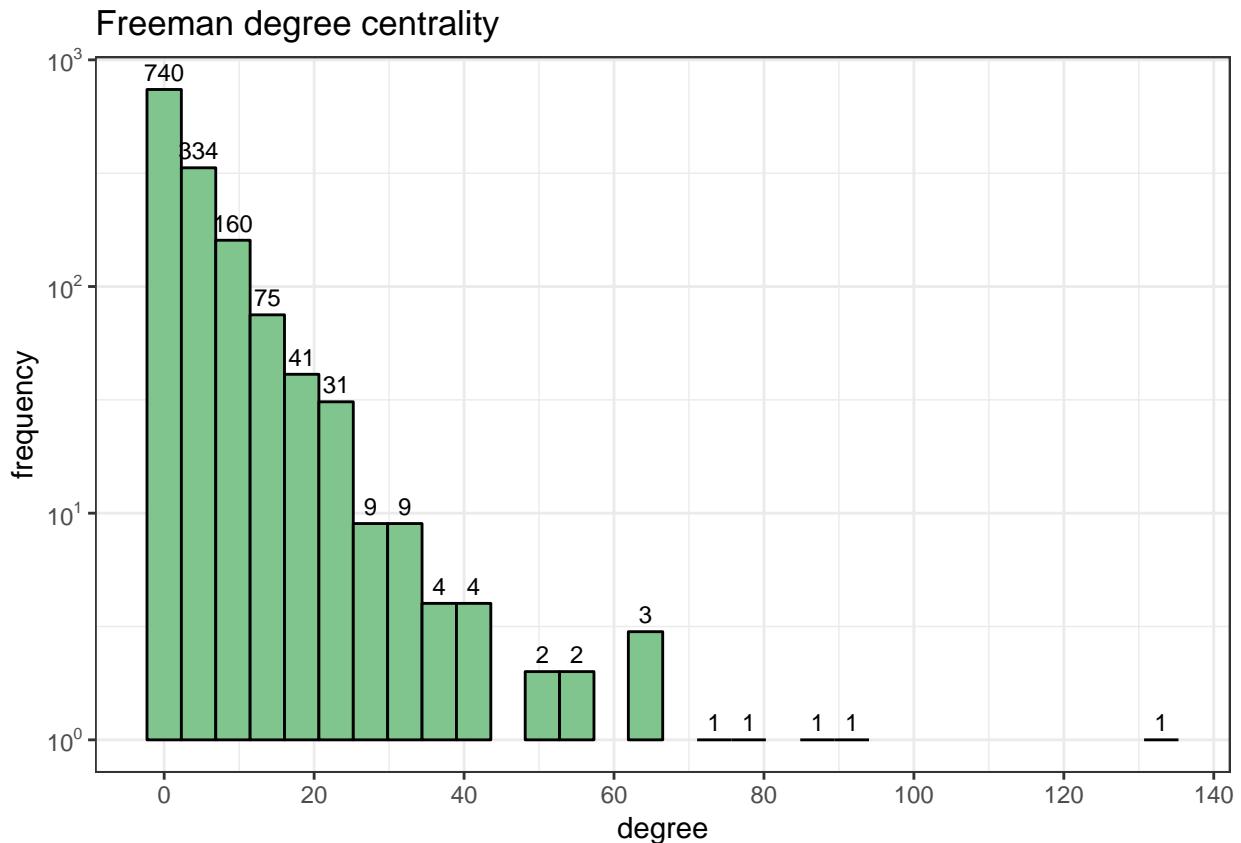
I nodi della rete che presenteranno molti collegamenti verranno definiti come **hub**.

Cominciamo mostrando la centralità dei nodi ricavata utilizzando la formula di Freeman:

$$C_D = \frac{\sum_{i=1}^N C_D(n^*) - C_D(i)}{(N-1)(N-2)}$$

```
degree <- centr_degree(graph, mode = "all", normalized = TRUE)

histogram_plot(degree$res, seq(0, 160, by = 20), "degree", "frequency",
               "Freeman degree centrality")
```



Vediamo come la maggior parte dei nodi ha un valore di degree centrality vicino a uno, questo è dovuto al fatto che il grado della maggior parte dei geni è esattamente uno.

Mostriamo quindi i 5 nodi con grado più elevato, che corrispondono agli hub.

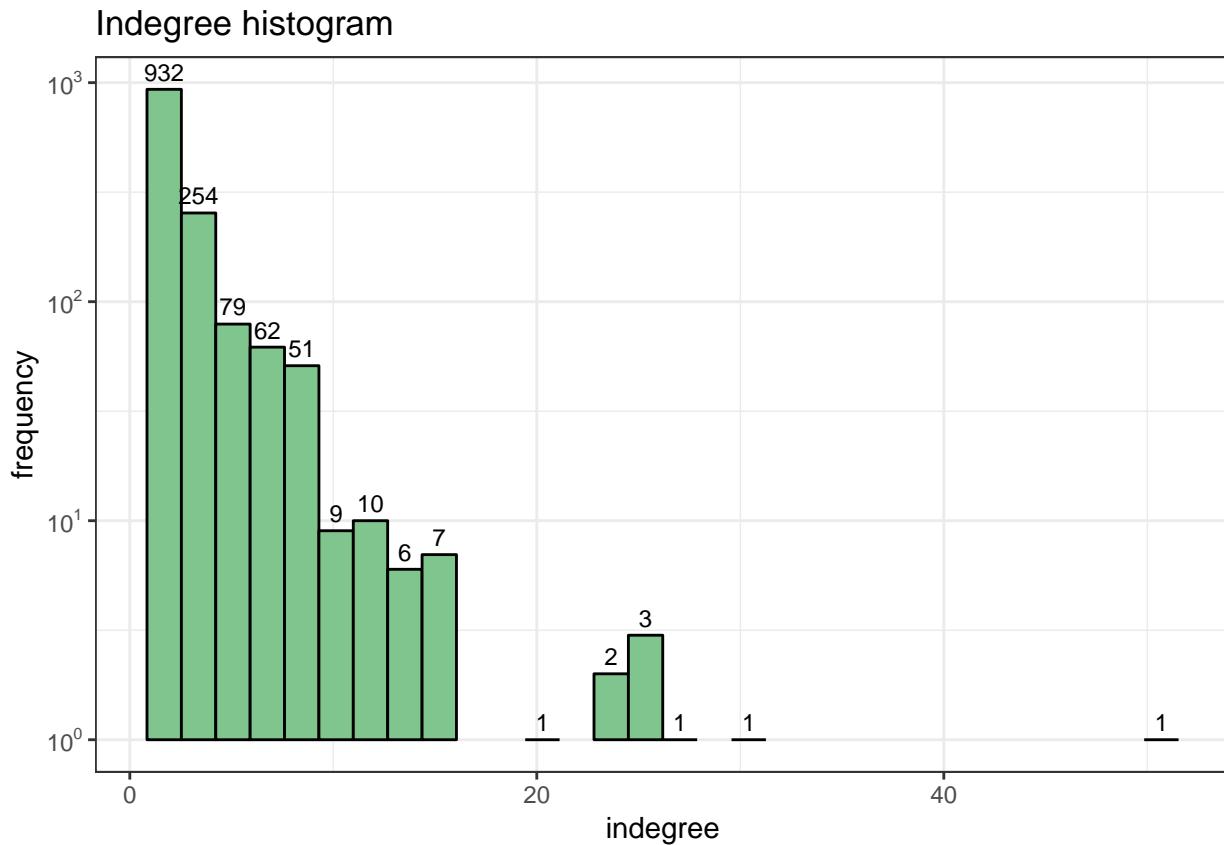
```
v(graph)$label[order(degree$res, decreasing=TRUE)][1:5]
```

```
## [1] "Colon cancer"      "Deafness"           "Leukemia"
## [4] "Breast cancer"     "Diabetes mellitus"
```

Essendo il grafo diretto è inoltre utile effettuare valutazioni riguardo l'in-degree e l'out-degree.

```
indegree <- centr_degree(graph, mode = "in", normalized = TRUE)
```

```
histogram_plot(indegree$res, seq(0, 160, by = 20), "indegree", "frequency",
               "Indegree histogram")
```

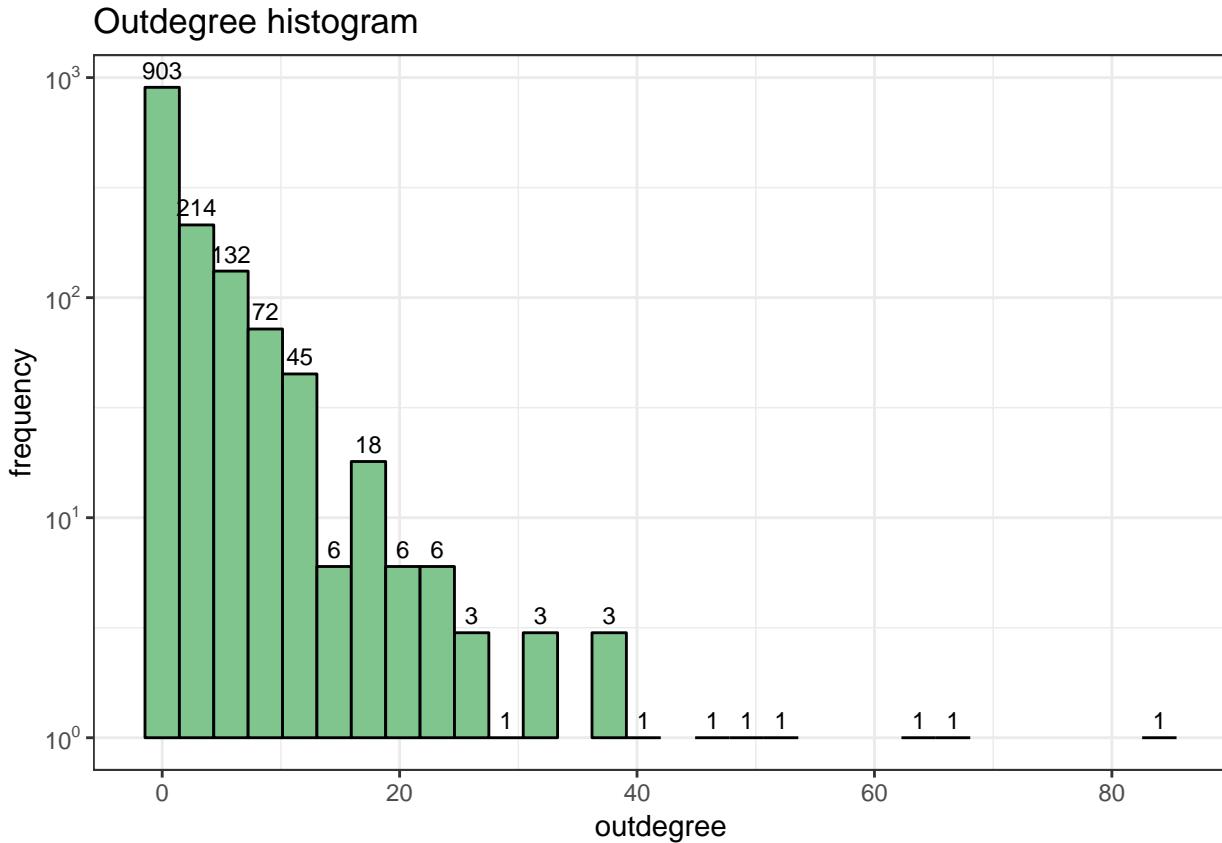


```
# node with the highest indegree
V(graph)$label[order(indegree$res, decreasing=TRUE)][1:5]

## [1] "Colon cancer"      "Breast cancer"     "Gastric cancer"
## [4] "Leukemia"          "Thyroid carcinoma"
# "Colon cancer", "Deafness", "Leukemia", "Breast cancer", "Diabetes mellitus"

outdegree <- centr_degree(graph, mode = "out", normalized = TRUE)

histogram_plot(outdegree$res, seq(0, 160, by = 20), "outdegree", "frequency",
               "Outdegree histogram")
```



```
# node with the highest outdegree
V(graph)$label[order(outdegree$res, decreasing=TRUE)][1:5]
```

```
## [1] "Colon cancer"      "Deafness"           "Leukemia"
## [4] "Diabetes mellitus" "Breast cancer"
```

Vediamo come tutti i 903 geni non hanno archi uscenti.

La centralizzazione in base al grado è una misura puramente locale, che può variare molto in base alla struttura del grafo, quindi potrebbe non essere sufficiente a descrivere l'influenza di un nodo sull'intera rete.

## 2.5 Betweenness

La misura di betweenness è una misura di centralità meno locale della precedente, basata sulla seguente formula:

$$C_B(i) = \sum_{j \neq k} \frac{g_{jk}(i)}{g_{jk}}$$

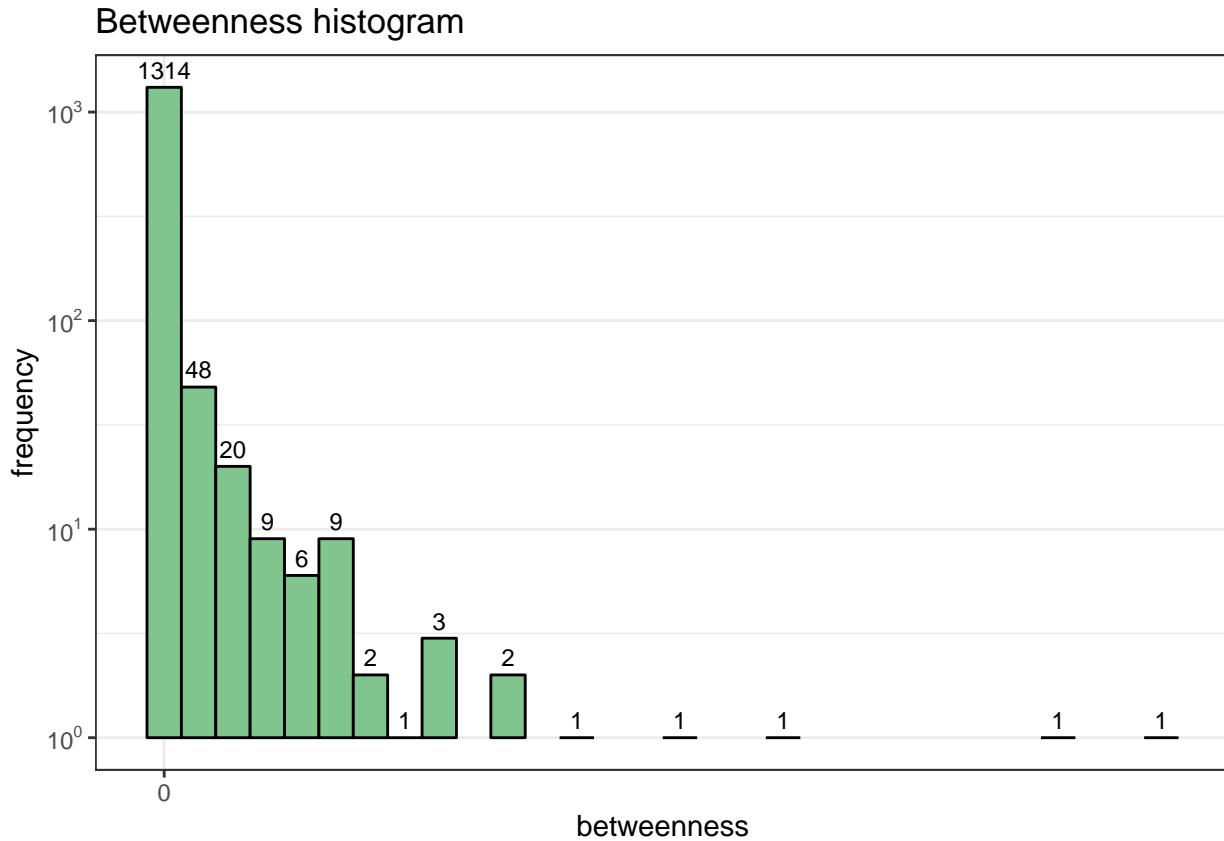
Dove  $g_{jk}(i)$  rappresenta il numero di shortest-path che passano per  $i$  tra  $j$  e  $k$ , e  $g_{jk}$  è il numero di shortest-path totali.

Essa verrà calcolata nella sua forma normalizzata

$$C'_B(i) = \frac{C_B(i)}{(n-1)(n-2)/2}$$

```
betweenness <- betweenness(graph, v = V(graph), directed = TRUE, normalized = TRUE)

histogram_plot(betweenness, c(0, 1), "betweenness", "frequency", "Betweenness histogram")
```



```
# node with the highest betweenness
V(graph)$label[order(betweenness, decreasing=TRUE)][1:5]

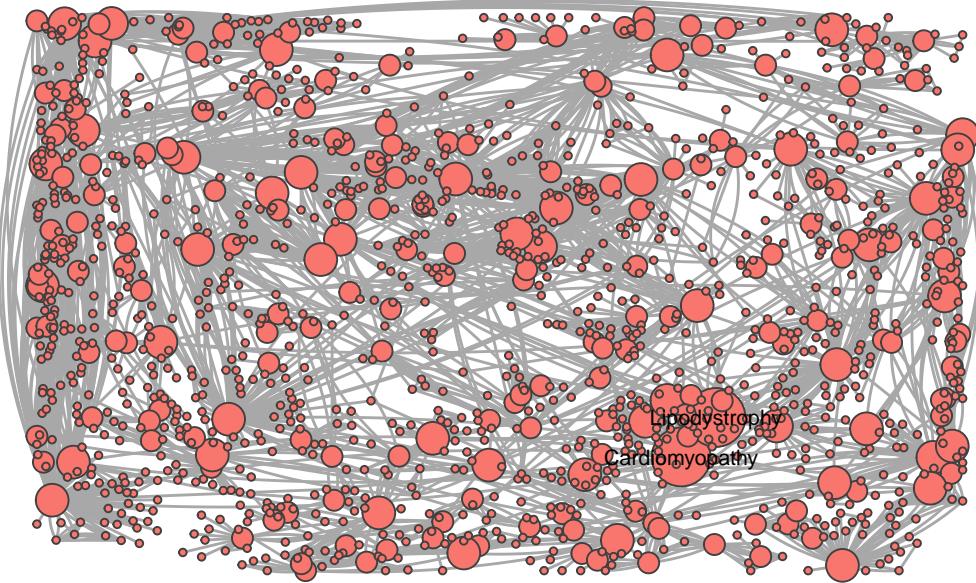
## [1] "Cardiomyopathy"      "Lipodystrophy"       "Diabetes mellitus"
## [4] "Glioblastoma"        "Deafness"

# "Cardiomyopathy" "Lipodystrophy"      "Diabetes mellitus" "Glioblastoma"      "Deafness"
```

Nodi con alta betweenness rappresentano gli hub all'interno della rete.

```
plot_graph(graph, "dh", betweenness, 0.14, 0.01, 0.0001, nodes$label, 0.14,
           "Human Disease Network Betweenness Centrality")
```

## Human Disease Network Betweenness Centrality

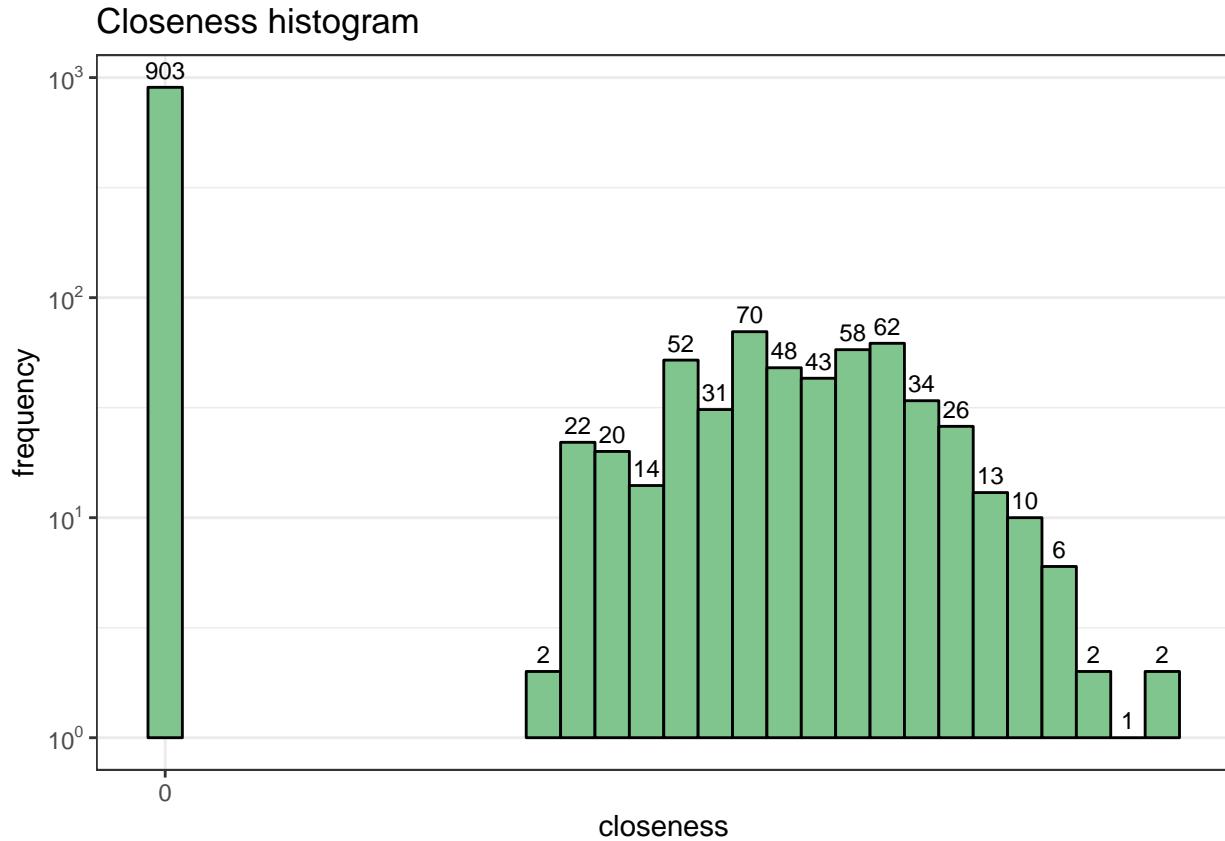


### 2.6 Closeness

La misura di closeness rappresenta l'efficienza di un nodo nello scambiare informazioni, ed è calcolata usando la seguente formula:

$$C_c(i) = \left[ \sum_{j=1}^n d(i,j) \right]^{-1}$$

```
closeness <- closeness(graph, v = V(graph), normalized = TRUE)
histogram_plot(closeness, c(0, 1), "closeness", "frequency", "Closeness histogram")
```



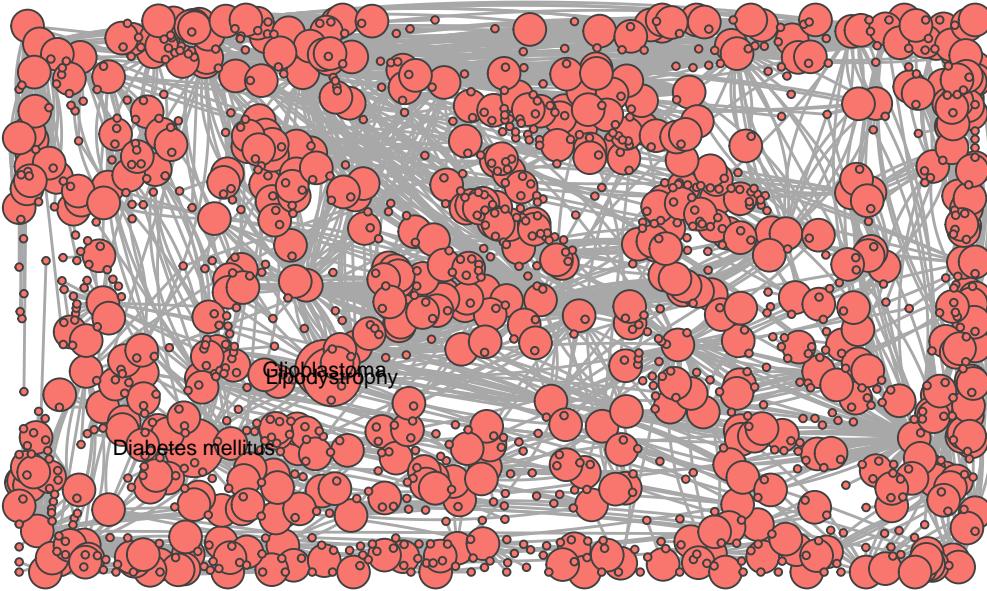
```
# node with the highest closeness
V(graph)$label[order(closeness, decreasing=TRUE)][1:5]

## [1] "Lipodystrophy"      "Diabetes mellitus" "Glioblastoma"
## [4] "Obesity"           "Cardiomyopathy"

# "Lipodystrophy" "Diabetes mellitus" "Glioblastoma" "Obesity" "Cardiomyopathy"

plot_graph(graph, "dh", closeness, 0.23, 0.09, 0.07, nodes$label, 0.23,
           "Human Disease Network Closeness Centrality")
```

## Human Disease Network Closeness Centrality



### 2.7 Eigenvector

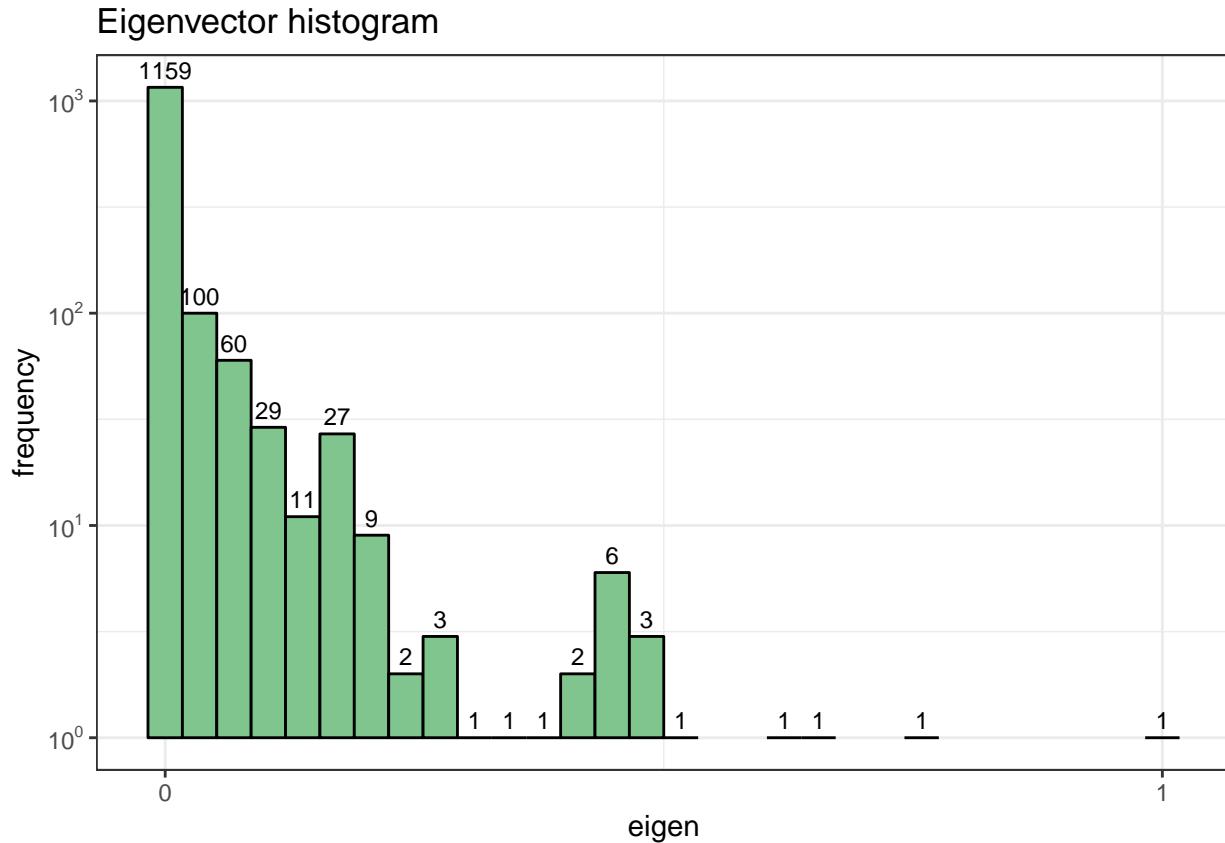
La centralità basata su autovettori rappresenta il concetto secondo il quale “un nodo è importante se è collegato ad altri nodi importanti” ed è calcolato tramite la seguente formula:

$$x_i = \frac{1}{\lambda} \sum_k a_{ki} x_k$$

con  $\lambda \neq 0$  e costante.

```
eigen <- eigen_centrality(graph, directed = TRUE)

histogram_plot(eigen$vector, c(0, 1), "eigen", "frequency", "Eigenvector histogram")
```



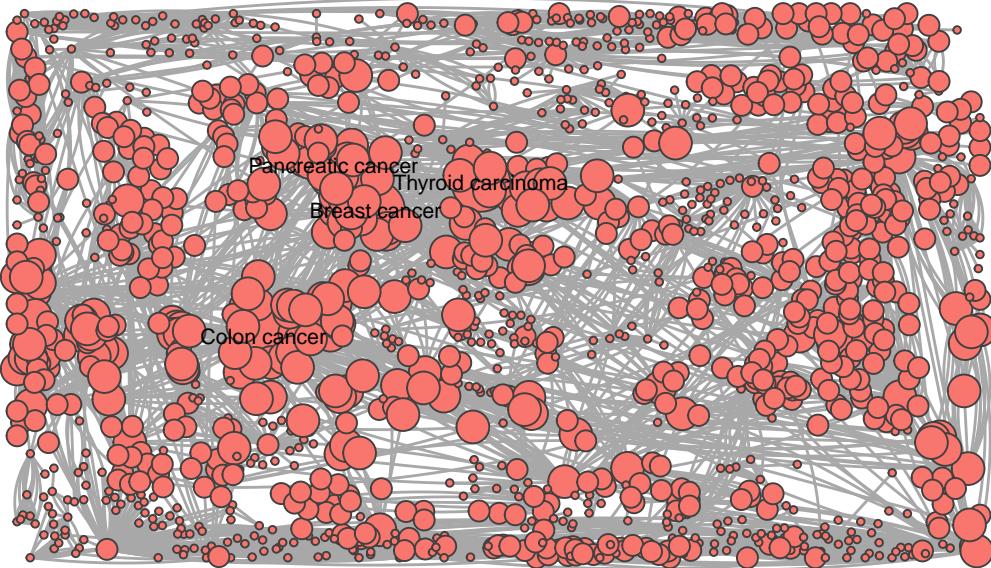
```
# node with the highest eigen
V(graph)$label[order(eigen$vector, decreasing=TRUE)][1:5]

## [1] "Colon cancer"      "Breast cancer"     "Thyroid carcinoma"
## [4] "Pancreatic cancer" "Hepatic adenoma"

# "Colon cancer"      "Breast cancer"     "Thyroid carcinoma" "Pancreatic cancer" "Hepatic adenoma"

plot_graph(graph, "dh", eigen$vector, 0.55, 0.04, 0.00005, nodes$label, 0.55,
           "Human Disease Network Eigenvector Centrality")
```

## Human Disease Network Eigenvector Centrality

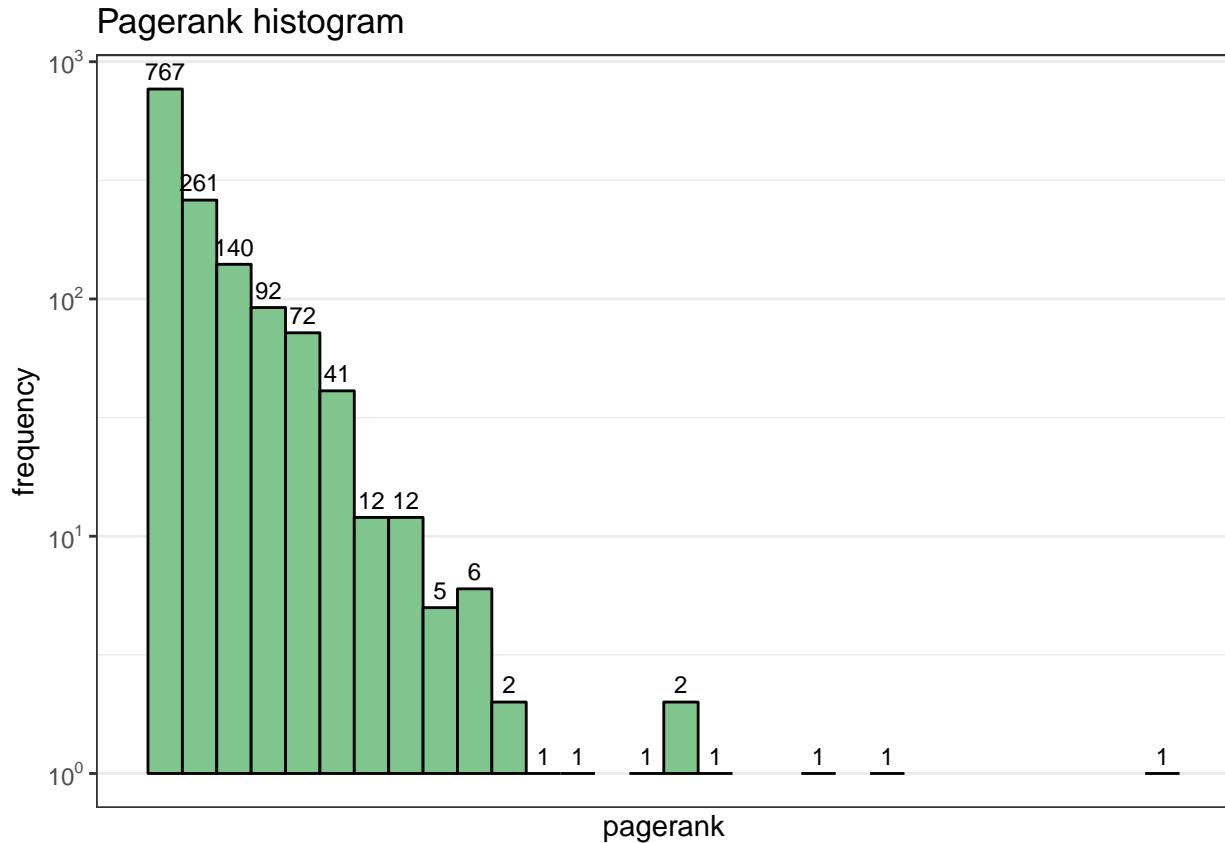


## 2.8 Pagerank

La misura di centralità basata su Pagerank risolve un problema particolare della misura precedente, per il quale se uno dei nodi della rete diventa una “autorità” allora passa tutta la sua centralità anche ai nodi vicini.

```
pagerank <- page_rank(graph, v = V(graph), directed = TRUE)
```

```
histogram_plot(pagerank$vector, c(0, 1), "pagerank", "frequency", "Pagerank histogram")
```

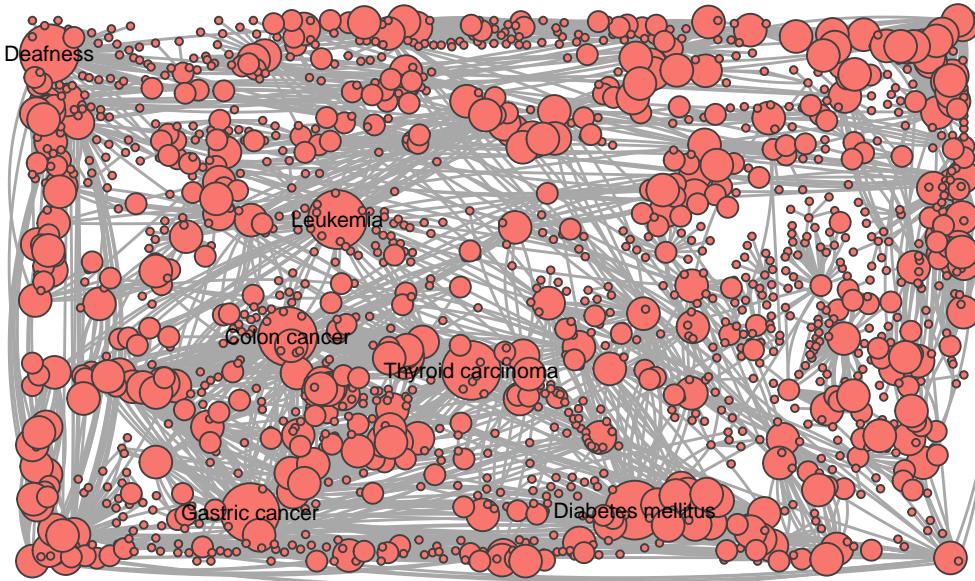


```
# node with the highest pagerank
V(graph)$label[order(pagerank$vector, decreasing=TRUE)][1:5]

## [1] "Colon cancer"      "Deafness"          "Leukemia"
## [4] "Diabetes mellitus" "Thyroid carcinoma"
# "Colon cancer" "Deafness" "Leukemia" "Diabetes mellitus" "Thyroid carcinoma"

plot_graph(graph, "dh", pagerank$vector, 0.003, 0.001, 0.0007, nodes$label, 0.003,
           "Human Disease Network Pagerank Centrality")
```

## Human Disease Network Pagerank Centrality



### 2.9 Coefficiente di clustering

Il **coefficiente di clustering** o transitività misura la probabilità che i vertici adiacenti di un vertice siano connessi.

Il coefficiente di clustering locale è una misura utile per valutare la densità locale della rete, il coefficiente globale invece ci dà un'idea su quanto sia clusterizzata la rete.

Più la rete è densa maggiore sarà il valore coefficiente di clustering. Questo accade in situazione con elevato numero di archi. La media può dare un'idea generale del grado di clustering.

```
global_transitivity <- transitivity(graph, type = "global", isolates = "zero")

local_transitivity <- transitivity(graph, type = "local", isolates = "zero")

transitivity <- data.frame(matrix(ncol = 1, nrow = 2))

colnames(transitivity) <- "Value"
rownames(transitivity) <- c("Network Local Transitivity",
                            "Network Global Transitivity")

transitivity[1,1] <- mean(local_transitivity)
transitivity[2,1] <- global_transitivity

pander(format(transitivity, digits = 3, justify="left"))
```

	Value
<b>Network Local Transitivity</b>	0.313
<b>Network Global Transitivity</b>	0.251

Vediamo che il valore del coefficiente di clustering globale è molto basso e questo può indicare che avremo

difficoltà nel valutare algoritmi di clustering su questa rete.

## 2.10 Grafo utilizzato per il clustering

### 2.10.1 Rimozione geni

Dalle analisi effettuate sulla rete è evidente come i geni all'interno della rete siano superflui, essendo le malattie collegate tra di loro se condividono almeno un gene, la presenza genica è ridondante e soprattutto aumenta enormemente il rischio di confondere i risultati degli algoritmi di community detection.

Abbiamo quindi deciso di lavorare sul grafo a cui vengono rimossi i geni per eseguire le valutazioni degli algoritmi.

### 2.10.2 Pesatura degli archi

La rete iniziale ha un attributo *weight* uguale ad uno per ogni arco, questo lo rende completamente inutile. Abbiamo deciso di pesare ogni arco dato il numero di geni condivisi dalle coppie di nodi connessi dall'arco in questione, questo rende il grafo sensatamente pesato e soprattutto ci da una motivazione ulteriore per il punto precedente.

### 2.10.3 Disorientamento del grafo

La rete iniziale è un grafo diretto che ha però una coppia di archi di uguale peso, per ogni malattia che condivide un gene. Abbiamo deciso di rimuovere uno dei due archi e disorientare il grafo, per ottenere un grafo esattamente identico, ma più facilmente gestibile e visualizzabile avendo dimezzato il numero di archi.

```
paste("N° di nodi del grafo finale: ", vcount(UD_nogenes))

## [1] "N° di nodi del grafo finale: 516"

paste("N° di archi del grafo finale: ", ecount(UD_nogenes))

## [1] "N° di archi del grafo finale: 1188"
```

## 2.11 Centrality Analytics

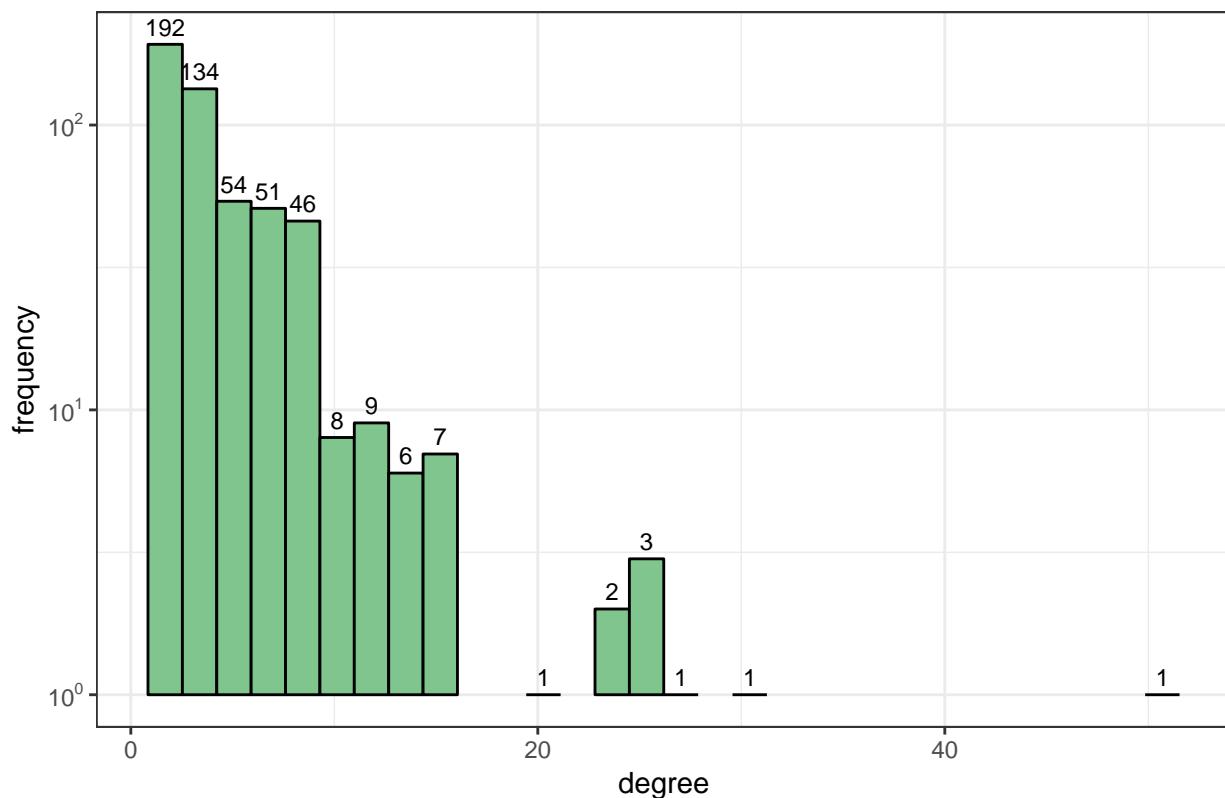
Ricalcoliamo le misure di centralità sul nuovo grafo.

### 2.11.1 Degree

```
UD_nogenes_degree <- centr_degree(UD_nogenes, mode = "all", normalized = TRUE)

histogram_plot(UD_nogenes_degree$res, seq(0, 160, by = 20), "degree", "frequency",
               "Degree histogram (nogenes)")
```

Degree histogram (nogenes)



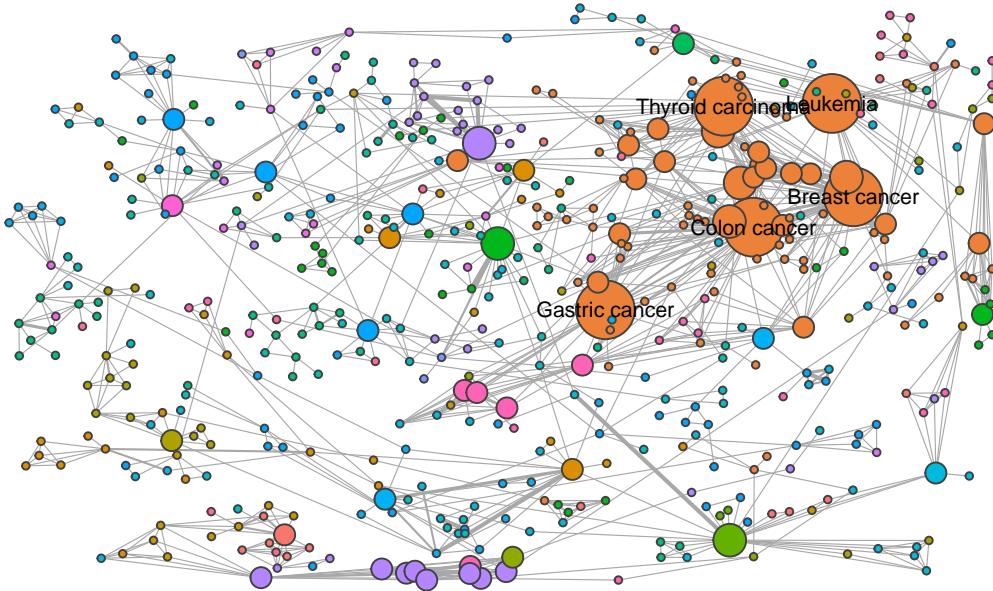
```
# nodes with the highest degree
V(graph)$label[order(UD_nogenes_degree$res, decreasing=TRUE)][1:5]

## [1] "Colon cancer"      "Breast cancer"     "Gastric cancer"
## [4] "Leukemia"          "Thyroid carcinoma"

# "Colon cancer", "Breast cancer", "Gastric cancer", "Leukemia", "Thyroid carcinoma"

plot_pretty_graph_centrality(UD_nogenes, "dh", UD_nogenes_degree$res, E(UD_nogenes)$weight,
                            25, 15, 8, nodes$label[nodes$X0 == "disease"], 25,
                            "Human Disease Network (nogenes) Degree Centrality")
```

## Human Disease Network (nogenes) Degree Central

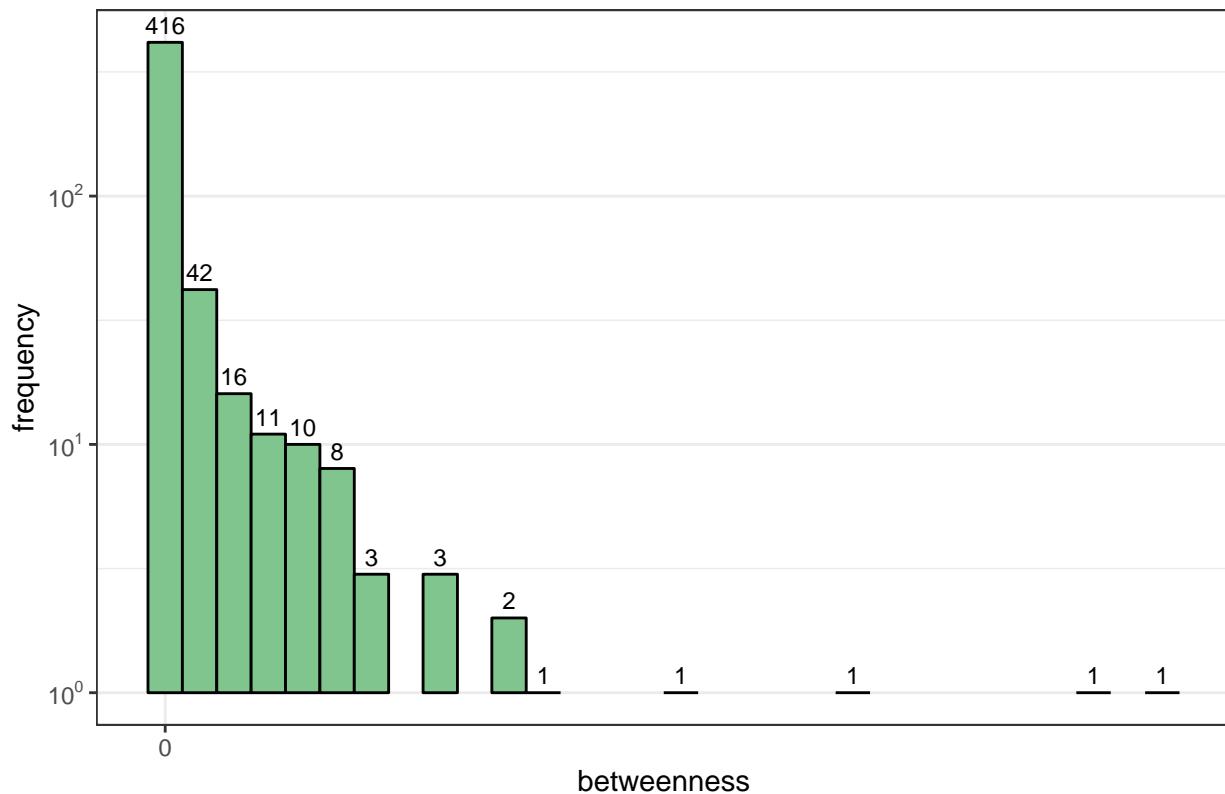


### 2.11.2 Betweenness

```
UD_nogenes_betweenness <- betweenness(UD_nogenes, v = V(UD_nogenes), directed = TRUE,
                                         normalized = TRUE, weights=E(UD_nogenes)$weight)

histogram_plot(UD_nogenes_betweenness, c(0, 1), "betweenness", "frequency",
               "Betweenness histogram (nogenes)")
```

Betweenness histogram (nogenes)



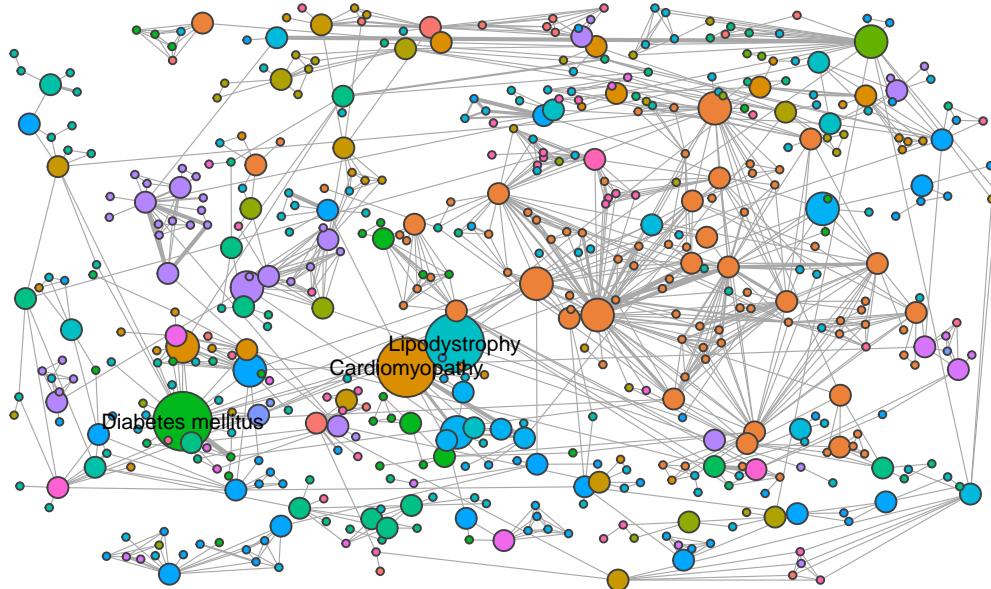
```
# nodes with the highest betweenness
V(graph)$label[order(UD_nogenes_betweenness, decreasing=TRUE)][1:5]

## [1] "Cardiomyopathy"      "Lipodystrophy"       "Diabetes mellitus"
## [4] "Glioblastoma"        "Myopathy"

# "Cardiomyopathy"      "Lipodystrophy"       "Diabetes mellitus" "Glioblastoma"      "Myopathy"

plot_pretty_graph_centrality(UD_nogenes, "dh", UD_nogenes_betweenness, E(UD_nogenes)$weight,
                            0.3, 0.1, 0.005, nodes$label[nodes$X0 == "disease"], 0.3,
                            "Human Disease Network (nogenes) Betweenness Centrality")
```

## Human Disease Network (nogenes) Betweenness C

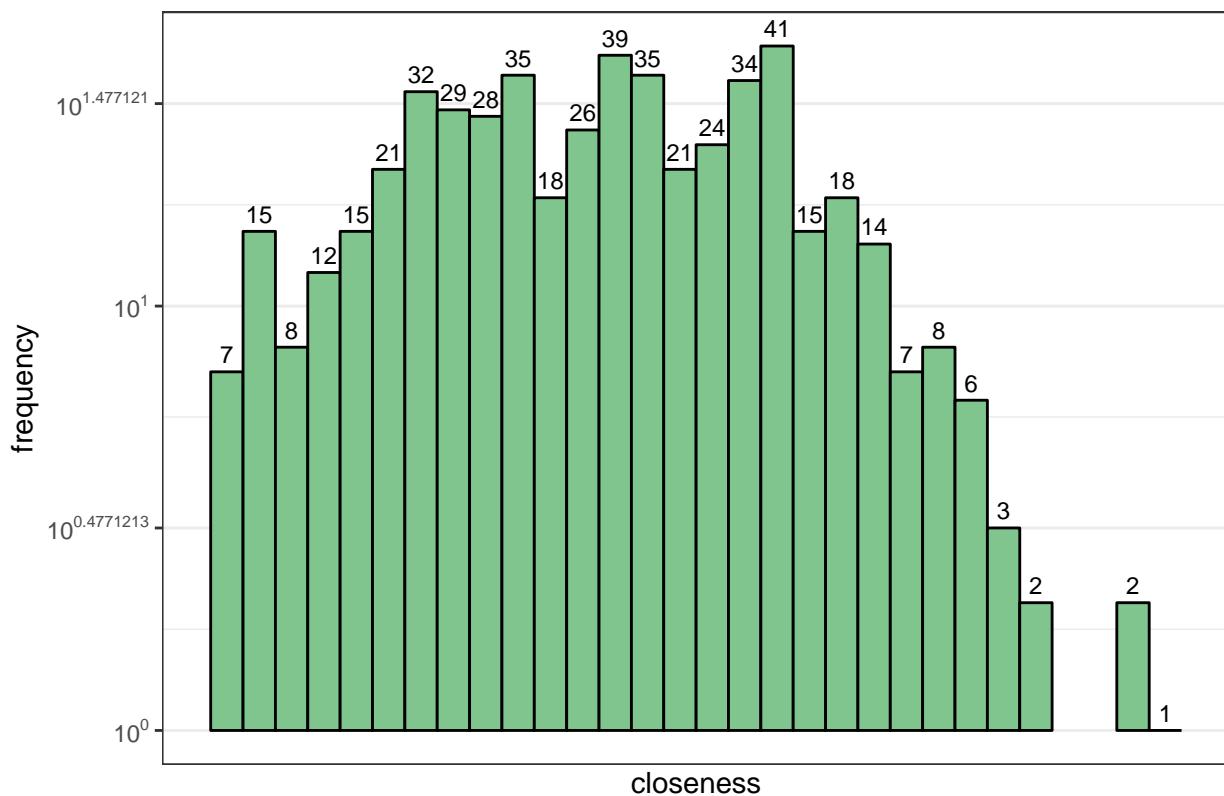


### 2.11.3 Closeness

```
UD_nogenes_closeness <- closeness(UD_nogenes, v = V(UD_nogenes), normalized = TRUE,
                                    weights=E(UD_nogenes)$weight)

histogram_plot(UD_nogenes_closeness, c(0, 1), "closeness", "frequency",
               "Closeness histogram (nogenes)")
```

Closeness histogram (nogenes)



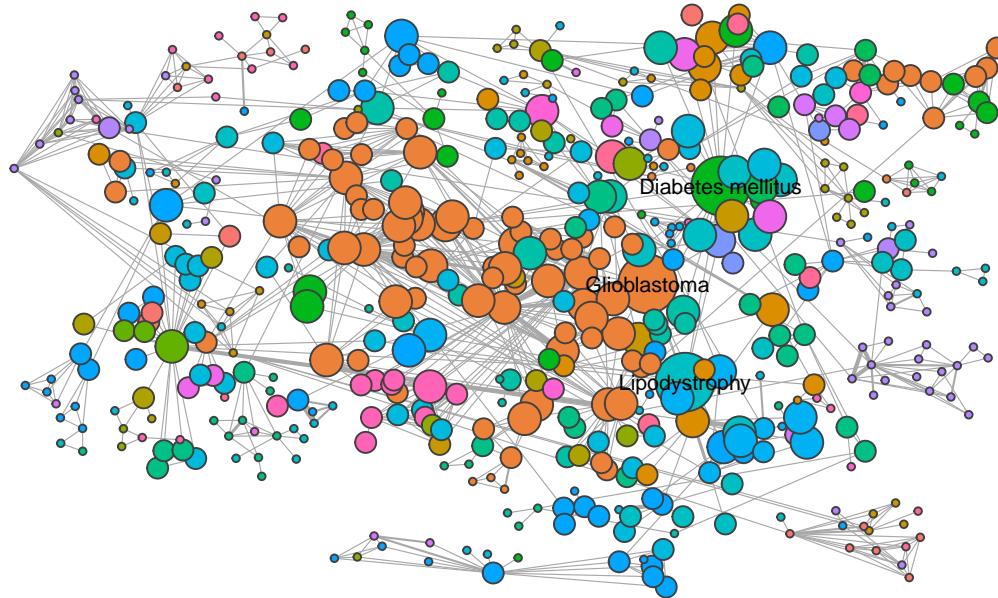
```
# nodes with the highest closeness
V(graph)$label[order(UD_nogenes_closeness, decreasing=TRUE)][1:5]

## [1] "Diabetes mellitus"   "Lipodystrophy"      "Glioblastoma"
## [4] "Cardiomyopathy"     "Insulin resistance"

# "Diabetes mellitus" "Lipodystrophy" "Glioblastoma" "Cardiomyopathy" "Insulin resistance"

plot_pretty_graph_centrality(UD_nogenes, "dh", UD_nogenes_closeness, E(UD_nogenes)$weight,
                            0.11, 0.09, 0.07, nodes$label[nodes$X0 == "disease"], 0.11,
                            "Human Disease Network (nogenes) Closeness Centrality")
```

## Human Disease Network (nogenes) Closeness Centrality

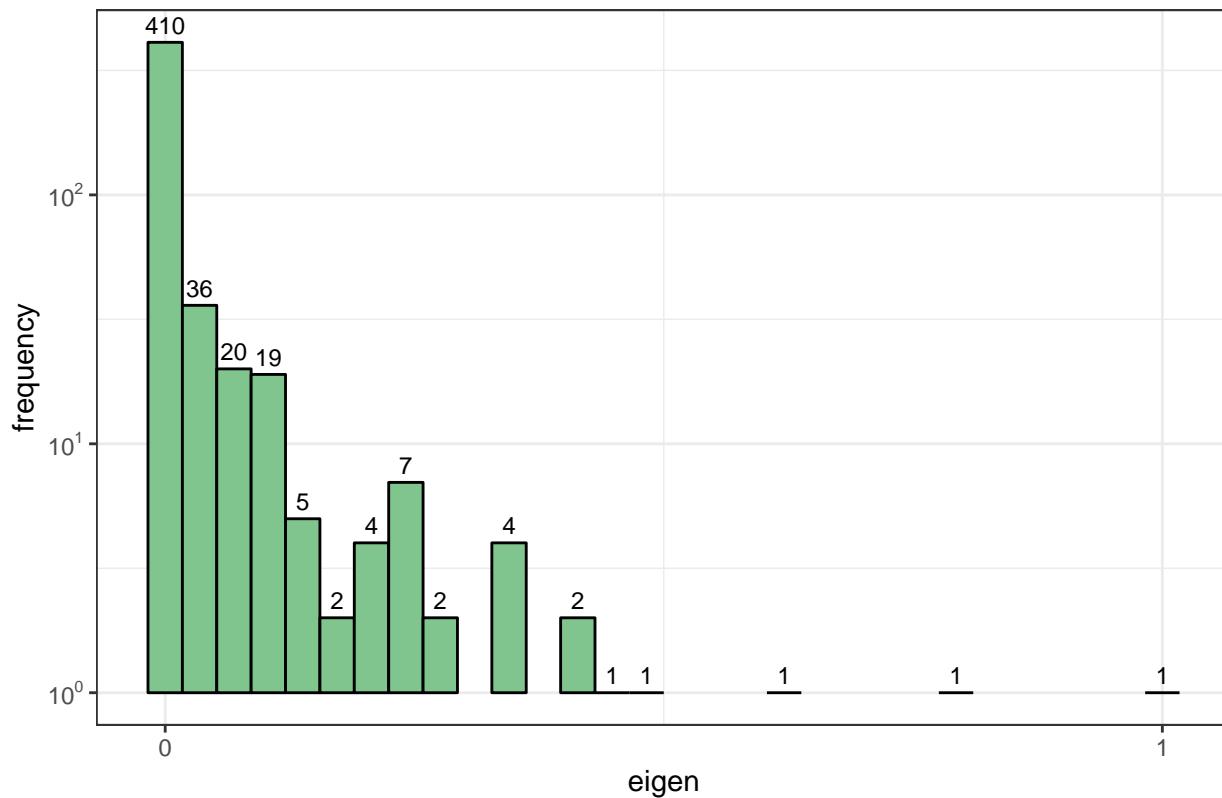


### 2.11.4 Eigenvector

```
UD_nogenes_eigen <- eigen_centrality(UD_nogenes, directed = TRUE, weights=E(UD_nogenes)$weight)

histogram_plot(UD_nogenes_eigen$vector, c(0, 1), "eigen", "frequency",
               "Eigenvector histogram (nogenes)")
```

### Eigenvector histogram (nogenes)

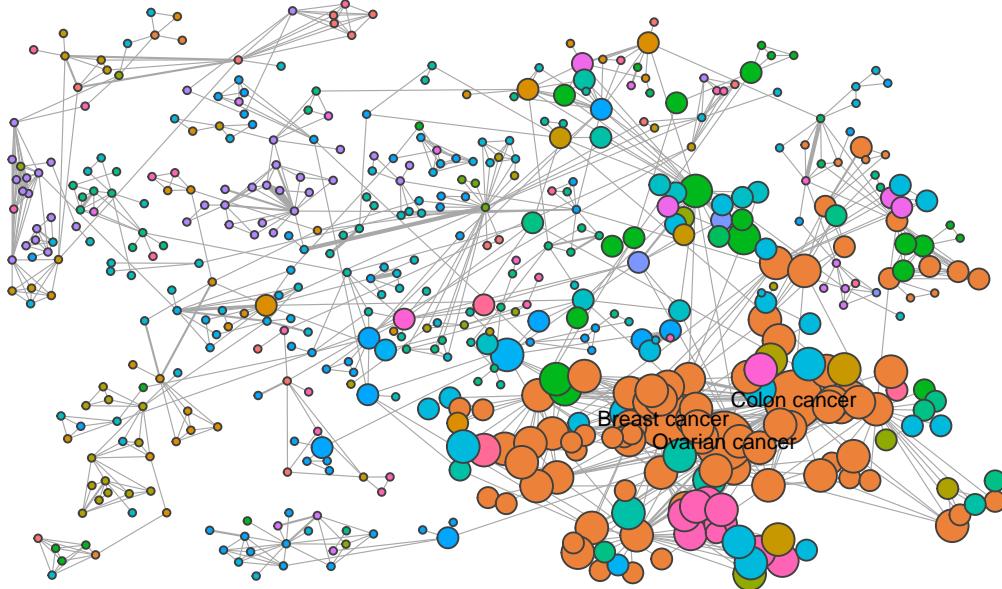


```
# nodes with the highest eigenvector
V(graph)$label[order(UD_nogenes_eigen$vector, decreasing=TRUE)][1:5]

## [1] "Colon cancer"      "Breast cancer"     "Ovarian cancer"
## [4] "Lymphoma"          "Pancreatic cancer"
#"Colon cancer" "Breast cancer" "Ovarian cancer" "Lymphoma" "Pancreatic cancer"

plot_pretty_graph_centrality(UD_nogenes, "dh", UD_nogenes_eigen$vector, E(UD_nogenes)$weight,
                            0.6, 0.04, 0.001, nodes$label[nodes$X0 == "disease"], 0.6,
                            "Human Disease Network (nogenes) Eigenvector Centrality")
```

# Human Disease Network (nogenes) Eigenvector Ce



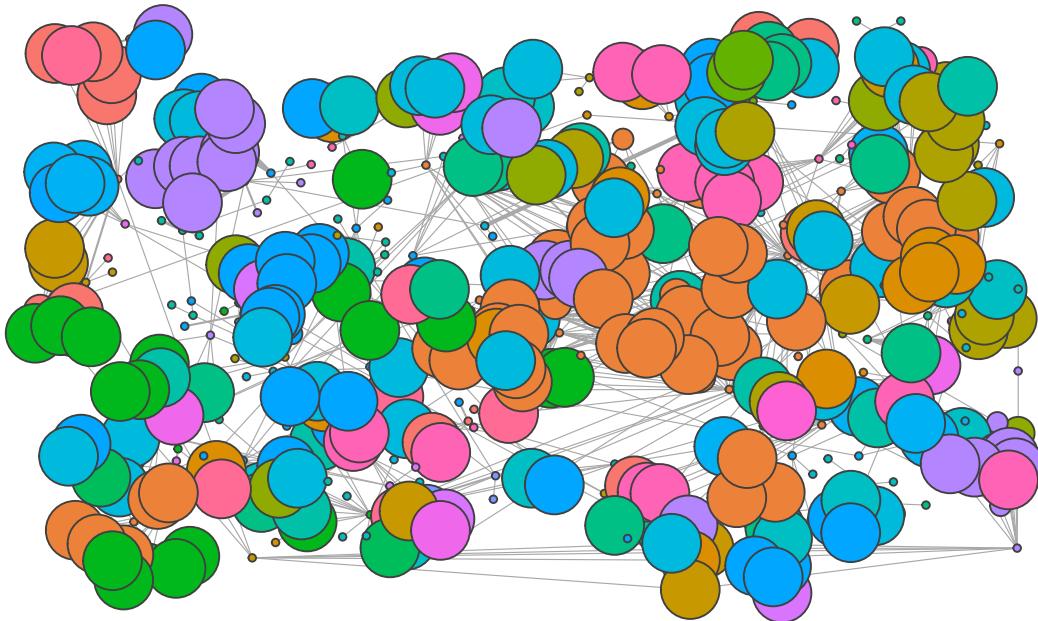
## 2.11.5 PageRank

Non calcoliamo PageRank in quanto non è una misura particolarmente esplicativa su grafi non diretti.

### 2.11.6 Coefficiente di clustering

La transitività misura la probabilità che i vertici adiacenti di un vertice siano connessi. Questa è talvolta chiamato anche **coefficiente di clustering**.

## Nogenes Weighted Clustering coeff



```
transitivity[1,1] <- mean(local_transitivity)
transitivity[2, 1] <- global_transitivity
transitivity[3,1] <- mean(weighted_transitivity)

pander(format(transitivity, digits = 3, justify="left"))
```

	Value
<b>Network Local Transitivity</b>	0.636
<b>Network Global Transitivity</b>	0.430
<b>Network Weighted Transitivity</b>	0.641

Le misure sono quasi raddoppiate rispetto alla controparte genica, questo ci fa pensare che gli accorgimenti addottati per migliorare la rete siano corretti.

# Clustering

## 3.1 Operazioni preliminari

Cominciamo caricando le librerie necessarie ed importando il grafo:

```
# source a set of functions
source("util.R")

# import libraries
libraries_list <- c("ggraph", "igraph", "dplyr", "readr", "DiagrammeR", "tidyverse",
                     "Cairo", "networkD3", "CINNA", "scales", "gridExtra", "leiden",
                     "pander", "MCL", "caret", "aricode")
import_libraries(libraries_list)

# import graph
edges <- read.csv(
  "../dataset/diseasome [Edges].csv",
  head = TRUE
)

nodes <- read.csv(
  "../dataset/diseasome [Nodes].csv",
  head = TRUE
)

nodes <- nodes %>% select(-timeset)
edges <- edges %>% select(-timeset, -label)

weight <- readRDS("edgeweights.rds")
edges$weight <- weight

graph <- graph.data.frame(edges, directed = FALSE, vertices = nodes)

UD_nogenes <- induced_subgraph(graph, which(nodes$X1 != "gene"))

# Rimuovo i doppi nodi
UD_nogenes <- igraph::simplify(UD_nogenes)
```

## 3.2 Groundtruth

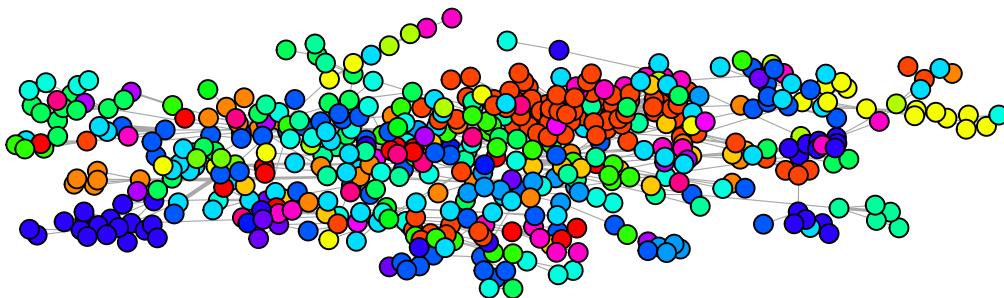
Le malattie nel grafo sono clusterizzate in macrogruppi medici (cancro, cardiovascolare, etc...), verifichiamo se questi cluster si trovano effettivamente vicini all'interno della rete.

```
disease_clusters <- unique(V(UD_nogenes)$X1)

i = 1
for (disease in disease_clusters)
{
  V(UD_nogenes)[V(UD_nogenes)$X1 == disease]$color = i
  i = i + 1
}

plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes$X1[nodes$X0 == "disease"],
                        E(UD_nogenes)$weight, "Human Disease Network Clusters")
```

## Human Disease Network Clusters

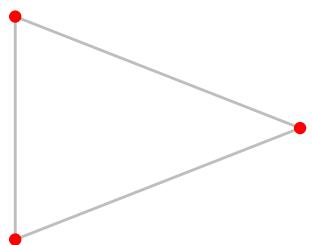
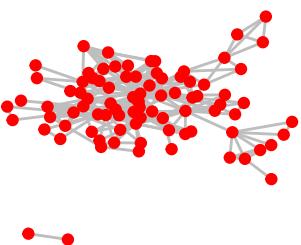
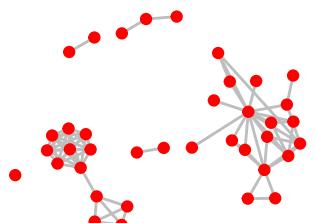
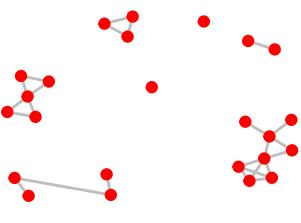
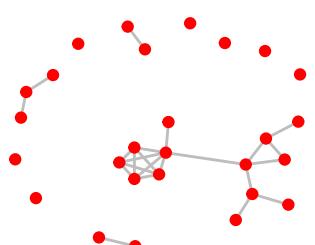
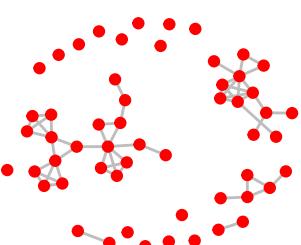
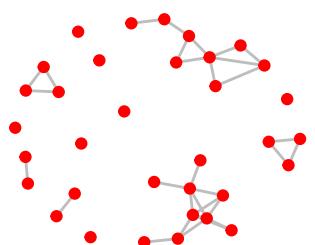
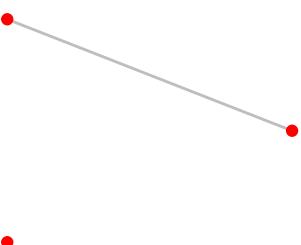


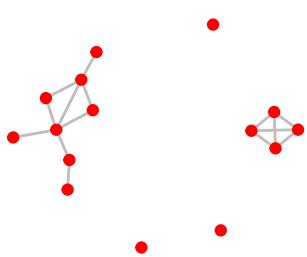
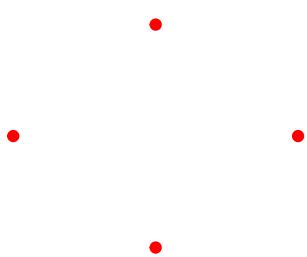
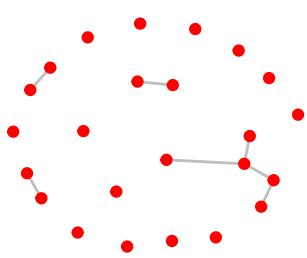
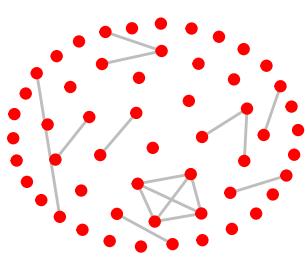
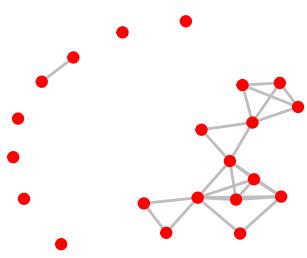
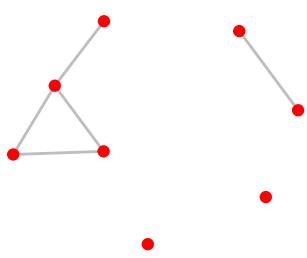
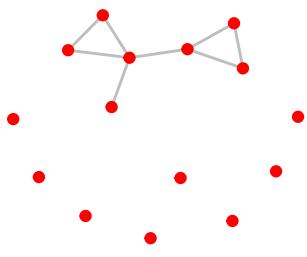
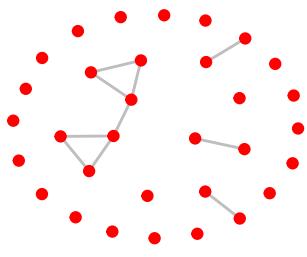
### Clusters

Bone	Developmental	Immunological	Nutritional	Skeletal
Cancer	Ear,Nose,Throat	Metabolic	Ophthalmological	Unclassified
Cardiovascular	Endocrine	Multiple	Psychiatric	
Connective tissue disorder	Gastrointestinal	Muscular	Renal	
Dermatological	Hematological	Neurological	Respiratory	

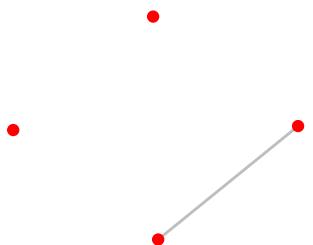
Dal grafo risultante vediamo un buon numero di nodi vicini tra di loro che condividono il tipo di malattia. Vediamo però anche un gran numero di nodi che sono lontani dal cluster di appartenenza!

Proviamo a plottare le componenti connesse che condividono la stessa malattia:

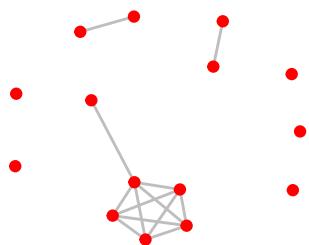
**Ear,Nose,Throat****Cancer****Ophthalmological****Endocrine****Cardiovascular****Neurological****Hematological****Nutritional**

**Muscular****Respiratory****Immunological****Multiple****Dermatological****Psychiatric****Connective tissue****Metabolic**

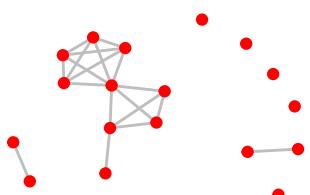
## Gastrointestinal



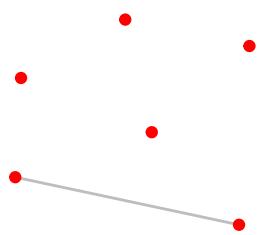
## Bone



## Skeletal



## Renal



## Developmental



## Unclassified

Notiamo subito che non tutte le malattie formano un cluster, questo potrebbe essere molto problematico in quanto, non avendo attributi sui nodi, possiamo solo limitarci a raggruppare nodi che sono vicini all'interno della rete.

Vediamo quante componenti connesse abbiamo per ogni cluster:

```
n.cluster <- 0
n.cluster.single <- 0

clusters <- list()
cont.cluster <- 1

for (g in graphs)
{
  component <- components(g)
  n.cluster = n.cluster + length(component$csizes[component$csizes > 0])
  n.cluster.single = n.cluster.single + length(component$csizes[component$csizes == 1])
```

```
for (i in 1:length(component$csize))
{
  comp <- component$membership[component$membership == i]
  cont.cluster = cont.cluster + 1
}
print(n.cluster)

## [1] 225
print(n.cluster.single)

## [1] 161
```

L'algoritmo ottimale dovrebbe essere in grado di distinguere 225 cluster basandosi unicamente sulla locazione del nodo all'interno della rete.

In particolare abbiamo 161 nodi i cui vicini non rappresentano la stessa malattia!

### 3.3 Algoritmi di clustering

Gli algoritmi utilizzati si basano principalmente sul concetto di community detection e graph partitioning, non avendo a disposizione attributi sui nodi è praticamente impossibile pensare di operare con algoritmi di clustering tradizionali.

Iniziamo costruendo una matrice utile per confrontare il cluster originale di ogni nodo con quello assegnatogli da uno degli algoritmi utilizzati:

```
nodes_results <- nodes %>% filter(X1 != "gene") %>% select(-X0)
nodes_results$Betweenness <- NA
nodes_results$Fastgreedy <- NA
nodes_results$Louvain <- NA
nodes_results$Spinglass <- NA
nodes_results$Markov <- NA
nodes_results$Leiden <- NA
nodes_results$Label_prop <- NA
nodes_results$Label_prop_init <- NA
nodes_results$Lead_eigen <- NA
```

#### 3.3.1 Regola di clustering

Ogni cluster viene etichettato utilizzando la label più frequente al suo interno:

```
label_cluster <- function(clusters)
{
  cluster_df <-
    as.data.frame(matrix(
      1:length(clusters),
      nrow = length(clusters),
      dimnames = list(NULL, "id")
    ))

  for (c in 1:length(clusters))
  {
    labels <- V(UD_nogenes)[clusters[[c]]]$X1
    labels <- labels[which(labels != "Multiple" & labels != "Unclassified")]

    if (length(labels) != 0)
    {
      value <- which.max(unlist(table(labels)))
      cluster_df$name[cluster_df$id == c] <- names(value)[1]
    } else
    {
      cluster_df$name[cluster_df$id == c] <- V(UD_nogenes)[clusters[[c]]]$X1[1]
    }
  }

  return(cluster_df)
}
```

### 3.4 Girvan–Newman

Il primo algoritmo di clustering che abbiamo utilizzato appartiene alla famiglia degli algoritmo **Hierarchy-centric community**. Esso è basato sull'utilizzo della misura di edge betweenness.

L'algoritmo di Girvan-Newman (2004) si fonda sull'idea che gli archi che connettono moduli separati della rete dovrebbero avere un alto valore di edge betweenness.

```
betweenness.communities <- cluster_edge_betweenness(UD_nogenes, directed = FALSE,
                                                    weights = E(UD_nogenes)$weight)

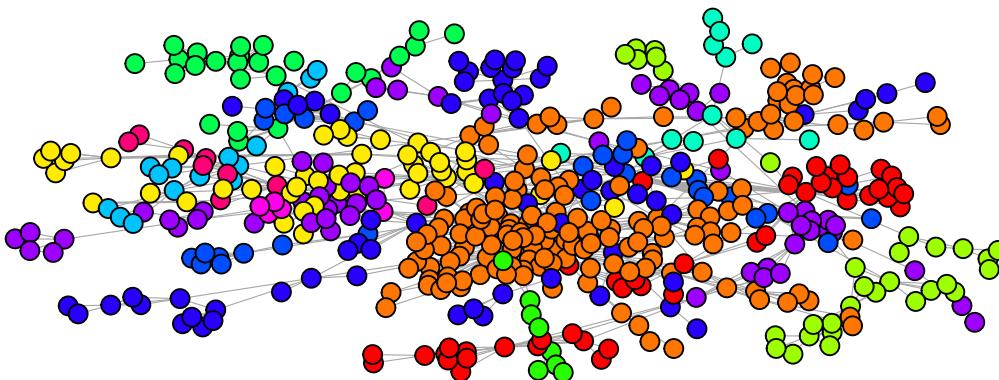
print(paste("N° di communities: ", max(betweenness.communities$membership)))

## [1] "N° di communities: 29"
```

Questo algoritmo distingue 29 cluster all'interno della rete. Ci aspettiamo quindi che i risultati in termini di purezza non siano particolarmente elevati in quanto non è ovviamente in grado di distinguere, ad esempio, i cluster di dimensione uno.

```
plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes_results$Betweenness,
                        E(UD_nogenes)$weight, "Girvan Newman Clusters")
```

### Girvan Newman Clusters



Clusters

- |                  |                  |                 |                    |            |
|------------------|------------------|-----------------|--------------------|------------|
| ● Bone           | ● Dermatological | ● Immunological | ● Neurological     | ● Skeletal |
| ● Cancer         | ● Endocrine      | ● Metabolic     | ● Ophthalmological |            |
| ● Cardiovascular | ● Hematological  | ● Muscular      | ● Renal            |            |

### 3.5 Fastgreedy

L'algoritmo fastgreedy (2004) cerca le communities all'interno della rete andando a massimizzare il valore di modularità, utilizzando un approccio bottom-up, rispetto a quello top-down utilizzato da Girvan–Newman. Inizialmente ogni nodo rappresenta una community e, iterativamente, ogni nodo viene unito in modo tale che l'unione sia localmente ottima (massimo aumento della modularità). L'algoritmo si ferma quando non è più possibile far aumentare la modularità.

```
fgreedy.communities <- cluster_fast_greedy(UD_nogenes, modularity = TRUE,
                                             weights = E(UD_nogenes)$weight)

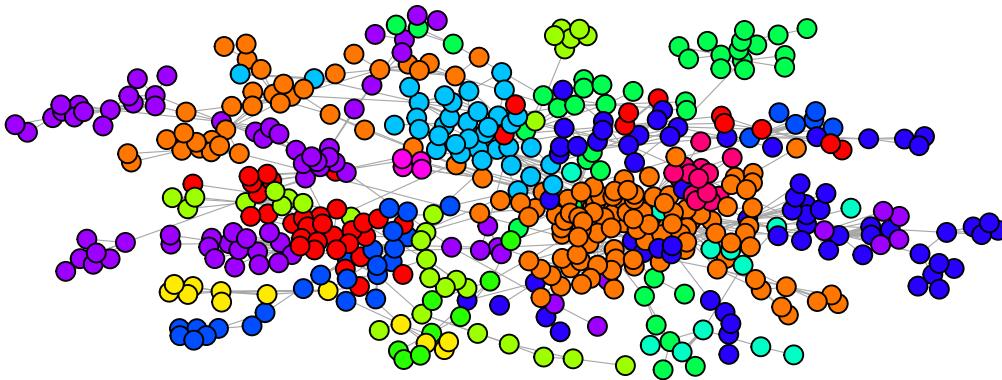
print(paste("N° di communities: ", max(fgreedy.communities$membership)))
```

```
## [1] "N° di communities: 26"
```

Notiamo che il risultato è simile a quello prodotto da Girvan–Newman, ma il numero di cluster è ancora minore e quindi molto probabilmente anche le performance saranno più basse.

```
plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes_results$Fastgreedy,
                         E(UD_nogenes)$weight, "Fastgreedy Clusters")
```

### Fastgreedy Clusters



#### Clusters

● Bone	● Dermatological	● Immunological	● Neurological	● Skeletal
● Cancer	● Endocrine	● Metabolic	● Ophthalmological	
● Cardiovascular	● Hematological	● Muscular	● Renal	

### 3.6 Louvain

L'algoritmo denominato Louvain (2008) possiamo vederlo come una evoluzione del precedente. Anche in questo caso l'obiettivo è quello di massimizzare la modularità, vengono seguiti gli stessi step dell'algoritmo precedente, dopo di che viene creato un grafo i cui nodi sono le community create e si ripete il procedimento fino a quando non è più possibile far aumentare la modularità:

```
louvain.communities <- cluster_louvain(UD_nogenes, weights = E(UD_nogenes)$weight)
```

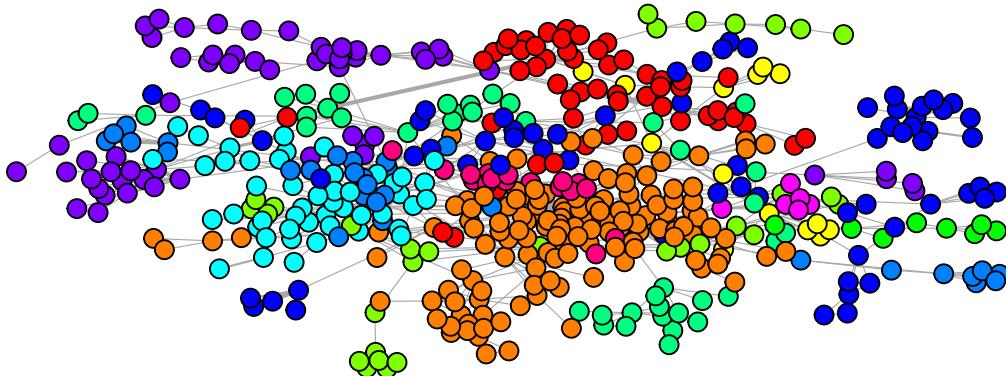
```
print(paste("N° di communities: ", max(louvain.communities$membership)))
```

```
## [1] "N° di communities: 27"
```

Anche in questo caso non ci aspettiamo che le performance differiscano particolarmente dagli algoritmi precedenti.

```
plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes_results$Louvain,
E(UD_nogenes)$weight, "Louvain Clusters")
```

## Louvain Clusters



Clusters

- |                  |                  |                 |                    |
|------------------|------------------|-----------------|--------------------|
| ● Bone           | ● Dermatological | ● Immunological | ● Ophthalmological |
| ● Cancer         | ● Endocrine      | ● Muscular      | ● Renal            |
| ● Cardiovascular | ● Hematological  | ● Neurological  | ● Skeletal         |

### 3.7 Spinglass

Questo algoritmo (2006) fa utilizzo di tecniche derivate dalla statistica fisica per costruire le community. Permette di specificare il parametro spins che idealmente rappresenta il numero k di cluster, il problema è che l'algoritmo cerca di riempire tutti i cluster, ma è possibile che non ci riesca e quindi il risultato mostrerà un numero di cluster molto ridotto. I risultati dell'algoritmo differiscono in base all'inizializzazione delle etichette, per questo motivo il processo di assegnazione delle etichette è stato ripetuto 100 volte per ottenere una stima più affidabile.

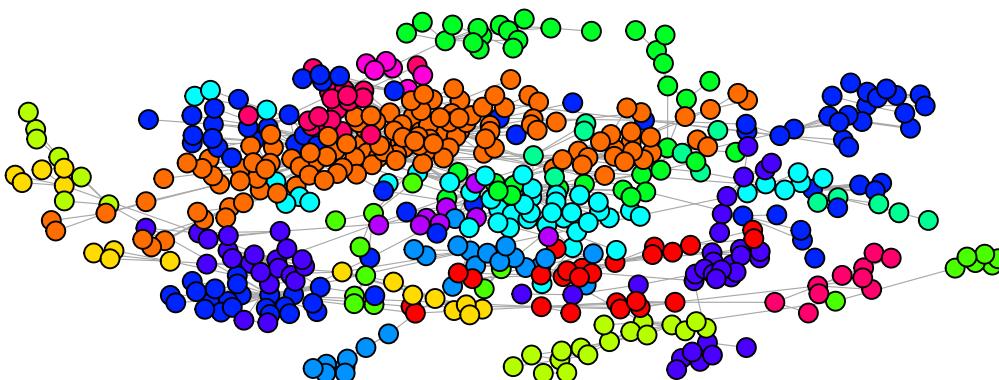
```
# 100 iterazioni
iteration_df <- readRDS("spinglass_iterations.rds")

for (n in 1:nrow(iteration_df))
{
  nodes_results$Spinglass[n] <- Mode(iteration_df[n, ])[[1]]
}
```

Il numero di cluster costruiti varia in base all'inizializzazione ed è circa 40.

```
plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes_results$Spinglass,
                        E(UD_nogenes)$weight, "Spinglass Clusters")
```

### Spinglass Clusters



Clusters

● Bone	● Dermatological	● Immunological	● Neurological	● Renal
● Cancer	● Endocrine	● Metabolic	● Ophthalmological	● Skeletal
● Cardiovascular	● Hematological	● Muscular	● Psychiatric	

### 3.8 Markov Cluster Algorithm

L'algoritmo MCL (2002) si basa sulla simulazione di percorsi stocastici sul grafo, sfruttando il paradigma di clustering secondo il quale le community hanno la seguente proprietà:

“Un cammino randomico su un grafo G che visita un cluster denso molto probabilmente uscirà dal cluster solamente dopo aver attraversato buona parte dei suoi vertici”. Il parametro inflation permette di aumentare la granularità, questo permette all'algoritmo di rilevare cluster più piccoli e quindi aumentare il numero di cluster rilevati.

```

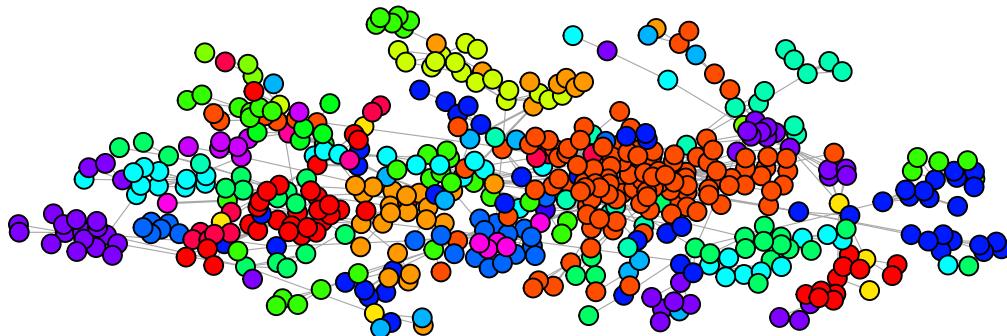
adjmat <- as_adj(UD_nogenes, type = "both", attr = "weight")
mcl.communities <- mcl(adjmat, addLoops = FALSE, inflation = 4, allow1 = TRUE)

print(paste("N° di communities: ", mcl.communities$K))

## [1] "N° di communities: 169"
plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes_results$Markov,
                         E(UD_nogenes)$weight, "Markov Clusters")

```

## Markov Clusters



### Clusters

● Bone	● Dermatological	● Hematological	● Muscular	● Psychiatric
● Cancer	● Developmental	● Immunological	● Neurological	● Renal
● Cardiovascular	● Endocrine	● Metabolic	● Nutritional	● Respiratory
● Connective tissue disorder	● Gastrointestinal	● Multiple	● Ophthalmological	● Skeletal

L'algoritmo rileva ben 169 communities, l'accuratezza ci dimostrerà se sono sensate, oppure i raggruppamenti sono molto randomici.

### 3.9 Leiden Algorithm

L'algoritmo Leiden (2019) è un ulteriore evoluzione dell'algoritmo Louvain mostrato in precedenza. I creatori dell'algoritmo hanno dimostrato come Louvain generi spesso delle community che non sono ottimali, inoltre questo algoritmo converge ad una soluzione ottima molto più rapidamente.

In particolare con questo algoritmo è possibile decidere a priori il numero di cluster e per questo motivo dovrebbe essere in grado di ottenere performance molto buone sul grafo in questione.

```
adjmat <- as_adj(UD_nogenes, type = "both", attr = "weight")
leiden.communities <- leiden(adjmat, resolution_parameter = 86)
print(max(leiden.communities))

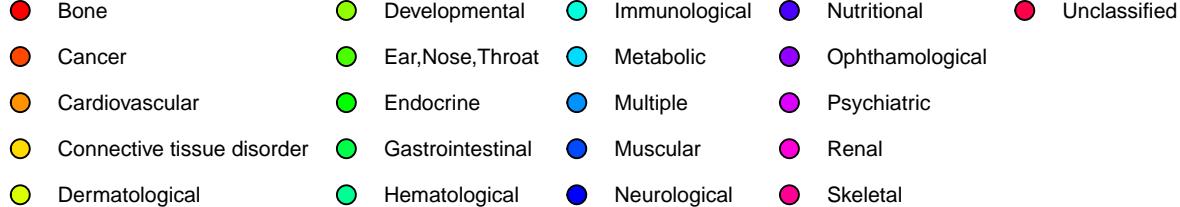
## [1] 225
leiden.communities <- make_clusters(UD_nogenes, membership = leiden.communities)
```

Settando il parametro di resolution è possibile far generare un numero di cluster pari al numero di componenti connesse distinte per malattia nel nostro grafo.

```
plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes_results$Leiden,
E(UD_nogenes)$weight, "Leiden Clusters")
```

### Leiden Clusters

Clusters



### 3.10 Label propagation

L'algoritmo di label propagation (2007) inizializza tutti i nodi con una label random e poi iterativamente aggiorna la label di ogni nodo basandosi su una votazione a maggioranza tra le label dei vicini. Il risultato dell'algoritmo dipende dall'inizializzazione, iteriamo quindi il procedimento 1000 volte per ottenere delle stime consistenti.

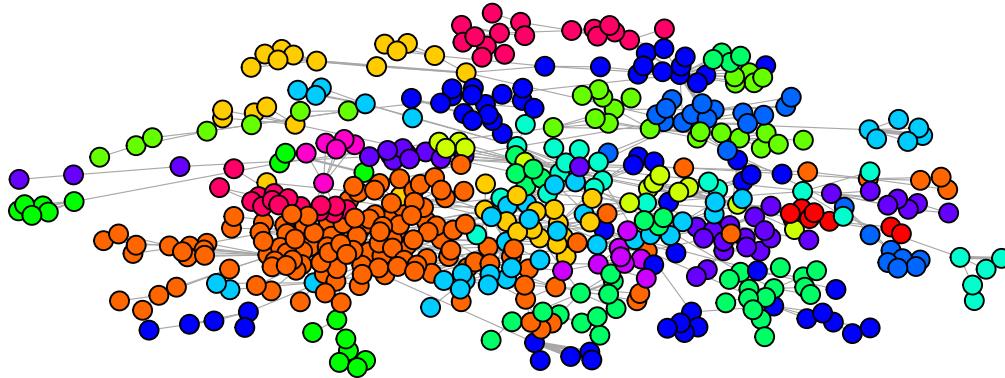
```
# 1000 iterazioni
iteration_df <- readRDS("label_prop_iterations.rds")

for (n in 1:nrow(iteration_df))
{
  nodes_results$Label_prop[n] <- Mode(iteration_df[n, ])[[1]]
}
```

L'algoritmo è molto semplice ed estremamente rapido, inoltre rileva ben 62 community che potrebbero essere indice di performance migliori.

```
plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes_results$Label_prop,
                         E(UD_nogenes)$weight, "Label Propagation Clusters")
```

## Label Propagation Clusters



### Clusters

- |                  |                              |                 |                    |               |
|------------------|------------------------------|-----------------|--------------------|---------------|
| ● Bone           | ● Connective tissue disorder | ● Hematological | ● Muscular         | ● Psychiatric |
| ● Cancer         | ● Dermatological             | ● Immunological | ● Neurological     | ● Renal       |
| ● Cardiovascular | ● Endocrine                  | ● Metabolic     | ● Ophthalmological | ● Skeletal    |

Viene poi testato l'algoritmo utilizzando una tecnica semi-supervised, inizializzando circa il 20% delle label e vedendo come si comporta.

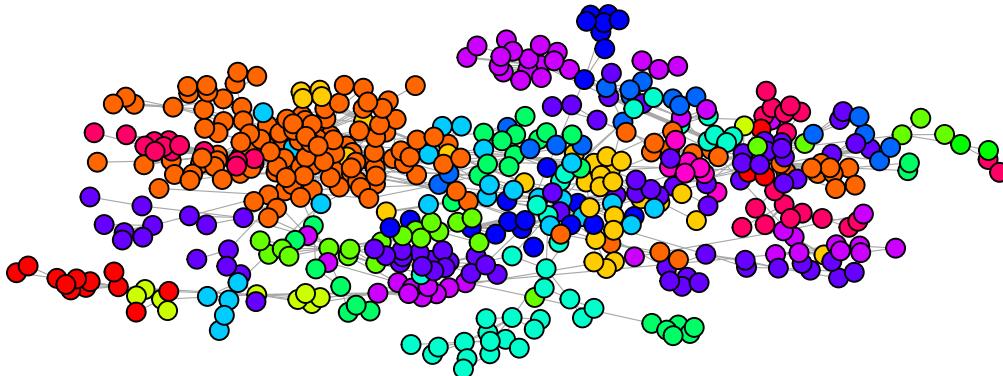
```
# 1000 iterazioni
iteration_df <- readRDS("label_prop_init_iterations.rds")

for (n in 1:nrow(iteration_df))
{
  nodes_results$Label_prop_init[n] <- Mode(iteration_df[n, ])[[1]]
}
```

Avendo inizializzato le label ci aspettiamo un buon aumento delle performance.

```
plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes_results$Label_prop_init,
E(UD_nogenes)$weight, "Label Propagation init Clusters")
```

## Label Propagation init Clusters



Clusters

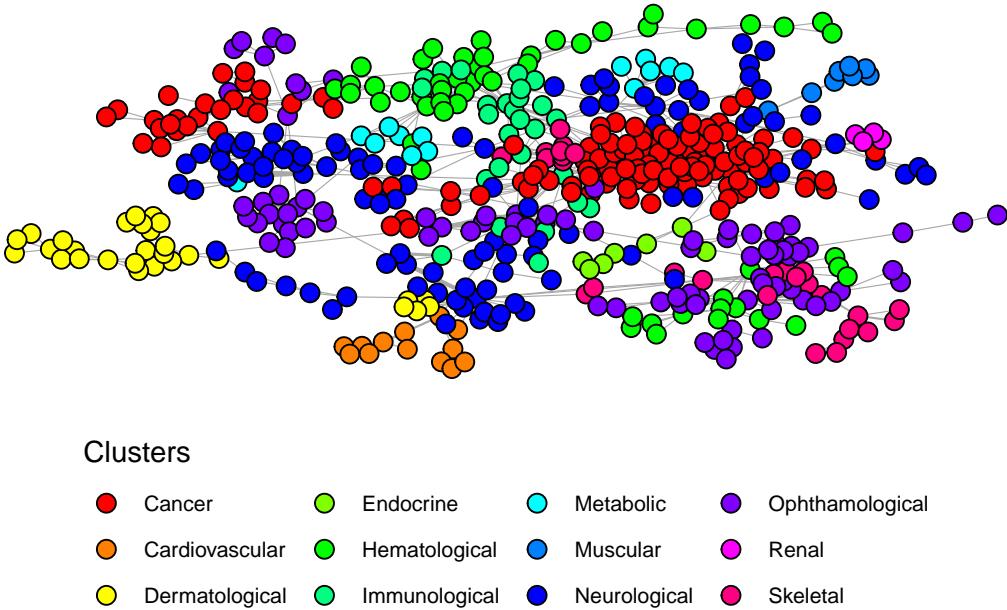
- |                  |                              |                 |                |                    |
|------------------|------------------------------|-----------------|----------------|--------------------|
| ● Bone           | ● Connective tissue disorder | ● Endocrine     | ● Metabolic    | ● Ophthalmological |
| ● Cancer         | ● Dermatological             | ● Hematological | ● Muscular     | ● Psychiatric      |
| ● Cardiovascular | ● Developmental              | ● Immunological | ● Neurological | ● Skeletal         |

### 3.11 Leading eigenvector

L'algoritmo in questione (2006) si basa sull'utilizzo di clustering divisivo con l'obiettivo di massimizzare la modularità. Ad ogni step separa il grafo in due componenti in modo tale che la separazione vada ad incrementare la modularità. La separazione è determinata valutando l'autovettore principale della matrice di modularità  $B = A - P$ .

```
lead_eigen.communities <- cluster_leading_eigen(UD_nogenes, weights = E(UD_nogenes)$weight)
print(paste("N° di communities: ", max(lead_eigen.communities$membership)))
## [1] "N° di communities: 27"
plot_pretty_graph_legend(UD_nogenes, "graphopt", nodes_results$Lead_eigen,
E(UD_nogenes)$weight, "Leading eigenvector Clusters")
```

## Leading eigenvector Clusters



### 3.12 Valutazione degli algoritmi

Le performance vengono misurate tramite valutazione con groundtruth.

#### 3.12.1 Accuracy

Valutiamo gli algoritmi in termini di accuratezza di classificazione dei nodi.

##### 3.12.1.1 Purity

La **purity** è un identificatore che indica quanto bene una community può rappresentare un'intera classe, calcolata osservando le etichette delle istanze. Essa viene calcolata come segue:

$$\frac{1}{N} \sum_{i=1}^k \max_j |C_i \cap L_j|$$

Abbiamo deciso di considerare i nodi con etichetta “Multiple” e “Unclassified”, come dei “Jolly”, facendo in modo che risultino sempre veri positivi.

```
compute_confmat <- function(groundtruth, communities.algo)
{
  l.union <- union(groundtruth, communities.algo)

  groundtruth[groundtruth == "Multiple"] <- communities.algo[groundtruth == "Multiple"]
  groundtruth[groundtruth == "Unclassified"] <- communities.algo[groundtruth == "Unclassified"]

  atable <-
    table(
```

```

    factor(groundtruth, l.union),
    factor(communities.algo, l.union)
  )

  return(confusionMatrix(atable))
}

purity_df <- data.frame(matrix(ncol = 23, nrow = 9))
colnames(purity_df) <- c(disease_clusters, "Purity")
rownames(purity_df) <- c("betweenness", "fastgreedy", "louvain", "spinglass", "markov",
                         "leiden", "label prop", "label prop init", "lead eigenvector")

fmeasure_df <- data.frame(matrix(ncol = 1, nrow = 9))

for (i in 4:length(nodes_results))
{
  f_measure <- 0
  confmat <- compute_confmtab(nodes_results$X1, nodes_results[, i])
  purity_df[i - 3, ] <- unname(c(diag(confmtab$table) / rowSums(confmtab$table)),
                                confmat$overall[1]))

  fmeasure_df[i - 3, ] <- sum(confmtab$byClass[, 7], na.rm = TRUE) / 22
}

purity_df$Multiple <- NULL
purity_df$Unclassified <- NULL

purity_df <- as.data.frame(t(purity_df))
colnames(purity_df) <- purity_df[1, ]

rownames(purity_df) <- c("Ear,Nose...", "Cancer", "Ophthalmolog", "Endocrine",
                         "Cardiovasc", "Neurologic", "Hematologic", "Nutritional",
                         "Muscular", "Respiratory", "Immunologic", "Dermatologic",
                         "Psychiatric", "Metabolic", "Gastrointest", "Bone", "Skeletal",
                         "Renal", "Development", "Connective tissue disorder", "Purity")

temp <- purity_df[14, ]
purity_df[14, ] <- purity_df[20, ]
purity_df[20, ] <- temp

colnames(purity_df) <- c("betweenness", "fastgreedy", "louvain", "spinglass", "markov",
                         "leiden", "label prop", "label prop init", "lead eigenvector")

panderOptions('table.split.table', 8*15)
# panderOptions('table.split.cells', 30)
pander(format(purity_df, digits = 3))

```

Table 3.1: Table continues below

	betweenness	fastgreedy	louvain	spinglass	markov	leiden	label prop
<b>Ear,Nose...</b>	0.000	0.000	0.000	0.000	0.000	1.000	0.000
<b>Cancer</b>	0.964	0.963	0.934	0.933	0.951	1.000	0.972
<b>Ophthalmolog</b>	0.867	0.915	0.867	0.860	0.902	0.902	0.800

	betweenness	fastgreedy	louvain	spinglass	markov	leiden	label prop
<b>Endocrine</b>	0.296	0.296	0.296	0.464	0.781	0.857	0.429
<b>Cardiovasc</b>	0.656	0.357	0.414	0.433	0.750	0.794	0.645
<b>Neurologic</b>	0.689	0.689	0.689	0.765	0.695	0.750	0.688
<b>Hematologic</b>	0.486	0.568	0.568	0.568	0.714	0.865	0.703
<b>Nutritional</b>	0.000	0.000	0.000	0.000	0.333	1.000	0.000
<b>Muscular</b>	0.895	0.895	0.895	0.882	1.000	0.750	0.882
<b>Respiratory</b>	0.000	0.000	0.000	0.000	0.400	0.000	0.000
<b>Immunologic</b>	0.417	0.417	0.556	0.417	0.583	0.708	0.583
<b>Dermatologic</b>	0.760	0.760	0.760	0.750	0.696	0.852	0.821
<b>Psychiatric</b>	0.000	0.000	0.000	0.556	0.556	0.500	0.556
<b>Metabolic</b>	0.000	0.000	0.000	0.000	0.375	0.909	0.000
<b>Gastrointest</b>	0.235	0.286	0.000	0.541	0.639	0.735	0.605
<b>Bone</b>	0.000	0.000	0.000	0.000	0.600	0.400	0.000
<b>Skeletal</b>	0.800	0.667	0.920	0.632	0.917	0.765	0.312
<b>Renal</b>	0.217	0.462	0.481	0.667	0.458	0.667	0.750
<b>Development</b>	0.455	0.400	0.455	0.455	0.500	0.500	0.455
<b>Connective tissue disorder</b>	0.000	0.000	0.000	0.000	0.333	0.588	0.529
<b>Purity</b>	0.614	0.612	0.607	0.653	0.748	0.814	0.692

	label prop init	lead eigenvector
<b>Ear,Nose...</b>	0.000	0.000
<b>Cancer</b>	0.991	0.941
<b>Ophthalmolog</b>	0.902	0.918
<b>Endocrine</b>	0.667	0.259
<b>Cardiovasc</b>	0.724	0.393
<b>Neurologic</b>	0.861	0.786
<b>Hematologic</b>	0.703	0.605
<b>Nutritional</b>	0.000	0.000
<b>Muscular</b>	0.882	0.267
<b>Respiratory</b>	0.000	0.000
<b>Immunologic</b>	0.720	0.308
<b>Dermatologic</b>	0.833	0.750
<b>Psychiatric</b>	0.556	0.000
<b>Metabolic</b>	0.250	0.000
<b>Gastrointest</b>	0.531	0.286
<b>Bone</b>	0.000	0.000
<b>Skeletal</b>	0.647	0.000
<b>Renal</b>	0.821	0.571
<b>Development</b>	0.000	0.400
<b>Connective tissue disorder</b>	0.562	0.000
<b>Purity</b>	0.764	0.581

### 3.12.1.2 F-measure

```
rownames(fmeasure_df) <- c("betweenness", "fastgreedy", "louvain", "spinglass", "markov",
                            "leiden", "label prop", "label prop init", "lead eigenvector")
```

```
colnames(fmeasure_df) <- c("F-Measure")
pander(fmeasure_df)
```

	F-Measure
<b>betweenness</b>	0.3391
<b>fastgreedy</b>	0.3429
<b>louvain</b>	0.3312
<b>spinglass</b>	0.4027
<b>markov</b>	0.6222
<b>leiden</b>	0.756
<b>label prop</b>	0.4457
<b>label prop init</b>	0.4928
<b>lead eigenvector</b>	0.2939

### 3.12.2 Adjusted Rand index

L'adjusted Rand index (ARI) misura l'accuratezza dei raggruppamenti costruiti dagli algoritmi utilizzando la seguente formula applicata sulla tabella di contingenza dei risultati dell'algoritmo:

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}}{\frac{1}{2} [\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] / \binom{n}{2}}$$

Dove  $a_i$  è la somma della riga  $i$ ,  $b_j$  è la somma della colonna  $j$  ed  $n$  è il numero totale di nodi nel grafo.

```
library(aricode)
ari_df <- data.frame(matrix(ncol = 1, nrow = 9))
colnames(ari_df) <- c("Adjusted Rand index")
rownames(ari_df) <- c("betweenness", "fastgreedy", "louvain", "spinglass", "markov",
                      "leiden", "label prop", "label prop init", "lead eigenvector")

for (i in 1:9)
{
  groundtruth <- nodes_results[, 3]
  clusters <- nodes_results[, 3 + i]
  groundtruth[groundtruth == "Multiple"] <- clusters[groundtruth == "Multiple"]
  groundtruth[groundtruth == "Unclassified"] <- clusters[groundtruth == "Unclassified"]
  ari_df[i, ] <- ARI(groundtruth, clusters)
}

pander(format(ari_df, digits = 3))
```

	Adjusted Rand index
<b>betweenness</b>	0.451
<b>fastgreedy</b>	0.477
<b>louvain</b>	0.460
<b>spinglass</b>	0.498
<b>markov</b>	0.597
<b>leiden</b>	0.707
<b>label prop</b>	0.536

Adjusted Rand index	
label prop init	0.628
lead eigenvector	0.414

### 3.12.3 Normalized Mutual Information

La Normalized Mutual Information rispetto alla purity è una misura più penalizzante, è calcolata con la seguente formula:

$$NMI(\Omega; C) = \frac{MI(\Omega; C)}{[H(\Omega) + H(C)]/2}$$

Dove  $MI$  rappresenta la Mutual Information tra i cluster ( $\Omega$ ) e le classi ( $C$ ), mentre  $H$  è la misura dell'entropia.

```
nmi_df <- data.frame(matrix(ncol = 1, nrow = 9))
colnames(nmi_df) <- c("Normalized Mutual Information")
rownames(nmi_df) <- c("betweenness", "fastgreedy", "louvain", "spinglass", "markov",
                      "leiden", "label prop", "label prop init", "lead eigenvector")

for (i in 1:nrow(nmi_df))
{
  groundtruth <- nodes_results[, 3]
  clusters <- nodes_results[, 3 + i]
  groundtruth[groundtruth == "Multiple"] <- clusters[groundtruth == "Multiple"]
  groundtruth[groundtruth == "Unclassified"] <- clusters[groundtruth == "Unclassified"]
  nmi_df[i, ] <- NMI(groundtruth, clusters)
}

pander(format(nmi_df, digits = 3))
```

Normalized Mutual Information	
betweenness	0.458
fastgreedy	0.475
louvain	0.472
spinglass	0.515
markov	0.642
leiden	0.726
label prop	0.551
label prop init	0.631
lead eigenvector	0.423

## 3.13 Distribuzione di centralità

Confrontiamo la distribuzione di centralità dell'intera rete delle malattie (nogenes) con quella dei clusters rilevati con l'algoritmo Leiden.

```
distribution[1,2] <- mean(leiden_degree[,1])
distribution[2,2] <- mean(leiden_betweenness[,1])
distribution[3,2] <- mean(leiden_closeness[,1])
```

```
distribution[4,2] <- mean(leiden_eigen[,1])
pander(format(distribution, digits = 3, justify="left"))
```

	Nogenes Network	Leiden Clusters
<b>Mean Degree</b>	4.6047	1.036
<b>Mean Betweenness</b>	0.0114	0.052
<b>Mean Closeness</b>	0.0747	0.278
<b>Mean Eigenvector</b>	0.0284	0.945

```
print(paste("Media dei pesi di Human Disease Network: ",
            format(mean(E(UD_nogenes)$weight), digits=3)))
## [1] "Media dei pesi di Human Disease Network: 2.17"
print(paste("Media della media dei pesi dei clusters individuati con l'algoritmo Leiden: ",
            format(mean(leiden_weights[,1]), digits=3)))
## [1] "Media della media dei pesi dei clusters individuati con l'algoritmo Leiden: 1.37"
```



# Conclusioni

Bla bla bla



# Bibliografia

- Blondel, Vincent D, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. "Fast Unfolding of Communities in Large Networks." *Journal of Statistical Mechanics: Theory and Experiment* 2008 (10). IOP Publishing: P10008. doi:10.1088/1742-5468/2008/10/p10008.
- Clauset, Aaron, M. E. J. Newman, and Cristopher Moore. 2004. "Finding Community Structure in Very Large Networks." *Phys. Rev. E* 70 (6). American Physical Society: 066111. doi:10.1103/PhysRevE.70.066111.
- E.J. Newman, Mark, and Michelle Girvan. 2004. "Finding and Evaluating Community Structure in Networks." *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics* 69 (March): 026113. doi:10.1103/PhysRevE.69.026113.
- Enright, A. J., S. Van Dongen, and C. A. Ouzounis. 2002. "An efficient algorithm for large-scale detection of protein families." *Nucleic Acids Research* 30 (7): 1575–84. doi:10.1093/nar/30.7.1575.
- Goh, Kwang-II, Michael E. Cusick, David Valle, Barton Childs, Marc Vidal, and Albert-László Barabási. 2007. "The Human Disease Network." *Proceedings of the National Academy of Sciences* 104 (21). National Academy of Sciences: 8685–90. doi:10.1073/pnas.0701361104.
- Newman, M. E. J. 2006. "Finding Community Structure in Networks Using the Eigenvectors of Matrices." *Phys. Rev. E* 74 (3). American Physical Society: 036104. doi:10.1103/PhysRevE.74.036104.
- Raghavan, Usha Nandini, Réka Albert, and Soundar Kumara. 2007. "Near Linear Time Algorithm to Detect Community Structures in Large-Scale Networks." *Phys. Rev. E* 76 (3). American Physical Society: 036106. doi:10.1103/PhysRevE.76.036106.
- Reichardt, Jörg, and Stefan Bornholdt. 2006. "Statistical Mechanics of Community Detection." *Phys. Rev. E* 74 (1). American Physical Society: 016110. doi:10.1103/PhysRevE.74.016110.
- Traag, Vincent, L Waltman, and Nees Jan van Eck. 2019. "From Louvain to Leiden: Guaranteeing Well-Connected Communities." *Scientific Reports* 9 (March): 5233. doi:10.1038/s41598-019-41695-z.



# Appendice

## A.1 Analisi sulle malattie contrassegnate come Unclassified:

```
unclassified_disease <- nodes_results %>% filter(X1 == "Unclassified")

unclassified_disease <- unclassified_disease %>% select(label, Betweenness, Fastgreedy,
  Louvain, Spinglass, Markov,
  Leiden, Label_prop,
  Label_prop_init)

col <- unclassified_disease$label

unclassified_disease <- as.data.frame(t(unclassified_disease))
colnames(unclassified_disease) <- unclassified_disease[1, ]
unclassified_disease <- unclassified_disease[-1, ]

colnames(unclassified_disease) <- col
```

```
panderOptions('table.split.table', 8*17)
pander(format(unclassified_disease, digits=3))
```

Table A.1: Table continues below

	Beta-2-adrenoreceptor agonist, reduced response to	Aquaporin-1 deficiency	Aneurysm, familial arterial	Benzene toxicity
<b>Betweenness</b>	Neurological	Hematological	Bone	Cancer
<b>Fastgreedy</b>	Neurological	Hematological	Ophthalmological	Cancer
<b>Louvain</b>	Neurological	Hematological	Bone	Cancer
<b>Spinglass</b>	Neurological	Hematological	Bone	Cancer
<b>Markov</b>	Endocrine	Hematological	Bone	Cancer
<b>Leiden</b>	Nutritional	Hematological	Connective tissue disorder	Cancer
<b>Label_prop</b>	Metabolic	Hematological	Connective tissue disorder	Cancer
<b>Label_prop_init</b>	Immunological	Hematological	Bone	Cancer

	Alcohol dependence	van Buchem disease	Placental abruption	Bannayan-Riley- Ruvalcaba syndrome	Carpal tunnel syndrome, familial
<b>Betweenness</b>	Cancer	Bone	Cardiovascular	Cancer	Metabolic
<b>Fastgreedy</b>	Cancer	Ophthalmological	Metabolic	Cancer	Hematological
<b>Louvain</b>	Cancer	Bone	Immunological	Cancer	Hematological
<b>Spinglass</b>	Psychiatric	Bone	Metabolic	Cancer	Hematological
<b>Markov</b>	Psychiatric	Bone	Cardiovascular	Cancer	Metabolic
<b>Leiden</b>	Nutritional	Unclassified	Cardiovascular	Unclassified	Hematological
<b>Label_prop</b>	Psychiatric	Bone	Cardiovascular	Cancer	Hematological
<b>Label_prop_init</b>	Psychiatric	Bone	Cardiovascular	Cancer	Hematological

Dalla precedente tabella, risulta evidente come alcune malattie, che inizialmente non avevano un cluster di appartenenza, vengano identificate dalla maggior parte degli algoritmo di clustering come appartenenti ad una determinata community.

Gli esempi più evidenti sono:

- **Aquaporin-1 deficiency:** viene identificata da tutti e 7 gli algoritmi come appartenente alla community *Hematological*. Verificando meglio attraverso la fonte [OMIM - Online Mendelian Inheritance in Man](<https://omim.org>), una trattazione sintetica di geni umani e fenotipi genetici, è possibile verificare la relazione di questo disturbo con i gruppi sanguigni.
- **Benzene toxicity:** viene indentificata da tutti gli algoritmi come appartenente al cluster *Cancer*. In questo caso questo tipo di malattia sembreremmo ricevere numerose influenze da Leukemia, Post-chemotherapy, Breast cancer etc.
- **Bannayan-Riley- Ruvalcaba syndrome:** viene identificata da quasi tutti gli algoritmi come appartenente alla community *Cancer*. Infatti questa malattia sembrerebbe avere un collegamento con il cancro al seno.

## A.2 Analisi sulle malattie contrassegnate come Multiple:

```
multiple_disease <- nodes_results %>% filter(X1 == "Multiple")
multiple_disease <- head(multiple_disease %>%
  select(label, Betweenness, Fastgreedy, Louvain, Spinglass,
  Markov, Leiden, Label_prop, Label_prop_init), 10)

col <- multiple_disease$label

multiple_disease <- as.data.frame(t(multiple_disease))
colnames(multiple_disease) <- multiple_disease[1, ]
multiple_disease <- multiple_disease[-1, ]

colnames(multiple_disease) <- col
```

Vengono mostrate nella tabella sottostante le prime 10 malattie che venivano identificate come Multiple e il nuovo cluster di appartenenza a seconda dell'algoritmo utilizzato.

```
panderOptions('table.split.table', 8*18)
pander(format(multiple_disease, digits=3))
```

Table A.3: Table continues below

	Fanconi anemia	Usher syndrome	Mitochondrial complex deciency	Dejerine-Sottas disease	Waardenburg syndrome
<b>Betweenness</b>	Cancer	Bone	Muscular	Neurological	Cancer
<b>Fastgreedy</b>	Cancer	Bone	Muscular	Neurological	Cancer
<b>Louvain</b>	Cancer	Bone	Muscular	Neurological	Cancer
<b>Spinglass</b>	Cancer	Neurological	Metabolic	Neurological	Cancer
<b>Markov</b>	Cancer	Neurological	Metabolic	Neurological	Multiple
<b>Leiden</b>	Multiple	Neurological	Metabolic	Neurological	Cancer
<b>Label_prop</b>	Cancer	Neurological	Metabolic	Neurological	Cancer
<b>Label_prop_init</b>	Cancer	Neurological	Neurological	Neurological	Cancer

	Stickler syndrome	Walker- Warburg syndrome	Rubenstein- Taybi syndrome	Waardenburg- Shah syndrome	Kallmann syndrome
<b>Betweenness</b>	Bone	Muscular	Cancer	Cancer	Cancer
<b>Fastgreedy</b>	Bone	Muscular	Cancer	Cancer	Skeletal
<b>Louvain</b>	Bone	Muscular	Cancer	Cancer	Skeletal
<b>Spinglass</b>	Bone	Muscular	Cancer	Cancer	Skeletal
<b>Markov</b>	Skeletal	Muscular	Cancer	Multiple	Cancer
<b>Leiden</b>	Bone	Muscular	Multiple	Neurological	Skeletal
<b>Label_prop</b>	Skeletal	Muscular	Cancer	Neurological	Skeletal
<b>Label_prop_init</b>	Skeletal	Muscular	Cancer	Neurological	Skeletal

In questo caso, risulta evidente come in alcuni casi le malattie non si inseriscano più in diversi cluster, ma alcune di queste presentano un unica community ricorrente.

In questo caso gli esempi che ci vengono in supporto sono:

- **Fanconi anemia:** viene indentificata come appartenente al cluster *Cancer*. Infatti tra le principali caratteristiche cliniche di questa malattia vi è un'alta predisposizione al cancro.
- **Waardenburg syndrome:** viene indentificata anche lei come appartenente al cluster *Cancer*.
- **Walker- Warburg syndrome:** viene indentificata da tutti gli algoritmi come appartenente alla community *Muscular*.
- **Rubenstein- Taybi syndrome:** viene identificata come appartenente alla community *Cancer*.
- **Dejerine-Sottas disease:** viene indentificata da tutti gli algoritmi come appartenente alla community *Neurological*.

In altri casi invece, come per **Usher syndrome**, **Mitochondrial complex deciency**, **Waardenburg-Shah syndrome**, **Kallmann syndrome** e **Stickler syndrome** multipli cluster sembrano influire sulle malattie.