

GBM analysis

Mattia Manna

2025-01-19

Indice

1	Importazione dati, pulizia e filtraggio	1
1.1	Importazione dati	1
1.2	Pulizia dati	1
1.3	Filtrare	1
1.4	Prendere count GBM e SANI e filtrare di nuovo	2
2	Calcolo dei DEGS sul TRAIN set	3
2.0.1	DEGS su tutto il dataframe con TMM	3
2.0.2	Visualize TMM parameters	5
2.1	Differential co-expressed network	6
3	Metriche di centralità	7
3.1	Istogrammi	8
3.1.1	Degree	8
3.1.2	Betweenness centrality	9
3.1.3	Closeness centrality	10
3.1.4	Eigenvector centrality	11
3.1.5	Local clustering coefficient	12
3.2	Degree centrality	13
3.3	Betweenness centrality	14
3.4	Closeness centrality	15
3.4.1	Ultimo 5% (0.95)	15
3.4.2	Primo 5 % (0.05)	15
3.5	Eigenvector centrality	16
3.6	Local Clustering coefficient	16

1 Importazione dati, pulizia e filtraggio

1.1 Importazione dati

Importare i dati salvati localmente.

```
tep.expr <- read.delim("~/Desktop/Università/Magistrale/Tesi/R/GSE68086/GSE68086_TEP_data_matrix.txt",
dim(tep.expr)
## [1] 57736 285

patients <- read.csv("~/Desktop/Università/Magistrale/Tesi/R/GSE68086/patients.csv", row.names=1, sep="
dim(patients)
## [1] 285 51

gene.info <- read.delim("~/Desktop/Università/Magistrale/Tesi/R/GSE68086/Human.GRCh38.p13.annot.tsv", r
dim(gene.info)
## [1] 39376 17
```

1.2 Pulizia dati

Siccome ci sono delle incongruenze tra il nome dei sample nei dataframe:

- patients, dataset contenente le informazioni per ogni sample
- tep.expr, dataset contenente la conta per i geni

sistemare queste incongruenze.

```
patients$sample_name <- sapply( patients$source_name_ch1, # Vettore al quale applicare la funzione
# Funzione da applicare ad ogni valore nel vettore
function(x) gsub("-", ".", x)
)
colnames(tep.expr)<- sapply(colnames(tep.expr), function(x) sub("X", "", x))
```

1.3 Filtrare

Filtrare per la condizione: almeno un sample con count > 5

```
tep.expr.filtr <- tep.expr %>% filter_all(any_vars(. >= 5))
dim(tep.expr.filtr)
```

```
## [1] 16347 285
```

1.4 Prendere count GBM e SANI e filtrare di nuovo

```
# Specificare la condizione da analizzare
condizione.da.analizzare = "GBM"

# Estrarre MALATI con la condizione specificata
condition.samples <- patients[patients$cancer.type.ch1==condizione.da.analizzare,]$sample_name

# Dimensioni del samples (controllo per la correttezza dei risultati)
n.condition.samples <- length(condition.samples)
print(n.condition.samples)
## [1] 40

# Estrarre il nome/codice dei sample SANI
healthy.samples <- patients[patients$cancer.type.ch1=="HC",]$sample_name

## Dimensioni del samples (controllo per la correttezza dei risultati)
length(healthy.samples)
## [1] 55

# Unire SANI e MALATI
tutti.samples <- c(condition.samples,healthy.samples)
length(tutti.samples)
## [1] 95

# Filtrare la matrice delle conte di nuovo perchè sono state selezionate solo alcune patologie
counts <- tep.expr.filtr[, tutti.samples]
counts <- counts %>% filter_all(any_vars(. >= 5))
dim(counts)
## [1] 14411    95
```

Estrarsi in sommario dei pazienti presi in considerazione.

```
patients.summary <- patients[patients$sample_name %in% colnames(counts),c("sample_name","cancer.type.ch1")]
patients.summary[1:5,]
```

```
##           sample_name cancer.type.ch1
## GSM1662606    VU280.GBM             GBM
## GSM1662607 VU284.GBM.vIII             GBM
## GSM1662608 VU306.GBM.vIII             GBM
## GSM1662609    VU360.GBM.WT            GBM
## GSM1662610 VU372.GBM.vIII             GBM
```

2 Calcolo dei DEGS sul TRAIN set

2.0.1 DEGS su tutto il dataframe con TMM

Numero degs su tutto il dataframe senza TMM (no divisione train e test).

```
group <- factor(c(rep("GBM",40),rep("HC",55)))

# Creazione dell'oggetto DGEList
oggettoDGEList <- DGEList(counts = counts, group = group)

oggettoDGEList <- calcNormFactors(oggettoDGEList, method = "TMM")

# Normalizzare con CPM e le standard lib sizes
normalized_counts <- cpm(oggettoDGEList,
                          normalized.lib.sizes = F,
                          prior.count= 1 ,
                          log = T)

TMMparameters <- TMMbyHand(counts)

normalized_counts <- as.data.frame(normalized_counts) # renderlo un dataframe

# build the design matrix
design <- model.matrix(~ 0 + group)

colnames(design) <- levels(group)

# compute common, trend, tagwise dispersion
y <- edgeR::estimateDisp(oggettoDGEList ,design = design,robust = F)

# fit the negative binomial GLM for each tag
fit <-edgeR:: glmFit(y, design=design)

# Definire il contrasto
contrast <- limma::makeContrasts(contrasts= "GBM - HC",levels = colnames(design))

# Test statistico
lrt <- edgeR::glmLRT(fit,contrast = contrast)

logCPM.threshold <- 3
FDR.threshold <- 0.01
## Estrarre l'oggetto che contiene le metriche per tutti gli n geni
metriche <- topTags(lrt, n = nrow(counts))$table

## Applicare le threshold e trovare i DEGs
DEGs.metriche <- metriche[(metriche$logCPM > logCPM.threshold)
                          &
                          (metriche$FDR < FDR.threshold), ]
```

```
nomiDEGS <- rownames(DEGs.metriche)
dim(DEGs.metriche)
```

```
## [1] 860 5
```

```
tep.expr.filtr.normDEGS <- normalized_counts[rownames(normalized_counts) %in%
                                              nomiDEGS,]
```

```
tep.expr.filtr.normDEGS <- dataframe.formato.classificazione(tep.expr.filtr.normDEGS)
```

```
# Esportare il dataframe
```

```
write.csv(tep.expr.filtr.normDEGS,
          file = "/Users/mattia/Desktop/Università/Magistrale/Tesi/Python TMM only for exploration/Data",
          row.names = T )
```

2.0.2 Visualize TMM parameters

The most interesting parameters are the reference sample and the smoothed factor.

```
TMMparameters$refColumn
```

```
## HD.35.2
```

```
##      84
```

```
TMMparameters$divisore
```

```
## [1] 0.9892687
```

2.1 Differential co-expressed network

```
DEGS.tep.expr.filtr.normalized.gbmVShc <- normalized_counts[rownames(normalized_counts) %in% rownames(D

matrix.condition1 <- DEGS.tep.expr.filtr.normalized.gbmVShc[, condition.samples]
matrix.condition2 <- DEGS.tep.expr.filtr.normalized.gbmVShc[, healthy.samples]
Zscores.gbmVShc <- evaluate_zscores(matrix.condition1,matrix.condition2)

# Soglia
z_threshold <- 3

output.gbmVShc <- differential_coexpression_network(Zscores.gbmVShc,z_threshold)

tep.expr.filtr.normHUBS <- normalized_counts[rownames(normalized_counts) %in%
                                             output.gbmVShc$hubs$gene,]

tep.expr.filtr.normHUBS <- dataframe.formato.classificazione(tep.expr.filtr.normHUBS)

# Esportare il dataframe
write.csv(tep.expr.filtr.normHUBS,
          file = "/Users/mattia/Desktop/Università/Magistrale/Tesi/Python TMM only for exploration/Data,
          row.names =T )
```

3 Metriche di centralità

```
matrice.adiacenza <- output.gbmVShc$adjacency.matrix
matrice.adiacenza <- abs(matrice.adiacenza)
```

```
# Creare il network igraph dalla matrice di adiacenza
network <- graph_from_adjacency_matrix(matrice.adiacenza)
```

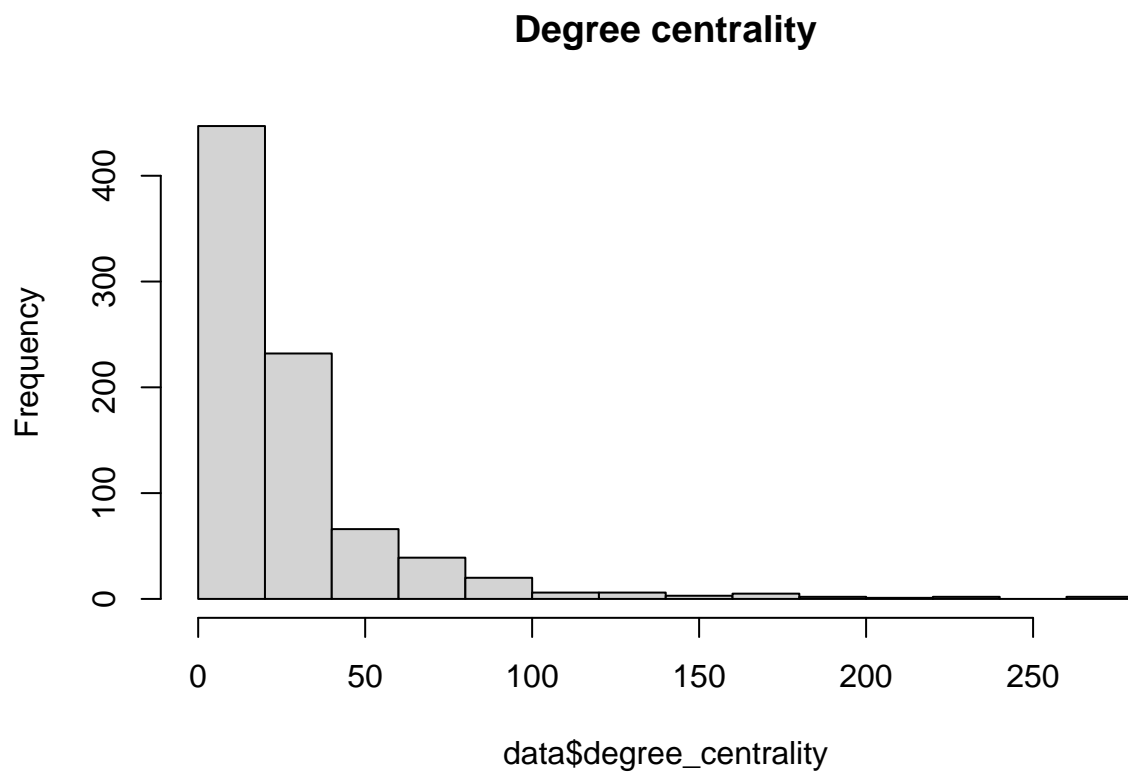
```
data <- data.frame(
  "Gene" = V(network)$name,
  "degree_centrality" = degree(network),
  "betweenness_centrality" = betweenness(network),
  "closeness_centrality" = closeness(network),
  "eigenvector_centrality" = eigen_centrality(network)$vector,
  "local_clustering_coefficient" = transitivity(network, type="local")
)
data[is.na(data)] <- 0
dim(data)
```

```
## [1] 831 6
```

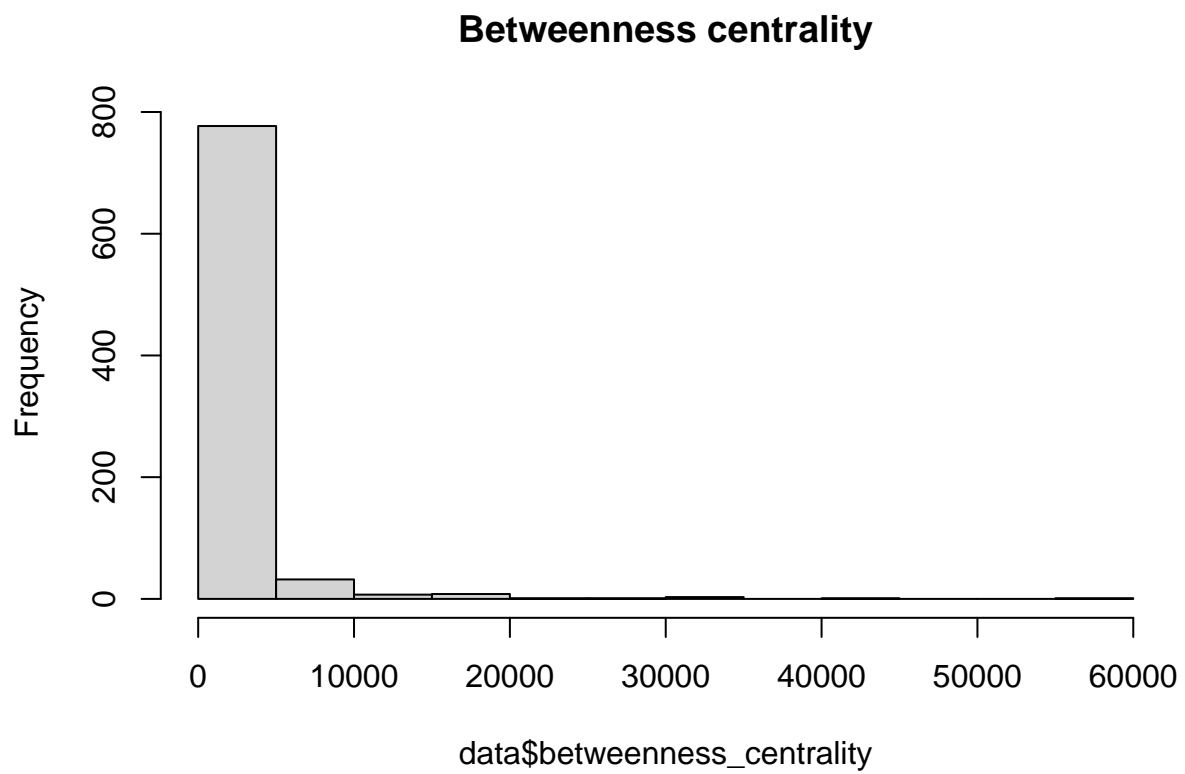
```
save(data, file = "/Users/mattia/Desktop/Università/Magistrale/Tesi/R/_Grafici/GBMmetrics_distribution.")
```


3.1 Istogrammi

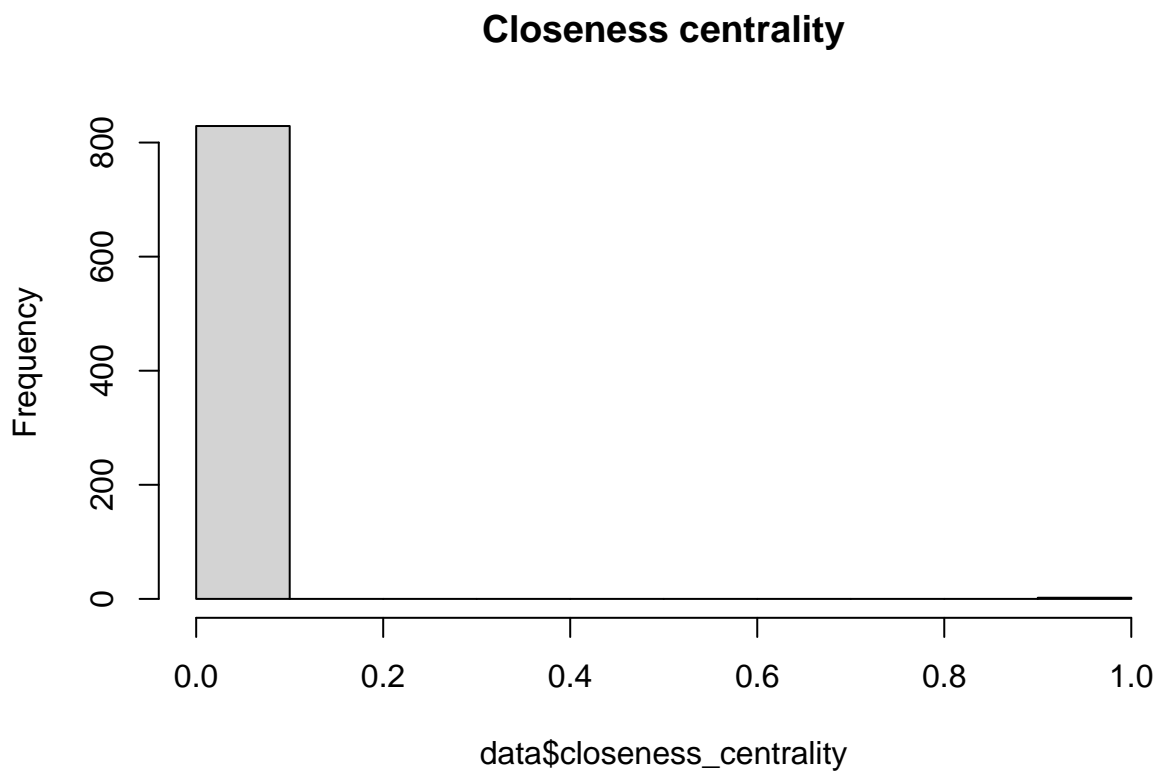
3.1.1 Degree



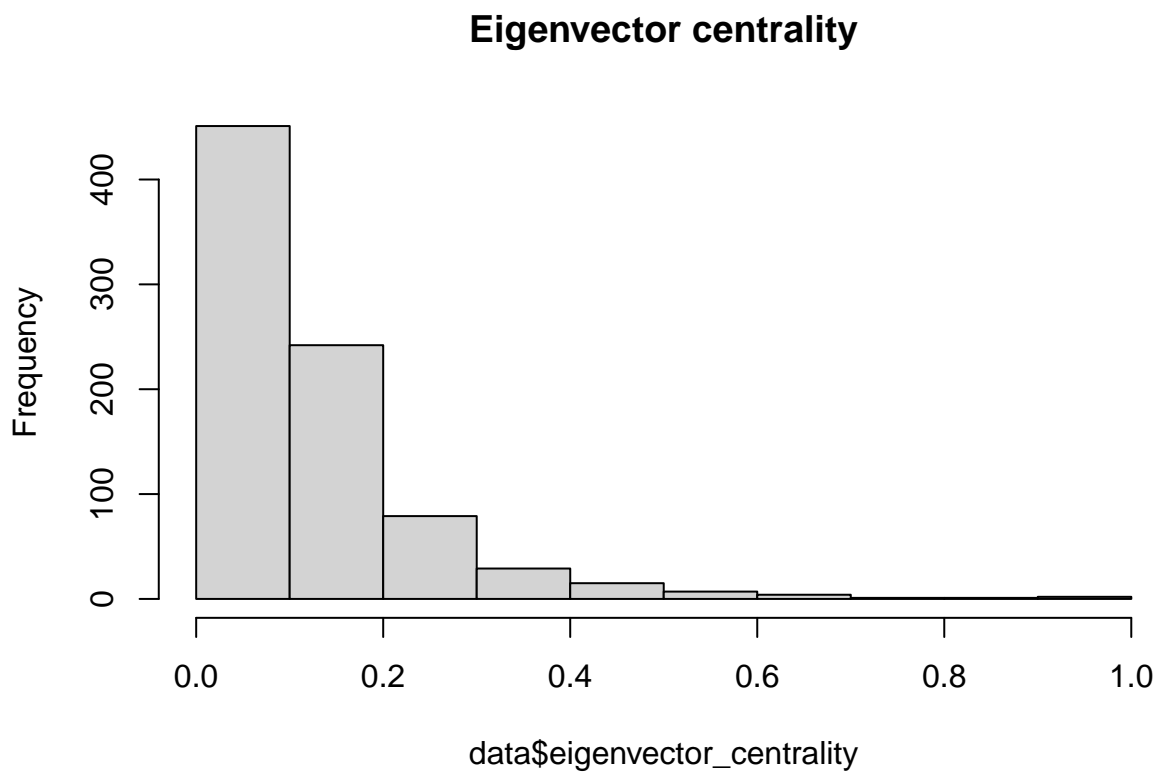
3.1.2 Betweenness centrality



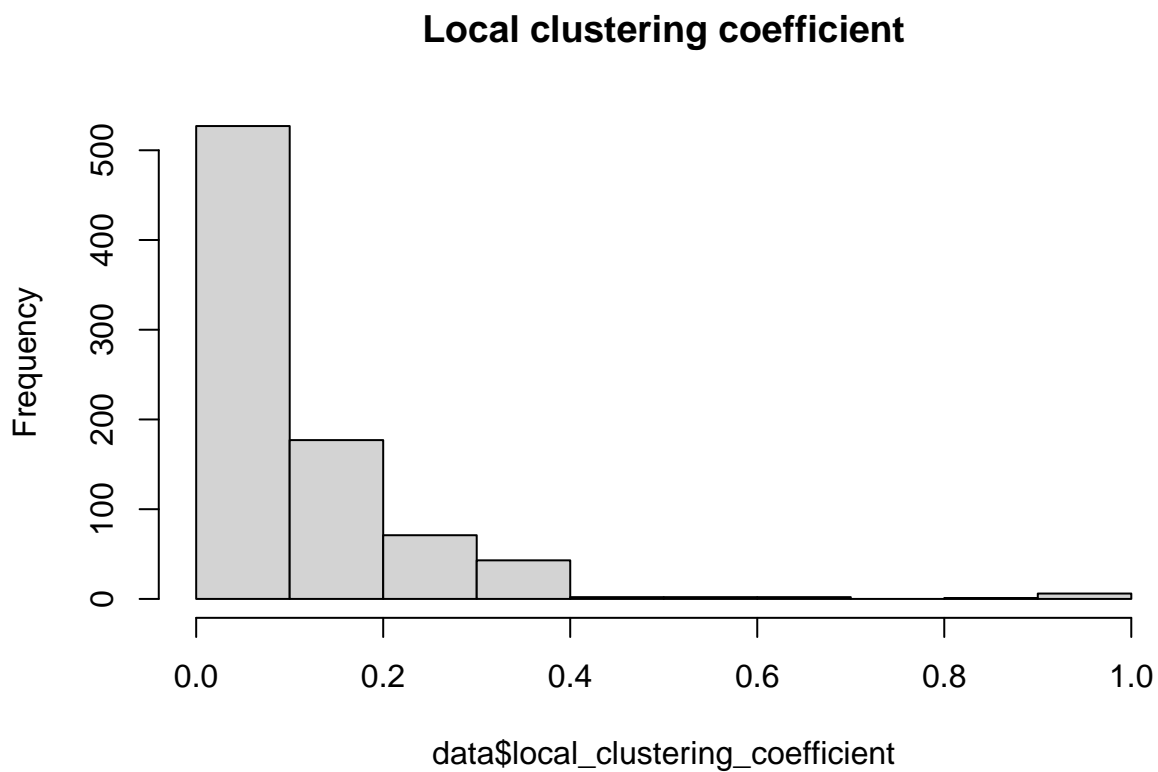
3.1.3 Closeness centrality



3.1.4 Eigenvector centrality



3.1.5 Local clustering coefficient



3.2 Degree centrality

```
soglia <- 0.95

metrica <- data$degree_centrality
q <- quantile(metrica, probs = c(soglia))

# Selezionare solo i geni con degree_centrality superiore al 95% percentile
genes <- data[metrica > q,]$Gene

# Estrarre solo i geni con degree_centrality superiore al 95% percentile
# dalla matrice delle conte filtrata e normalizzata
tep.expr.filtr.normDEGREE <- normalized_counts[rownames(normalized_counts) %in%
                                                genes,]

dim(tep.expr.filtr.normDEGREE)
## [1] 42 95
nrow(tep.expr.filtr.normDEGREE) # Numero geni selezionati tramite metrica
## [1] 42

tep.expr.filtr.normDEGREE <- dataframe.formato.classificazione(tep.expr.filtr.normDEGREE)
dim(tep.expr.filtr.normDEGREE)
## [1] 95 43

# Esportare il dataframe
write.csv(tep.expr.filtr.normDEGREE,
          file = "/Users/mattia/Desktop/Università/Magistrale/Tesi/Python TMM only for exploration/Data",
          row.names = T )
```

3.3 Betweenness centrality

```
soglia <- 0.95

metrica <- data$betweenness_centrality
q <- quantile(metrica, probs = c(soglia))

# Selezionare solo i geni con eigenvector_centrality superiore al 95% percentile
genes <- data[metrica > q,]$Gene

# Estrarre solo i geni con eigenvector_centrality superiore al 95% percentile
# dalla matrice delle conte filtrata e normalizzata
tep.expr.filtr.normBETWEENNESS <- normalized_counts[rownames(normalized_counts) %in%
                                                    genes,]

dim(tep.expr.filtr.normBETWEENNESS)
## [1] 42 95
nrow(tep.expr.filtr.normBETWEENNESS) # Numero geni selezionati tramite metrica
## [1] 42

tep.expr.filtr.normBETWEENNESS <- dataframe.formato.classificazione(tep.expr.filtr.normBETWEENNESS)
dim(tep.expr.filtr.normBETWEENNESS)
## [1] 95 43

# Esportare il dataframe
write.csv(tep.expr.filtr.normBETWEENNESS,
          file = "/Users/mattia/Desktop/Università/Magistrale/Tesi/Python TMM only for exploration/Data",
          row.names = T )
```

3.4 Closeness centrality

3.4.1 Ultimo 5% (0.95)

```
soglia <- 0.95

metrica <- data$closeness centrality
q <- quantile(metrica, probs = c(soglia))

# Selezionare solo i geni con eigenvector centrality superiore al 95% percentile
genes <- data[metrica > q,]$Gene

# Estrarre solo i geni con eigenvector centrality superiore al 95% percentile
# dalla matrice delle conte filtrata e normalizzata
tep.expr.filtr.normCLOSENESS <- normalized_counts[rownames(normalized_counts) %in%
                                                    genes,]

dim(tep.expr.filtr.normCLOSENESS)
## [1] 42 95
nrow(tep.expr.filtr.normCLOSENESS) # Numero geni selezionati tramite metrica
## [1] 42

tep.expr.filtr.normCLOSENESS <- dataframe.formato.classificazione(tep.expr.filtr.normCLOSENESS)
dim(tep.expr.filtr.normCLOSENESS)
## [1] 95 43

# Esportare il dataframe
write.csv(tep.expr.filtr.normCLOSENESS,
          file = "/Users/mattia/Desktop/Università/Magistrale/Tesi/Python TMM only for exploration/Data",
          row.names = T )
```

3.4.2 Primo 5 % (0.05)

Ha due code quindi considerare anche il primo 5%.

```
soglia <- 0.05
metrica <- data$closeness centrality
q <- quantile(metrica, probs = c(soglia))
genes <- data[metrica < q,]$Gene
tep.expr.filtr.normCLOSENESS_last5 <- normalized_counts[rownames(normalized_counts) %in%
                                                         genes,]

dim(tep.expr.filtr.normCLOSENESS_last5)
## [1] 42 95
nrow(tep.expr.filtr.normCLOSENESS_last5) # Numero geni selezionati tramite metrica
## [1] 42

tep.expr.filtr.normCLOSENESS_last5 <- dataframe.formato.classificazione(tep.expr.filtr.normCLOSENESS_la
dim(tep.expr.filtr.normCLOSENESS_last5)
## [1] 95 43

# Esportare il dataframe
write.csv(tep.expr.filtr.normCLOSENESS_last5,
          file = "/Users/mattia/Desktop/Università/Magistrale/Tesi/Python TMM only for exploration/Data",
          row.names = T )
```


3.5 Eigenvector centrality

```
soglia <- 0.95

metrica <- data$eigenvector_centrality
q <- quantile(metrica, probs = c(soglia))

# Selezionare solo i geni con eigenvector_centrality superiore al 95% percentile
genes <- data[metrica > q,]$Gene

tep.expr.filtr.normEIGEN <- normalized_counts[rownames(normalized_counts) %in%
                                              genes,]

dim(tep.expr.filtr.normEIGEN)
## [1] 42 95
nrow(tep.expr.filtr.normEIGEN) # Numero geni selezionati tramite metrica
## [1] 42

tep.expr.filtr.normEIGEN <- dataframe.formato.classificazione(tep.expr.filtr.normEIGEN)
dim(tep.expr.filtr.normEIGEN)
## [1] 95 43

# Esportare il dataframe
write.csv(tep.expr.filtr.normEIGEN,
          file = "/Users/mattia/Desktop/Università/Magistrale/Tesi/Python TMM only for exploration/Data",
          row.names = T )
```

3.6 Local Clustering coefficient

```
soglia <- 0.95

metrica <- data$local_clustering_coefficient
q <- quantile(metrica, probs = c(soglia))

# Selezionare solo i geni con eigenvector_centrality superiore al 95% percentile
genes <- data[metrica > q,]$Gene

tep.expr.filtr.normCLUSTER <- normalized_counts[rownames(normalized_counts) %in%
                                              genes,]

dim(tep.expr.filtr.normCLUSTER)
## [1] 27 95
nrow(tep.expr.filtr.normCLUSTER) # Numero geni selezionati tramite metrica
## [1] 27

tep.expr.filtr.normCLUSTER <- dataframe.formato.classificazione(tep.expr.filtr.normCLUSTER)
dim(tep.expr.filtr.normCLUSTER)
## [1] 95 28

# Esportare il dataframe
write.csv(tep.expr.filtr.normCLUSTER,
          file = "/Users/mattia/Desktop/Università/Magistrale/Tesi/Python TMM only for exploration/Data",
          row.names = T )
```