# Trabalho 2 | AR

Rede Aleatórias & Comunidades

André Silvestre Nº 104532 Eliane Gabriel Nº 103303
Maria João Lourenço Nº 104716 Margarida Pereira Nº 105877
Umeima Mahomed Nº 99239

16 de janeiro de 2024

## Questão 1

### Implementação do Algoritmo de Geração de Redes Aleatórias Passeio Aleatório

```
# Função para gerar redes aleatórias utilizando o Modelo Passeio Aleatório
generate_random_network <- function(initial_size) {

  # Configuração inicial de uma clique completa com n nodos
  rn <- make_full_graph(initial_size, directed = F)

  # Cada uma das redes geradas deve ter 200 nodos
  # Para isso geramos (200-valor inicial de nodos da clique)
  for (i in 1:(200-initial_size)) {

    # Adiciona um novo nodo i (1º Passo)
    rn <- add_vertices(rn, 1)

    # Escolhe um nodo existente aleatoriamente e ligá-lo ao novo nodo (2º Passo)
    selected_node <- sample(1:(vcount(rn)-1), 1)      # Escolher o nodo j
    nn <- neighbors(rn, selected_node)                # Nodos Adjacentes de j
    rn <- add_edges(rn, c(selected_node,vcount(rn)))  # Ligar j a i

    # 3º Passo:
    # Cada uma das restantes ligações une o nodo adicionado a um dos adjacentes
    # do nodo escolhido no Passo 2 com probabilidade 'p'
    # ou une o novo nodo a um nodo escolhido aleatoriamente com probabilidade '1 - p'

    # Adiciona mais 2 ligações (para perfazer as 3 que são impostas no enunciado)
    for (j in 1:2) {
      if (runif(1) < 0.8) {
        selected_node_1 <- sample(setdiff(nn, neighbors(rn, vcount(rn))), 1)
        rn <- add_edges(rn, c(vcount(rn), selected_node_1))
      }
      else {
        # Para garantir que está a escolher um nodo diferente do:
        # Novo nodo e seus vizinhos e do escolhido no Passo 2
```

```
        nn_i <- neighbors(rn, vcount(rn))
        selected_node_2 <- sample(setdiff(1:(vcount(rn)-1), c(selected_node,nn_i)), 1)
        rn <- add_edges(rn, c(selected_node_2,vcount(rn)))
      }
    }
  }

  return(rn)
}
```

```
# Parâmetros das Redes Aleatórias
initial_size_1 <- 10
initial_size_2 <- 20
```

### a) Gerar e caracterizar 10 redes aleatórias a partir de clique com 10 nodos

```
# a) Gerar e caracterizar 10 redes aleatórias a partir de clique com 10 nodos
networks_1 <- lapply(1:10,function(i) generate_random_network(initial_size_1))

# Caracterizar as redes geradas quanto à distância média (<l>),
#                                    ao coeficiente de clustering da rede (C)
#                                    e à existência de hubs (K)
distances_1 <- sapply(networks_1, function(net) mean_distance(net))
coef_clustering_1 <- sapply(networks_1, function(net) transitivity(net, type = "global"))
hubs_1 <- sapply(networks_1,
                 function(net){
                   mean(degree(net, mode = "all")*degree(net, mode = "all")/
                          mean(degree(net, mode = "all"))^2)})

result_1 <- data.frame(Rede = 1:10,
                       Distancia_Media = round(distances_1, 4),
                       Coeficiente_Clustering = round(coef_clustering_1, 4),
                       K = round(hubs_1, 4))
```

```r
cat("Resultados para clique com 10 nodos:")
```

```
## Resultados para clique com 10 nodos:
```

```r
ftable <- flextable(result_1)
ftable <- border_remove(x = ftable) %>%
  hline(i= 1, part = "header", border = fp_border(color = "gray", width = 2)) %>%
  hline_bottom(part = "body", border = fp_border(color = "grey", width = 1)) %>%
  vline(j=1:3, border = fp_border(color = "white", width = 5)) %>%
  align(j= 1:4, align = "center", part = "all") %>%
  bg(j = 2:4, bg = "darkgrey", part = "header") %>%
  color(j = 2:4, color = "white", part = "header") %>%
  set_header_labels(Rede = "Rede",
                    Distancia_Media = "Distância Média (<l>)",
                    Coeficiente_Clustering = "Coeficiente de Clustering (C)",
                    K = 'Heterogenidade (K)') %>%
  bold(bold = TRUE, part = "header") %>%
  bold(j = 1, bold = TRUE, part = "body") %>%
  color(j = 1, color = "darkgrey", part = "all") %>%
  autofit()
ftable
```

| Rede | Distância Média (<l>) | Coeficiente de Clustering (C) | Heterogenidade (K) |
|---|---|---|---|
| 1 | 3.2217 | 0.2663 | 1.7410 |
| 2 | 3.2431 | 0.2669 | 1.6186 |
| 3 | 3.3264 | 0.2670 | 1.6509 |
| 4 | 3.3539 | 0.2864 | 1.5972 |
| 5 | 3.3677 | 0.2933 | 1.5443 |
| 6 | 3.1559 | 0.2504 | 1.8095 |
| 7 | 3.2735 | 0.2738 | 1.6199 |
| 8 | 3.1357 | 0.2526 | 1.7828 |
| 9 | 3.2399 | 0.2714 | 1.6881 |
| 10 | 3.2759 | 0.2791 | 1.6633 |

### b) Gerar e caracterizar 10 redes aleatórias a partir de clique com 20 nodos

```r
# b) Gerar e caracterizar 10 redes aleatórias a partir de clique com 20 nodos
networks_2 <- lapply(1:10, function(i) generate_random_network(initial_size_2))
# Caracterizar as redes geradas quanto à distância média (<l>),
#                                       ao coeficiente de clustering da rede (C)
#                                       e à existência de hubs (K)
distances_2 <- sapply(networks_2, function(net) mean_distance(net))
coef_clustering_2 <- sapply(networks_2, function(net) transitivity(net, type = "global"))
hubs_2 <- sapply(networks_2,
                 function(net){
                   mean(degree(net, mode = "all")*degree(net, mode = "all")/
                          mean(degree(net, mode = "all"))^2)})

result_2 <- data.frame(Rede = 1:10,
                       Distancia_Media = round(distances_2, 4),
                       Coeficiente_Clustering = round(coef_clustering_2, 4),
                       K = round(hubs_2, 4))
```

```r
cat("Resultados para clique com 20 nodos:")
```

```
## Resultados para clique com 20 nodos:
```

```r
ftable <- flextable(result_2)
ftable <- border_remove(x = ftable) %>%
  hline(i= 1, part = "header", border = fp_border(color = "gray", width = 2)) %>%
  hline_bottom(part = "body", border = fp_border(color = "grey", width = 1)) %>%
  vline(j=1:3, border = fp_border(color = "white", width = 5)) %>%
  align(j= 1:4, align = "center", part = "all") %>%
  bg(j = 2:4, bg = "darkgrey", part = "header") %>%
  color(j = 2:4, color = "white", part = "header") %>%
  set_header_labels(Rede = "Rede",
                    Distancia_Media = "Distância Média (<l>)",
                    Coeficiente_Clustering = "Coeficiente de Clustering (C)",
                    K = 'Heterogenidade (K)') %>%
  bold(bold = TRUE, part = "header") %>%
  bold(j = 1, bold = TRUE, part = "body") %>%
  color(j = 1, color = "darkgrey", part = "all") %>%
  autofit()
ftable
```

| Rede | Distância Média (<l>) | Coeficiente de Clustering (C) | Heterogenidade (K) |
|---|---|---|---|
| 1 | 3.0596 | 0.4273 | 2.1499 |
| 2 | 3.0143 | 0.4089 | 2.2299 |
| 3 | 3.1070 | 0.4319 | 2.1105 |
| 4 | 3.0222 | 0.4230 | 2.1479 |
| 5 | 3.0139 | 0.4136 | 2.2004 |
| 6 | 2.9723 | 0.4094 | 2.2492 |
| 7 | 3.0223 | 0.4197 | 2.1664 |
| 8 | 3.0242 | 0.4275 | 2.1664 |
| 9 | 3.0660 | 0.4196 | 2.1790 |
| 10 | 3.0197 | 0.4208 | 2.1451 |

### c) Comparar os resultados obtidos em a) e b)

```
# c) Comparar os resultados obtidos em a) e b)
result_a_b <- data.frame(result_1, result_2[2:4])

ftable <- flextable(result_a_b)
ftable <- border_remove(x = ftable) %>%
  add_header_row(values = c("",
                            "Clique com 10 nodos",
                            "Clique com 20 nodos"), colwidths = c(1,3,3)) %>%
  hline(i= 2, part = "header", border = fp_border(color = "gray", width = 2)) %>%
  hline_bottom(part = "body", border = fp_border(color = "grey", width = 1)) %>%
  vline(j=1:6, border = fp_border(color = "white", width = 5)) %>%
  align(j= 1:7, align = "center", part = "all") %>%
  bg(i=2, j = 2:4, bg = "#DF6613", part = "header") %>%
  bg(i=2, j = 5:7, bg = "navy", part = "header") %>%
  color(i=1, j = 2:4, color = "#DF6613", part = "header") %>%
  color(i=1, j = 5:7, color = "navy", part = "header") %>%
  color(i=2, j = 2:7, color = "white", part = "header") %>%
  set_header_labels(Rede = "Rede", Distancia_Media = "<l>", Distancia_Media.1 = "<l>",
                    Coeficiente_Clustering = "C", Coeficiente_Clustering.1="C",
                    K = 'K', K.1 = 'K') %>%
  bold(bold = TRUE, part = "header") %>%
  bold(j = 1, bold = TRUE, part = "body") %>%
  color(j = 1, color = "darkgrey", part = "all") %>%
  autofit()
ftable
```

| | Clique com 10 nodos | | | Clique com 20 nodos | | |
|---|---|---|---|---|---|---|
| Rede | <l> | C | K | <l> | C | K |
| 1 | 3.2217 | 0.2663 | 1.7410 | 3.0596 | 0.4273 | 2.1499 |
| 2 | 3.2431 | 0.2669 | 1.6186 | 3.0143 | 0.4089 | 2.2299 |
| 3 | 3.3264 | 0.2670 | 1.6509 | 3.1070 | 0.4319 | 2.1105 |
| 4 | 3.3539 | 0.2864 | 1.5972 | 3.0222 | 0.4230 | 2.1479 |
| 5 | 3.3677 | 0.2933 | 1.5443 | 3.0139 | 0.4136 | 2.2004 |
| 6 | 3.1559 | 0.2504 | 1.8095 | 2.9723 | 0.4094 | 2.2492 |
| 7 | 3.2735 | 0.2738 | 1.6199 | 3.0223 | 0.4197 | 2.1664 |
| 8 | 3.1357 | 0.2526 | 1.7828 | 3.0242 | 0.4275 | 2.1664 |
| 9 | 3.2399 | 0.2714 | 1.6881 | 3.0660 | 0.4196 | 2.1790 |
| 10 | 3.2759 | 0.2791 | 1.6633 | 3.0197 | 0.4208 | 2.1451 |

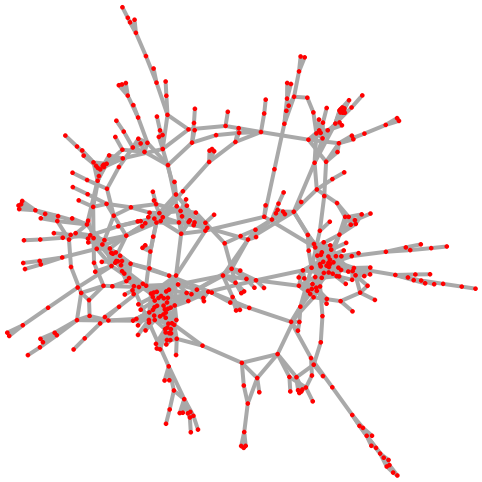
# Questão 2

## Componente Gigante da Rede de Contactos Sociais Diretos

```
# Importar a rede de um ficheiro .txt
rede <- read_graph("trab_links.txt", format = "edgelist", directed=F)

# Encontrar a componente gigante da rede
components <- clusters(rede)
giant_component <- induced.subgraph(rede,
                                    which(components$membership == which.max(components$csize)))

# Representar graficamente a componente gigante da rede
plot(giant_component,
     vertex.color= "red",vertex.label=NA, vertex.size=2,
     vertex.frame.color=NA, edge.width= 2, )
```



```
cat("Número de nodos na componente gigante:", vcount(giant_component), "\n")
```

```
## Número de nodos na componente gigante: 496
```

```
cat("Número de ligações na componente gigante:", ecount(giant_component), "\n")
```

```
## Número de ligações na componente gigante: 984
```

## Métodos de detecção de comunidades

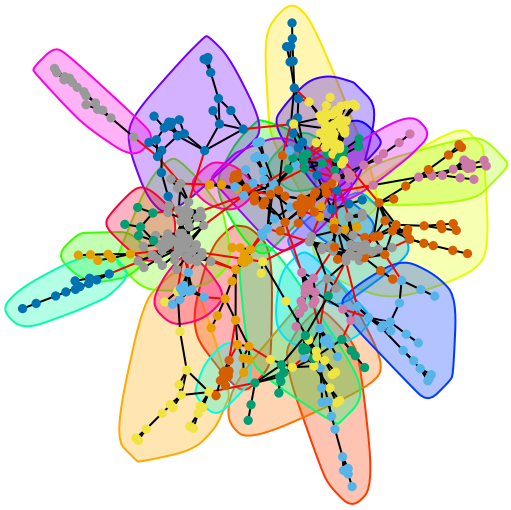
### 1 | Remoção de Pontes

```
# Iterar 5 vezes o algoritmo para verificar os resultados
results_list_cluster_edge_betweenness <- list()
for (i in 1:5) {
  set.seed(i*123) # Para reprodutibilidade
  cebd <- cluster_edge_betweenness(giant_component)
  results_list_cluster_edge_betweenness[[i]] <- data.frame(
    Método = paste("Remoção de Pontes [", i, "]", sep=""),
    Número_de_Comunidades = length(cebd),
    Dimensão_de_Cada_Comunidade = toString(table(membership(cebd)), collapse = ','),
    Modularidade = round(modularity(cebd), 3)
  )
}

# Combinar os resultados num único dataframe
all_results_cluster_edge_betweenness <- do.call(rbind,
                                                results_list_cluster_edge_betweenness)
t(all_results_cluster_edge_betweenness)
```

```
##                              [,1]
## Método                       "Remoção de Pontes [1]"
## Número_de_Comunidades        "27"
## Dimensão_de_Cada_Comunidade "18, 17, 18, 18, 18, 33, 16, 62, 6, 20, 7, 15, 11, 12, 18, 30, 5, 18, 11
## Modularidade                 "0.84"
##                              [,2]
## Método                       "Remoção de Pontes [2]"
## Número_de_Comunidades        "27"
## Dimensão_de_Cada_Comunidade "18, 17, 18, 18, 18, 33, 16, 62, 6, 20, 7, 15, 11, 12, 18, 30, 5, 18, 11
## Modularidade                 "0.84"
##                              [,3]
## Método                       "Remoção de Pontes [3]"
## Número_de_Comunidades        "27"
## Dimensão_de_Cada_Comunidade "18, 17, 18, 18, 18, 33, 16, 62, 6, 20, 7, 15, 11, 12, 18, 30, 5, 18, 11
## Modularidade                 "0.84"
##                              [,4]
## Método                       "Remoção de Pontes [4]"
## Número_de_Comunidades        "27"
## Dimensão_de_Cada_Comunidade "18, 17, 18, 18, 18, 33, 16, 62, 6, 20, 7, 15, 11, 12, 18, 30, 5, 18, 11
## Modularidade                 "0.84"
##                              [,5]
## Método                       "Remoção de Pontes [5]"
## Número_de_Comunidades        "27"
## Dimensão_de_Cada_Comunidade "18, 17, 18, 18, 18, 33, 16, 62, 6, 20, 7, 15, 11, 12, 18, 30, 5, 18, 11
## Modularidade                 "0.84"
```

```
plot(cebd, giant_component, vertex.label=NA, vertex.size=4,
     vertex.frame.color=NA, edge.width= 1)
```



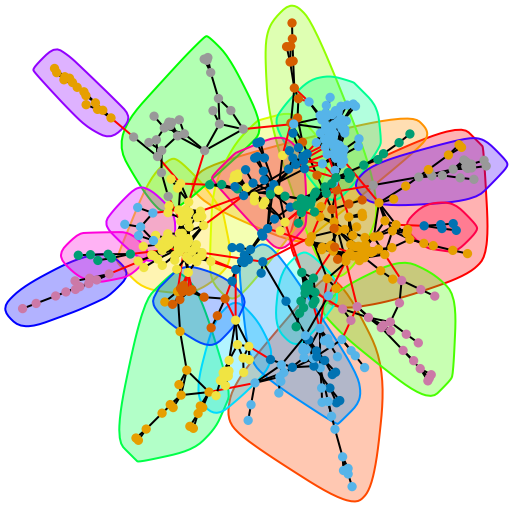
### 2 | Otimização de Modularidade (*Fast Greedy*)

```
# Iterar 5 vezes o algoritmo para verificar os resultados
results_list_cluster_fast_greedy <- list()
for (i in 1:5) {
  set.seed(i*123) # Para reprodutibilidade
  cfgr <- cluster_fast_greedy(giant_component)
  results_list_cluster_fast_greedy[[i]] <- data.frame(
    Método = paste("Fast Greedy [", i, "]", sep=""),
    Número_de_Comunidades = length(cfgr),
    Dimensão_de_Cada_Comunidade = toString(table(membership(cfgr)), collapse = ','),
    Modularidade = round(modularity(cfgr), 3)
  )
}

# Combinar os resultados num único dataframe
all_results_cluster_fast_greedy <- do.call(rbind,
                                           results_list_cluster_fast_greedy)
t(all_results_cluster_fast_greedy)
```

```
##                             [,1]
## Método                      "Fast Greedy [1]"
## Número_de_Comunidades       "21"
## Dimensão_de_Cada_Comunidade "60, 35, 30, 63, 37, 19, 20, 26, 18, 57, 16, 18, 15, 11, 11, 16, 14, 6, 
## Modularidade                "0.838"
##                             [,2]
## Método                      "Fast Greedy [2]"
## Número_de_Comunidades       "21"
## Dimensão_de_Cada_Comunidade "60, 35, 30, 63, 37, 19, 20, 26, 18, 57, 16, 18, 15, 11, 11, 16, 14, 6, 
## Modularidade                "0.838"
##                             [,3]
## Método                      "Fast Greedy [3]"
## Número_de_Comunidades       "21"
## Dimensão_de_Cada_Comunidade "60, 35, 30, 63, 37, 19, 20, 26, 18, 57, 16, 18, 15, 11, 11, 16, 14, 6, 
## Modularidade                "0.838"
##                             [,4]
## Método                      "Fast Greedy [4]"
## Número_de_Comunidades       "21"
## Dimensão_de_Cada_Comunidade "60, 35, 30, 63, 37, 19, 20, 26, 18, 57, 16, 18, 15, 11, 11, 16, 14, 6, 
## Modularidade                "0.838"
##                             [,5]
## Método                      "Fast Greedy [5]"
## Número_de_Comunidades       "21"
## Dimensão_de_Cada_Comunidade "60, 35, 30, 63, 37, 19, 20, 26, 18, 57, 16, 18, 15, 11, 11, 16, 14, 6, 
## Modularidade                "0.838"
```

```
plot(cfgr, giant_component, vertex.label=NA, vertex.size=4,
     vertex.frame.color=NA, edge.width= 1)
```



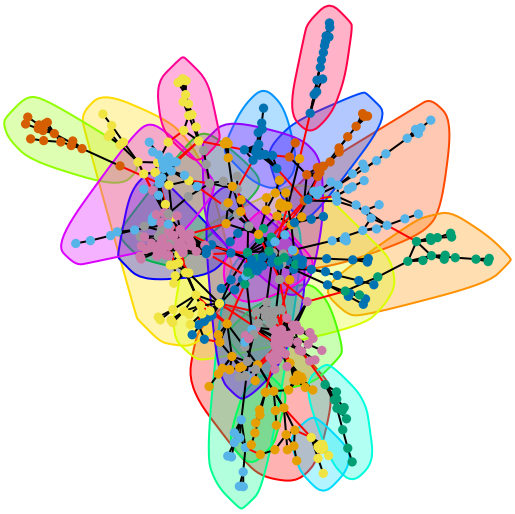
### 3 | Algoritmo de Louvain

```
# Iterar 5 vezes o algoritmo para verificar os resultados
results_list_cluster_louvain <- list()
for (i in 1:5) {
  set.seed(i*123) # Para reprodutibilidade
  clor <- cluster_louvain(giant_component)
  results_list_cluster_louvain[[i]] <- data.frame(
    Método = paste("Algoritmo de Louvain [", i, "]", sep=""),
    Número_de_Comunidades = length(clor),
    Dimensão_de_Cada_Comunidade = toString(table(membership(clor)), collapse = ','),
    Modularidade = round(modularity(clor), 3)
  )
}

# Combinar os resultados num único dataframe
all_results_cluster_louvain <- do.call(rbind,
                                       results_list_cluster_louvain)
t(all_results_cluster_louvain)
```

```
##                              [,1]
## Método                       "Algoritmo de Louvain [1]"
## Número_de_Comunidades        "21"
## Dimensão_de_Cada_Comunidade "27, 12, 31, 18, 20, 48, 16, 18, 46, 6, 46, 20, 18, 11, 16, 24, 25, 26, 4
## Modularidade                 "0.846"
##                              [,2]
## Método                       "Algoritmo de Louvain [2]"
## Número_de_Comunidades        "20"
## Dimensão_de_Cada_Comunidade "27, 12, 35, 18, 32, 66, 16, 18, 46, 6, 47, 20, 18, 11, 16, 16, 25, 26, 1
## Modularidade                 "0.845"
##                              [,3]
## Método                       "Algoritmo de Louvain [3]"
## Número_de_Comunidades        "21"
## Dimensão_de_Cada_Comunidade "32, 12, 35, 18, 21, 61, 16, 18, 46, 6, 18, 11, 16, 16, 17, 44, 25, 26, 3
## Modularidade                 "0.846"
##                              [,4]
## Método                       "Algoritmo de Louvain [4]"
## Número_de_Comunidades        "19"
## Dimensão_de_Cada_Comunidade "34, 12, 35, 18, 23, 55, 16, 18, 46, 6, 47, 27, 18, 11, 16, 20, 25, 40, 1
## Modularidade                 "0.844"
##                              [,5]
## Método                       "Algoritmo de Louvain [5]"
## Número_de_Comunidades        "21"
## Dimensão_de_Cada_Comunidade "32, 35, 18, 36, 54, 16, 46, 19, 6, 16, 11, 12, 16, 16, 45, 25, 26, 27, 1
## Modularidade                 "0.843"
```

```
plot(clor, giant_component, vertex.label=NA, vertex.size=4,
     vertex.frame.color=NA, edge.width= 1)
```



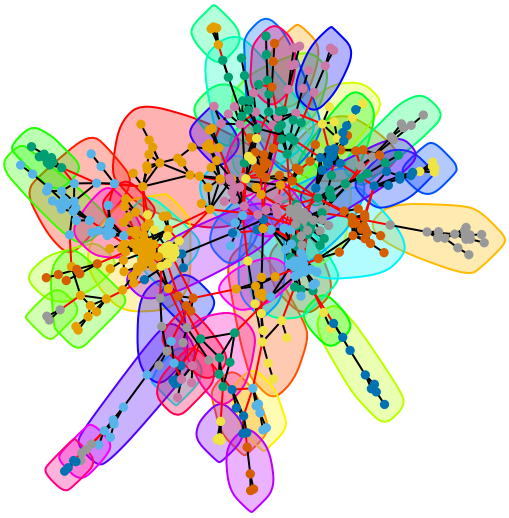
### 4 | Propagação de Etiquetas

```r
# Iterar 5 vezes o algoritmo para verificar os resultados
results_list_cluster_label_prop <- list()
for (i in 1:5) {
  set.seed(i*123)
  clpr <- cluster_label_prop(giant_component)
  results_list_cluster_label_prop[[i]] <- data.frame(
    Método = paste("Propagação de Etiquetas [", i, "]", sep=""),
    Número_de_Comunidades = length(clpr),
    Dimensão_de_Cada_Comunidade = toString(table(membership(clpr)), collapse = ','),
    Modularidade = round(modularity(clpr), 3)
  )
}
all_results_cluster_label_prop <- do.call(rbind, results_list_cluster_label_prop)

t(all_results_cluster_label_prop)
```

```
##                              [,1]
## Método                       "Propagação de Etiquetas [1]"
## Número_de_Comunidades        "58"
## Dimensão_de_Cada_Comunidade "9, 22, 12, 12, 9, 4, 5, 12, 3, 47, 6, 13, 7, 5, 6, 23, 7, 5, 3, 11, 8, 
## Modularidade                 "0.790"
##                              [,2]
## Método                       "Propagação de Etiquetas [2]"
## Número_de_Comunidades        "56"
## Dimensão_de_Cada_Comunidade "13, 9, 8, 5, 8, 5, 3, 7, 3, 47, 6, 12, 7, 8, 6, 21, 9, 11, 3, 11, 11, 1
## Modularidade                 "0.804"
##                              [,3]
## Método                       "Propagação de Etiquetas [3]"
## Número_de_Comunidades        "53"
## Dimensão_de_Cada_Comunidade "13, 12, 11, 13, 6, 7, 5, 12, 19, 4, 57, 6, 7, 6, 16, 13, 7, 6, 3, 9, 11
## Modularidade                 "0.794"
##                              [,4]
## Método                       "Propagação de Etiquetas [4]"
## Número_de_Comunidades        "57"
## Dimensão_de_Cada_Comunidade "14, 10, 9, 11, 5, 13, 5, 6, 53, 6, 13, 8, 6, 13, 9, 6, 3, 11, 11, 12, 18
## Modularidade                 "0.795"
##                              [,5]
## Método                       "Propagação de Etiquetas [5]"
## Número_de_Comunidades        "57"
## Dimensão_de_Cada_Comunidade "27, 10, 7, 9, 5, 7, 5, 16, 46, 6, 13, 6, 8, 6, 7, 3, 8, 11, 12, 2, 16, 
## Modularidade                 "0.804"
```

```r
plot(clpr, giant_component, vertex.label=NA, vertex.size=4,
     vertex.frame.color=NA, edge.width= 1)
```



```
# Combinar todos os resultados num único dataframe
resultados_completos <- bind_rows(all_results_cluster_edge_betweenness,
                                  all_results_cluster_fast_greedy,
                                  all_results_cluster_louvain,
                                  all_results_cluster_label_prop)
ftable <- flextable(resultados_completos)
ftable <- border_remove(x = ftable) %>%
  hline(i= 1, part = "header", border = fp_border(color = "navy", width = 2)) %>%
  hline_bottom(part = "body", border = fp_border(color = "navy", width = 1)) %>%
  vline(j=1:3, border = fp_border(color = "white", width = 5)) %>%
  align(j= 1:4, align = "center", part = "all") %>%
  bg(i=1, j = 2:4, bg = "navy", part = "header") %>%
  color(i=1, j = 2:4, color = "white", part = "header") %>%
  set_header_labels(Número_de_Comunidades = "Número de Comunidades",
                    Dimensão_de_Cada_Comunidade = "Dimensão das Comunidade",
                    Modularidade = 'Modularidade') %>%
  bold(bold = TRUE, part = "header") %>%  bold(j = 1, bold = TRUE, part = "body") %>%
  color(i=1, j = 1, color = "navy", part = "header") %>%
  autofit() %>% hrule(rule = "exact", part = "all")
ftable
```

| Método | Número de Comunidades | |
|---|---|---|
| **Remoção de Pontes [1]** | 27 | 18 |
| **Remoção de Pontes [2]** | 27 | 18 |
| **Remoção de Pontes [3]** | 27 | 18 |
| **Remoção de Pontes [4]** | 27 | 18 |
| **Remoção de Pontes [5]** | 27 | 18 |
| **Fast Greedy [1]** | 21 | |
| **Fast Greedy [2]** | 21 | |
| **Fast Greedy [3]** | 21 | |
| **Fast Greedy [4]** | 21 | |
| **Fast Greedy [5]** | 21 | |
| **Algoritmo de Louvain [1]** | 21 | |
| **Algoritmo de Louvain [2]** | 20 | |
| **Algoritmo de Louvain [3]** | 21 | |
| **Algoritmo de Louvain [4]** | 19 | |
| **Algoritmo de Louvain [5]** | 21 | |
| **Propagação de Etiquetas [1]** | 58 | 9, 22, 12, 12, 9, 4, 5, 12, 3, 47, 6, 13, 7, |
| **Propagação de Etiquetas [2]** | 56 | 13, 9, 8, 5, 8, 5, 3, 7, 3, 47, 6, 12, 7, 8, |
| **Propagação de Etiquetas [3]** | 53 | 13, 12, 11, 13, 6, 7, 5, 12, 19, 4, 57 |
| **Propagação de Etiquetas [4]** | 57 | 14, 10, 9, 11, 5, 13, 5, 6, 53, 6, 13, 8, 6 |
| **Propagação de Etiquetas [5]** | 57 | 27, 10, 7, 9, 5, 7, 5, 16, 46, 6, 13, 6, 8 |