INSTITUTO UNIVERSITÁRIO DE LISBOA

Análise de Redes - Trabalho de Grupo n.º 2

Ciência de Dados - PL - 3º ano | Professora: Maria João Frazão Lopes

Catarina Castanheira, 92478 João Martins, 93259 Joel Paula, 93392

03/01/2022

QUESTÃO 1:

> Suponha que pretende gerar uma rede aleatória não orientada com 100 nodos e grau médio aproximadamente igual a 4. Qual deve ser a probabilidade utilizada na geração da rede? Gere esta rede.

O grau médio seria de aproximadamente 100, se todos os nodos estivessem ligados entre si. Neste caso teremos uma probabilidade de 4% - que cada nodo esteja ligado apenas a 4 dos 100 nodos.

É possível determinar o grau médio de uma rede conhecendo o número de nodos e a probabilidade de geração utilizada:

$\langle k \rangle = p(N-1)$

Isto significa que, resolvendo a equação em função de p, temos:

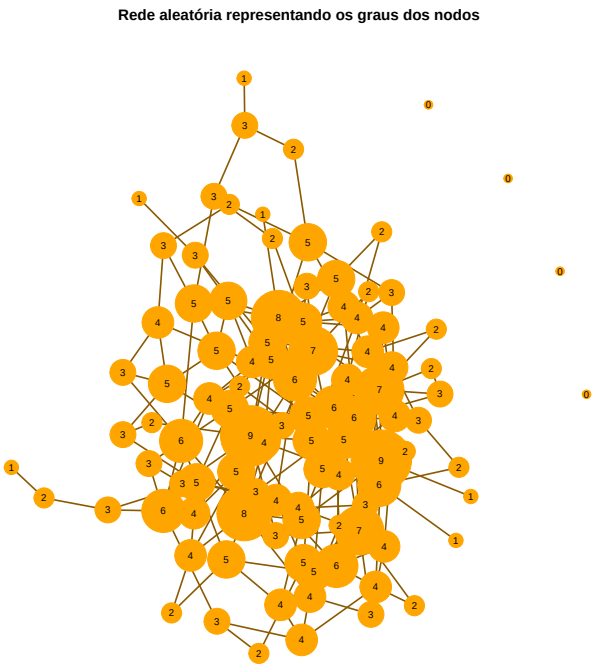
$p = \langle k \rangle / (N-1)$

No nosso caso concreto, o objectivo é obtermos $\langle k \rangle = 4$, tendo $N = 100$. Logo, a probabilidade que deverá ser usada na geração da rede aleatória é:

$p = 4/(100-1) \Leftrightarrow p = 4/99 = 0.04$ (aprox.)

Ou seja, para obtermos uma rede aletória de 100 nodos e um grau médio de 4, deveremos usar uma probabilidade de 4% na geração das suas ligações.

```
set.seed(42)
graph1 <- sample_gnp(100, 0.04)
degrees <- degree(graph1, mode="all")
par(mar=c(0,0,1,0))
# Gráfico com grau
plot(graph1,  vertex.size=3+degrees*2, edge.color="orange4"
     , edge.width=2
     , vertex.label=degrees
     , vertex.label.cex=0.75
     , vertex.label.color="black"
     , vertex.label.family="sans"
     , vertex.frame.color="orange"
     , vertex.color = "orange"
     , main="Rede aleatória representando os graus dos nodos"
     )
```



Caracterize esta rede quanto ao grau médio dos nodos, à conectividade, distância média e existência de triângulos. Aplique ainda métodos de identificação de comunidades.

Grau Médio

```
# grau médio
mean(degree(graph1))
```

```
## [1] 3.76
```

O grau médio é próximo de 4, tal como esperaríamos.

Conectividade

```
degree(graph1)
```

```
##   [1] 2 5 2 4 4 5 3 5 5 4 5 6 6 3 9 5 2 1 4 5 2 6 4 1 1 5 0 2 3 4 2 4 3 4 7 6 1
##  [38] 1 0 0 2 4 5 0 2 3 5 4 2 4 8 3 3 4 3 2 5 3 6 8 6 3 4 2 5 7 4 3 3 2 4 7 3 2
##  [75] 3 6 3 4 2 9 2 5 4 5 5 5 2 3 1 3 5 4 3 5 4 3 5 4 5 4
```

Vemos que existem 4 nodos com grau zero, o que indica que não têm qualquer ligação. Neste caso estamos perante uma rede desconexa.

Portanto, existem 4 nodos isolados e uma componente gigante.

```
components(graph1)
```

```
## $membership
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1
##  [38] 1 3 4 1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## $csize
## [1] 96  1  1  1  1
##
## $no
## [1] 5
```

Associação

```
# Associação de grau
assortativity_degree(graph1)
```

```
## [1] -0.08082738
```

É negativo mas muito próximo de zero, por isso não é um rede associativa mas também não se pode concluir que é Não associativa.

Vamos olhar agora para o método de medição da associação de grau com base no grau médio dos nodos adjacentes:

```
knn(graph1)$knnk
```

```
## [1] 5.166667 4.781250 4.666667 4.952381 4.740000 4.428571 4.095238 4.812500
## [9] 4.277778
```

A função tem uma tendência decrescente, mas não estritamente decrescente, oscilando. Por isso mantemos que não é uma rede associativa mas também não pode ser classificada como não associativa.

Distância média

```
# distância média
mean_distance(graph1)
```

```
## [1] 3.474123
```

```
log10(100)
```

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

```
#diâmetro
diameter(graph1)
```

```
## [1] 8
```

A distância média é grande, já que se afasta substancialmente de $log10(N)$.

A maior distância entre nodos (conectados) é de 8.

Existência de triângulos

```
# Coeficiente de clustering
transitivity(graph1, type="global")
```

```
## [1] 0.03453237
```

É um número baixo de triângulos, já que o coeficiente de *clustering* é um rácio entre o número de triângulos e o número total de ternos conexos e este é baixo.

Identificação de comunidades

Usando o método do corte mínimo:

```
min_cut(graph1, value.only = F)
```

```
## $value
## [1] 0
##
## $cut
## + 0/188 edges from ff27438:
##
## $partition1
## + 96/100 vertices, from ff27438:
##  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
## [20]  20  21  22  23  24  25  26  28  29  30  31  32  33  34  35  36  37  38  41
## [39]  42  43  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61
## [58]  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
## [77]  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
## [96] 100
##
## $partition2
## + 4/100 vertices, from ff27438:
## [1] 27 39 40 44
```

Temos um conjunto com os nodos ligados (a componente gigante) e um outro com os nodos isolados. Com um corte de dimensão zero.

Usando o método das cliques:

```
sapply(cliques(graph1),length)
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
##  [38] 1 2 1 2 2 1 2 1 2 2 2 1 2 2 2 2 1 2 1 2 1 2 2 1 2 1 2 1 2 2 2 2 1 2 2 1 2
##  [75] 1 2 2 2 1 2 2 2 1 2 2 1 2 1 2 1 2 2 2 1 2 2 1 2 2 2 1 2 2 1 2 1 2 1 2 1 2
## [112] 2 2 1 2 2 1 2 2 2 1 2 2 2 2 1 2 2 2 2 2 1 2 2 1 2 1 2 1 2 2 2 2 2 1 2 2 1
## [149] 2 3 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 3 2 1 2 2 2 2 1
## [186] 2 3 2 2 1 2 2 1 2 2 3 2 2 2 2 2 2 1 2 2 2 1 2 2 1 2 2 2 1 2 2 1 2 2 2 2 1
## [223] 2 2 2 2 1 2 2 2 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 1 2
## [260] 2 2 1 2 2 3 2 2 1 2 2 2 2 1 2 2 1 2 3 2 2 2 1 2 2 2 2 2 2 1 2 2 3 2 3 2 2
```

As cliques parecem fazer divisões muito pequenas, de 1, 2 ou 3 nodos. Sendo que as maiores cliques têm 3 nodos. O número excessivo de comunidades é um dos problemas conhecidos deste algoritmo.

```
largest_cliques(graph1)
```

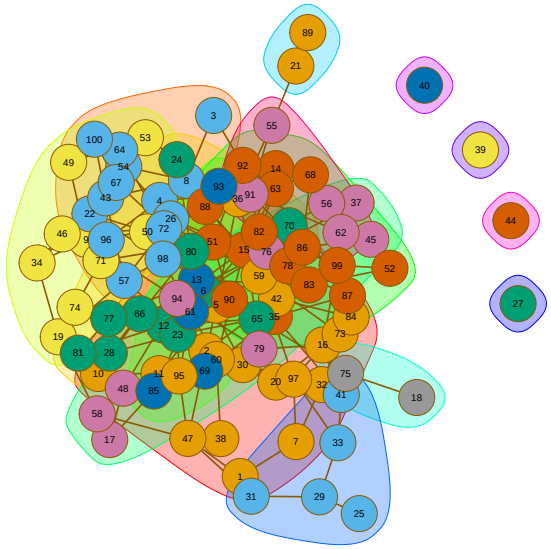
```
## [[1]]
## + 3/100 vertices, from ff27438:
## [1] 92 82 63
##
## [[2]]
## + 3/100 vertices, from ff27438:
## [1] 92 82 51
##
## [[3]]
## + 3/100 vertices, from ff27438:
## [1] 85 61 69
##
## [[4]]
## + 3/100 vertices, from ff27438:
## [1] 85 61 60
##
## [[5]]
## + 3/100 vertices, from ff27438:
## [1] 84 32 73
##
## [[6]]
## + 3/100 vertices, from ff27438:
## [1] 80  6 66
##
## [[7]]
## + 3/100 vertices, from ff27438:
## [1] 76  6 72
##
## [[8]]
## + 3/100 vertices, from ff27438:
## [1] 59  5 36
```

Função que mostra dados sobre *clusters*:

```
show.cluster <- function(g1, cl) {
  plot(cl, g1
           , edge.color="orange4"
           , edge.width=2
           , vertex.label.cex=0.75
           , vertex.label.color="black"
           , vertex.label.family="sans"
           , vertex.frame.color="orange4"
           )
  print(c("Número de clusters:", length(cl)))
  print("Tamanho dos clusters:")
  print(sizes(cl))
  print(c("modularidade:", modularity(cl)))
  print("Pertença a clusters:")
  membership(cl)
}
```

Verificando o método de clustering pela remoção de pontes:

```
show.cluster(graph1, cluster_edge_betweenness(graph1))
```



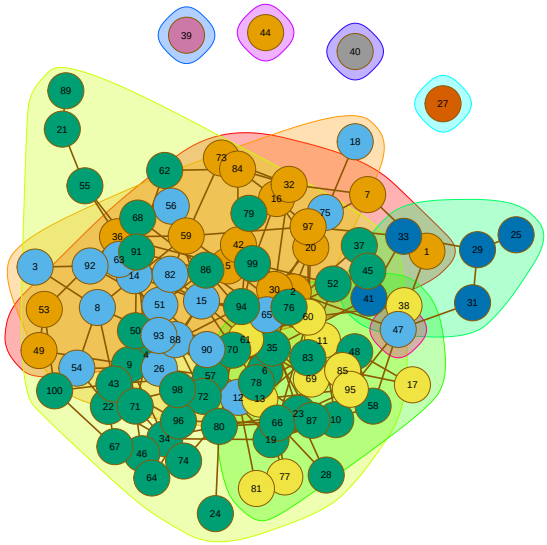
```
## [1] "Número de clusters:" "15"
## [1] "Tamanho dos clusters:"
## Community sizes
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
## 20 14 11  9  5 16  8  2  2  5  1  1  1  1  4
## [1] "modularidade:"     "0.473276935264826"
## [1] "Pertença a clusters:"

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

```
##  [26]  2 11  3 10  1 10  1 10  4  6  1  7  1 12 13 10  1  2 14  7  4  1  7  4  4
##  [51]  6  6  4  2 15 15  2  7  1  1  5 15  6  2  3  3  2  6  5  3  4  2  1  4  8
##  [76]  7  3  6  7  3  3  6  6  1  5  6  6  6  9  6 15  6  5  7  1  2  1  2  6  2
```

Usando o método de clustering pela propagação de etiquetas:

```
set.seed(42)
show.cluster(graph1, cluster_label_prop(graph1))
```



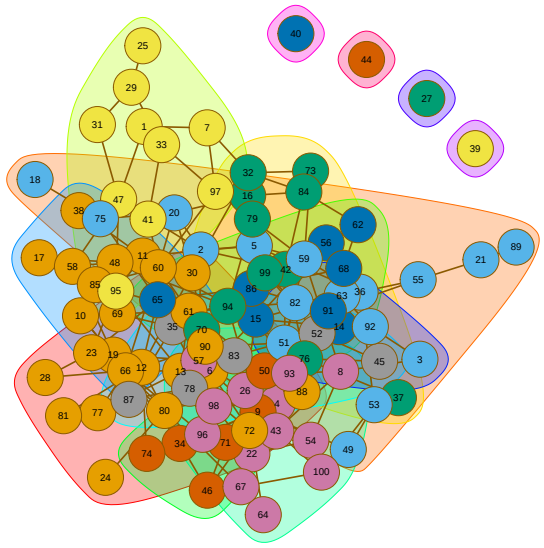
```
## [1] "Número de clusters:" "10"
## [1] "Tamanho dos clusters:"
## Community sizes
##  1  2  3  4  5  6  7  8  9 10
## 16 18 45 11  5  1  1  1  1  1
## [1] "modularidade:"     "0.391381846989588"
## [1] "Pertença a clusters:"

##   [1]  1  1  2  3  1  3  1  2  3  3  4  2  4  2  2  1  4  2  3  1  3  3  3  3  5
##  [26]  2  6  3  5  1  5  1  5  3  3  1  3  4  7  8  5  1  3  9  3  3 10  3  1  3
##  [51]  2  3  1  2  3  2  3  3  1  4  4  3  2  3  2  3  3  3  4  3  3  3  1  3  2
##  [76]  3  4  3  3  3  4  2  3  1  4  3  3  2  3  2  3  2  2  3  4  3  1  3  3  3
```

Usámos o `set.seed(42)` para estabilizar numa mesma solução, para análise.

Usando o método da otimização de modularidade:

```
show.cluster(graph1, cluster_fast_greedy(graph1))
```



```
## [1] "Número de clusters:" "14"
## [1] "Tamanho dos clusters:"
## Community sizes
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14
## 13 12 11 10  8  6 13  6 12  5  1  1  1  1
## [1] "modularidade:"     "0.473970122227252"
## [1] "Pertença a clusters:"

##   [1]  4  2 10  7  2  1  4  7  6  9  9  1  1  5  5  3  9  2  9  2  2  7  1  1  4
##  [26]  7 11  1  4  1  4  3  4  6  8  2  3  9 12 13  4  3  7 14  8  6  4  9  2  6
##  [51] 10  8  2  7  2  5  7  9  2  9  9  5 10  7  5  9  7  5  9  3  6  1  3  6  2
##  [76]  3  1  8  3  1  1 10  8  3  9  5  8  1  2  1  5 10  7  3  4  7  4  7  3  7
```

QUESTÃO 2:

Utilize o programa seguinte para gerar a rede aleatória rn2:

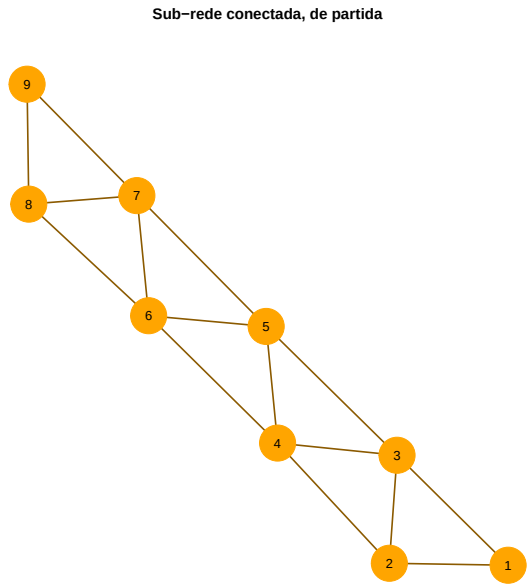
```
set.seed(42)
rn2 <- graph(edge=c(1,2,1,3,2,3,3,4,3,5,4,5,5,6,5,7,6,7,7,8,7,9,8,9,2,4,4,6,6,8)
             ,n=100
             ,directed=F);
x = 9;
y = 15;
for (i in 1:91) {
  new <- floor(runif(1,min=1,max=x));
  nn <- neighbors(rn2,new);
  x = x+1;
  y = y+1;
```

```
  rn2 <- add_edges(rn2,c(new,x));
  newr <- runif(1);
  y = y+1;
  if (newr < 0.75) {
    new1 <- floor(runif(1,min=1,max=degree(rn2,new,mode="all")));
    rn2 <- add_edges(rn2,c(x,nn[new1]))}
  else {
    new2 <- new;
    while (new==new2) new2<-floor(runif(1,min=1,max=x-1));
    rn2 <- add_edges(rn2,c(new2,x))};
}
```

Qual o método utilizado nesta geração? Justifique.

O método utilizado nesta geração é o Modelo do Passeio Aleatório. Neste método, começamos a criação de uma rede aleatória a partir de uma pequena sub-rede - no programa temos uma subrede conexa composta por 9 nodos, com a representação abaixo:

```
set.seed(42)
plot(graph(edge=c(1,2,1,3,2,3,3,4,3,5,4,5,5,6,5,7,6,7,7,8,7,9,8,9,2,4,4,6,6,8),directed=F)
     , edge.color="orange4"
     , edge.width=2
     , vertex.label.color="black"
     , vertex.label.family="sans"
     , vertex.frame.color="orange"
     , vertex.color = "orange"
     , main="Sub-rede conectada, de partida")
```



Na prática, a rede gerada pelo programa contém já os 100 nodos que se pretende utilizar, mas os restantes 91 não têm, na fase inicial, ainda qualquer ligação. Com a construção da rede aleatória eles serão iterativa-

mente adicionados. Em cada iteração, ou seja, por cada um dos 91 nodos que serão adicionados à subrede conexa já existente, temos:

- o método selecciona aleatoriamente a partir de uma distribuição uniforme um dos nodos da subrede ("new"), e identifica quais os respectivos nodos adjacentes. Cria entretanto uma ligação entre o nodo seleccionado e o próximo a ser integrado na sub-rede (ou seja, um nodo do subconjunto que têm grau zero; na primeira iteração, por exemplo, cria uma nova ligação entre o nodo 10 e um dos da subrede conexa);
- é gerado um número aleatório que dita uma probabilidade (p = 0.75 e q = 0.25), que é usada da seguinte forma:
  - se obtivermos p (com probabilidade de 75%), então é criada mais uma ligação entre o novo nodo e um dos adjacentes de "new" (já identificado no início da iteração) - criando um triângulo;
  - se obtivermos q (1-p), então é criada mais uma ligação entre o novo nodo e um dos já pertencentes à subrede conexa, seleccionado aleatoriamente a partir de uma distribuição uniforme;

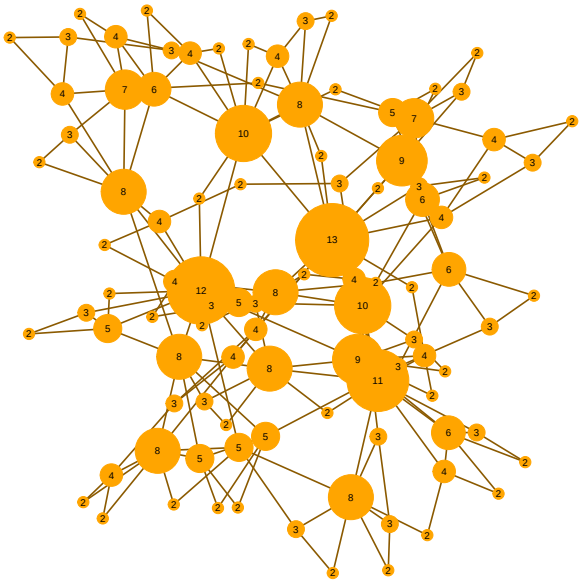
Este método termina quando todos os 91 nodos tiverem sido adicionados à rede através da criação das ligações.

Este é um método de geração de redes aleatórias que não escolhe os nodos adjacentes com base no seu grau. Esta escolha é aleatória.

> Caracterize esta rede quanto ao grau médio dos nodos, à conectividade, distância média e existência de triângulos. Aplique ainda métodos de identificação de comunidades.

```
set.seed(42)
degrees <- degree(rn2, mode="all")
par(mar=c(0,0,1,0))
# Gráfico com grau
plot(rn2,  vertex.size=degrees*2, edge.color="orange4"
     , edge.width=2
     , vertex.label=degrees
     , vertex.label.cex=0.75
     , vertex.label.color="black"
     , vertex.label.family="sans"
     , vertex.frame.color="orange"
     , vertex.color = "orange"
     , main="Rede aleatória 2 representando os graus dos nodos"
     )
```

Rede aleatória 2 representando os graus dos nodos



```
# Grau médio
mean(degree(rn2))
```

```
## [1] 3.94
```

O grau médio é próximo de 4, em semelhança à primeira rede.

```
# Conectividade
degree(rn2)
```

```
##   [1]  8  8 12  8  9 11 10 13  8 10  4  8  4  9  8  5  5  5  4  6  6  4  3  6  4
##  [26]  7  3  5  6  2  8  2  4  4  4  3  4  5  3  4  2  7  3  2  5  4  3  2  3  2
##  [51]  3  3  2  4  3  3  3  2  3  2  2  2  3  2  4  2  2  2  3  2  2  2  3  2  3
##  [76]  3  2  3  2  2  2  2  2  2  2  4  3  2  2  2  2  2  2  2  2  2  2  2  2  2
```

Não existem nodos com grau igual a 0, o que quer dizer que existe um caminho entre qualquer par de nodos. Neste caso estamos perante uma rede conexa.

```
# Associação de grau
assortativity_degree(rn2)
```

```
## [1] 0.004358032
```

É muito próximo de zero, por isso não é um rede associativa mas também não se pode concluir que é Não associativa.

```
knn(rn2)$knnk
```

```
##  [1]      NaN 6.050000 5.333333 5.500000 5.100000 5.416667 4.214286 5.392857
##  [9] 6.000000 6.050000 6.454545 5.583333 5.923077
```

```
# Distância média
mean_distance(rn2)
```

```
## [1] 3.675758
```

```
log10(100)
```

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

A distância média é grande, já que se afasta substancialmente de $log10(N)$.

```
# Diâmetro
diameter(rn2)
```

```
## [1] 7
```

A maior distância entre nodos (conectados) é de 7.

```
# Existência de triângulos
transitivity(rn2, type = "global")
```

```
## [1] 0.2491694
```

O rácio é bastante superior a zero, embora não se aproxime de 1 - revela a existência de bastantes triângulos.

Identificação de comunidades

Usando o método do corte mínimo:

```
set.seed(42)
min_cut(rn2, value.only = F)
```

```
## $value
## [1] 2
##
## $cut
## + 2/197 edges from 00673bf:
## [1] 37--53 31--53
##
## $partition1
## + 1/100 vertex, from 00673bf:
## [1] 53
##
```

```
## $partition2
## + 99/100 vertices, from 00673bf:
##  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
## [20]  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
## [39]  39  40  41  42  43  44  45  46  47  48  49  50  51  52  54  55  56  57  58
## [58]  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77
## [77]  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96
## [96]  97  98  99 100
```

Mais uma vez, este método separa a rede em dois com dimensão dois - uma componente gigante e um nodo isolado.

Usando o método das cliques:

```
sapply(cliques(rn2),length)
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 2 2
##  [38] 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2 1 2 1 2 2 1 2 2 1 2 1 2 1 2 1 2 1 2 2
##  [75] 2 1 2 2 1 2 2 1 2 2 1 2 1 2 1 2 1 2 2 1 2 1 2 1 2 1 2 1 2 2 1 2 2
## [112] 1 2 2 2 1 2 2 1 2 3 2 3 2 3 2 3 3 2 2 2 2 2 2 1 2 3 2 1 2 3 2 1 2 3 2 1 2
## [149] 3 2 1 2 3 2 1 2 3 2 1 2 3 2 2 1 2 2 1 2 3 2 1 2 3 2 1 2 3 2 1 2 3 2 1 2 3
## [186] 2 1 2 3 2 1 2 3 2 1 2 3 2 1 2 2 1 2 3 2 3 2 1 2 3 2 2 1 2 2 1 2 3 2 1 2 3
## [223] 2 1 2 3 2 2 1 2 3 2 1 2 3 3 2 2 1 2 3 2 1 2 3 2 3 2 2 1 2 2 3 2 1 2 3 3 2
## [260] 2 1 2 3 2 3 2 2 1 2 3 2 3 2 3 2 3 2 2 1 2 3 2 2 2 1 2 3 2 3 3 2 2 2 2 1 2
## [297] 3 2 3 2 2 2 1 2 3 3 2 2 2 1 2 3 3 2 2 3 2 3 2 3 3 2 2 2 1 2 3 2 3 3 2 3 2
## [334] 3 3 2 2 2 2 1 2 2 3 2 3 3 2 3 2 3 3 2 2 2 1 2 3 3 3 3 2 3 2 3 2 3 2 3 2 2
## [371] 2 2
```

As cliques parecem fazer divisões muito pequenas, de 1, 2 ou 3 nodos. Sendo que as maiores cliques têm 3 nodos. O número excessivo de comunidades é um problema conhecido deste algoritmo.

```
largest_cliques(rn2)
```

```
## [[1]]
## + 3/100 vertices, from 00673bf:
## [1] 98 31 49
##
## [[2]]
## + 3/100 vertices, from 00673bf:
## [1] 97 12 78
##
## [[3]]
## + 3/100 vertices, from 00673bf:
## [1] 96 54 59
##
## [[4]]
## + 3/100 vertices, from 00673bf:
## [1] 95  4 75
##
## [[5]]
## + 3/100 vertices, from 00673bf:
## [1] 94 15 25
##
```

```
## [[6]]
## + 3/100 vertices, from 00673bf:
## [1] 93  5 86
##
## [[7]]
## + 3/100 vertices, from 00673bf:
## [1] 92 15 17
##
## [[8]]
## + 3/100 vertices, from 00673bf:
## [1] 91 16 19
##
## [[9]]
## + 3/100 vertices, from 00673bf:
## [1] 90  4  6
##
## [[10]]
## + 3/100 vertices, from 00673bf:
## [1] 89 31 76
##
## [[11]]
## + 3/100 vertices, from 00673bf:
## [1] 86  5  6
##
## [[12]]
## + 3/100 vertices, from 00673bf:
## [1] 85 40 69
##
## [[13]]
## + 3/100 vertices, from 00673bf:
## [1] 83 21 73
##
## [[14]]
## + 3/100 vertices, from 00673bf:
## [1] 82 26 46
##
## [[15]]
## + 3/100 vertices, from 00673bf:
## [1] 81  5 57
##
## [[16]]
## + 3/100 vertices, from 00673bf:
## [1] 80  1 43
##
## [[17]]
## + 3/100 vertices, from 00673bf:
## [1] 79 28 38
##
## [[18]]
## + 3/100 vertices, from 00673bf:
## [1] 78 12 65
##
## [[19]]
## + 3/100 vertices, from 00673bf:
```

```
## [1] 76 31 55
##
## [[20]]
## + 3/100 vertices, from 00673bf:
## [1] 75  2  4
##
## [[21]]
## + 3/100 vertices, from 00673bf:
## [1] 74  3 10
##
## [[22]]
## + 3/100 vertices, from 00673bf:
## [1] 73 21 47
##
## [[23]]
## + 3/100 vertices, from 00673bf:
## [1] 72 10 65
##
## [[24]]
## + 3/100 vertices, from 00673bf:
## [1] 71 18 36
##
## [[25]]
## + 3/100 vertices, from 00673bf:
## [1] 68 29 51
##
## [[26]]
## + 3/100 vertices, from 00673bf:
## [1] 67  9 27
##
## [[27]]
## + 3/100 vertices, from 00673bf:
## [1] 65 10 12
##
## [[28]]
## + 3/100 vertices, from 00673bf:
## [1] 64 28 38
##
## [[29]]
## + 3/100 vertices, from 00673bf:
## [1] 61 24 35
##
## [[30]]
## + 3/100 vertices, from 00673bf:
## [1] 60 42 56
##
## [[31]]
## + 3/100 vertices, from 00673bf:
## [1] 59 33 54
##
## [[32]]
## + 3/100 vertices, from 00673bf:
## [1] 57  5  7
##
```

```
## [[33]]
## + 3/100 vertices, from 00673bf:
## [1] 56 14 42
##
## [[34]]
## + 3/100 vertices, from 00673bf:
## [1] 51  6 29
##
## [[35]]
## + 3/100 vertices, from 00673bf:
## [1] 50  3 18
##
## [[36]]
## + 3/100 vertices, from 00673bf:
## [1] 49 17 31
##
## [[37]]
## + 3/100 vertices, from 00673bf:
## [1] 48 15 25
##
## [[38]]
## + 3/100 vertices, from 00673bf:
## [1] 46 24 26
##
## [[39]]
## + 3/100 vertices, from 00673bf:
## [1] 45 20 42
##
## [[40]]
## + 3/100 vertices, from 00673bf:
## [1] 44 10 35
##
## [[41]]
## + 3/100 vertices, from 00673bf:
## [1] 43  1 26
##
## [[42]]
## + 3/100 vertices, from 00673bf:
## [1] 42 14 20
##
## [[43]]
## + 3/100 vertices, from 00673bf:
## [1] 41 29 37
##
## [[44]]
## + 3/100 vertices, from 00673bf:
## [1] 40  1 26
##
## [[45]]
## + 3/100 vertices, from 00673bf:
## [1] 39  3  4
##
## [[46]]
## + 3/100 vertices, from 00673bf:
```

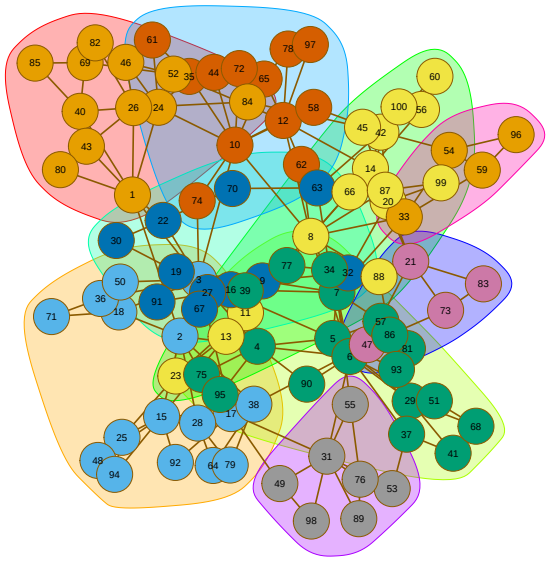
```
## [1] 38  2 28
##
## [[47]]
## + 3/100 vertices, from 00673bf:
## [1] 37  6 29
##
## [[48]]
## + 3/100 vertices, from 00673bf:
## [1] 36  3 18
##
## [[49]]
## + 3/100 vertices, from 00673bf:
## [1] 35 10 24
##
## [[50]]
## + 3/100 vertices, from 00673bf:
## [1] 34  6  7
##
## [[51]]
## + 3/100 vertices, from 00673bf:
## [1] 29  5  6
##
## [[52]]
## + 3/100 vertices, from 00673bf:
## [1] 28  2 15
##
## [[53]]
## + 3/100 vertices, from 00673bf:
## [1] 27  9 16
##
## [[54]]
## + 3/100 vertices, from 00673bf:
## [1] 26  1 24
##
## [[55]]
## + 3/100 vertices, from 00673bf:
## [1] 23 11 13
##
## [[56]]
## + 3/100 vertices, from 00673bf:
## [1] 22  1  3
##
## [[57]]
## + 3/100 vertices, from 00673bf:
## [1] 21 14 20
##
## [[58]]
## + 3/100 vertices, from 00673bf:
## [1] 19  9 16
##
## [[59]]
## + 3/100 vertices, from 00673bf:
## [1] 18  2  3
##
```

```
## [[60]]
## + 3/100 vertices, from 00673bf:
## [1] 16  7  9
##
## [[61]]
## + 3/100 vertices, from 00673bf:
## [1]  8 87 14
##
## [[62]]
## + 3/100 vertices, from 00673bf:
## [1]  8 66 14
##
## [[63]]
## + 3/100 vertices, from 00673bf:
## [1]  8 62 12
##
## [[64]]
## + 3/100 vertices, from 00673bf:
## [1]  8 14 12
##
## [[65]]
## + 3/100 vertices, from 00673bf:
## [1]  8 13 11
##
## [[66]]
## + 3/100 vertices, from 00673bf:
## [1]  8 12 10
##
## [[67]]
## + 3/100 vertices, from 00673bf:
## [1]  8 11  9
##
## [[68]]
## + 3/100 vertices, from 00673bf:
## [1]  8  7 33
##
## [[69]]
## + 3/100 vertices, from 00673bf:
## [1] 8 7 9
##
## [[70]]
## + 3/100 vertices, from 00673bf:
## [1] 8 7 6
##
## [[71]]
## + 3/100 vertices, from 00673bf:
## [1] 5 7 6
##
## [[72]]
## + 3/100 vertices, from 00673bf:
## [1] 5 4 6
##
## [[73]]
## + 3/100 vertices, from 00673bf:
```

```
## [1] 5 4 3
##
## [[74]]
## + 3/100 vertices, from 00673bf:
## [1] 4 2 3
##
## [[75]]
## + 3/100 vertices, from 00673bf:
## [1] 3 1 2
```

Verificando o método da remoção de pontes:

```
show.cluster(rn2, cluster_edge_betweenness(rn2))
```

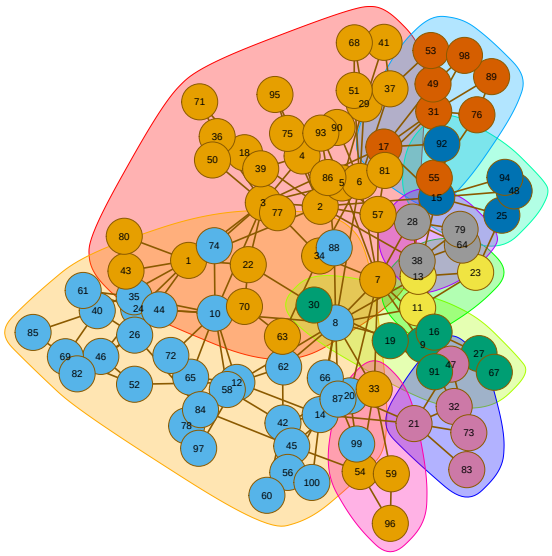


```
## [1] "Número de clusters:" "9"
## [1] "Tamanho dos clusters:"
## Community sizes
##  1  2  3  4  5  6  7  8  9
## 12 16 19 15 11 12  4  7  4
## [1] "modularidade:"     "0.628552655311912"
## [1] "Pertença a clusters:"

##   [1] 1 2 2 3 3 3 3 4 5 6 4 6 4 4 2 5 2 2 5 4 7 5 4 1 2 1 5 2 3 5 8 5 9 3 6 2 3
##  [38] 2 3 1 3 4 1 6 4 1 7 2 8 2 3 1 8 9 8 4 3 6 9 4 6 6 5 2 6 4 5 3 1 5 2 6 7 6
##  [75] 3 8 3 6 2 1 3 1 7 1 1 3 4 4 8 3 5 2 3 2 3 9 6 8 4 4
```

Usando o método de propagação de etiquetas:

```
set.seed(42)
show.cluster(rn2, cluster_label_prop(rn2))
```



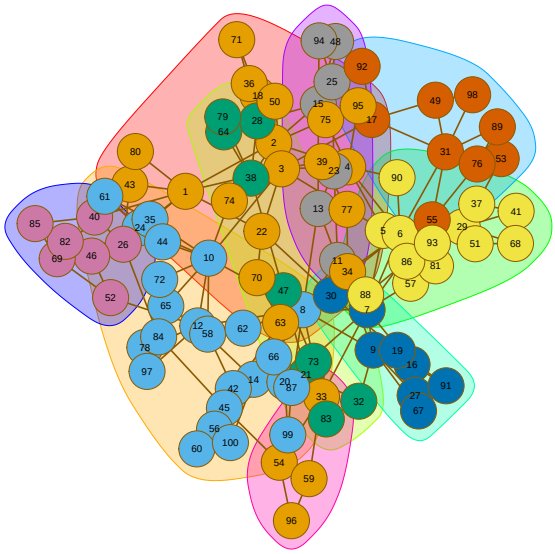
```
## [1] "Número de clusters:" "9"
## [1] "Tamanho dos clusters:"
## Community sizes
##  1  2  3  4  5  6  7  8  9
## 31 33  7  3  5  8  5  4  4
## [1] "modularidade:"     "0.564521116235925"
## [1] "Pertença a clusters:"

##   [1] 1 1 1 1 1 1 1 2 3 2 4 2 4 2 5 3 6 1 3 2 7 1 4 2 5 2 3 8 1 3 6 7 9 1 2 1 1
##  [38] 8 1 2 1 2 1 2 2 2 7 5 6 1 1 2 6 9 6 2 1 2 9 2 2 2 1 8 2 2 3 1 2 1 1 2 7 2
##  [75] 1 6 1 2 8 1 1 2 7 2 2 1 2 2 6 1 3 5 1 5 1 9 2 6 2 2
```

Usámos o `set.seed(42)` para estabilizar numa mesma solução, para análise.

Usando o método da otimização de modularidade:

```
show.cluster(rn2, cluster_fast_greedy(rn2))
```



```
## [1] "Número de clusters:" "9"
## [1] "Tamanho dos clusters:"
## Community sizes
##  1  2  3  4  5  6  7  8  9
## 19 24  9 13  8  9  7  7  4
## [1] "modularidade:"     "0.616867221520782"
## [1] "Pertença a clusters:"

##   [1] 1 1 1 1 4 4 5 2 5 2 8 2 8 2 8 5 6 1 5 2 3 1 8 2 8 7 5 3 4 5 6 3 9 1 2 1 4
##  [38] 3 1 7 4 2 1 2 2 7 3 8 6 1 4 7 6 9 6 2 4 2 9 2 2 2 1 3 2 2 5 4 7 1 1 2 3 1
##  [75] 1 6 1 2 3 1 4 7 3 2 7 4 2 4 6 4 5 6 4 8 1 9 2 6 2 2
```

QUESTÃO 3:

Compare e comente os resultados obtidos nas questões anteriores.

Que diferenças conseguimos observar entre a questão 1 e a 2? O que é que dá para concluir?

- CC sobre a associação de grau
- CC sobre a distância média
- CC sobre o diâmetro
- CC sobre a existência de triângulos
- JP sobre as comunidades
- JM o impacto que o método *random walk* tem na construção de uma rede aleatória; vantagens e desvantagens dos métodos usados em 1 e 2