Lista 1 - Matemática das Redes Complexas

Redes Complexas para Ciência da Computação

Luben Miguel, Rodrigo Lassance

```r
# pacotes sendo usados
library(igraph)
library(tidyverse)
library(Matrix)
library(knitr)
```

Exercício 1

Todas as redes utilizadas ao longo desta lista foram obtidas na página https://networkrepository.com. Os respectivos nomes atribuídos a cada uma das redes ao longo do texto dizem respeito às suas denominações originais encontradas no *site*. As redes selecionadas foram:

- **ca-CSphd**: Rede de orientação alunos de PhD da ciência da computação, vértices representando cada indivíduo e arestas indicando se houve alguma relação de orientação entre os indivíduos;
- **rt-israel**: Rede de retweets na hashtag de política sobre Israel, vértices representando os usuários e arestas indicando caso algum dos usuários tenha feito retweet da postagem do outro;
- **inf-openflights**: Rede de transferência de aeroportos, vértices representando os aeroportos e arestas indicando se de um aeroporto é possível viajar para o outro.

Contudo, antes de dar início às descrições sobre as redes, é necessário primeiro garantir que todas elas são direcionadas. Isso será feito através da avaliação de simetria da matriz de adjacência. Caso qualquer uma delas for simétrica, é sinal de que a rede não é direcionada.

```r
# pre-processamento e importacao dos dados
# lendo os dados de phd da computacao
phd <- Matrix::readMM("dados/ca-CSphd.mtx")
```

```r

# lendo os dados dos tweets sobre israel
israel <- read.table("dados/rt_israel.edges", sep=",", header = FALSE) |>
  select(-3) |>
  as.matrix()

# lendo os dados da rede de aeroportos
airports <- read.table("dados/inf-openflights.edges", sep=" ", header = FALSE) |>
  select(-3) |>
  as.matrix()

# grafo da rede de phds de ciencia da computacao
phd_graph <- graph_from_adjacency_matrix(phd,
                                         mode = "directed")

# grafo dos tweets de israel
israel_graph <- israel |>
  graph_from_edgelist(directed=TRUE)

israel_adj <- israel_graph |>
  as_adj()

airports_graph <- airports |>
  graph_from_edgelist(directed=TRUE)

airports_adj <- airports_graph |>
  as_adj()

# checando se sao simetricas
c(isSymmetric(phd), isSymmetric(israel_adj), isSymmetric(airports_adj))
```

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

Assim, podemos então concluir que as matrizes das três redes não apresentam simetria, portanto são todas direcionadas.

Agora, descreveremos brevemente as redes elencadas a partir das informações contidas nos dados, buscando também levantar possíveis problemas de pesquisa em que cada uma poderia ser usada.

- **ca-CSphd**: a rede possui 1882 vértices (indivíduos) e 1740 arestas (relações de orientação). Algumas possíveis temáticas que essa rede poderia informar são:

identificação do histórico de orientações (quem orientou quem ao longo do tempo), inferência acerca das áreas de estudo dos diversos orientadores com base nos projetos realizados por seus orientandos (ou mesmo do próprio orientador do indivíduo em análise), identificação de comunidades de pesquisa da área da Ciência da Computação.

- **rt-israel.php**: a rede possui 3698 vértices (usuários) e 4175 arestas (retweets). Desse modo, poderíamos buscar identificar usuários com maior "influência" na hashtag (como contas institucionais, jornalistas ou mesmo usuários com um maior número de seguidores), assim como avaliar se o usuário pode ser um bot (caso sua taxa de retweets no tempo seja humanamente impossível, por exemplo).

- **inf-openflights**: a rede possui 2939 vértices (aeroportos) e 30501 arestas (rotas). Assim, ela poderia ser utilizada para identificar pólos comerciais da região como um todo (vértices que recebem maior número de voos) e estimar possíveis comunidades dentro dessa rede.

Exercício 2

As matrizes de adjacências foram calculadas no exercício 1, visto que isso foi necessário para avaliar o número de arestas. Agora, vamos checar se alguma delas apresenta pesos:

```r
# phd
all(phd == 0 | phd == 1)
```

```
[1] TRUE
```

```r
# israel
all(israel_adj == 0 | israel_adj == 1)
```

```
[1] TRUE
```

```r
# aeroportos
all(airports_adj == 0 | airports_adj == 1)
```

```
[1] TRUE
```

Com isso, concluímos que nenhuma delas tem peso, visto que as entradas da matriz se resumem em 0's e 1's.

A seguir, devemos transformar todas as três redes em não-direcionadas. Para que isso seja possível, faz-se necessário identificar todas as entradas da matriz que são iguais a 1 e, caso o elemento transposto seja 0, igualá-lo a 1 também. Por exemplo, se $A_{ji} = 1$, precisamos garantir que $A_{ij} = 1$ também.

```
# criando novamente as matrizes de adjancecias
# funcao que transforma grafos direcionadas em nao direcionados
transforma_adj <- function(mat_adj){
  new_mat <- matrix(nrow = nrow(mat_adj),
                    ncol = ncol(mat_adj))

  # montando a parte inferior da matriz
  # sera 1 se pelo menos ha uma direcao entre os vertices
  new_mat[lower.tri(mat_adj)] <- ((mat_adj[lower.tri(mat_adj)] +
                                  t(mat_adj)[lower.tri(mat_adj)]) >= 1) + 0

  # igualando com a parte superior
  new_mat[upper.tri(mat_adj)] <- t(new_mat)[upper.tri(new_mat)]

  # igualando com a diagonal
  diag(new_mat) <- diag(mat_adj)
  return(as(new_mat, "sparseMatrix"))
}

# phd simetrica
phd_sym <- transforma_adj(phd)
isSymmetric(phd_sym)
```

```
[1] TRUE
```

```
# israel simetrica
israel_sym <- transforma_adj(israel_adj)
isSymmetric(israel_sym)
```

```
[1] TRUE
```

```r
# aeroportos simetrica
airports_sym <- transforma_adj(airports_adj)
isSymmetric(airports_sym)
```

```
[1] TRUE
```

```r
# lista das matrizes transformadas
mat_list <- list("phd" = phd_sym,
                 "israel" = israel_sym,
                 "aeroportos" = airports_sym)

# lista das matrizes originais
original_list <-  list("phd" = phd,
                       "israel" = israel_adj,
                       "aeroportos" = airports_adj)
```

Garantimos assim que as novas matrizes são simétricas, portanto não direcionadas. Feito isso, computamos agora o número de elementos positivos de $X = A^3$e $Y = A^4$ para as redes selecionadas e o exibimos na seguinte tabela:

```r
calc_X <- function(mat){
  ((mat %*% mat %*% mat) > 0) |>
    sum()
}

calc_Y <- function(mat){
  (mat %*% mat %*% mat %*% mat > 0) |>
    sum()
}

X_list <- mat_list |> map_dbl(function(x){
  x |> calc_X()
})

Y_list <- mat_list |> map_dbl(function(x){
  x |> calc_Y()
})

# tabela de elementos da matriz com valor nao nulo
tibble(rede = names(mat_list),
```

```r
            X = X_list,
            Y = Y_list) |>
  knitr::kable(format = "latex",
               booktabs = TRUE,
               escape = FALSE,
               col.names = c("Rede",
                             "Elementos positivos em $A^3$",
                             "Elementos positivos em $A^4$"),
               caption = "Número de elementos positivos em cada multiplicação de matriz")
  kableExtra::kable_styling(full_width = FALSE,
                            latex_options = "hold_position")
```

Tabela 1: Número de elementos positivos em cada multiplicação de matriz

Rede	Elementos positivos em A^3	Elementos positivos em A^4
phd	22351	40714
israel	242479	1403868
aeroportos	2609541	5699665

Exercício 3

Primeiramente, consideramos que, por força do vértice, o enunciado está se referindo ao valor da diagonal das matrizes de cocitação e acoplamento bibliográfico. Assim, como isso se refere ao in-degree e ao out-degree, teremos:

```r
return_cocit <- function(mat){
  mat %*% t(mat)
}

return_acopbib <- function(mat){
  t(mat) %*% mat
}


in_degree <- function(mat){
  (mat %*% t(mat)) |>
    diag()
}

```

```
out_degree <- function(mat){
  (t(mat) %*% mat) |>
    diag()
}

max_cocit <-  original_list |>
  map(function(x){
  in_degree(x) |>
    which.max()
})


max_acopbib <- original_list |>
  map(function(x){
  out_degree(x) |>
    which.max()
})

tibble(rede = names(original_list),
        X = max_cocit,
        Y = max_acopbib) |>
kable(format = "latex",
      booktabs = TRUE,
      escape = FALSE,
      col.names = c("Rede",
                    "Maior força (cocitação)",
                    "Maior força (acoplamento bibliográfico)"),
      caption = "Vértices de maior força na cocitação e acoplamento bibliográfico (1ª inte
  kableExtra::kable_styling(full_width = FALSE,
                            latex_options = "hold_position")
```

Tabela 2: Vértices de maior força na cocitação e acoplamento bibliográfico (1ª interpretação)

Rede	Maior força (cocitação)	Maior força (acoplamento bibliográfico)
phd	2	1
israel	1832	1391
aeroportos	53	53

Já, entendendo força do vértice como a soma dos pesos de cada vizinho do vértice de interesse para as matrizes de cocitação e acoplamento bibliográfico, obtemos os seguintes vértices:

```r
max_forca_cocit <-  original_list |>
  map(function(x){
  mat = return_cocit(x)
  diag(mat) = 0
  mat |>
  rowSums() |>
    which.max()
})


max_forca_acopbib <- original_list |>
  map(function(x){
  mat = return_acopbib(x)
  diag(mat) = 0
  mat |>
  rowSums() |>
    which.max()
})

tibble(rede = names(original_list),
        X = max_forca_cocit,
        Y = max_forca_acopbib) |>
kable(format = "latex",
      booktabs = TRUE,
      escape = FALSE,
      col.names = c("Rede",
                      "Maior força (cocitação)",
                      "Maior força (acoplamento bibliográfico)"),
      caption = "Vértices de maior força na cocitação e acoplamento bibliográfico (2ª inte
  kableExtra::kable_styling(full_width = FALSE,
                            latex_options = "hold_position")
```

Tabela 3: Vértices de maior força na cocitação e acoplamento bibliográfico (2ª interpretação)

Rede	Maior força (cocitação)	Maior força (acoplamento bibliográfico)
phd	20	149
israel	2000	744
aeroportos	53	53

Exercício 4

Entre as três redes elencadas, aquela que de julga ser acíclica é a de orientações de PhD. Assim, para verificar que isso de fato é o caso, iremos avaliar a soma dos módulos dos autovalores da matriz de adjacência. Caso $\sum_{i=1}^{V} |\lambda_i| = 0$, sendo V o número de vértices da rede, podemos então garantir que é acíclica. Assim, usando a função *eigen* do R, retornamos a diagonal da matriz de autovalores Λ de A para a rede de phd e obtemos:

```r
# testando se eh aciclico pela soma do modulo dos autovalores
phd |>
  eigen() |>
  pluck("values") |>
  abs() |>
  sum()
```

```
[1] 0
```

Ou seja, como $\sum_{i=1}^{V} |\lambda_i| = 0$, não existem ciclos na rede de PhD's de Ciência da Computação.

Exercício 5

A reciprocidade é definida pela seguinte fórmula (https://en.wikipedia.org/wiki/Reciprocity_(network_science)):

$$\rho = \frac{\sum_{i \neq j}(A_{ij} - \bar{A})(A_{ji} - \bar{A})}{\sum_{i \neq j}(A_{ij} - \bar{A})^2} \ ,$$

tal que $\bar{A} = \sum_{i \neq j} \frac{A_{ij}}{N(N-1)}$ é a proporção de conexões diretas entre os diferentes vértices do grafo. Assim, para computar tal medida, fazemos:

```r
compute_rho <- function(mat){
  N <- nrow(mat)
  mat_mean <- (1/(N * (N - 1)))*sum(mat[row(mat) != col(mat)])
  rho <- ((mat - mat_mean)*(t(mat) - mat_mean))[row(mat) != col(mat)] |>
    sum()
  return(rho/sum(((mat - mat_mean)^2)[row(mat) != col(mat)]))
}

# computando rho para as duas matrizes ciclicas
compute_rho(israel_adj)
```

```
[1] 0.00448763
```

```r
compute_rho(airports_adj)
```

```
[1] 0.9719346
```

Podemos perceber que a medida de reciprocidade fornece resultados opostos para as redes. Enquanto que na rede de retweets a reciprocidade é baixa, na de aeroportos é consideravelmente alta. Isso faz sentido dado que no Twitter é raro que um usuário retweete o conteúdo de outra pessoa que o retweetou, enquanto que é bastante comum para aeroportos que, quando um leva para o outro, a volta também costuma ser possível.