Exercícios 1 - Redes Complexas

Luben Miguel, Rodrigo Lassance

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

Exercício 1

Primeiramente, importamos as redes e checamos se são direcionadas pela simetria da matriz de adjacência:

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

# 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
```

```r
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 eh direcionado
isSymmetric(israel_adj)
```

```
[1] FALSE
```

```r
isSymmetric(airports_adj)
```

```
[1] FALSE
```

```r
isSymmetric(phd)
```

```
[1] FALSE
```

Checada que são direcionadas, as redes que selecionamos foram:

- Rede de orientação de PHD's da ciência da computação: Retirada de https://networkrepository.com/ca-CSphd.php, a rede tem 1882 vértices e 1740 arestas, cada vértice representa um indivíduo e cada aresta representa quem esse indivíduo ja orientou. Essa rede modela relações de orientação dos estudantes de Doutorado em computação, assim, a partir desta, podemos traçar histórico de orientações, analisar área de estudo dos diversos orientadores através de possíveis informações de seus orientandos, ou de seu próprio orientador, identificação de comunidades de pesquisa da área da computação.

- Rede de retweets na hashtag de política sobre Israel: Retirada de https://networkrepository.com/rt-israel.php, a rede tem 3698 vértices e 4175 arestas, cada vértice representa um usuário e cada aresta representa um retweet sobre um tweet da hashtag específica. Como essa rede modela retweets de uma hashtag política, podemos identificar usuários de maior influência na hashtag, como por exemplo, contas institucionais, jornalistas ou simplesmente usuários com mais seguidores.
- Rede de transferência de aeroportos Retirada de https://networkrepository.com/inf-openflights.php, a rede tem 2939 vértices e 30501 arestas cada vértice representa um aeroporto e cada aresta representa pelo menos uma rota de ida viável para cada outro aeroporto. Essa rede modela as diferentes rotas entre diferentes aeroportos por uma dada região de análise, o que nos permite identificar polos comerciais dessa região e estimar possíveis comunidades dentro dessa rede.

Exercício 2

As matrizes de adjacências foram calculadas no exercício 1. Vamos checar se são sem pesos:

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

```
[1] TRUE
```

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

```
[1] TRUE
```

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

```
[1] TRUE
```

Sendo sem peso, vamos agora transforma-las em matrizes simétrica, removendo a informação da direção. Ou seja, vamos transformar A_{ji} em 1 se $A_{ij} = 1$ e vice e versa.

```r
# 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"))
}

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

```
[1] TRUE
```

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

```
[1] TRUE
```

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

```
[1] TRUE
```

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

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

Transformadas em matrizes simétricas, 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),
           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")
```

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

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

Exercício 3

Para computar a matriz de cocitação e acoplamento bibliografico e retornar os vértices de maior in-degree e out-degree, teremos:

```
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()
})

```

```r
tibble(rede = names(original_list),
       X = max_cocit,
       Y = max_acopbib) |>
  knitr::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
  kableExtra::kable_styling(full_width = FALSE,
                            latex_options = "hold_position")
```

Table 2: Vértices de maior força na cocitação e acoplamento bibliográfico para cada rede

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

Exercício 4

Para testar se a matriz é aciclica, basta checar se a soma dos módulos dos autovalores $\sum_{i=1}^{V} |\lambda_i|$ é 0, sendo V o número de vértices da rede. 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 ciclios 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:

```
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(airports_adj)
```

```
[1] 0.9719346
```

```
compute_rho(israel_adj)
```

```
[1] 0.00448763
```