Introdução aos Grafos Aleatórios

Adrian Hinojosa

Departamento de Estatística, Universidade Federal de Minas Gerais

Sumário

CAPíTULO 1

Introdução

Estas notas tem por objetivo discutir os conceitos básicos de rede ou grafo aleatório e das aplicações ao estudo das redes sociais. Modelos probabilísticos de redes aleatórias. Estatísticas descritivas das redes aleatórias: grau, subgrafos, conetividade, etc . Estimadores de máxima verosimilhança e Bayesianos assim como a implementação no software R.

Usaremos alguns algoritmos dos programas igraph, network, ergm e bergm. para serem utilizados em alguns exemplos.

Os objetivos deste trabalho são:

- Introdução aos Grafos, tipos de grafos, propriedades.
- Análises Descritivas dos grafos.
- Modelos de grafos Aleatórios, Grafos Aleatórios Exponenciais.
- Estimadores de Máxima Verosimilhança para grafos aleatórios.
- Estimadores Bayesianos para grafos aleatórios.
- Tópicos de regressão linear e previsão. Outros Tópicos.

Estas notas estão baseadas no livro *Statistical Analysis of Network Data: Methods and Models*, do Kolaczyk, Eric D., ver [7].

CAPíTULO 2

Introdução aos Grafos: tipos de grafos e propriedades

Um famoso problema histórico de matemática, que se tornou uma legenda popular conhecida como as *sete pontes de Königsberg*.

Cortado pelo rio Prególia, a cidade de Königsberg (território de Prússia) possuía duas grandes ilhas que, juntas, formaram um complexo que naquela época continha sete pontes. Foi discutida a possibilidade de atravessar todas as pontes sem repetir nenhuma ponte. Em 1736, o matemático Leonard Euler provou, de forma simples, que não havia como satisfazer essas condições. Criando o primeiro gráfico da história, Euler transformou as estradas em linhas retas e suas interseções em pontos. primeiro grafo de la historia, Euler transformou os caminhos em retas e suas interseções em pontos.

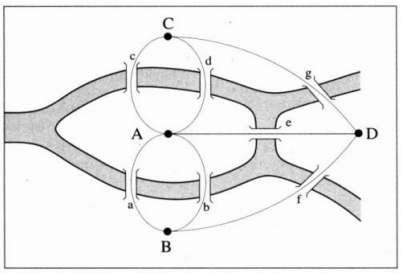

FIGURA 1. As sete pontes de Königsberg

Algumas aplicações de grafos:

- Redes sociais: Facebook, Twitter, etc.
- Epidemiologia, saúde pública.
- Comunicações: celulares, redes de computadores
- Biologia molecular, genética.
- Inteligência Artificial e novos paradigmas da teoria da informação.
- Etc.

Para resolver problemas práticos, como os mencionados acima, um grafo é a melhor solução para representar a relação entre os objetos de um determinado conjunto, onde alguns pares de objetos, vértices, também chamados de nós ou pontos, são conectados por bordas, também conhecidas como linhas ou arcos.

Tal como as equipes de futebol, ou estilos de música e gêneros cinematográficos, os grafos também possuem características que os separam em diferentes categorias, como serão apresentadas a seguir.

0.1. Definição de grafos. Considere o conjunto de vértices V, que é finito com n elementos, $V = \{1, \ldots, n\}$. Para este conjunto vamos definir o conjunto de arestas $E \subseteq V \times V$.

DEFINIÇÃO 0.1. *Um grafo G é um par ordenado de conjuntos: $G = (V, E)$. O conjunto de todos os possíveis grafos com n vértices será denotado por $\mathcal{G}_n$, isto é:*

$$\mathcal{G}_n = \{G : G = (V, E), \quad |V| = n, \quad E \subseteq V \times V\}$$

EXEMPLO 0.2. *Considere o conjunto de vértices $V = \{1, \ldots, 7\}$ e as arestas $E = \{(1,4), (1,5), (2,3), (2,7), (3,6)(4,1), (4,7), (5,6), (5,3,)(7,7)\}$, então, podemos representar o grafo $G = (V, E)$ por médio de pontos e setas:*

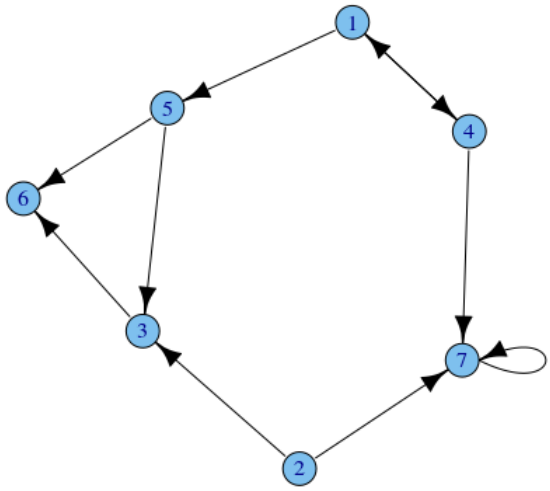

FIGURA 2. Grafo G com $n = 7$ vértices

O **tamanho** de um grafo é o número de arestas do grafo e a **ordem** é o numero de vértices, assim, no exemplo anterior o tamanho do grafo é $|E| = 10$ e a ordem é $|V| = 7$. Usualmente os grafos classificam-se pelo tamanho em **dispersos** (sparce graph) se $|E| \approx |V|$ ou densos se $|E| \approx |V|^2$.

1. Tipos de grafos

Um grafo não dirigido é aquele em que os vértices não são ordenados, o seja, as arestas não possuem orientação. Assim, por exemplo, se suprimimos no grafo anterior as direciones, então obtemos:

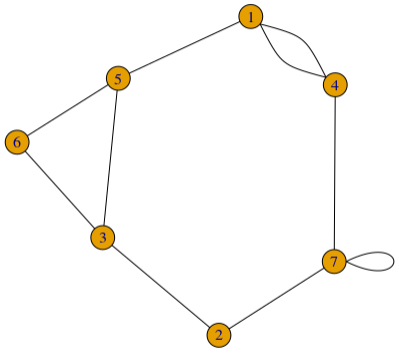

FIGURA 3. Grafo G não dirigido com $n = 7$ vértices

Este grafo é um exemplo de um multigrafo com **loops**, isto é, vértices com múltiplas arestas e arestas que conectam o mesmo vértice. A seguir vamos considerar apenas grafos que não tem estas características. Para isso vamos definir um grafo **simples** como aquele grafo que não tem loops ou arestas múltiplas, em particular vamos também considerar nestas notas somente grafos não dirigidos.

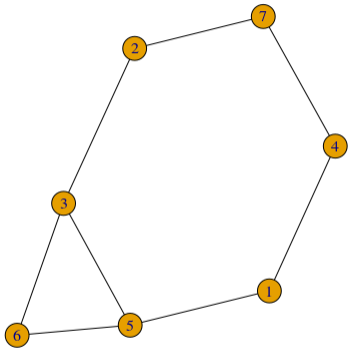

FIGURA 4. Grafo simples G (não dirigido) com $n = 7$ vértices

No caso dos grafos simples com n vértices, vale que o número destes grafos é: $|\mathcal{G}_n| = 2^{\binom{n}{2}}$, pois o número de possíveis pares de vértices é $\binom{n}{2}$ e o número de subconjuntos deste conjunto de pares é $2^{\binom{n}{2}}$; lembre que um grafo é um subconjunto de pares. Vamos a definir o grau $d(i)$ de um vértice $i \in V$ como o número de vértices conectados com este vértice por médio de uma aresta que pertence ao grafo.

$$d(i) = \sum_{j:(i,j)\in E} 1.$$

Assim no grafo da Figura 4 temos que, $d(1) = 2$, $d(2) = 2$, etc. Podemos representar esta função por médio do seguinte gráfico:

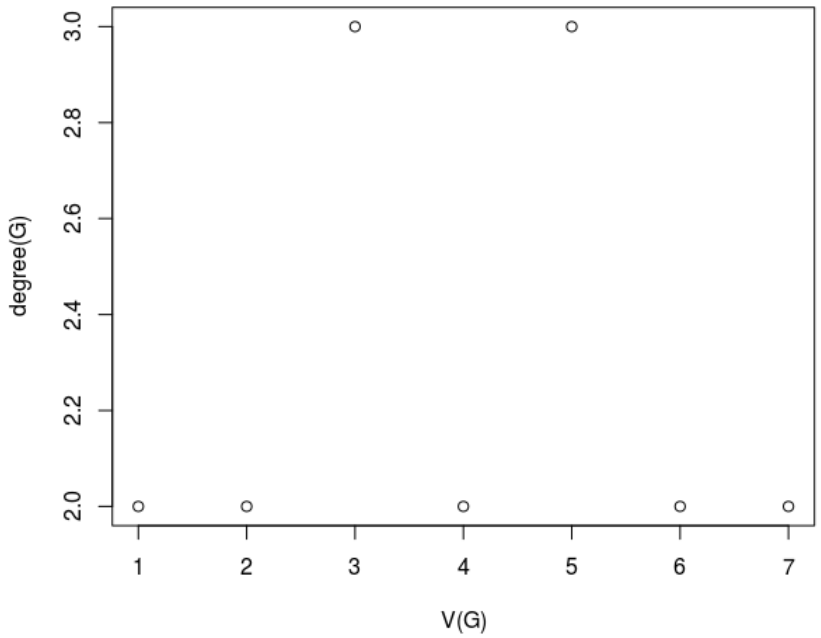

FIGURA 5. Grau do grafo G

A distribuição do grau de um grafo é a frequência relativa dos graus do grafo. No exemplo anterior somente ha vértices com graus 2 (com frequência relativa de $0,7$) e 3 (com frequência relativa de $0,3$); através de um gráfico a representamos na Figura 6.

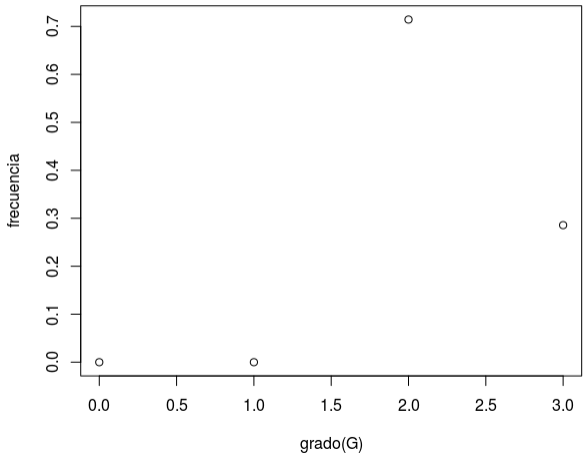

FIGURA 6. A distribuição do grau do grafo G

Definimos a media do grau como:

$$d(G) = \frac{1}{|V|} \sum_{v \in V} d(v).$$

Então, vale que:

$$|E| = \frac{1}{2} d(G)|V| \tag{1.1}$$

Dizemos que dois vértices são adjacentes se uma aresta do grafo conecta a ambos. Outra maneira de representar um grafo é usar esta noção de adjacência para definir a **matriz de adjacência** $A = (a_{i,j})$, na cual as linhas e as colunas representam os vértices do grafo, e os coeficientes $a_{i,j} = 1$ ou 0 de acordo com as correspondentes linha i e coluna j sejam adjacentes, isto é, formem uma aresta do grafo: $(i, j) \in E$. No grafo do exemplo anterior a correspondente matriz de adjacência A seria:

```
 > A
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    0    1    1    0    0    0    0
[2,]    1    0    0    0    0    1    0
[3,]    1    0    0    0    1    0    1
[4,]    0    0    0    0    1    1    0
[5,]    0    0    1    1    0    0    1
[6,]    0    1    0    1    0    0    0
[7,]    0    0    1    0    1    0    0
```

FIGURA 7. Matriz de adjacência A

Observe que esta matriz A é simétrica pois o grafo G é simples, logo não é dirigido.

Outra matriz importante é a matriz **Laplaciana**, do grafo: $L = D - A$, com D a matriz diagonal com o grau de cada vértice na diagonal: $D =$diag$(d(i))$.

```
> L
   1  4  5  2  3  7  6
1  2 -1 -1  0  0  0  0
4 -1  2  0  0  0 -1  0
5 -1  0  3  0 -1  0 -1
2  0  0  0  2 -1 -1  0
3  0  0 -1 -1  3  0 -1
7  0 -1  0 -1  0  2  0
6  0  0 -1  0 -1  0  2
```

FIGURA 8. Laplaciano L

Dizemos que L está normalizado se

$$L_{i,j} = \begin{cases} -\frac{a_{i,j}}{\sqrt{d(i)d(j)}} & \text{si } i \neq j \\ 1 & \text{si } i = j \end{cases}$$

neste caso, $L = D^{-1/2}(I - A)D^{-1/2}$ com $D^{-1/2} =$diag$(1/\sqrt{d(i)})$ (onde colocamos zero se $d(i) = 0$), do exemplo anterior temos:

```
> L
           1    4          5          2          3    7          6
1  1.0000000 -0.5 -0.4082483  0.0000000  0.0000000  0.0  0.0000000
4 -0.5000000  1.0  0.0000000  0.0000000  0.0000000 -0.5  0.0000000
5 -0.4082483  0.0  1.0000000  0.0000000 -0.3333333  0.0 -0.4082483
2  0.0000000  0.0  0.0000000  1.0000000 -0.4082483 -0.5  0.0000000
3  0.0000000  0.0 -0.3333333 -0.4082483  1.0000000  0.0 -0.4082483
7  0.0000000 -0.5  0.0000000 -0.5000000  0.0000000  1.0  0.0000000
6  0.0000000  0.0 -0.4082483  0.0000000 -0.4082483  0.0  1.0000000
```

FIGURA 9. Laplaciano normalizado L

O **espectro** do Laplaciano é o conjunto de auto-valores do Laplaciano normalizado, neste exemplo:

```
 > Espectro
[1] 0.00 0.34 0.53 1.20 1.39 1.62 1.91
```

FIGURA 10. Espectro de L

É possível provar que os autovalores estão limitados entre 0 e 2, e que o primeiro autovalor é sempre 0. O número de subgrafos conexos do grafo corresponde ao número de autovalores zero. Os maiores autovalores estão associados a formação de 'anéis' no grafo.

1.1. Classes importantes de grafos: Um grafo **regular** é um grafo no qual todos os vértices possuem o mesmo número de ligações com os outros vértices, o seja, todos os vértices tem o mesmo grau. Um grafo regular com vértices de grau k é chamado **grafo k-regular** ou gráfico regular de grau k. Por exemplo, G com 5 vértices é 2-regular:

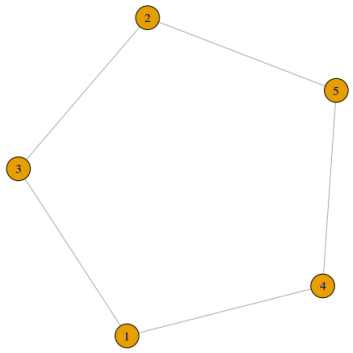

FIGURA 11. Grafo 2-regular G

Este grafo também é chamado de **2-ciclo**, C_2, e é um **caminho fechado** que não tem **interseções**. Agora considere G com 5 vértices é 4-regular:

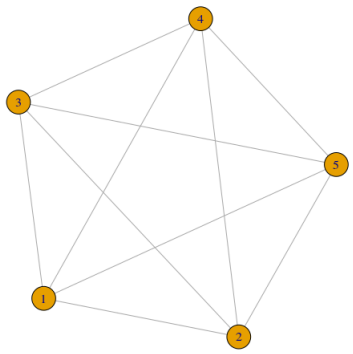

FIGURA 12. Grafo 4-regular G

Este grafo é um grafo **completo**, K_5, é dizer que cada vértice tem uma aresta ligando com todos os outros vértices. Os grafos completos tem a característica de que

cada par de vértices tem uma aresta que os une. Finalmente, observe nestes exemplos que um grafo com n vértices e k-regular não podemos ter n e k impares, um deles tem que ser par (use a relação (1.1)).

Um **caminho** é um conjunto de vértices $v_1, \ldots, v_k \in V$ tal que o conjunto de arestas $(v_1, v_2), (v_2, v_3), \ldots, (v_{\ell-1}, v_\ell), (v_\ell, v_{\ell+1}), \ldots, (v_{k-1}, v_k)$ está em E; uma **geodésica** entre dos vértices é o caminho de menor tamanho entre eles. Um grafo é chamado de **grafo conexo** o **conectado** se todo par de vértices do grafo pode unir-se por um caminho. O grafo da Figura (4) é conexo:

```
 > is.connected(G)
[1] TRUE
```

Mais, se removemos as arestas $(1, 5)$ e $(2, 3)$, obtemos um grafo desconexo:

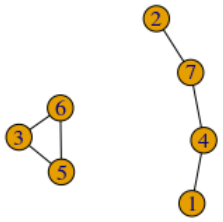

FIGURA 13. Grafo desconexo G

verificamos que é desconexo:

```
 > is.connected(G)
[1] FALSE
```

Também podemos verificar isto usando o espectro deste grafo, é possível mostrar que neste caso terá dois autovalores zero:

```
> Espectro
[1] 0.0 0.0 0.5 1.5 1.5 1.5 2.0
```

FIGURA 14. Espectro de L

Um grafo **bipartido** é um grafo no cual os vértices estão separados em dois conjuntos, V_1 e V_2 e as arestas somente conectam vértices de conjuntos separados. Na Figura (15) podemos observar um grafo bipartido:

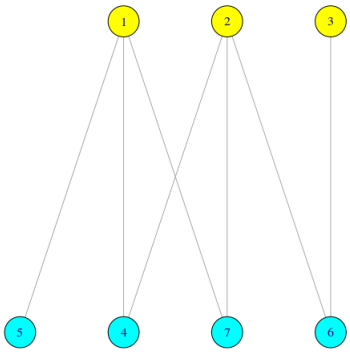

FIGURA 15. Grafo bipartido g

Uma **árvore** é um grafo que não contem ciclos será k-**árvore** se for k-regular. Por exemplo:

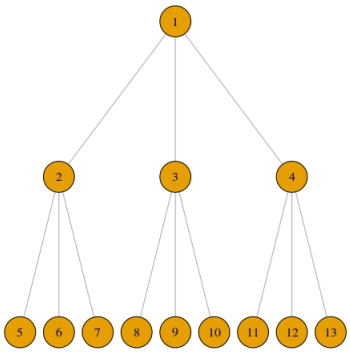

FIGURA 16. 3-árvore com 13 vértices T_3

Os vértices de grau 1 são chamados **folhas**.

Árvores desconexos são conhecidos como **florestas**

Uma n-**estrela**, S_n, é um grafo que é uma árvore com n vértices e $n-1$ folhas. Por exemplo:

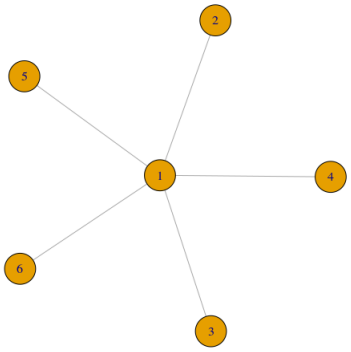

FIGURA 17. 6-estrela S_6

Finalmente, um **triângulo** é um grafo completo com 3 vértices, K_3:

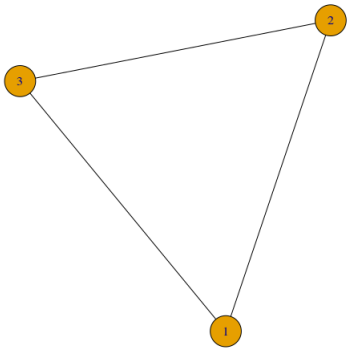

FIGURA 18. Triângulo

2. Análise descritiva dos grafos

As propriedades estruturais dos grafos são características de interesse em dois sentidos. Em primeiro lugar, desde o ponto de vista das aplicações as redes sociais se interessam com a conectividade do grafo e o papel de certos vértices centrais. Desde o ponto de vista matemático, é possível mostrar que a quantidade de certos subgrafos caracteriza a um grafo. Vamos a descrever três destas características: centralidade, transitividade e conectividade.

2.1. Centralidade. A centralidade de um vértice é caracterizada pela importância desse vértice num grafo. Podemos considerar, em principio, que basta observar seu grau, pois esse é um indicativo de quantos vértices estão conectados com ele. Observe que $0 \leq d(v) \leq |V| - 1$. Assim, teríamos que os vértices com maior grau são mais centrais. Vejamos um exemplo, considere o grafo G da Figura (19), com $|V| = 20$ vértices:

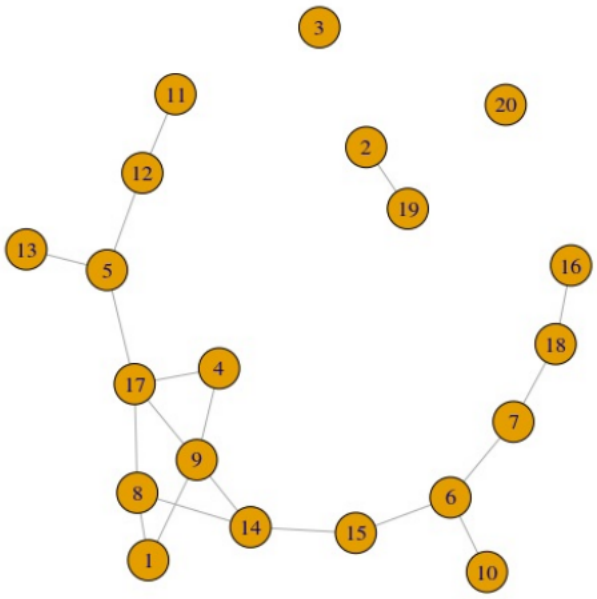

FIGURA 19. O grafo G

O grau de cada vértice é:

```
> degree(G)
6  7  1  8  9  4 10  5 12 11 13 14 15 17 18 16  2 19  3 20
3  2  2  3  4  2  1  3  2  1  1  3  2  4  2  1  1  1  0  0
```

e a distribuição do grau é:

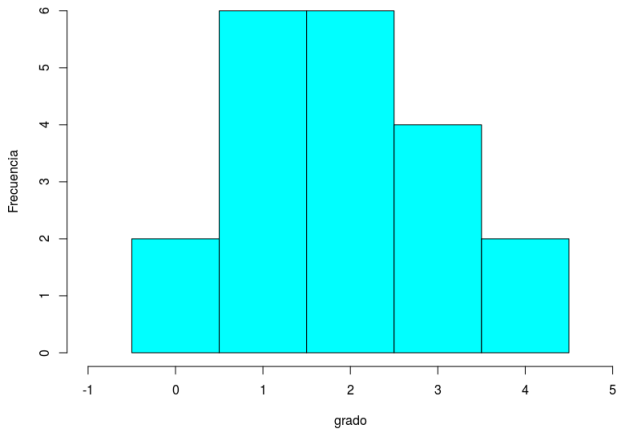

FIGURA 20. Grau do grafo G

como podemos observar existem 2 vértices com grau 4, que é o máximo. Estos dois vértices são os mais conectados.

Outra medida central para vértices é a **centralidade por proximidade**:

$$c_C(v) = \frac{1}{\sum_{u \in V} d(v,u)}, \quad v \in V \tag{2.2}$$

Donde $d(v,u)$ é a distancia (número de arestas) da geodésica entre u e v, aqui iremos supor que o grafo é conexo.

Os valores da proximidade para o maior subgrafo conexo do exemplo anterior ($G1 \subset G$, $V(G1) = V(G) - \{2,3,19,20\}$) são:

```
> closeness(G)
         6          7          1          8          9          4         10
0.01923077 0.01612903 0.01851852 0.02325581 0.02380952 0.02000000 0.01515152
         5         12         11         13         14         15         17
0.01886792 0.01538462 0.01265823 0.01492537 0.02380952 0.02173913 0.02222222
        18         16
0.01351351 0.01136364
```

ou graficamente também pode ser observado:

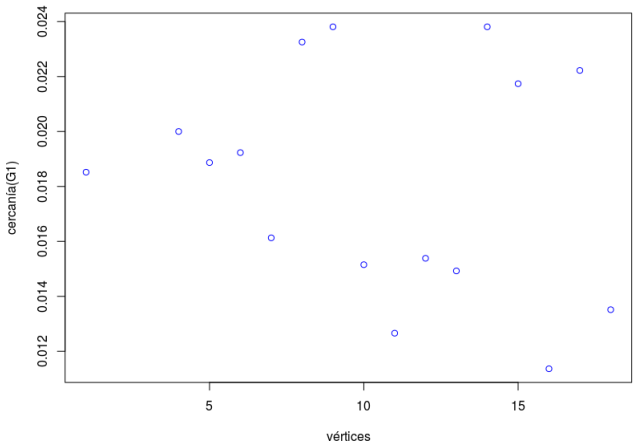

FIGURA 21. Proximidade dos vértices do grafo $G1 \subset G$

A Figura (21) mostra que os vértices 9 e 14 são os que estão mais próximos aos demais vértices.

Agora, podemos considerar que um vértice é central usando a media de todos os caminhos que usam esse vértice, a **centralidade por mediação** está definida como:

$$c_M(v) = \sum_{s \neq u \neq v \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}, \quad v \in V. \tag{2.3}$$

Onde $\sigma(s,t|v)$ é o número de geodésicas entre s e t que passam por v e $\sigma(s,t)$ é uma constante normalizadora definida por $\sigma(s,t) = \sum_v \sigma(s,t|v)$.

Os valores da mediação para $G1$ são:

```
> betweenness(G)
        6         7         1         8         9         4        10
47.0000000 26.0000000  0.3333333 23.5000000 31.5000000  0.0000000  0.0000000
        5        12        11        13        14        15        17
38.0000000 14.0000000  0.0000000  0.0000000 54.3333333 50.0000000 45.3333333
       18        16
14.0000000  0.0000000
```

graficamente:

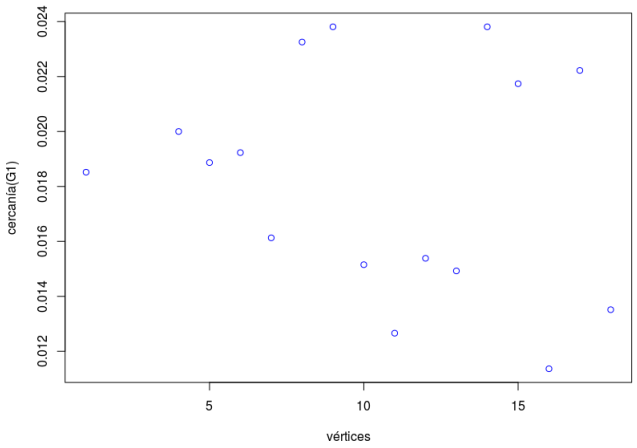

FIGURA 22. Mediação dos vértices do grafo $G1 \subset G$

Aqui, observamos que os vértices 9 e 14 são os que tem um papel central maior que os demais vértices.

Estas duas definições de centralidade podem ser estendidas para arestas e ainda existe uma noção de centralidade usando o espectro.

2.2. Transitividade. A transitividade está relacionada com uma característica das redes sociais que poderíamos resumir como: "o amigo de meu amigo é meu amigo"; esta propriedade costuma aparecer em redes coesivas, é dizer em redes com grau alto de agregação.

Num grafo simples a **transitividade** é também chamada o **coeficiente de agregação** e está definida como:

$$cl(G) = \frac{3\tau_{K_3}(G)}{\tau_{S_2}(G)}. \tag{2.4}$$

Onde $\tau_{K_3}(G)$ é o número de triângulos, K_3, contidos no grafo e $\tau_{S_2}(G)$ é o número de 2-estrelas, S_2, contidas no grafo. Observe que cada 2-estrelas é um possível triângulo, basta que esteja a aresta que une as dois pontas da estrela para formar o triângulo. logo o coeficiente de agregação, $cl(G)$, mede a proporção de possíveis triângulos que são triângulos. Como cada 2-estrela é contabilizada 3 vezes, em cada possível triângulo, teríamos que dividir para 3 o número delas, $\tau_{S_2}(G)/3$.

O coeficiente de agregação é um dos mais conhecidos índices para grafos. Vejamos qual é o coeficiente de agregação nos grafos, G e $G1$ dos exemplos:

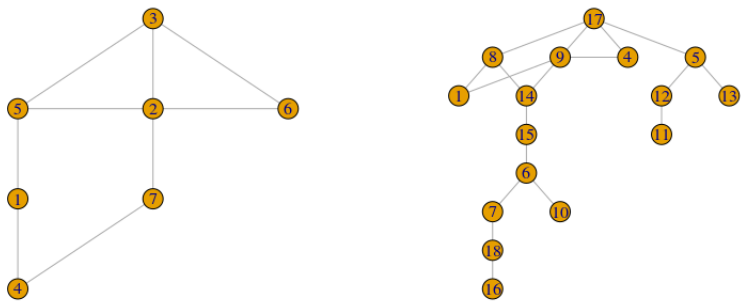

FIGURA 23. Transitividade dos grafos G e $G1$, $cl(G) = 0.27$, $cl(G1) = 0.1$.

Claramente o grafo $G1$ é menos agregado, pois tem más vértices e porque entre estes vértices extras tem más 2-estrelas que não são triângulos.

2.3. Conectividade. É de interesse nas aplicações em grafos a localização das **comunidades** dentro de um grafo, isto é, subgrafos que tenham seus vértices conexos, em algum sentido. Vamos a ver como se faz isto usando uma técnica de hierarquização. Suponha que para um grafo $G = (V, E)$ existe uma partição de V em subconjuntos disjuntos $\mathcal{V} = \{V_1, \ldots, V_K\}$. A partir de $\mathcal{V}$ definimos $f_{i,j} = f_{i,j}(\mathcal{V})$ como a fração de arestas de E que tem um vértice em V_i e outro vértice no V_j. Usando $f_{i,j}$ definimos a **modularidade** de $\mathcal{V}$ como:

$$\text{mod } (\mathcal{V}) = \sum_{k=1}^{K} \left[f_{k,k} - f_{k,k}^{*} \right]^2 \tag{2.5}$$

Onde $f_{i,j}^{*} = f_{k,+} f_{+,k}$ e $f_{k,+} = \sum_{j=1}^{K} f_{k,j}$, $f_{+,k} = \sum_{i=1}^{K} f_{i,k}$. Observe que num grafo simples, isto é, não dirigido, vale que $f_{k,+} = f_{+,k}$. Esta seleção de f^{*} corresponde a um grafo com o mesmo grau que o grafo original, mas com as arestas colocadas aleatoriamente, sem levar em consideração a partição $\mathcal{V}$. O problema consiste agora em encontrar entre todas as partições $\mathcal{V}$, com um número fixo K de elementos, aquela que minimiza $\text{mod } (\mathcal{V})$.

Segue um exemplo, considere o grafo $G1$ do exemplo da seguinte Figura (24):

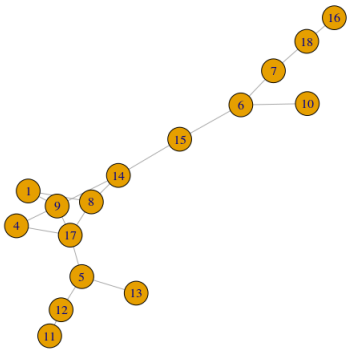

FIGURA 24. Grafo $G1$.

Usando o algoritmo 'fastgreedy' do igraph, obtemos:

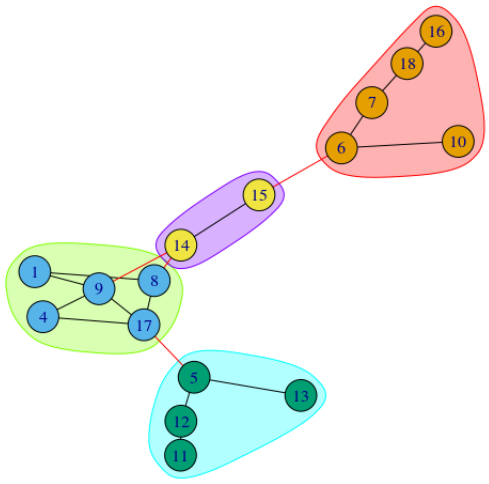

FIGURA 25. Partição do grafo $G1$ em quatro 'comunidades'.

3. Usando R

Nesta seção descrevemos os comandos do R que foram utilizados para representar as figuras desta seção Observe que é necessário carregar a libraria do pacote **igraph** do R para poder usar os comandos.

(1) Figura 2:

```
 G <- graph.formula(1-+4, 1-+5, 2-+3, 2-+7, 3-+6, 4-+1,
 4-+7, 5-+6, 5-+3, 7-+7,simplify = FALSE)
plot(G,edge.color="black",layout=layout.fruchterman.reingold)
```

(2) Figura 3:

```
 G <- graph.formula(1-4, 1-5, 2-3, 2-7, 3-6, 4-1,
 4-7, 5-6, 5-3, 7-7,simplify = FALSE)
plot(G,edge.color="black",layout=layout.fruchterman.reingold)
```

(3) Figura 4:

```
 G <- graph.formula(1-4, 1-5, 2-3, 2-7, 3-6, 4-1,
 4-7, 5-6, 5-3)
plot(G,edge.color="black",layout=layout.fruchterman.reingold)
```

(4) Figura 5:

```
G <- graph.formula(1-4, 1-5, 2-3, 2-7, 3-6, 4-1,
4-7, 5-6, 5-3)
degree(G)
plot(V(G),degree(G))
```

(5) Figura 6:

```
G <- graph.formula(1-4, 1-5, 2-3, 2-7, 3-6, 4-1,
4-7, 5-6, 5-3)
degree_distribution(G)
max(degree(G))
degree_distribution(G)
plot(0:max(degree(G)),degree_distribution(G),
xlab = "grau(G)",ylab = "frecuencia")
```

(6) Figura 7:

```
G <- graph.formula(1-4, 1-5, 2-3, 2-7, 3-6, 4-1,
4-7, 5-6, 5-3)
A<-as_adj(G, type = "both",  attr = NULL,edges = FALSE,
names = FALSE, sparse =FALSE)
A
```

(7) Figura 8:

```
G <- graph.formula(1-4, 1-5, 2-3, 2-7, 3-6, 4-1,
4-7, 5-6, 5-3)
L<-laplacian_matrix(G, normalized = FALSE, weights = NULL,
sparse = FALSE)
L
```

(8) Figura 9:

```
G <- graph.formula(1-4, 1-5, 2-3, 2-7, 3-6, 4-1,
4-7, 5-6, 5-3)
L<-laplacian_matrix(G, normalized = TRUE, weights = NULL,
sparse = FALSE)
L
```

(9) Figura 10:

```
G <- graph.formula(1-4, 1-5, 2-3, 2-7, 3-6, 4-1, 4-7, 5-6, 5-3)
L<-laplacian_matrix(G, normalized = TRUE, weights = NULL,
sparse = FALSE)
Espectro<-sort(round(eigen(L)$values,2))
Espectro
```

(10) Figura 11:

```
G<-sample_k_regular(5, 2, directed = FALSE, multiple = FALSE)
plot(G)
```

(11) Figura 12:

```
G<-sample_k_regular(5, 4, directed = FALSE, multiple = FALSE)
plot(G)
```

(12) Figura 13:

```
G <- graph.formula(1-4,  2-7, 3-6, 4-1, 4-7, 5-6, 5-3)
plot(G,edge.color="black",layout=layout.fruchterman.reingold,
vertex.size=20,edge.arrow.size=20)
```

(13) Figura 15:

```
g <- sample_bipartite(3,4, type = "gnp", p=0.5, directed = FALSE)

plot(g, layout = layout_as_bipartite,
     vertex.color=c("yellow","cyan")[V(g)$type+1],
     vertex.size=20)
```

(14) Figura 14:

```
G <- graph.formula(1-4,  2-7, 3-6, 4-1, 4-7, 5-6, 5-3)
L<-laplacian_matrix(G, normalized = TRUE, weights = NULL,
sparse = FALSE)
Espectro<-sort(round(eigen(L)$values,2))
Espectro
```

(15) Figura 16:

```
T<-make_tree(13, 3, mode = "undirected")
plot(T,vertex.size=20,layout=layout_as_tree(T, root=c(1)))
```

(16) Figura 17:

```
S<-make_star(6, mode = "undirected")
plot(S,layout=layout.fruchterman.reingold,vertex.size=20 )
```

(17) Figura 18:

```
G<-graph.formula(1-- 2,2-- 3,3-- 1)
plot(G,edge.color="black",layout=layout.fruchterman.reingold,
vertex.size=20,edge.arrow.size=20)
```

(18) Figura 19:

```
G<-graph.formula(6-- 7,1-- 8,1-- 9,4-- 9,6--10,5--12,11--12,
5--13,8--14,9--14,6--15,14--15,4--17,5--17,8--17,9--17,7--18,
16--18, 2--19)
G<-add_vertices(G,2)
V(G)$name[19:20]<-c(3,20)
plot(G,layout=layout.fruchterman.reingold,vertex.size=20 )
```

(19) Figura 20:

```
grau<-degree(G)
hist(grau,col="cyan",
     xlab="grau", ylab="Frecuencia",main = "",
     breaks = c(0:(max(grau)+1)-0.5), xlim = c(-1,(max(grau)+1)))
```

(20) Figura 21:

```
G1<-graph.formula(6-- 7,1-- 8,1-- 9,4-- 9,6--10,5--12,11--12,5--13,
8--14,9--14, 6--15,14--15,4--17,5--17,8--17,9--17,7--18,16--18)
closeness(G1)
plot(V(G1)$name,closeness(G1),col="blue",
     ylab="cercanía(G1)", xlab="vértices",main = "")
```

(21) Figura 22:

```
G1<-graph.formula(6-- 7,1-- 8,1-- 9,4-- 9,6--10,5--12,11--12,5--13,
8--14,9--14,6--15,14--15,4--17,5--17,8--17,9--17,7--18,16--18)
betweenness(G1)
plot(V(G1)$name,closeness(G),col="blue",
     ylab="cercanía(G1)", xlab="vértices",main = "")
```

(22) Figura 23:

```
G <- graph.formula(1-4, 1-5, 2-3, 2-7, 3-6, 4-1, 4-7, 5-6, 5-3)
transitivity(G)
G1<-graph.formula(6-- 7,1-- 8,1-- 9,4-- 9,6--10,5--12,11--12,5--13,
8--14,9--14, 6--15,14--15,4--17,5--17,8--17,9--17,7--18,16--18)
transitivity(G1)
```

(23) Figura 25:

```
G1<-graph.formula(6-- 7,1-- 8,1-- 9,4-- 9,6--10,5--12,11--12,5--13,
8--14,9--14,6--15,14--15,4--17,5--17,8--17,9--17,7--18,16--18)
plot(G1)
kc<-fastgreedy.community(G1)
length(kc)
sizes(kc)
membership(kc)
plot(kc, G1)
```

CAPíTULO 3

Grafos Aleatórios: Modelos e Inferência

Neste capítulo vamos a discutir alguns modelos clássicos de grafos aleatórios. Os primeiros modelos foram introduzidos por Erdös e Rényi em 1959, eram grafos simples em que cada aresta era incluída no grafo com igual probabilidade.

Existem basicamente dois tipos de modelos; um é mais teórico-matemático e é usado para representar, qualitativamente, fenômenos de grandes redes como por exemplo, os grafos de 'escala livre' ou 'o mundo pequeno', como no modelo de 'conexão preferencial'. Outro tipo de modelos mais usados nas aplicações são modelos paramétricos, como o modelo exponencial, que é o modelo que vamos discutir aqui.

1. Modelos de Grafos Aleatórios

Vamos a considerar os grafos simples, é possível estender estas definições para outros grafos dirigidos, multígrafos, bipartidos.

DEFINIÇÃO 1.1. *Um modelo aleatório para grafos esta definido pela coleção:*

$$\{\mathbb{P}_\theta(G), G \in \mathcal{G}_n, \theta \in \Theta\}.$$

Onde $\mathcal{G}_n = \{G = (E, V), |V| = n, G\ simples\}$ *é uma coleção de grafos simples de ordem* n *que é chamada* **espaço amostral**, *e para cada* $\theta \in \Theta$*:*

$$\mathbb{P}_\theta : \mathcal{G}_n \to [0, 1],$$

é uma distribuição de probabilidades em $\mathcal{G}_n$*. Usualmente o* **espaço paramétrico** Θ *é um conjunto 'regular' contido em algum* $\mathbb{R}^k$

A principal dificuldade de trabalhar com modelos aleatórios para grafos consiste em que devemos definir o valor da probabilidade $\mathbb{P}$ para cada grafo $G = (E, V)$, (com $|V| = n$), do espaço amostral, $\mathcal{G}_n$, e o número de grafos deste espaço é finito mas muito grande, $|\mathcal{G}_n| = 2^{\binom{n}{2}} \approx 2^{n^2}$ e o número de vértices n é grande.

Outra característica determinante nestos modelos é o tamanho dos grafos que estamos considerando, grafos **dispersos** (sparce graph) se $|E| \approx |V|$ ou **densos** se $|E| \approx |V|^2$.

Esta dificuldade e outras considerações nos levam a restringir a definição de $\mathbb{P}$ à um conjunto menor, para isso consideramos certas funções chamadas **estatísticas** (ou observações) η do grafo:

$$\eta : \mathcal{G}_n \to \mathbb{R}.$$

Por exemplo, considerando $\eta(G) = |E| =$ 'o número de arestas do grafo', podemos restringir $\mathcal{G}_n$ aos grafos G que tenham um tamanho fixo m, isto é, o novo espaço amostral $\mathcal{G}^*$ sera:

$$\mathcal{G}^* = \{G = (E, V), |V| = n, \eta(G) = m, G \text{ simples}\}.$$

EXEMPLO 1.2. *Considere $n = 3$ vértices, neste caso $\mathcal{G}_3 = \{G1, \ldots, G8\}$ está formado por $2^{\binom{3}{2}} = 8$ grafos:*

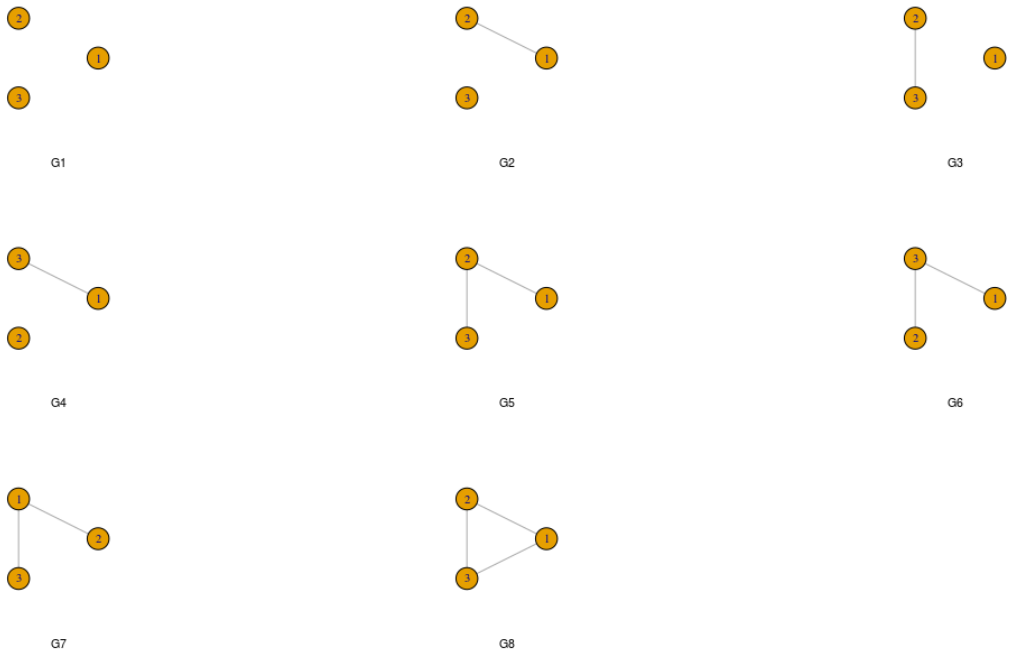

FIGURA 1. Os 8 grafos contidos em $\mathcal{G}_3$

Podemos definir uma distribuição de probabilidades em $\mathcal{G}_3$ fazendo:

$$\mathbb{P}_\theta(Gi) = \frac{\theta^i}{\sum_{k=1}^{8} \theta^k} = \frac{\theta^i(1-\theta)}{\theta(1-\theta^7)}, \quad i = i = 1, \ldots, 8, \quad \theta \in \mathbb{R}^+.$$

Assim $\mathbb{P}(G5) = \frac{\theta^5(1-\theta)}{\theta(1-\theta^7)}$. Observe que se $\theta < 1$ então os grafos mais densos ($G8$ por exemplo) tem menos probabilidade. No entanto se $\theta > 1$ ocorre o oposto.

Agora se fazemos $\eta = |E|$ e fixamos o conjunto de grafos que tem $\eta = 2$ vértices, isto é, estamos interessados em uma característica 'estrutural': ter dois vértices. Assim obtemos $\mathcal{G}^* = \{G5, G6, G7\}$

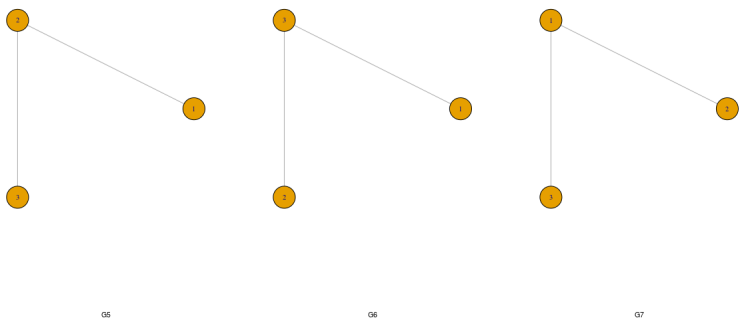

FIGURA 2. $G5, G6, G7$

Neste caso poderíamos calcular as probabilidades destes grafos, usando a probabilidade condicional, por exemplo:

$$\mathbb{P}^*(G7) = \mathbb{P}_\theta(G7|\eta = 2) = \frac{\mathbb{P}_\theta(G7 \cap \eta = 2)}{\mathbb{P}_\theta(\eta = 2)} = \frac{\frac{\theta^7(1-\theta)}{\theta(1-\theta^7)}}{\sum_{k=5}^{7} \frac{\theta^k(1-\theta)}{\theta(1-\theta^7)}} = \frac{\theta^7}{\theta^5 + \theta^6 + \theta^7}.$$

Observe que o evento $\{\eta = 2\}$ está formado pelos grafos com dois arestas e que nosso espaço amostral tem agora somente 3 grafos.

Poderíamos também escolher outras características η como o número de triângulos, o grau, etc. e fixar este número.

1.1. Modelos Clássicos. Nesta seção descrevemos quatro modelos: Uniforme, Erdös-Rényi, Conexão Preferencial e Exponencial.

1.1.1. *Modelo Uniforme.* Neste modelo todos los grafos tem igual probabilidade:

$$\mathbb{P}_U(Gi) = \frac{1}{|\mathcal{G}_n|}.$$

Para este modelo η é uma **Variável Aleatória** com função de distribuição de probabilidade:

$$F_{\eta,\mathcal{G}_n}(t) = \frac{\#\{G \in \mathcal{G}_n, \eta(G) \leq t\}}{|\mathcal{G}_n|},$$

onde $\#\{\cdot\}$ é a cardinalidade de um conjunto e $0 \leq t \in \mathbb{R}$. Esta Variável Aleatória consiste em que vamos à observar esta característica η para cada grafo de $\mathcal{G}_n$ e medimos que tão provável é ela.

EXEMPLO 1.3. *Considere o espaço amostral $\mathcal{G}_3$ dos grafos com três vértices. Neste caso é fácil ver que:*

$$\mathbb{P}_U(Gi) = \frac{1}{8}, \quad i = 1, \ldots, 8$$

Por outro lado, observe que $\eta(G) = |E(G)| = 0, 1, 2, 3$ é uma variável aleatória discreta com distribuição:

$$\mathbb{P}(\eta = 0) = \frac{1}{8}, \quad \mathbb{P}(\eta = 1) = \frac{3}{8},$$

$$\mathbb{P}(\eta = 2) = \frac{3}{8} \quad \mathbb{P}(\eta = 3) = \frac{1}{8}.$$

Em geral, suponha que o número de vértices é fixo, $|V| = n$ e que também fixamos o número de arestas, $|E| = m$. Definimos o modelo Uniforme com m arestas, $G(n, m)$, como a distribuição de probabilidades que atribui igual probabilidade aos grafos pertencentes à $\mathcal{G}_{n,m} = \{G = (E, V) : |V| = n, |E| = m\}$, isto é:

$$\mathbb{P}(G) = \frac{1}{\binom{N}{m}}, \quad G \in \mathcal{G}_{n,m},$$

com $N = \binom{n}{2}$ o número de arestas possíveis num grafo com n vértices.

Para simular grafos do modelo $G(n, m)$ podemos usar o programa R. Um exemplo, simulado, de um grafo aleatório Uniforme com $|V| = 30$ e $|E| = 100$ fixo:

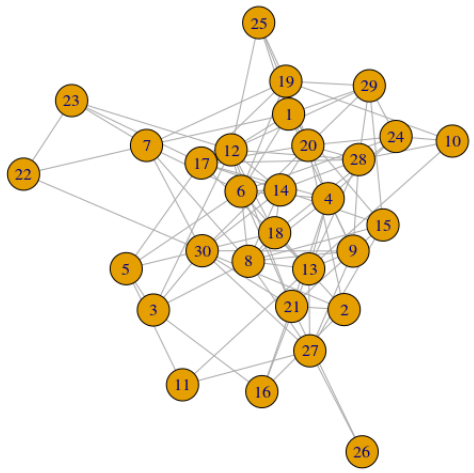

FIGURA 3. Grafo G Uniforme com $|V| = 30$ e $|E| = 100$.

1.1.2. *Modelo Erdös-Rényi.* O modelo de Erdös-Rényi é uma generalização do modelo uniforme e é o modelo mais simples de grafo aleatório. Foi introduzido em 1959 por Paul Erdös em um dos trabalhos mais citados sobre combinatória. Erdös formulou neste trabalho a ideia de usar probabilidade positiva de um evento para provar um resultado de existência na matemática combinatória, para isso introduz pela primeira vez na matemática a ideia de um grafo aleatório. Este método de prova será conhecido depois como o 'Método Probabilístico'.

A construção simplesmente consiste em dar uma probabilidade p ao evento de que cada aresta pertença ao grafo, e isto independentemente das outras arestas: $\mathbb{P}(e \in G) = p$.

A probabilidade para $G = (E, V) \in \mathcal{G}_n$ no modelo de Erdös, $G(n, p)$, é então:

$$\mathbb{P}_p(G) = p^{|E|}(1-p)^{\binom{n}{2}-|E|}.$$

Observe que a probabilidade de cada grafo G com m arestas seria $\mathbb{P}(G) = p^m(1-p)^{\binom{n}{2}-m}$.

Podemos demostrar uma relação entre os modelos Uniforme $G(n, m)$ e Erdös-Rényi $G(n, p)$. Se a 'densidade de vértices' se mantêm igual a $p = \frac{m}{\binom{n}{2}}$ então ambos modelos coincidem quando n é suficientemente grande, isto ocorre pela lei dos grandes números.

Para simular grafos do modelo $G(n, p)$ podemos usar o programa R. Um exemplo de um grafo simulado para o modelo de Erdös-Rényi, com $|V| = 10$:

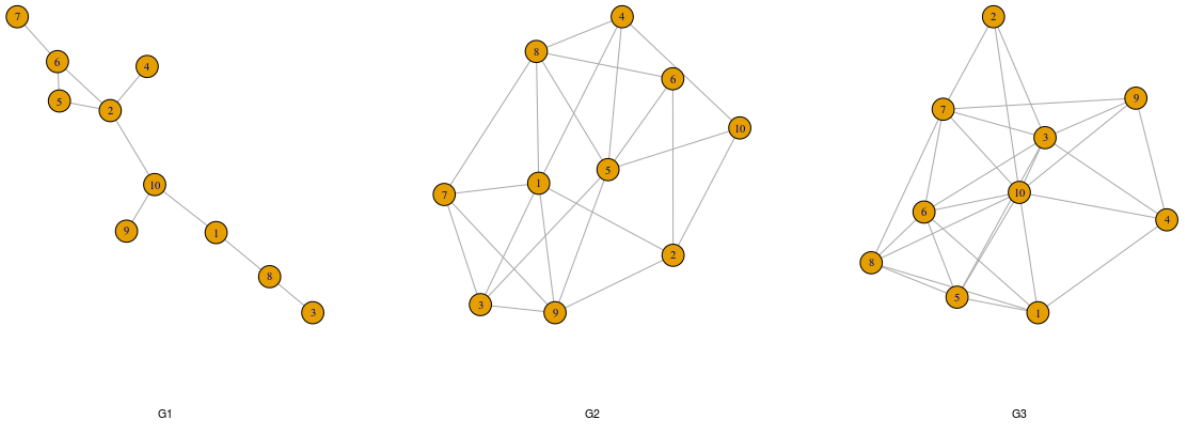

FIGURA 4. Grafo G Erdös-Rényi com $|V| = 10$, $p = 0.3$, $p = 0.5$ e $p = 0.7$

Observe que o número de arestas neste modelo é aleatório, em media é $\mathbb{E}|E| = \binom{n}{2}p$. Assim na figura (4) o número médio de arestas é:

$\mathbb{E}_{p=0.3}|E| = 13.5$, $\mathbb{E}_{p=0.5}|E| = 22.5$ e $\mathbb{E}_{p=0.7}|E| = 31.5$

1.1.3. *Modelo Conexão Preferencial.* Esta família de modelos é uma das mais importantes na literatura de modelos de redes, pois representa de maneira aproximada o comportamento de redes como a internet. Este é um modelo dinâmico, no sentido de que o número de vértices não é fixo, neste modelo vamos dinamicamente adicionando arestas e vértices com um mecanismo que descreveremos a continuação.

Uma das propriedades importantes deste modelo é que a distribuição do grau 'decai' como uma lei de potencias, (lembre que nossos grafos são aleatórios logo o grau dos vértices de cada grafo não é fixo), isto é, se $d(v_k)$ é o grau do k-essimo vértice, quando os graus estão ordenados em forma crescente, então:

$$d(v_k) \approx \frac{1}{k^\alpha}.$$

Este fenômeno é chamado de escala libre (scale-free), e significa que com probabilidade pequena, não nula, existem vértices com um grau muito grande, isto é, vértices

que estão conectados com uma grande quantidade de outros vértices. Este fenômeno é conhecido como 'mundo pequeno' (small-world), pois significa que usando caminhos pequenos é possível conectar a todos os vértices. Em estúdios empíricos, tem-os encontrado que o tamanho desses caminhos é 6, como consequência nas redes sociais necessitamos apenas 6 pessoas para conectarnos com qualquer outra pessoa.

A seguir veremos um modelo de grafos que possuiem estas características, é o modelo de 'Barabási–Albert', ou modelo BA. A ideia é criar vértices que estejam conectados aos vértices com grau maior:

- A rede começa com n_0 vértices ($|V(G_n)| = n_0$.
- Um novo vértice i_n é criado em cada passo n. Com probabilidade proporcional ao grau de cada uno dos vértices já existentes, assim, criamos uma aresta que une este vértice com os anteriores:

$$\mathbb{P}((i_n, j) \in V(G_{n+1})) = \frac{d(j)}{\sum_{k \in V(G_n)} d(k)}, \quad j \in V(G_n).$$

Vejamos um exemplo simulado de um grafo BA com $n = 500$ vértices e com poderes $\alpha = 0.1$ e $\alpha = 0.001$, α é a potencia del decaimento.:

FIGURA 5. Grafo G do modelo BA com $|V| = 500$, $\alpha = 0.1$ e $\alpha = 0.001$

Histogramas dos graus para os grafos anteriores, são mostrados a seguir,

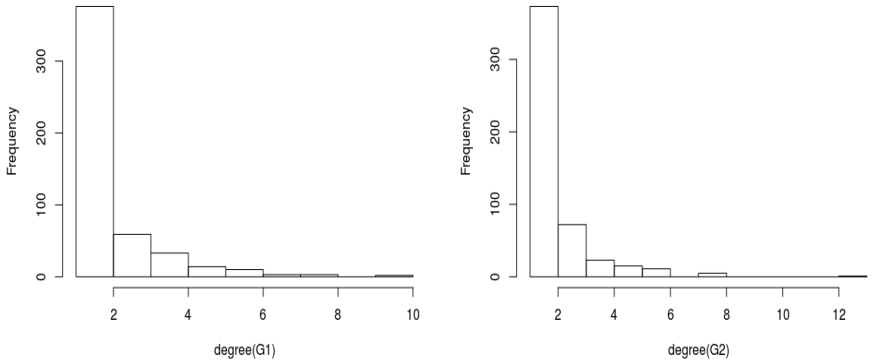

FIGURA 6. Histogramas dos Grafos G do modelo BA com $|V| = 500$, $\alpha = 0.1$ e $\alpha = 0.001$

1.1.4. *Modelo Exponencial Aleatório ERGM.* O **Modelo Exponencial Aleatório para Grafos (ERGM)** é o modelo paramétrico mais usado nos estudos de redes sociais, sua importância radica em que permite introduzir observações das características estruturais dos grafos na distribuição de probabilidade, isto é, a probabilidade de um grafo depende de características tais como o número de arestas, triângulos, estrelas, etc. que estão presentes no grafo.

Este modelo foi introduzido por Holland e Leinhardt em 1981, ver [5], e em sua forma general por Frank e Strauss em 1986, ver [4]. O estudo matematico e rigoroso deste modelo foi desenvolvido por Diaconis et. al. em 2013, ver [2], nesse trabalho mostra-se a lei dos grades números para distribuições de grafos , assim como também a transição de fase para uma família importante de ERGMs.

Para descrever este modelo, considere um conjunto de características: $\eta_1, \ldots, \eta_k$ de um grafo, estas podem ser o número de arestas, triângulos, estrelas, etc. e suponha que associamos parâmetros: $\theta_1, \ldots, \theta_k$ a cada uma delas. Então definimos a distribuição de probabilidades deste modelo como:

$$\mathbb{P}_\Theta(G) = \frac{e^{\sum_{i=1}^k \theta_i \cdot \eta_i(G)}}{Z(\Theta)}, \quad G \in \mathcal{G}_n,$$

onde $\Theta = (\theta_1, \ldots, \theta_k)$ são os parâmetros e $Z(\Theta)$ é uma constante normalizadora chamada função de partição:

$$Z(\Theta) = \sum_{G \in \mathcal{G}_n} e^{\sum_{i=1}^k \theta_i \cdot \eta_i(G)}.$$

O grafo de Erdös-Rényi é um caso particular de este modelo, ele corresponde a usar somente o número de vértices do grafo no modelo:

$$\mathbb{P}_\Theta(G) = \frac{e^{\theta \cdot |E(G)|}}{Z(\theta)}, \quad G \in \mathcal{G}_n.$$

Observe que $\eta(g) = |E(G)|$. Neste caso, é fácil achar o valor da constante normalizadora: $Z(\theta) = (1 + e^{\theta})^n$. Assim, temos que a probabilidade de ter uma aresta é $p = \frac{e^{\theta}}{1+e^{\theta}}$.

Em geral é um problema difícil encontrar o valor de $Z(\Theta)$, no lugar disso, são utilizados algoritmos aproximados, ao menos quando n é pequeno.

EXEMPLO 1.4. *Considere os grafos com $n = 10$ vértices, usaremos duas características: o número de arestas ($\eta_1(G)$) e o número de triângulos ($\eta_2(G)$) presentes no grafo.*

Vejamos que ocorre se fixamos o parâmetro que controla o número de triângulos $\theta_2 = 0.3$ e variamos θ_1 ($\theta_1 = -1$, $\theta_1 = 1$) que controla o número de arestas. Lembre que a probabilidade de ter uma aresta é $p = \frac{e^{\theta_1}}{1+e^{\theta_1}}$ (obtemos $p_1 = 0.27$ e $p_2 = 0.73$)

Usando o programa 'ergm' de R obtemos os grafos $G1$ ($\theta_1 = -1$, $\theta_2 = 0.3$) e $G2$ ($\theta_1 = -1$, $\theta_2 = 0.3$):

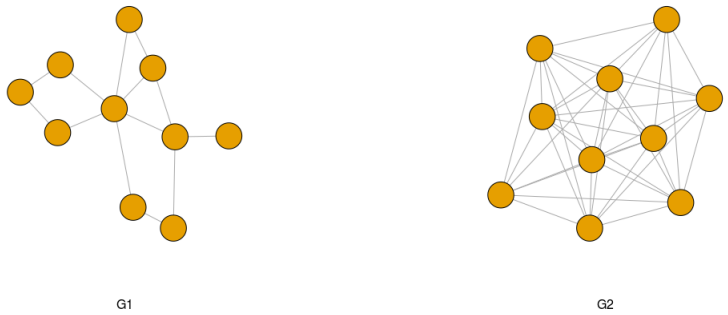

FIGURA 7. Grafos $G1$ e $G2$ do modelo ERGM com $\theta_1 = -1, 1$, $\theta_2 = 0.3$

Observe o número de arestas e triângulos que se formaram:

```
> summary(G1 ~ edges + triangle)
  edges triangle
     13        2

> summary(G2 ~ edges + triangle)
  edges triangle
     41       90
```

Isto significa que ao aumentar as arestas (edges) aumentamos o número de triângulos, isto é ambas características estão correlacionadas no grafo

Os histogramas dos graus para os grafos anteriores:

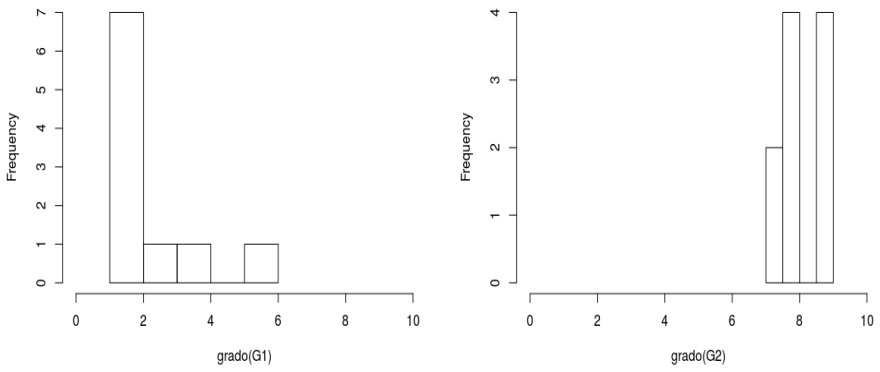

FIGURA 8. Distribuição do grau de $G1$ e $G2$

Como vemos o grafo $G2$ é mas conectado, seu índice de agregação (transitividade o densidade de triângulos) é quatro vezes maior que o de $G1$:

```
 > transitivity(G1)
[1] 0.2
> transitivity(G2)
[1] 0.9060403
```

1.1.5. *Simulação de Grafos Aleatórios.* Os algoritmos para simular modelos de grafos aleatórios tem o problema relacionado com o número extremadamente grande de grafos que pertencem ao espaço amostral. Por exemplo, no caso do modelo ERGM, pode-se percever na função de partição $Z(\Theta)$:

$$Z(\Theta) = \sum_{G \in \mathcal{G}_n} e^{\sum_{i=1}^{k} \theta_i \cdot \eta_i(G)}.$$

A soma tem 2^{n^2} termos. Por isto existem algoritmos aproximados para encontrar $Z(\Theta)$, vamos a explicar brevemente em que consiste um deles.

Neste algoritmo usamos a *Lei de los grandes números* para encontrar $Z(\Theta)$, usando o fato de que podemos simular com facilidade grafos de Erdös-Rényi, com $p = 0,5$, estes grafos correspondem à $\Theta_0 = (0, \ldots, 0)$.

Agora observamos a seguinte relação:

$$\begin{aligned}\frac{Z(\Theta)}{Z(\Theta_0)} &= \sum_{G\in\mathcal{G}_n} \frac{e^{\sum_{i=1}^k \theta_i\cdot\eta_i(G)}}{Z(\Theta_0)},\\ &= \sum_{G\in\mathcal{G}_n} \frac{e^{\sum_{i=1}^k \theta_i\cdot\eta_i(G)} e^{\sum_{i=1}^k 0\cdot\eta_i(G)}}{Z(\Theta_0)},\\ &= \sum_{G\in\mathcal{G}_n} \mathbb{P}_{\Theta_0}(G) e^{\sum_{i=1}^k \theta_i\cdot\eta_i(G)},\\ &= \sum_{G\in\mathcal{G}_n} \mathbb{P}_{\Theta_0}(G) e^{\Theta^t\cdot\eta(G)},\\ &= \mathbb{E}_{\Theta_0} e^{\Theta^t\cdot\eta(G)}.\end{aligned}$$

Observe que para $p = 0,5$, vale que $Z(\Theta_0) = (1+e^0)^n = 2^n$, logo: $Z(\Theta) = 2^n \cdot \mathbb{E}_{\Theta_0} e^{\Theta^t\cdot\eta(G)}$.

Suponha que simulamos m de esses grafos: $G_1, \dots G_m$, pois é fácil simular de $\mathbb{P}_{\Theta_0}(\cdot)$ que são grafos de Erdös-Rényi, então, usando a lei de los grandes números, vale que:

$$\lim_{m\to\infty} \frac{e^{\Theta^t\cdot\eta(G_1)} + \dots + e^{\Theta^t\cdot\eta(G_m)}}{m} = \mathbb{E}_{\Theta_0} e^{\Theta^t\cdot\eta(G)}.$$

Em principio funciona, mas ocorrem outros problemas quando n é grande e deixa de funcionar esta ideia.

EXEMPLO 1.5. *Considere o exemplo anterior com* $\Theta = (-1, 0.3)$ *, com* $m = 100$ *grafos de Erdös-Rényi com* $p = 0.5$*, usando o seguinte código R para implementar as simulações:*

```
 m=100
g.sim <- simulate(network(10,,density=0.1,directed=FALSE)
                  ~ edges + triangle, coef=c(0, 0),nsim=m)
S<-print(g.sim,stats.print=TRUE)
eta<-attr(S,"stats")
theta=c(-1,0.3)
Theta<-matrix(rep(theta, m),nrow = m)
s<-c()
for(i in 1:m){
s[i]<-exp(Theta[i,]%*%eta[i,])
}
Z_Theta<-2^10*(sum(s)/m)
Z_Theta
```

Obtemos $Z(\Theta) = 2462486387$

2. Estimadores de Máxima Verosimilhança (ERGM)

Após a construção de modelos de grafos aleatórios vem o problema, empírico, de encontrar o valor dos parâmetros destes modelos a partir das observações, que na *Estatística* se conhece como *Inferência Paramétrica*. Inicialmente vamos descrever o paradigma clássico de estimação paramétrica que se conhece como 'Método da Máxima

Verosimilhança', depois, discutiremos como aplicamos este método para o parâmetro θ del modelo ERGM.

2.0.1. *Estimadores de Máxima Verosimilhança.* Suponha que temos m observações independentes: $x_1, \dots x_m$ de um modelo aleatório com distribuição $\mathbb{P}_\theta$, para esta 'amostra' definimos a *verosimilhança* como a função:

$$L := L(\theta, x_1, \dots x_m) = \prod_{i=1}^{m} \mathbb{P}_\theta(x_i)$$

O método da máxima verosimilhança consiste em encontrar um θ que maximize a função L. θ também seria o máximo do logaritmo de L, então, definimos o estimador de máxima verosimilhança (EMV) do parâmetro θ como o valor:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \log L(\theta, x_1, \dots x_m) = \arg\max_{\theta \in \Theta} \ell(\theta, \underset{\sim}{X}).$$

Para distribuições (da família exponencial) que tem a forma, $\mathbb{P}_\theta(x) = ce^{\theta \cdot T(x)}$, vale a seguinte propriedade:

$$\mathbb{E}_{\hat{\theta}} T(X) = T(\underset{\sim}{X}) \tag{2.6}$$

2.0.2. *Estimadores de Máxima Verosimilhança para ERGM.* Para o modelo ERGM temos uma única observação, o grafo G_o, com verosimilhança dada por:

$$\mathbb{P}_\theta(G_o) = \frac{e^{\theta \cdot \eta(G_o)}}{Z(\theta)},$$

e o EMV de θ é

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \ell(\theta, G_o).$$

O problema de avaliar $\ell(\theta, G_o)$ é que necessitamos o valor da função de partição $Z(\theta)$, vamos a usar a mesma ideia de simulações para achar una expressão aproximada para $\ell(\theta, G_o)$, e suponha que podemos simular da distribuição $\mathbb{P}_{\theta_0}$. Definimos $r(\theta, \theta_0) = \ell(\theta, G_o) - \ell(\theta_0, G_o)$.

Observe que:

$$\begin{aligned} \ell(\theta, G_o) - \ell(\theta_0, G_o) &= \log\left(\frac{e^{\theta \cdot \eta(G_o)}}{Z(\theta)}\right) - \log\left(\frac{e^{\theta_0 \cdot \eta(G_o)}}{Z(\theta_0)}\right) \\ &= (\theta - \theta_0) \cdot \eta(G_o) - \log\left(\frac{Z(\theta)}{Z(\theta_0)}\right). \end{aligned}$$

Como visto antes simulamos m grafos $G_1, \dots G_m$ usando $\mathbb{P}_{\theta_0}$ e utilizando a lei dos grandes números podemos aproximar o valor de

$$\log\left(\frac{Z(\theta)}{Z(\theta_0)}\right) \approx \log\left(\frac{1}{m}\sum_{i=1}^{m} e^{(\theta - \theta_0) \cdot \eta(G_i)}\right).$$

Finalmente,, usamos o valor aproximado para definir:

$$r_m(\theta, \theta_0) = (\theta - \theta_0) \cdot \eta(G_o) - \log\left(\frac{1}{m}\sum_{i=1}^{m} e^{(\theta - \theta_0) \cdot \eta(G_i)}\right).$$

Assim, o EMV aproximado para o modelo é:

$$\hat{\theta}_m = \arg\max_{\theta \in \Theta} r_m(\theta, \theta_0).$$

Como o modelo pertence à família exponencial, então, o EMV $\hat{\theta}$ satisfaz a relação (2.6),

$$\mathbb{E}_{\hat{\theta}}\eta(X) = \eta(G_o).$$

O programa **ergm** do R implementa esta solução. Veja o seguinte exemplo.

EXEMPLO 2.1. *Usaremos como exemplo o grafo $G2$ simulado a partir de um ERGM com estatísticas: o número de vértices $\eta_1(G) = |E|$ e $\eta_1(G) = |$triângulos$(G)|$ com $\theta = (\theta_1, \theta_2) = (1, 0.3)$:*

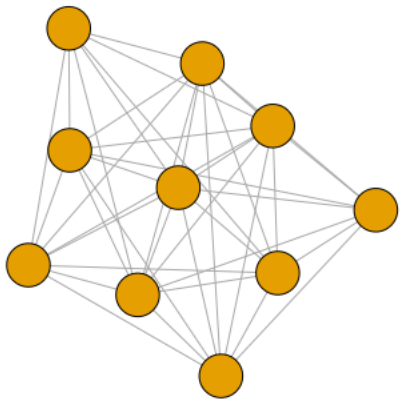

FIGURA 9. Grafo $G2$ do modelo ERGM com $\theta_1 = 1$, $\theta_2 = 0.3$

Para este grafo a saída do programa é:

```
theta_est<-ergm(G2 ~ edges+ triangle,control=control.ergm(MCMC.burnin=100))
summary(theta_est)

==========================
Summary of model fit
==========================

Formula:   G2 ~ edges + triangle

Iterations:  2 out of 20

Monte Carlo MLE Results:
         Estimate Std. Error MCMC % p-value
edges      0.0197     2.4155      0   0.994
triangle   0.3822     0.3916      0   0.334
```

O valor do EMV é $\hat{\theta} = (0.0197, 0.3822)$. A estimação não foi boa (p-valor para θ_1 é 0.994, muito alto, e p-valor para θ_2 é 0.334, baixo, mais não aceitável, pois aceitamos valores menores que 0.025), isto era esperado pois, estamos utilizando uma aproximação bastante regular para a verosimilhança. Podemos ver que utilizando técnicas Bayesianas podemos melhorar o resultado.

Avaliamos a estimação usando a distribuição de outras características: grau, arestas comuns e geodésicas de um conjunto de grafos simulados com os parâmetros estimados.

```
gofG2 <- gof(theta_est)

par(mfrow=c(1,3))
par(oma=c(0.5,2,1,0.5))
plot(gofG2)
```

o resultado é dado por:

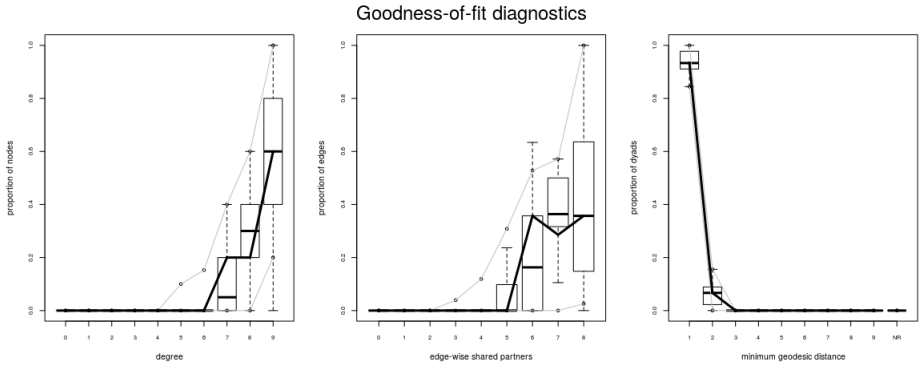

FIGURA 10. Bondade do ajuste (goodness of fit: gof) para o valor estimado $\hat{\theta} = (0.0197, 0.3822)$

Podemos observar na Figura (10) que estas outras características do grafo $G2$ (a linha mais obscura dos gráficos correspondentes ao grau, arestas comuns e geodésicas) se encontram dentro dos valores esperados das simulações (as linhas cinça dos gráficos).

3. Estimadores Bayesianos (ERGM)

A estadística Bayesiana se baseia no famoso teorema de Bayes:

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}.$$

Supondo que o parâmetro que será estimado θ é aleatório e que sua distribuição é $\mathbb{P}(\theta)$, podemos expressar a verosimilhança como $L(\theta, \underset{\sim}{X}) = \mathbb{P}(\underset{\sim}{X}|\theta)$ e reformular o problema original de achar θ ao problema de achar a distribuição de θ condicional à informação observada $\underset{\sim}{X}$,

$$\mathbb{P}(\theta|\underset{\sim}{X}) = \frac{\mathbb{P}(\underset{\sim}{X}|\theta)\mathbb{P}(\theta)}{\mathbb{P}(\underset{\sim}{X})}.$$

Nesta expressão $\pi(\theta) := \mathbb{P}(\theta)$ é chamada de distribuição **a priori** e, o objetivo consiste em encontrar a distribuição **a posteriori** $\pi(\theta|\underset{\sim}{X}) := \mathbb{P}(\theta|\underset{\sim}{X})$. Observe que $\mathbb{P}(\underset{\sim}{X})$ não depende de θ, logo é constante, assim, a relação de Bayes pode ser expressa como:

$$\mathbb{P}(\theta|\underset{\sim}{X}) = cL(\theta, \underset{\sim}{X})\pi(\theta),$$

onde c é uma constante. Existem dois métodos para encontrar uma expressão para a a posteriori. O primeiro consiste em encontrar uma distribuição que tenha a mesma expressão que a posteriori, neste caso dizemos que ambas, a priori e a posteriori são conjugadas, o que em general não ocorre.

O segundo método, mais general, consiste em encontrar um mecanismo que nos permite simular m observações da posteriori: $\theta_1, \ldots, \theta_m$, $\theta_i \sim \pi(\theta|\underset{\sim}{X})$. Dependendo do mecanismo de simulação (usualmente simulamos um processo estocástico) vale que:

$$\pi(\theta|\underset{\sim}{X}) \approx F_m(\theta|\underset{\sim}{X}) := \frac{1}{m}\sum_{i=1}^{m} \delta_{\theta_i}(\theta),$$

onde $\delta_a(x) = 1$ se $a = x$ e $\delta_a(x) = 0$, caso contrario. A função $F_m(\cdot)$ é chamada distribuição empírica. Usualmente a relação anterior é consequência da *ergodicidade* do processo (isto é da unicidade da distribuição estacionaria).

Um método que nos permite simular os θ's é o método de Metropolis-Hastings, que consiste em usar uma cadeia de Markov que seja fácil de simular e, digamos, que tenha probabilidade de transição $q(\theta'|\theta)$, então o objetivo consiste em perturbar esta cadeia de modo tal que a nova cadeia tenha a distribuição estacionaria desejada, isto é, a distribuição a posteriori. Suponha que a nova cadeia tem probabilidade de transição $k(\theta'|\theta) = q(\theta'|\theta)\alpha(\theta'|\theta)$. Se vale que:

$$k(\theta'|\theta)\pi(\theta|\underset{\sim}{X}) = k(\theta|\theta')\pi(\theta'|\underset{\sim}{X}),$$

isto é, que a cadeia modificada, com probabilidade de transição $k(\theta'|\theta)$, é 'reversível' para $\pi(\theta|\underset{\sim}{X})$. Então, temos necessariamente que $\pi(\theta|\underset{\sim}{X})$ é a única distribuição estacionaria para $k(\theta'|\theta)$. Esta é uma propriedade das cadeias de Markov muito utilizada na simulação de processos.

Usando as ideias anteriores o algoritmo de Metropolis-Hastings, que nos permite simular os θ's necessários para encontrar a distribuição empírica $F_m(\cdot)$, resume-se aos seguintes passos:

Passo 1 Simulamos θ_0 da distribuição a priori $\pi(\theta)$,
Passo 2 Supondo que tenhamos simulado θ_n, simulamos um candidato θ^* usando $q(\theta, \theta_n)$.
Passo 3 Avaliamos:

$$\alpha(\theta'|\theta) = \min\left(\frac{\pi(\theta^*|\underset{\sim}{X})q(\theta^*, \theta_n)}{\pi(\theta_n|\underset{\sim}{X})q(\theta_n, \theta^*)}, 1\right) = \min\left(\frac{cL(\theta^*, \underset{\sim}{X})\pi(\theta^*)q(\theta^*, \theta_n)}{cL(\theta_n, \underset{\sim}{X})\pi(\theta_n)q(\theta_n, \theta^*)}, 1\right).$$

Observe que a constante c cancela-se. Geralmente usamos uma cadeia tal que $q(u,u') = q(u',u)$ pelo que a expressão anterior se simplifica:

$$\alpha(\theta'|\theta) = \min\left(\frac{L(\theta^*, \underset{\sim}{X})\pi(\theta^*)}{L(\theta_n, \underset{\sim}{X})\pi(\theta_n)}, 1\right)$$

Passo 4 Simulamos uma observação u da distribuição Uniforme e fazemos:

$$\theta_{n+1} = \begin{cases} \theta^*, & \text{se } u \leq \alpha(\theta'|\theta), \\ \theta_n, & \text{se } u > \alpha(\theta'|\theta). \end{cases}$$

Passo 5 Volvemos ao passo 2.

Os passos são iterados até que os θ's estejam 'estacionários'. O algoritmo foi chamado da 'revolução computacional' por Persi Diaconis, ver [8].

3.0.1. *Estimadores Bayesianos (ERGM).* Consideramos um grafo observado G_o que segue o modelo ERGM com verosimilhança:

$$L(\theta, G_o) = \mathbb{P}_\theta(G_o) = \frac{e^{\theta \cdot \eta(G_o)}}{Z(\theta)}$$

e distribuição a priori $\pi(\theta)$ queremos encontrar a distribuição a posteriori $\pi(\theta|G_o)$ usando o algoritmo de Metropolis-Hastings. Suponha que os θ's sejam gerados usando uma cadeia de Markov simétrica: $q(\theta, \theta') = q(\theta', \theta)$. Então, para avaliar o $\alpha(\theta'|\theta)$ do algoritmo temos que avaliar:

$$\frac{L(\theta^*, G_o)\pi(\theta^*)}{L(\theta_n, G_o)\pi(\theta_n)} = \frac{\pi(\theta^*)e^{(\theta^* - \theta_n)\cdot\eta(G_o)}}{\pi(\theta_n)} \frac{Z(\theta_n)}{Z(\theta^*)},$$

Como podemos ver a dificuldade de implementar isto, consiste em avaliar $Z(\theta_n)/Z(\theta^*)$ e como foi visto, isso da origem a um error muito grande. Caimo et. al. (2012) modificaram este algoritmo, usando uma simulação auxiliar para θ, veremos no exemplo este resultado.

EXEMPLO 3.1. *Usaremos como exemplo o grafo $G2$ simulado a partir de um ERGM com estadísticas: o número de vértices $\eta_1(G) = |E|$ e $\eta_1(G) = |$triângulos$(G)|$ com $\theta = (\theta_1, \theta_2) = (1, 0.3)$; em lugar de procurar $\hat{\theta}$, vamos a supor que, 'a priori', este valor é representado por uma distribuição, a priori, $\pi(\theta_1, \theta_2) \sim$ Uniforme em $[0,2] \times [0,1]$ e queremos a distribuição empírica da a posteriori $\pi((\theta_1, \theta_2)|G_o)$. Para isso usaremos o programa* **bergm** *do R.*

```
library("ergm")
library("bergm")
g.sim <- simulate(network(10,,density=0.1,directed=FALSE) ~ edges + triangle,
coef=c(0, 0))

G2 <- simulate( ~ edges + triangle, coef=c(1, 0.3),directed=FALSE,nsim=1,
                basis=g.sim,seed=786,control=control.simulate(
                  MCMC.burnin=10000,
                  MCMC.interval=100))
theta_post <- bergm(G2 ~ edges + triangle,
             burn.in=10,
             aux.iters=500,
             main.iters=500,
             gamma=1)
```

```
bergm.output(theta_post)
```

e os resultados são:

```
  Overall posterior density estimate:
          theta1 (edges) theta2 (triangle)
Post. mean       3.377747       -0.02407964
Post. sd         5.224419        0.73015466

 Overall acceptance rate: 0.31
```

Podemos observar que as medias da distribuição empírica $F_m(\theta_1, \theta_2)$ são,

$$(\bar{\theta_1}, \bar{\theta_2}) = (3,377747; -0,02407964),$$

não muito diferentes dos valores originais, lembre que agora os θ's são aleatórios. Observe que a probabilidade de aceitação de cada θ foi de 0,31.

A seguir pode ser observada a convergência das cadeias para a estacionaridade:

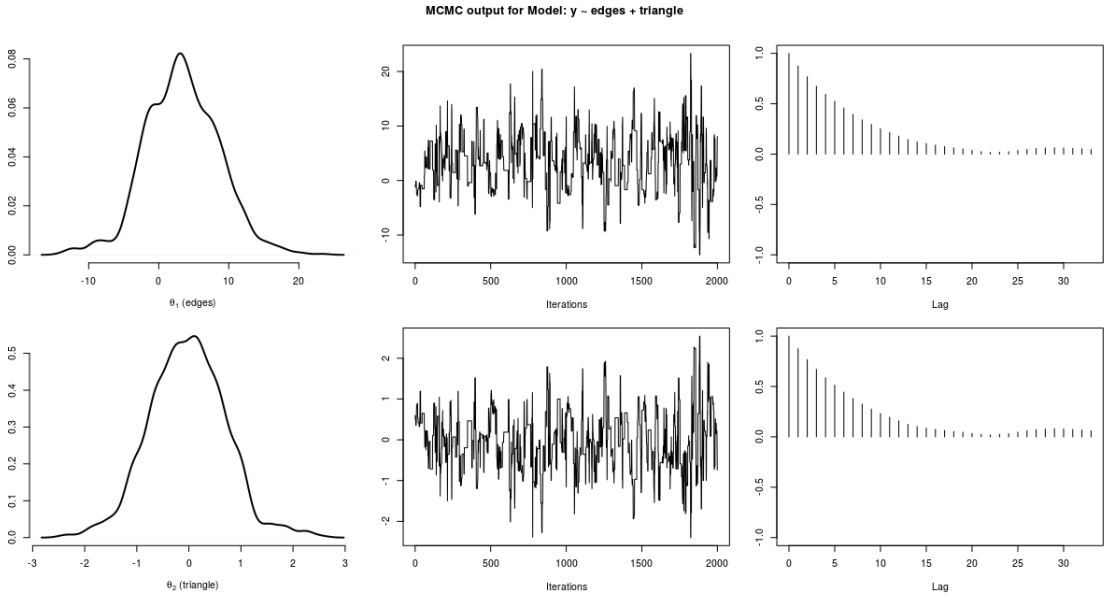

FIGURA 11. Distribuição a posteriori e convergência do processo $(\theta_1(n), \theta_2(n))$

4. Usando R

Neste capitulo utilizamos os programas do R: igraph, intergraph, network, ergm e bergm.

(1) Figura 1:

```
G1 <- graph.formula(1,2,3)
G2 <- graph.formula(1--2,3)
G3 <- graph.formula(1,2--3)
G4 <- graph.formula(1--3,2)
G5 <- graph.formula(1--2--3)
G6 <- graph.formula(1--3--2)
```

```
G7 <- graph.formula(2--1--3)
G8 <- graph.formula(1--2--3,3--1)
graphics.off()
par(mfrow=c(3,3))
plot(G1, xlab='G1',vertex.size=55,layout=layout_in_circle)
plot(G2, xlab='G2',vertex.size=55,layout=layout_in_circle)
plot(G3, xlab='G3',vertex.size=55,layout=layout_in_circle)
plot(G4, xlab='G4',vertex.size=55,layout=layout_in_circle)
plot(G5, xlab='G5',vertex.size=55,layout=layout_in_circle)
plot(G6, xlab='G6',vertex.size=55,layout=layout_in_circle)
plot(G7, xlab='G7',vertex.size=55,layout=layout_in_circle)
plot(G8, xlab='G8',vertex.size=55,layout=layout_in_circle)
plot(5,5,type="n",axes=FALSE,ann=FALSE,xlim=c(0,10),
ylim =c(0,10))
```

(2) Figura 2:

```
graphics.off()
par(mfrow=c(1,3))
plot(G5, xlab='G5',vertex.size=35,layout=layout_in_circle)
plot(G6, xlab='G6',vertex.size=35,layout=layout_in_circle)
plot(G7, xlab='G7',vertex.size=35,layout=layout_in_circle)
```

(3) Figura 3:

```
G<-erdos.renyi.game(30, 100, type ="gnm", directed = FALSE,
                 loops = FALSE)
plot(G)
```

(4) Figura 4:

```
G1<-erdos.renyi.game(10, 0.3, type ="gnp", directed = FALSE,
                 loops = FALSE,)
G2<-erdos.renyi.game(10, 0.5, type ="gnp", directed = FALSE,
                      loops = FALSE,)
G3<-erdos.renyi.game(10, 0.7, type ="gnp", directed = FALSE,
                      loops = FALSE)


graphics.off()
par(mfrow=c(1,3))
plot(G1,xlab='G1')
plot(G2,xlab='G2')
plot(G3,xlab='G3')
```

(5) Figuras 5 e 6:

```
graphics.off()
G1 <- sample_pa(500, power=0.1,directed = FALSE)
G2 <- sample_pa(500, power=0.001,directed = FALSE)
graphics.off()
par(mfrow=c(1,2))
plot(G1,vertex.label=NA,vertex.size=3)
plot(G2,vertex.label=NA,vertex.size=3)
graphics.off()
par(mfrow=c(1,2))
par(mfrow=c(1,2))
hist(degree(G1),main='')
hist(degree(G2),main='')
```

(6) Figuras 7 e 8:

```
library("ergm")
g.sim <- simulate(network(10,,density=0.1,directed=FALSE)
~ edges + triangle, coef=c(0, 0))
summary(g.sim ~ edges + triangle)
G1 <- simulate( ~ edges + triangle, coef=c(-1, 0.3),
directed=FALSE,nsim=1,basis=g.sim,seed=234,
control=control.simulate(
  MCMC.burnin=10000,
  MCMC.interval=100))
summary(G1 ~ edges + triangle)
G2 <- simulate( ~ edges + triangle, coef=c(1, 0.3),
directed=FALSE,nsim=1,basis=g.sim,seed=786,
control=control.simulate(
                 MCMC.burnin=10000,
                 MCMC.interval=100))
summary(G2 ~ edges + triangle)
g1<-asIgraph(G1)
g2<-asIgraph(G2)

par(mfrow=c(1,2))
plot(g1,vertex.label=NA,vertex.size=25,xlab='G1')
plot(g2,vertex.label=NA,vertex.size=25,xlab='G2')
par(mfrow=c(1,2))
hist(degree(g1),main='', xlab='grau(G1)',
xlim = c(0,10),ylim = c(0,7))
hist(degree(g2),main='', xlab='grau(G2)',
xlim = c(0,10),ylim = c(0,7))
```

CAPíTULO 4

Modelos de Regressão para Grafos e Modelos Dinâmicos

Neste capítulo discutiremos alguns modelos de regressão para grafos: vizinhos próximos, campos de Markov, espectrais e por Kernels. Um exemplo sobre genética sera apresentado e ser utilizados parte dos códigos de R do libro [7].

Além disso discutiremos um modelo dinâmico para grafos, o modelo de epidemiologia SIR (susceptível, infectado e recuperado), neste modelo veremos como influi na velocidade de propagação da doença quando se considera a infeção propagando-se num grafo.

1. Modelos de Regressão ou Previsão para Grafos

Estes modelos são preditivos no sentido de que uma variável aleatória Y é explicada, ou prevista, por outra variável aleatória X, isto é, como função de X. Esta função usualmente é linear, de aí o nome de 'regressão linear'. Existem muitas variedades de modelos de regressão e para o caso multilinear a variável X é trocada por um vetor, usando um grafo podemos generalizar este modelo.

No caso de aplicações de regressão usando grafos, as variáveis estão indexadas com os vértices do grafo, como num campo de vectores, a variável tem um valor em cada vértice. A previsão consiste em que saber o valor da variável num vértice a partir dos valores dos vértices conectados a ele.

1.1. Regressão Linear. No caso mais simples, não aleatório, temos um conjunto de pontos $(x_1, y_1), \ldots, (x_n, y_n)$ no plano e queremos encontrar uma reta $Y = \hat{\alpha} + \hat{\beta}X$ cujos pontos minimizam a distância ao conjunto de pontos observados. $\hat{\alpha}$ é chamado o interceptor e $\hat{\beta}$ a inclinação.

Encontrar $(\hat{\alpha}, \hat{\beta})$ corresponde a resolver o problema de minimização:

$$\min_{(\alpha,\beta)} \{(Y - \alpha - \beta X)^t (Y - \alpha - \beta X)\},$$

onde $X = (x_1, \ldots, x_n)$ e $Y = (y_1, \ldots, y_n)$.

Como $(Y - \alpha + \beta X)^t (Y - \alpha - \beta X) = ||Y - \alpha - \beta X||_2^2$, chama-se a este problema de minimização quadrática, este valor é a distancia euclídea entre Y e $\alpha + \beta X$. Lembre que distância$(a, b) = ||a - b||_2 = \sum_i (a_i - b_i)^2$, $a, b \in \mathbb{R}^n$.

No caso aleatório, os pontos $(x_1, y_1), \ldots, (x_n, y_n)$ são observações do vector aleatório (X, Y), cujas marginais X e Y tem uma relação de dependência: $Y = \alpha + \beta X + \varepsilon$, aqui, ε é uma variável aleatória com esperança $\mathbb{E}(\varepsilon) = 0$ e variância $\mathbb{V}(\varepsilon) = \sigma^2 > 0$, logo $\mathbb{E}(Y|X = x) = \alpha + \beta x$ por isso dizemos que este modelo é o melhor preditor lineal para Y.

A solução deste problema consiste em minimizar:

$$\min_{(\alpha,\beta)} \mathbb{E}\{(Y-\alpha-\beta X)^t(Y-\alpha-\beta X)\},$$

Observe que ECM:$= \mathbb{E}\{(Y-\alpha-\beta X)^t(Y-\alpha-\beta X)\} = \mathbb{E}(Y-\alpha-\beta X)^2$, o **erro quadrático médio** é a quantia a ser minimizada.

1.2. Regressão Vizinhos Próximos. A regressão por vizinhos próximos, NN (Nearest Neirghbor), é um tipo de regressão local. Suponha que temos como dado um grafo $G=(V,E)$ e que em cada vértice $i \in V$ temos uma variável X_i com a mesma distribuição . Mais desconhecemos a relação de dependência entre estas variáveis, sabemos que estão ligadas pelo grafo G.

Se temos observações $\{x_i, i \in V\}$ nosso objetivo é predizer o valor de cada x_i a partir dos valores de x no seus vizinhos $\mathcal{N}_i = \{j \in V : (i,j) \in E\}$, o preditor NN para x_i neste caso está definido por:

$$\hat{x}_i = \frac{\sum_{j\in\mathcal{N}_i} x_j}{|\mathcal{N}_i|}.$$

EXEMPLO 1.1. *Vamos a usar um exemplo da genética, os dados consistem de uma rede de 241 interações (arestas) entre 134 proteínas (vértices) pertencentes à um organismo, Saccharomyces cerevisiae, cada proteína tem um atributo ($(ICSC)_i = 1$ o 0) que indica se a 'cascada de sinalização intracelular' (intracellular signaling cascade) está presente o não, esta é uma característica de comunicação intracelular. Estos dados foram colectados por [6]. Este grafo é chamado $ppi.CC$,*

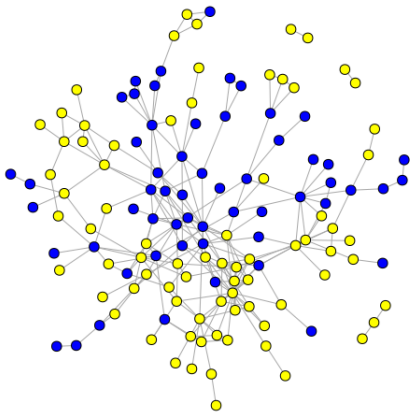

FIGURA 1. Grafo $ppi.CC$ de interações entre proteínas de *Saccharomyces cerevisiae*: $ICSC = 1$ (amarelo) e $ICSC = 0$ (azul)

Se usamos o grafo $ppi.CC$ e a relação de vizinhos próximos descrita antes obtemos um grafo $ppi.CC.nn$ igual ao anterior pero com valores de ICSC previstos por esta relação NN, podemos comparar os dois gráficos:

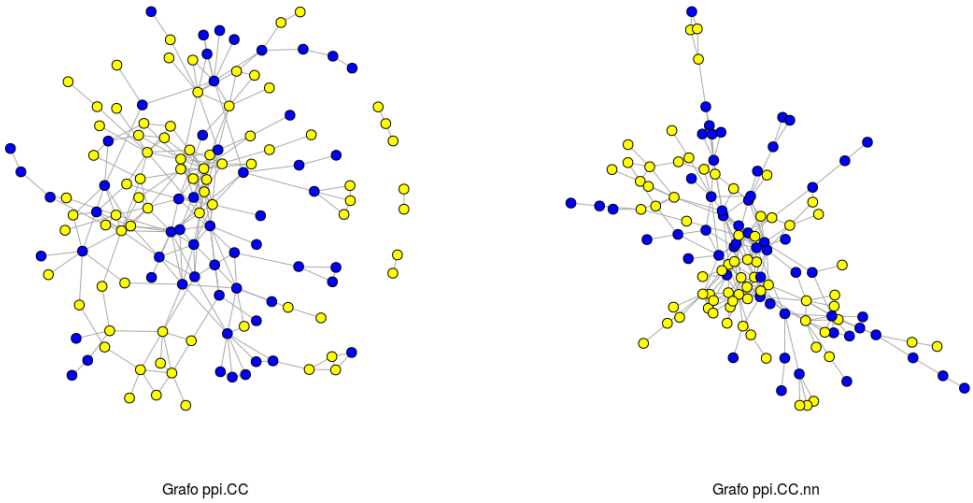

FIGURA 2. Grafos $ppi.CC$ e $ppi.CC.nn$

Para verificar que tanto coincidem os valores dos vértices vamos a contar as coincidências em cada vértice:

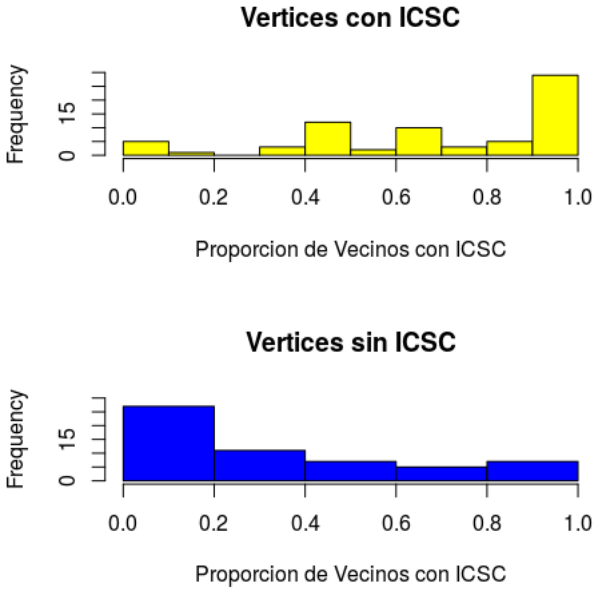

FIGURA 3. Coincidências entre os valores dos vértices $ppi.CC$ e $ppi.CC.nn$

Observe que valores próximos a 1 (cor amarelo) tem uma frequência alta de coincidência, isto é, poucos dos vértices que tem valor perto de zero são amarelos em $ppi.CC$. (o mesmo acontece para $(ICSC)_i = 0$ em $ppi.CC$). Vejamos qual a porcentagem de mal classificados:

```
  #### %de error: não coincidencias
nn.pred <- as.numeric(nn.ave > 0.5)
mean(as.numeric(nn.pred != V(ppi.CC.gc)$ICSC))
[1] 0.2598425
```

O erro de predição é 25,9 $\%$, que é baixo.

1.3. Regressão Linear para Campos de Markov. Nesta subseção vamos definir um modelo aleatório para as observações x_i dos vértices, este modelo é chamado **Campo Aleatório Marcoviano** .

Seja $G = (E, V)$ um grafo com $|V| = n$ e $X = (X_1, \ldots, X_n)$ um conjunto de variáveis aleatórias definidas em V, dizemos que X é um Campo Aleatório Markoviano (MRF: Markov Random Field em inglês) se:

(1) $\mathbb{P}(X = x) > 0$ para todo $x \in \mathbb{R}^n$.

(2) É valida a propriedade de Markov:

$$\mathbb{P}(X_i = x_i | X_{(-i)} = x_{(-i)}) = \mathbb{P}(X_i = x_i | X_{\mathcal{N}_i} = x_{\mathcal{N}_i})$$

com $X_{(-i)} = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n)$ e $X_{\mathcal{N}_i}$ é X restrito aos vértices $j \in \mathcal{N}_i = \{j \in V : (i, j) \in E\}$

A propriedade (2) diz que a distribuição de X_i, depende somente dos vizinhos do vértice i, isto é, ele é Markov, esta distribuição que depende somente dos vizinhos é chamada de 'especificação local' do modelo.

O modelo MRF é comum na estatística espacial ou nos modelos para o processamento digital de imagens, origina-se no trabalho de Besag (1976), ver [1]. Uma propriedade dos MRF, o teorema de Hammersley–Clifford, que permite representar X como:

$$\mathbb{P}(X = x) = \frac{e^{U(x)}}{Z} \tag{1.7}$$

onde $U(x)$ é uma função real, chamada 'Potencial', e $Z = \sum_x e^{U(x)}$ é a função de partição, podemos ver que este modelo inclui os modelos exponenciais. Através desta propriedade é possível que posamos fazer modelos de regressão lineal. Distribuições do tipo (1.7) são chamadas **medidas de Gibbs** em honor ao físico J. W. Gibbs que as descreveu no seculo XIX.

É possível mostrar que $U(x) = \sum_{c \in \mathcal{C}} U_c(x)$ onde $\mathcal{C}$ é o conjunto de *cliques* ou grafos completos K_n contidos em G. Por exemplo, um vértice é K_1 (o grafo completo com 0 arestas e um vértice), uma aresta é K_2 (o grafo completo com 1 aresta e 2 vértices), um triângulo é K_3 (o grafo completo com 3 arestas e 3 vértices), etc. Em consequência a equação (1.7) diz que a distribuição de X depende dos valores de $U(x)$ nestes 'cliques'.

O modelo MRF que vamos utilizar é chamado de **Modelo Auto-Logístico** e corresponde a um potencial que somente depende dos valores nos vértices (K_1) e nas arestas (K_2), além do mais, os valores de x_i podem ser só 1 o 0, isto é,

$$U(x) = \sum_{i \in V} \alpha x_i + \sum_{(i,j) \in E} \beta x_i x_j.$$

È possível provar que nesse caso a especificação local do Modelo Auto-logístico é dada por:

$$\mathbb{P}_{\alpha,\beta}(X_i = 1 | X_{\mathcal{N}_i} = x_{\mathcal{N}_i}) = \frac{e^{\alpha + \beta \sum_{j \in \mathcal{N}_i} x_j}}{1 + e^{\alpha + \beta \sum_{j \in \mathcal{N}_i} x_j}}.$$

Então, vale que:

$$\log \frac{\mathbb{P}_{\alpha,\beta}(X_i = 1 | X_{\mathcal{N}_i} = x_{\mathcal{N}_i})}{\mathbb{P}_{\alpha,\beta}(X_i = 0 | X_{\mathcal{N}_i} = x_{\mathcal{N}_i})} = \alpha + \beta \sum_{j \in \mathcal{N}_i} x_j.$$

Assim é possível fazer regressão linear para as especificações locais.

EXEMPLO 1.2. *Continuando com o exemplo das interações com as proteínas, vamos usar o modelo auto-logístico para predizer os valores de $ICSC$ no grafo $ppi.CC$ e logo comparar os resultados com o grafo realmente observado; primeiro, temos que encontrar α e β, para isto usamos o programa* **ngspatial** *de estatística espacial do R , usando o comando* **autologistic** *obtemos:*

```
 m1.mrf <- autologistic(formula1, A=A,control=list(confint="none"))
m1.mrf$coefficients
(Intercept)         eta
  0.2004949   1.1351942
```

Logo, os coeficientes são $\alpha = 0,2$ e $\beta = 1,13$ uma comparação gráfica dos grafos:

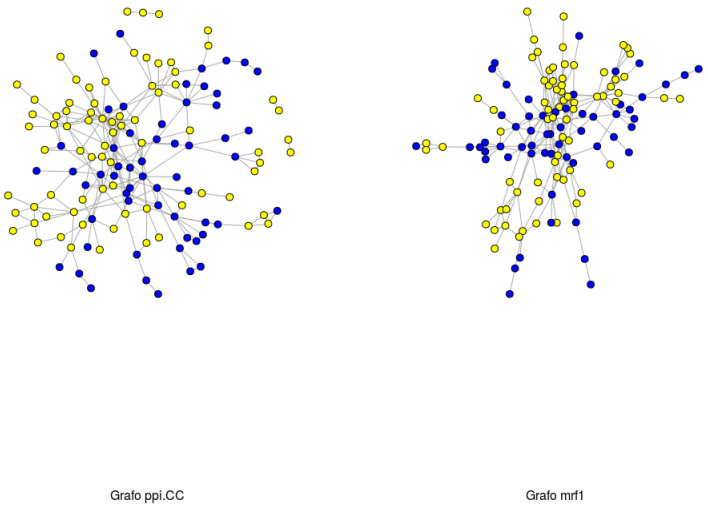

FIGURA 4. Grafos $ppi.CC$ e $mrf1$

como antes, vamos verificar quantos vértices coincidem:

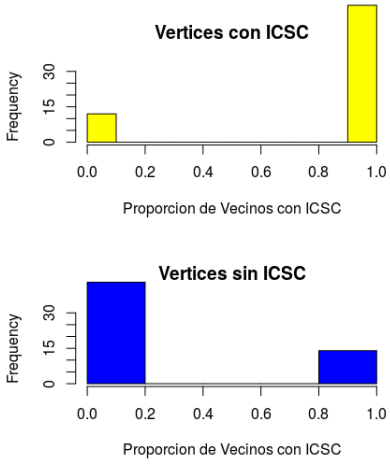

FIGURA 5. Coincidências entre os valores dos vértices $ppi.CC$ e $mrf1$

A porcentagem de não coincidências neste caso é de 20%, que é mais baixa que do modelo de vizinhos próximos.

```
  > mean(as.numeric(mrf1.pred != V(ppi.CC.gc)$ICSC))
[1] 0.2047244
```

Finalmente, como em todos os modelos de regressão linear podemos pesquisar o que tanto é explicado pelo modelo:

```
  ####explicação do  modelo
> assortativity(ppi.CC.gc, X+1, directed=FALSE)
[1] 0.3739348
```

Os parâmetros do modelo explicam 37,39% da informação contida nos dados do grafo $ppi.CC.gc.$

1.4. Regressão por Kernels para Grafos. Os modelos de predição que usam Kernels são conhecidos nas pesquisas sobre modelos de auto-aprendizado ou *Machine Leraning*, são regressões locais, usualmente se usa uma rede para estimar os parâmetros locais e logo se faz predição do comportamento da nova informação. Um **Kernel** $K = (k_{i,j})_{\{i,j\in V\}}$ é uma matriz simétrica semi-definida positiva, isto é, $k_{i,j} = k_{j,i}$ e $x^tKx \geq 0$ para todo $x \in \mathbb{R}^{|V|}$; além disso satisfaz para todo subconjunto de m vértices $V^{(m)} = \{v_{i_1}, \ldots, v_{i_m}\}$ a restrição de K a esse conjunto $K^{(m)} = (k_{i,j})_{\{i,j\in V^{(m)}\}}$, ela também é simétrica e semi-definida positiva.

Uma propriedade das matrizes definidas positivas (respetivamente semi-definidas positivas), um Kernel é semi-definida positiva, é que elas definem uma distância (respetivamente quase-distância) entre os vectores:

$$d(x, y) = (x - y)^t K(x - y),$$

esta distância é uma norma, pois K é simétrica.

Isto nos permite considerar o problema de regressão local:

$$\min_{\alpha}\{(X - K^{(m)}\alpha)^t(X - K^{(m)}\alpha) + \lambda\alpha^t(K^{(m)})^{-1}\alpha\}. \tag{1.8}$$

Para usar esta regressão em um grafo G, vamos usar o Laplaciano $L = D - A$ do grafo que é uma matriz simétrica e semi-definida positiva , onde A é a matriz de Adjacência e $D = \mathrm{diag}(d(i))$. Fazemos $K = L^{-1}$ (esta é uma pseudo inversa pois existem autovalores zero) que também é simétrica y semi-definida positiva. Vale a 'decomposição espectral':

$$K = \Phi^t\Delta\Phi^t,$$

donde Φ esta formado pelos auto-vetores de K e $\Delta = \mathrm{diag}(\delta_i)$ são os auto-valores de $K = L^{-1}$, isto é, $\gamma_i = \delta_i^{-1}$ são os auto-valores de L o Laplaciano de G (se existem auto-valores zero, fazemos$\delta_i := 0$):

$$L^{-1} = \sum_{i\in V} f(\gamma_i)\phi^t\phi,$$

com $f(x) = 1/x, x \neq 0$ e $f(x) = 0, x = 0$.

Usando esta decomposição podemos mostrar que,

$$\alpha^tK^{-1}\alpha = h^tLh = \sum_{(i,j)\in E}(h_1 - h_j)^2,$$

onde $h = \Phi\alpha$ é uma combinação linear dos auto-vetores. Logo na minimização serão favorecidos vértices que sejam adjacentes.

EXEMPLO 1.3. *Vamos usar a regressão local para predizer os valores de* $ICSC$ *no grafo* $ppi.CC$ *. Primeiro, vejamos a distribuição dos* $f(\gamma_i)$*:*

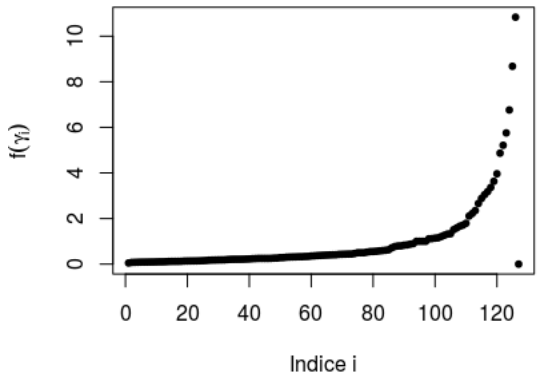

FIGURA 6. Distribuição dos $f(\gamma_i)$

Podemos ver o efeito de selecionar os auto-vetores maiores em cada vértice do grafo:

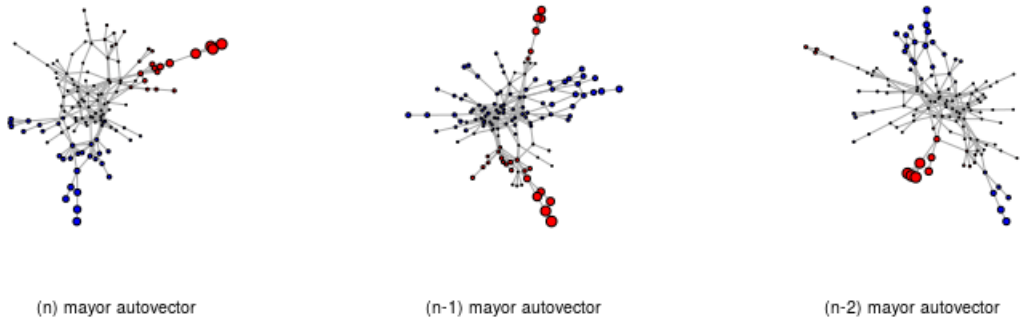

FIGURA 7. Auto-vetores maiores no radio de cada vértice do grafo

Finalmente o preditor usando kernels no grafo $ppi.CC$ *:*

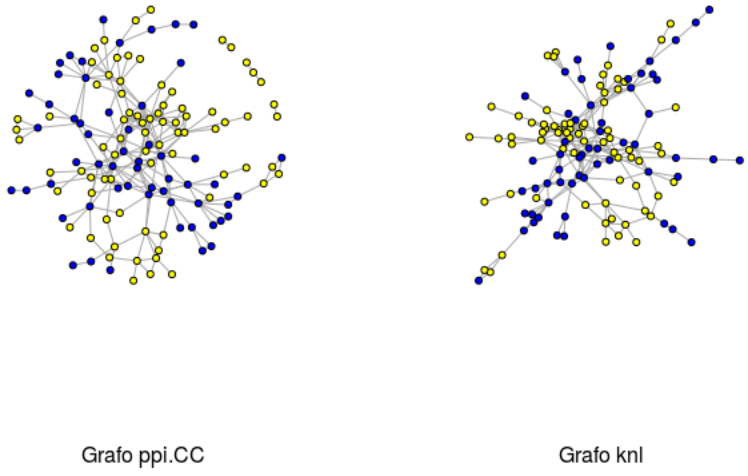

FIGURA 8. Grafos $ppi.CC.gc$ e knl

a comparação entre os dos:

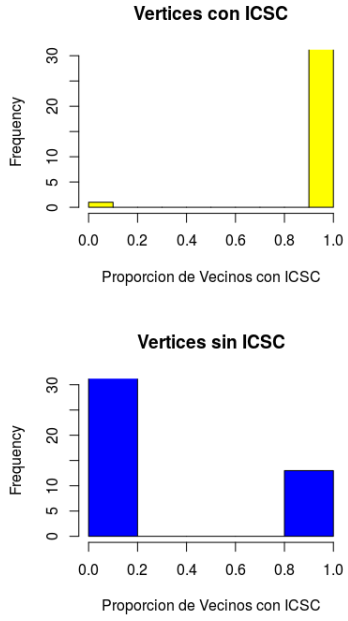

FIGURA 9. Coincidências entre $ppi.CC.gc$ e knl

a porcentagem de errores foi 11%

```
 > mean(as.numeric(m1.svm.fitted != V(ppi.CC.gc)$ICSC))
[1] 0.1102362
```

2. Outros Tópicos

Entre inúmeros tópicos sobre aplicações em grafos aleatórios podemos mencionar a análise de fenômenos como os grafos de escala-libre, como é o comportamento dos diferentes modelos de previsão, nos exemplos só temos considerado grafos finitos e relativamente pequenos, embora que é conhecido que a medida que os grafos aumentam de tamanho, fenômenos tais como: degeneração, transição de fase o criticalidade são relevantes, isto é, algumas das características mudam completamente.

Um dos tópicos que discutiremos é o efeito do uso de um grafo aleatório na dispersão de alguma doença. A continuação discutimos o modelo de infeção SIR. A seguir vamos usar o exemplo e os códigos de R do libro de [7] para este modelo.

2.1. Modelos Dinâmicos. Um dos modelos padrão da Epidemiologia é o processo SIR, **Susceptíveis-Infectados-Recuperados**. Neste processo observamos ao longo do tempo a evolução de uma doença, primeiro, o número de infectados na população cresce, para logo ir decrescendo a medida que vão ficando menos pessoas susceptíveis de serem infectadas pois se tem imunizado depois da recuperação, como está descrito esquematicamente no seguinte gráfico:

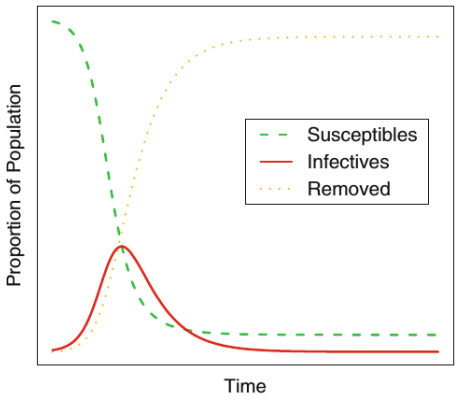

FIGURA 10. Evolução de uma infecção (vermelho), gráfico tomado de [7]

Para construir o processo, que descreve a evolução no gráfico, considere uma população de $N+1$ indivíduos, e digamos que em qualquer instante essa população decompõe-se em três tipos: Susceptíveis (S), Infectados (I) e Recuperados (R), assim teríamos que: $N+1 = N_S(t) + N_I(t) + N_R(t)$, onde $N_S(t)$ (resp.$N_I(t)$ $N_R(t)$) é o número de Susceptíveis (resp. Infectados, Recuperados) no instante t.

No instante inicial $N_I(0) = 1$ e $N_S(0) = N$, a evolução subsequente é governada por uma cadeia de Markov à tempo contínuo, isto é, por um processo estocástico que tem mudanças somente em certos tempos aleatórios, as mudanças estão governados

pelas seguintes probabilidades de transição:

$$\begin{aligned}\mathbb{P}(N_S(t+\delta t)=s-1, N_I(t+\delta t)=i+1|N_S(t)=s, N_I(t)=i) &\approx \beta si\delta t,\\ \mathbb{P}(N_S(t+\delta t)=s, N_I(t+\delta t)=i-1|N_S(t)=s, N_I(t)=i) &\approx \gamma i\delta t \text{ e}\\ \mathbb{P}(N_S(t+\delta t)=s, N_I(t+\delta t)=i|N_S(t)=s, N_I(t)=i) &\approx 1-(\beta s+\gamma)i\delta t.\end{aligned} \tag{2.9}$$

onde δt significa que esta transição acontece em um tempo infinitesimal depois de t, β é a taxa de infeção e γ a taxa de recuperação. O comportamento deste processo esta descrito na Figura (10).

O problema do modelo assim descrito é que as interações espaciais não existem, o que significa que um infectado pode contagiar a qualquer susceptível, o que não é realista. Uma maneira de melhorar o modelo é introduzir efeitos espaciais por meio de um grafo $G=(E,V)$ que descreva as relações entre os indivíduos. Para isso, consideramos o processo $X_i(t)=0,1,2$, com $i\in V$, onde 0 (respectivamente 1, 2) se o vértice i esta susceptível (respectivamente infectado, recuperado). O processo $X_i(t)$ obedece:

$$\mathbb{P}(X(t+\delta t)=x'|X(t)=x)=\begin{cases}\beta M_i(x)\delta t & \text{se } x_i=0 \text{ e } x_i'=1\\ \gamma\delta t & \text{se } x_i=1 \text{ e } x_i'=2\\ 1-[\beta M_i(x)+\gamma]\delta t & \text{se } x_i=2 \text{ e } x_i'=2\end{cases} \tag{2.10}$$

onde $M_i(x)$ é o número de vizinhos do vértice i que estão infectados $x_j=1, j\in\mathcal{N}_i$. Finalmente $N_S(t), N_I(t)$ e $N_R(t)$ são a soma dos vértices com valores 0, 1 ou 2.

Simulamos este processo usando o programa **SIR** do R, com $\beta = 0.5$ e $\gamma = 1$ e com grafos simulados dos modelos de Erdös-Rényi, Barabasi-Albert (BA) e Watts-Strogatz (este é outro modelo de escala-livre), para obter:

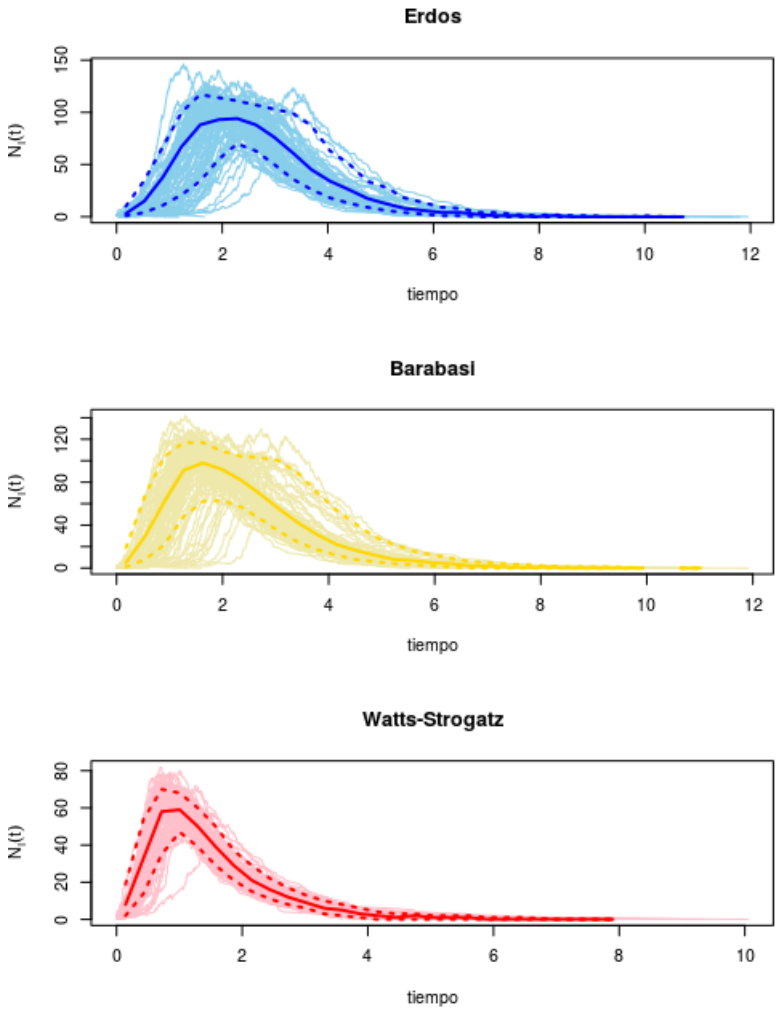

FIGURA 11. Evolução de uma infeção num grafo aleatório, exemplo tomado de [7]

3. Usando R

Utilizamos neste capitulo os programas do R: sand, ngspatial, kernlab e sir.

(1) Figura 1:

```
###datos
set.seed(42)
library(sand)
data(ppi.CC)
```

```
ppi.CC<-upgrade_graph(ppi.CC)
summary(ppi.CC)
### 10 primeros vertices
V(ppi.CC)$ICSC[1:10]
###visualizacion
V(ppi.CC)[ICSC == 1]$color <- "yellow"
V(ppi.CC)[ICSC == 0]$color <- "blue"
graphics.off()
plot(ppi.CC, vertex.size=5, vertex.label=NA)
```

(2) Figura 2:

```
####vizinhos  proximos
clu <- clusters(ppi.CC)
ppi.CC.gc <- induced.subgraph(ppi.CC,
                              clu$membership==which.max(clu$csize))
nn.ave <- sapply(V(ppi.CC.gc),
                 function(x) mean(V(ppi.CC.gc)[nei(x)]$ICSC))
class(nn.ave)
###grafo predecido
nn<-ppi.CC.gc
nn$ICSC<-nn.ave
V(nn)[ICSC == 1]$color <- "yellow"
V(nn)[ICSC == 0]$color <- "blue"
graphics.off()
par(mfrow=c(1,2))
plot(ppi.CC, vertex.size=5, vertex.label=NA,xlab="Grafo ppi.CC ")
plot(nn, vertex.size=5, vertex.label=NA, xlab="Grafo ppi.CC.nn ")
```

(3) Figura 3:

```
#### porcentage de coincidencias
par(mfrow=c(2,1))
hist(nn.ave[V(ppi.CC.gc)$ICSC == 1], col="yellow",
     ylim=c(0, 30), xlab="Proporcion de Vecinos com ICSC",
     main="Vertices com ICSC")
hist(nn.ave[V(ppi.CC.gc)$ICSC == 0], col="blue",
     ylim=c(0, 30), xlab="Proporcion de Vecinos com ICSC",
     main="Vertices sin ICSC")
```

(4) Figura 4:

```
##### campos markovianos
set.seed(42)
library(sand)
data(ppi.CC)
#### grafo G aqui corresponde a ppi.CC
ppi.CC<-upgrade_graph(ppi.CC)
V(ppi.CC)[ICSC == 1]$color <- "yellow"
V(ppi.CC)[ICSC == 0]$color <- "blue"
library(ngspatial)

clu <- clusters(ppi.CC)
ppi.CC.gc <- induced.subgraph(ppi.CC,
                              clu$membership==which.max(clu$csize))
X <- V(ppi.CC.gc)$ICSC
```

```
A <- get.adjacency(ppi.CC.gc, sparse=FALSE)
formula1 <- X ~ 1
#### Modelo autologistico
m1.mrf <- autologistic(formula1, A=A,control=list(confint="none"))
m1.mrf$coefficients

#####prediccion
mrf1.pred <- as.numeric((m1.mrf$fitted.values > 0.5))
####comparacion
mrf1<-ppi.CC.gc
mrf1$ICSC<-mrf1.pred
V(mrf1)[ICSC == 1]$color <- "yellow"
V(mrf1)[ICSC == 0]$color <- "blue"
graphics.off()
par(mfrow=c(1,2))
plot(ppi.CC, vertex.size=5, vertex.label=NA,xlab="Grafo ppi.CC ")
plot(mrf1, vertex.size=5, vertex.label=NA, xlab="Grafo mrf1 ")
```

(5) Figura 5:

```
par(mfrow=c(2,1))
hist(mrf1.pred[V(ppi.CC.gc)$ICSC == 1], col="yellow",
     ylim=c(0, 30), xlab="Proporcion de Vecinos com ICSC",
     main="Vertices com ICSC")
hist(mrf1.pred[V(ppi.CC.gc)$ICSC == 0], col="blue",
     ylim=c(0, 30), xlab="Proporcion de Vecinos com ICSC",
     main="Vertices sin ICSC")
```

(6) Figura 6:

```
###### autovectores mayores
e.vec1 <- e.L$vectors[, (nv-1)]
v1.colors <- character(nv)
v1.colors[e.vec1 >= 0] <- "red"
v1.colors[e.vec1 < 0] <- "blue"
v1.size <- 15 * sqrt(abs(e.vec1))
l1 <- layout.fruchterman.reingold(ppi.CC.gc)
#####
e.vec2 <- e.L$vectors[, (nv-2)]
v2.colors <- character(nv)
v2.colors[e.vec2 >= 0] <- "red"
v2.colors[e.vec2 < 0] <- "blue"
v2.size <- 15 * sqrt(abs(e.vec2))
l2 <- layout.fruchterman.reingold(ppi.CC.gc)
plot(ppi.CC.gc, layout=l2, vertex.color=v2.colors,
     vertex.size=v2.size, vertex.label=NA, xlab=c("(n-1) mayor autovector"))
#####
e.vec3 <- e.L$vectors[, (nv-3)]
v3.colors <- character(nv)
v3.colors[e.vec3 >= 0] <- "red"
v3.colors[e.vec3 < 0] <- "blue"
v3.size <- 15 * sqrt(abs(e.vec3))
l3 <- layout.fruchterman.reingold(ppi.CC.gc)
#####
```

```
graphics.off()
par(mfrow=c(1,3))
plot(ppi.CC.gc, layout=l1, vertex.color=v1.colors,
     vertex.size=v1.size, vertex.label=NA, xlab=c("(n) mayor autovector"))
plot(ppi.CC.gc, layout=l2, vertex.color=v2.colors,
     vertex.size=v2.size, vertex.label=NA, xlab=c("(n-1) mayor autovector"))
plot(ppi.CC.gc, layout=l3, vertex.color=v3.colors,
     vertex.size=v3.size, vertex.label=NA, xlab=c("(n-2) mayor autovector"))
```

(7) Figura 7:

```
     library(kernlab)
K1.tmp <- e.L$vectors %*% diag(f.e.vals) %*%
  t(e.L$vectors)
K1 <- as.kernelMatrix(K1.tmp)
m1.svm <- ksvm(K1, X, type="C-svc")
m1.svm.fitted <- fitted(m1.svm)
####comparación
knl<-ppi.CC.gc
knl$ICSC<-m1.svm.fitted
V(knl)[ICSC == 1]$color <- "yellow"
V(knl)[ICSC == 0]$color <- "blue"
graphics.off()
par(mfrow=c(1,2))
plot(ppi.CC, vertex.size=5, vertex.label=NA,xlab="Grafo ppi.CC ")
plot(knl, vertex.size=5, vertex.label=NA, xlab="Grafo knl ")
```

(8) Figura 8:

```
par(mfrow=c(2,1))
hist(m1.svm.fitted[V(ppi.CC.gc)$ICSC == 1], col="yellow",
     ylim=c(0, 30), xlab="Proporcion de Vecinos com ICSC",
     main="Vertices com ICSC")
hist(m1.svm.fitted[V(ppi.CC.gc)$ICSC == 0], col="blue",
     ylim=c(0, 30), xlab="Proporcion de Vecinos com ICSC",
     main="Vertices sin ICSC")
```

(9) Figura 11:

```
     #########modelo SIR com grafos
####simulacion de grafos
library(sir)
gl <-list()
gl$ba<- barabasi.game(250, m=5, directed=FALSE)
gl$er<- erdos.renyi.game(250, 1250, type=c("gnm"))
gl$ws<- watts.strogatz.game(1, 100, 12, 0.01)
######tasas de infeccion e recuperacion
beta <- 0.5
gamma <- 1
####  numero de simulaciones
ntrials <- 100
######simulacion de SIR
sim <- lapply(gl, sir, beta=beta, gamma=gamma,
              no.sim=ntrials)
#####porcentajes de infectados
```

```
graphics.off()
par(mfrow=c(3,1))
plot(sim$er, xlab="tiempo", main="Erdos")
plot(sim$ba, color="palegoldenrod",
     median_color="gold", quantile_color="gold",
     xlab="tiempo", main="Barabasi")
plot(sim$ws, color="pink", median_color="red",
     quantile_color="red",xlab="tiempo",
     main="Watts-Strogatz")
```

Referências

[1] J. Besag. Spatial Interaction and the Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society Series B (Methodological)*, 36, 1974.

[2] S. Chatterjee and P. Diaconis. Estimating and understanding exponential random graph models. *Ann. Statist.*, 41(5):2428–2461, 2013.

[3] R. Diestel. *Graph theory*. Graduate Texts in Mathematics. Springer, 3rd edition, 2006.

[4] O. Frank and D. Strauss. Markov Graphs. *Journal of the American Statistical Association*, 81, 09 1986.

[5] P. W. Holland and S. Leinhardt. A Method for Detecting Structure in Sociometric Data. *American Journal of Sociology*, 76, 11 1970.

[6] X. Jiang, E. D. Kolaczyk, N. Nariai, M. Steffen, and S. Kasif. Integration of relational and hierarchical network information for protein function prediction. *BMC Bioinformatics*, 9, 12 2008.

[7] E. D. Kolaczyk. *Statistical Analysis of Network Data: Methods and Models*. Springer Series in Statistics. Springer, 2009.

[8] P. D. L. Saloff-Coste. What Do We Know about the Metropolis Algorithm? *Journal of Computer and System Sciences*, 57, 1998.