VISIBILITY GRAPHS

Prof. Dirceu de Freitas Piedade Melo

COM APLICAÇÕES NO R-STUDIO

INTRODUÇÃO **2**
Por que transformar séries temporais em redes?
Dinâmica da oficina
Pré-requisitos

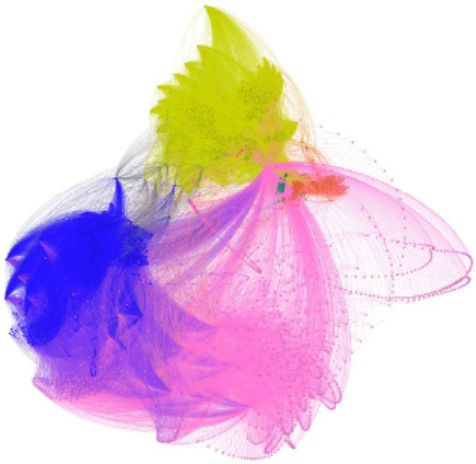
Agradecimentos **4**

Visibility Graphs **5**
Noção geométrica
Definição Matemática

Fluxograma do algoritmo Visibility Graph **8**

Sobre o RStudio **9**
Baixando e instalando programas e pacotes
Teclas de Atalho
Área de Trabalho do RStudio
Diretório de Trabalho
Gerando Séries no R

SIMULAÇÕES NO RStudio **16**
PREPARAÇÃO
EXPERIMENTO 1 - VISIBILIY GRAPHS A PARTIR DE SÉRIES DO R
EXPERIMENTO 2
Experimento 3



INTRODUÇÃO

POR QUE TRANSFORMAR SÉRIES TEMPORAIS EM REDES?

Uma série temporal é um conjunto de informações numéricas, compiladas, registradas ou observadas em domínio do tempo, organizadas cronologicamente. As aplicações das séries temporais estão inseridas em diversas áreas do conhecimento, como medicina (eletroencefalograma), economia (preços diários de ações), meteorologia (precipitação pluviométrica), engenharia química (processos químicos industriais), epidemiologia (ocorrências semanais de dengue), engenharia de som (sinal de áudio).

Nesta oficina nós vamos trabalhar a transformação de séries temporais em visibility graphs usando a modelagem computacional na linguagem R. Apesar do caráter operacional da oficina, vale ressaltar que a importância de conhecer esse tipo de transformação. Podemos destacar basicamente duas:

- permite a aproximação entre a análise de séries temporais e a análise de redes complexas
- acrescenta novas métricas para a análise de séries temporais

Algumas pesquisas já mostram resultados dessa aproximação. Lacasa et al. (2008) mostram que a estrutura de séries periódicas, aleatórias e fractais é preservada nos seus respectivos visibility graphs; Segundo Lacasa et al. (2009) é possível quantificar correlações de longo alcançe e fractalidade em séries temporais através do estudo da distribuição de grau do visibility graph gerado.

Ao final dessa oficina você poderá reproduzir esses experimentos e tentar usar esses modelos de análise com séries temporais de seu interesse.

DINÂMICA DA OFICINA

Primeiramente iremos apresentar noções básicas que permitirão a compreensão do processo de mapeamento, depois mostraremos a idéia computacional. Por último vamos trabalhar com os scripts R dentro do RStudio, explorando séries pseudo-aleatórias, séries de movimento browniano, séries periódicas, séries de fibonacci e séries de sinais de áudio.

Mostraremos como gerar séries temporais no RStúdio e posteriomente transformá-las em visibility graphs. Também mostraremos como importar séries temporais alocadas em um diretório, transformá-las em visibility graphs.

Iremos também mostrar como fazer a visualização das redes usando três tipos de algoritmos, seguido do cálculo e visualização das principais propriedades de redes complexas.

PRÉ-REQUISITOS

Para acompanhar as atividades práticas é necessário:

- instalar o R

- instalar o RStudio

É desejável, mas não obrigatório:

- ter algum conhecimento de séries temporais, grafos e redes complexas
- ter familiaridade com o R ou RStudio

Agradecimentos

Agradeço ao grupo de pesquisa Fuxicos e Boatos em ter me motivado a produzir esse material, a partir da grande motivação adquirida ao receber o convite para a realização da oficina.

De forma igualmente especial gostaria de agradecer ao prof. Cleônidas Tavares de Souza Junior por sua colaboração na revisão e escrita dos scrips em R.

Visibility Graphs

O visibility graph é um grafo gerado a partir de uma série temporal. Lacasa et al. (2008) apresentam um critério de mapeamento que usa uma abordagem geométrica para estabelecer a relação entre dois pontos de uma série temporal e seus dois respectivos vértices no grafo criado.

Como consequência, as redes geradas por visibility graphs herdam a estrutura das suas respectivas séries temporais. Os autores dessa abordagem propõem que, com a transformação de séries em grafos, os métodos clássicos de análise de redes complexas podem ser aplicados para caracterizar séries temporais sob um novo ponto de vista.

NOÇÃO GEOMÉTRICA

A idéia geométrica do mapeamento com visibility graphs pode ser comparada à visualização entre dois observadores localizados no topo de montanhas distintas, onde cada observador é um ponto da série temporal. Se os observadores conseguem se ver, eles irão corresponder a dois pontos interligados na rede. Caso contrário serão dois pontos sem ligação.

A visualização entre os pontos consecutivos pode ser considerada como trivial. Então esses pontos sempre correponderão a vértices interligados por uma aresta (1 e 2, 2 e 3, 3 e 4, etc.). Consequentemente o mapeamento de uma série temporal por visibility graphs sempre vai gerar redes conectadas (grafos com padrão hamiltoniano).

Se pensarmos em observadores sobre os pontos 1 e 2, 1 e 3, 1 e 4, por exemplo, não é difícil notar que existe visualização mútua. Já entre os pontos 1 e 5 não existe visualização, pois ao traçarmos a linha visual entre esses pontos, haverá uma interrupção nessa linha devido a montanha cujo ”pico”é o ponto 4. Se usarmos essa metodologia para todos os elementos da série temporal, teremos pares de pontos que correspondem a pares de vértices do grafo que estão interligados, uma vez que não apresentam interrupção nas suas linhas visuais, e pares de pontos que não irão gerar vértices interconectados por uma aresta. A Figura 1 mostra o visibility graph gerado a partir da série V e a representação, em linha pontilhada, de doispontos da série que não possuem visibilidade, e que, consequentemente, não formam um par de vértices ligados por uma aresta.

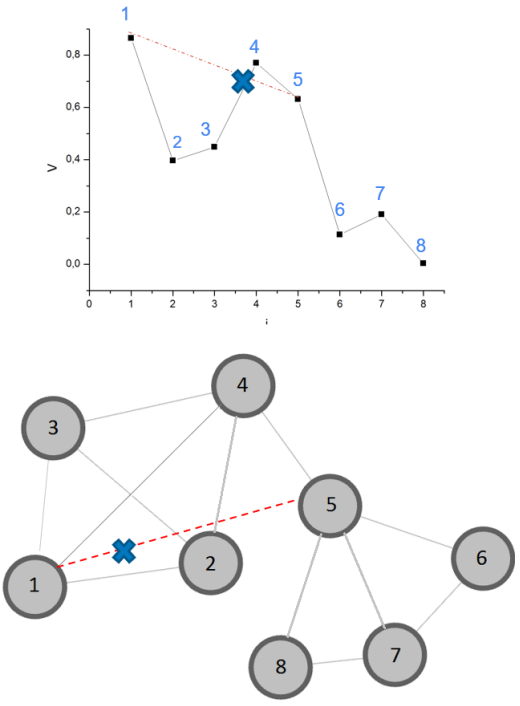


Figura 1: A série V, seu respectivo visibility graph, e a ilustração de dois pontos da série que não geram conexão no grafo

Observando as linhas visuais entre os pontos 5, 6, 7 e 8, podemos notar que não existe, entre eles, outro tipo de visibilidade que não seja aquela que resulta na ligação trivial entre os vértices.

DEFINIÇÃO MATEMÁTICA

Dada uma série temporal $\{V_1, V_2, ..., V_n\}$. O critério de criação de um visibility graph a partir de uma série temporal é o seguinte:

Dois valores artitrários (x_A, V_A) e (x_B, V_B) terão visibilidade, e consequentemente se tornarão dois vértices conectados de um grafo associado, se todos os valores (t_c, V_c) localizados entre eles satisfazem:

$$\frac{V_B - V_A}{x_B - x_A} < \frac{V_B - V_C}{x_B - x_C} \tag{1}$$

A Equação 1 define matematicamente o critério mapeamento dos visibility graphs. Essa equação pode ser vista como uma comparação entre as tangentes das retas AB e BC, onde

$$tgAB = \frac{V_B - V_A}{x_B - x_A} \tag{2}$$

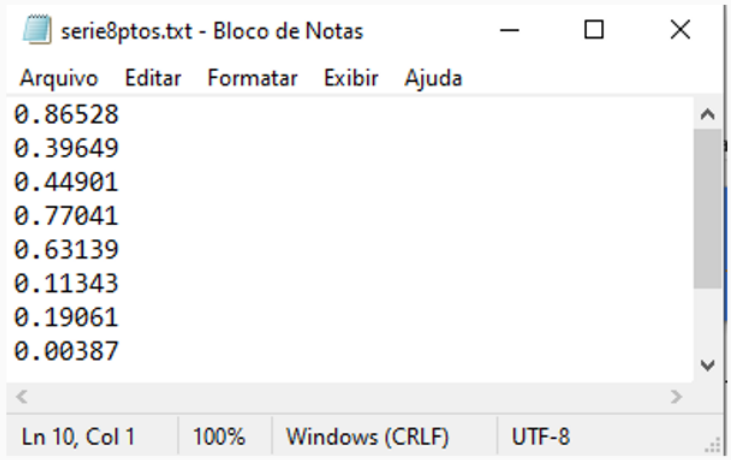
e

$$tgBC = \frac{V_B - V_C}{x_B - x_C}. \tag{3}$$

Toda vez que $tg(AB) < tg(BC_i)$, para todo C_i entre A e B, existe visibilidade entre os pontos A e B da série, e no grafo será criada uma aresta $E(V_a, V_b)$, caso contrário não haverá visibilidade entre A e B na série, tampouco será criada uma aresta no visibility graph.

Exemplo 1

A Figura 2 mostra o arquivo TXT da série temporal apresentada na Figura **??** e sua respectiva tabela com a posição de cada ponto. Vamos avaliar numericamente a visibilidade entre alguns pontos.



x_i	V_i
1	0.86528
2	0.39649
3	0.44901
4	0.77041
5	0.63139
6	0.11343
7	0.19061
8	0.00387

Figura 2: Arquivo da série temporal e sua respectiva tabela posicional

Primeiro vamos avaliar a visibilidade entre os pontos nas posições $x_A = 1$ e $x_B = 4$, tendo como pontos intermediários $x_{C1} = 2$ e $x_{C2} = 3$. Daí calculamos:

$$tgAB = \frac{V_B - V_A}{x_B - x_A} = \frac{0.77041 - 0.86528}{4 - 1} = -0.032 \tag{4}$$

$$tgBC_1 = \frac{V_B - V_{C1}}{x_B - x_{C1}} = \frac{0.77041 - 0.39649}{4 - 2} = 0.18696 \tag{5}$$

$$tgBC_2 = \frac{V_B - V_{C2}}{x_B - x_{C2}} = \frac{0.77041 - 0.44901}{4 - 3} = 0.3214 \tag{6}$$

Como podemos observar $tgAB < tgBC_1$ e $tgAB < tgBC_2$. Isso mostra que os pontos A e B possuem visibilidade mútua e desse modo eles irão corresponder a dois vértices interligados por uma aresta no visibility graph.

Graficamente:

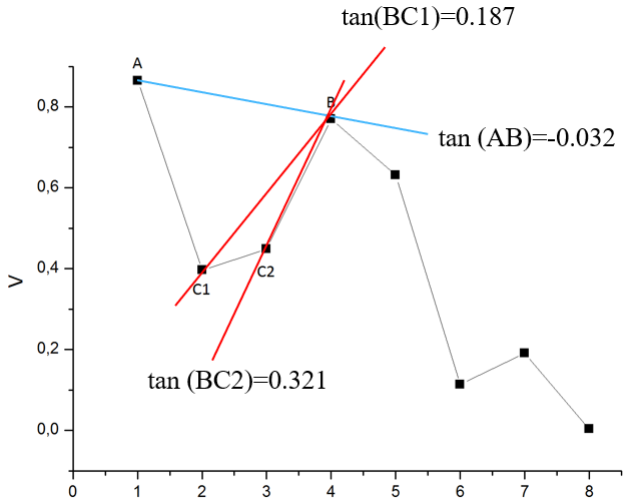


Figura 3: Exemplo de visibilidade entre dois pontos. Retas AB, BC_1, BC_2, BC_3 e suas respectivas tangentes

Exemplo 2

Tomando agora $x_A = 1$ e $x_B = 5$, tendo como pontos intermediários $x_{C1} = 2$, $x_{C2} = 3$ e $x_{C3} = 4$. A Figura 4 mostra as retas formadas por esses pontos e suas respectivas tangentes. Apesar de $tg(AB) < tg(BC1)$ e $tg(AB) < tg(BC_2)$, temos que $tg(AB) > tg(BC_3)$, o que não satisfaz o critério de visibilidade definido na Equação 1. De onde concluímos que os pontos $x_A = 1$ e $x_B = 5$ não irão formar um par de vértices conectados no visibility graph.

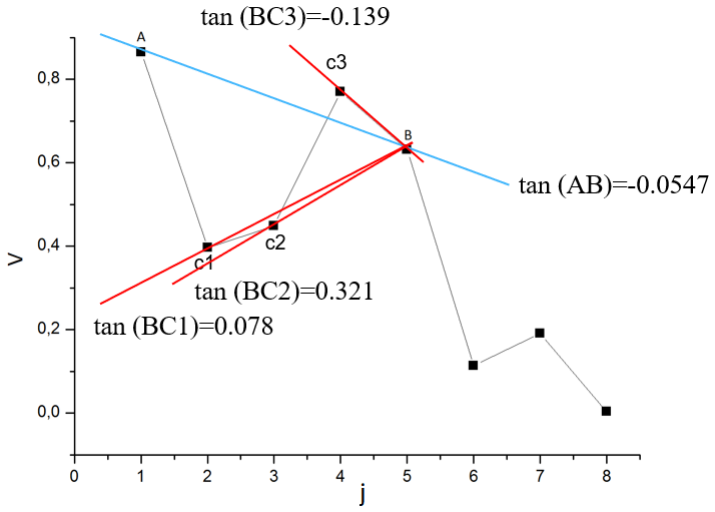


Figura 4: Exemplo de não visibilidade entre dois pontos. Retas AB, BC_1, BC_2, BC_3 e suas respectivas tangentes

Fluxograma do algoritmo Visibility Graph

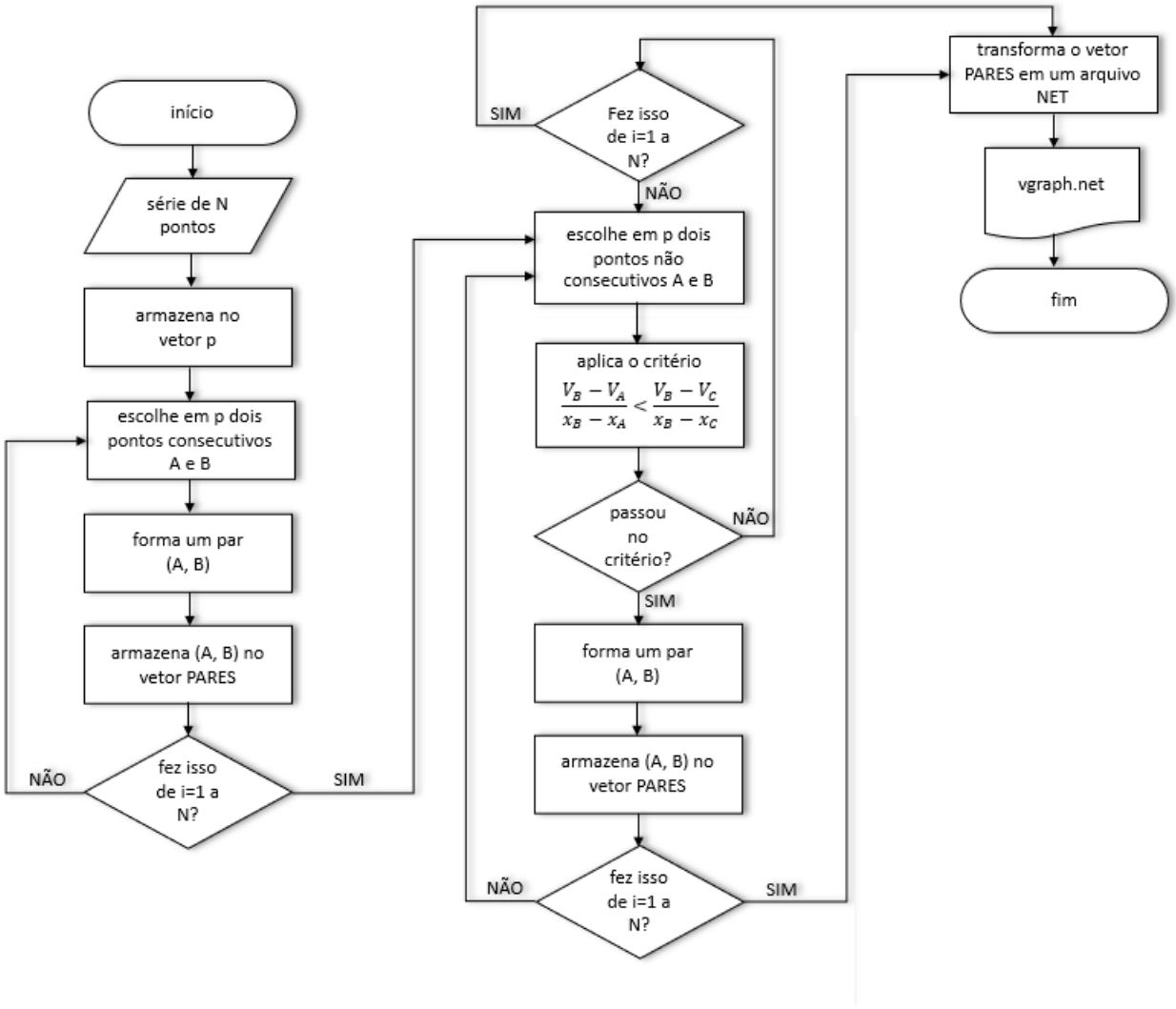


Figura 5: Algoritmo de mapeamento de uma série em um visibility graph

Sobre o RStudio

O R é uma linguagem de programação de código aberto, dedicada à manipulação, análise e visualização de dados. O RStudio é um ambiente de desenvolvimento integrado para a linguagem R. O RStudio foi pensado para trazer recursos facilitadores ao usuário na utilização das potencialidades do R. Maiores detalhes podem ser encontrados em R Core Team (2020).

BAIXANDO E INSTALANDO PROGRAMAS E PACOTES

Antes de instalar o RStudio, faz-se necessário baixar e instalar o R acessando `https://www.r-project.org/`. Nesse link você vai encontrar o repositório CRAN com várias bibliotecas e pacotes. Outro site bastante útil é o `https://www.rdocumentation.org/`, onde se pode acessar uma vasta documentação sobre o R.

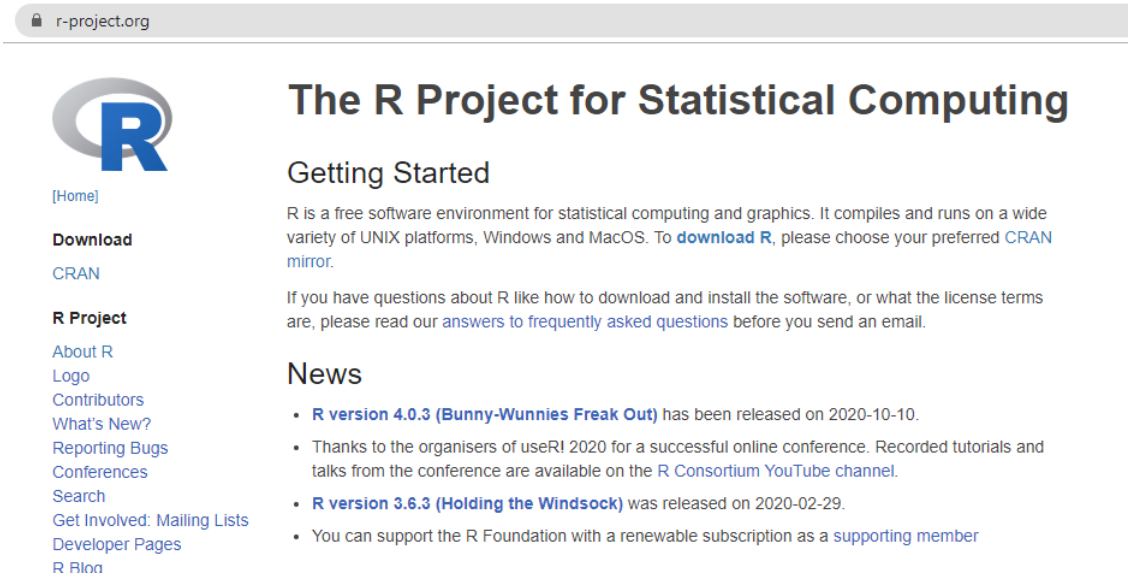


Figura 6: Página web do R Project

O download do RStudio pode ser realizado acessando `https://rstudio.com/products/rstudio/download/`.

A instalação do R e do RStudio é simples: basta clicar no instalador e seguir as instruções.

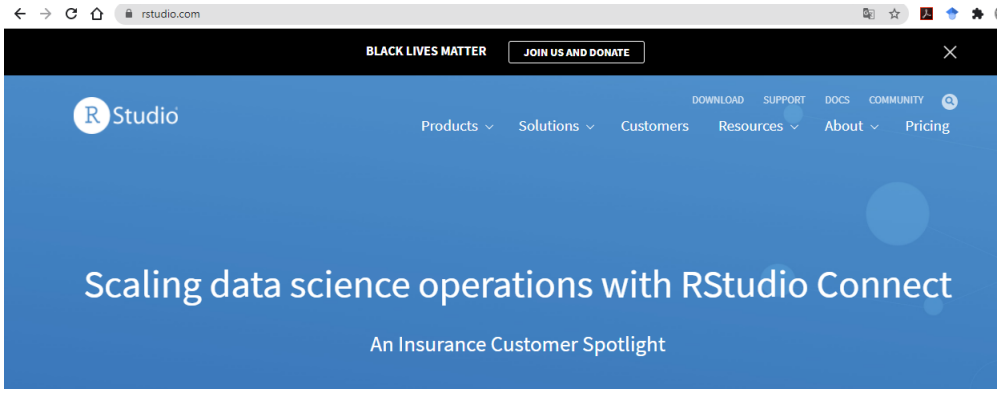


Figura 7: Página web do RStudio

Pacote igraph

Segundo o Csardi and Nepusz (2006):

O *igraph* é uma biblioteca e pacote R para análise de rede. Os principais objetivos da biblioteca IGRAPH é proporcionar um conjunto de tipos de dados e funções para 1) A aplicação livre de dor de algoritmos de gráfico, 2) rápido manuseamento de grandes gráficos, com milhões de vértices e arestas, 3) que permite prototipagem rápida por meio de alta linguagens de nível como R.

Instalando os pacotes para a oficina

Nessa oficina trabalharemos com os pacotes igraph,somebm, cowplot e poweRlaw. A seguir daremos o exemplo da instalação do pacote igraph. depois siga o mesmo procedimento para instalar os outros pacotes.

Vá em ”Painel de Plots e Pacotes”na área de trabalho e na aba ”pacotes”(1) clique em ”install”(2) (Figura 8).

Depois digite ”igraph”na janela ”Packages”e depois clique em ”install”.

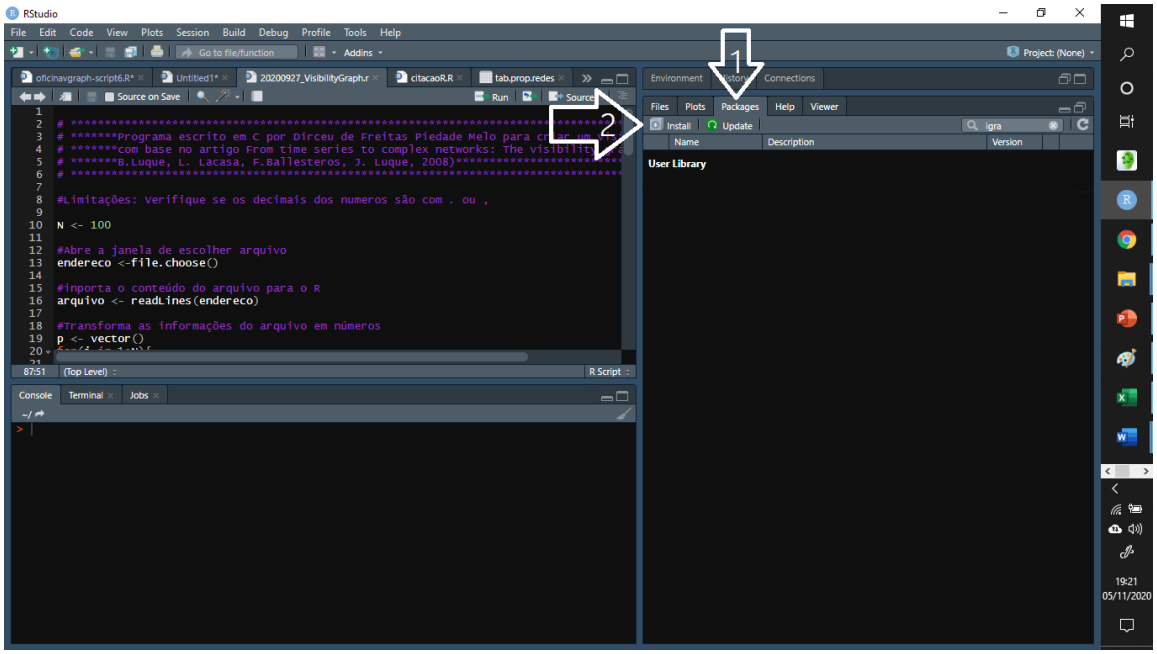


Figura 8: instalação do pacote igraph - passos 1 e 2

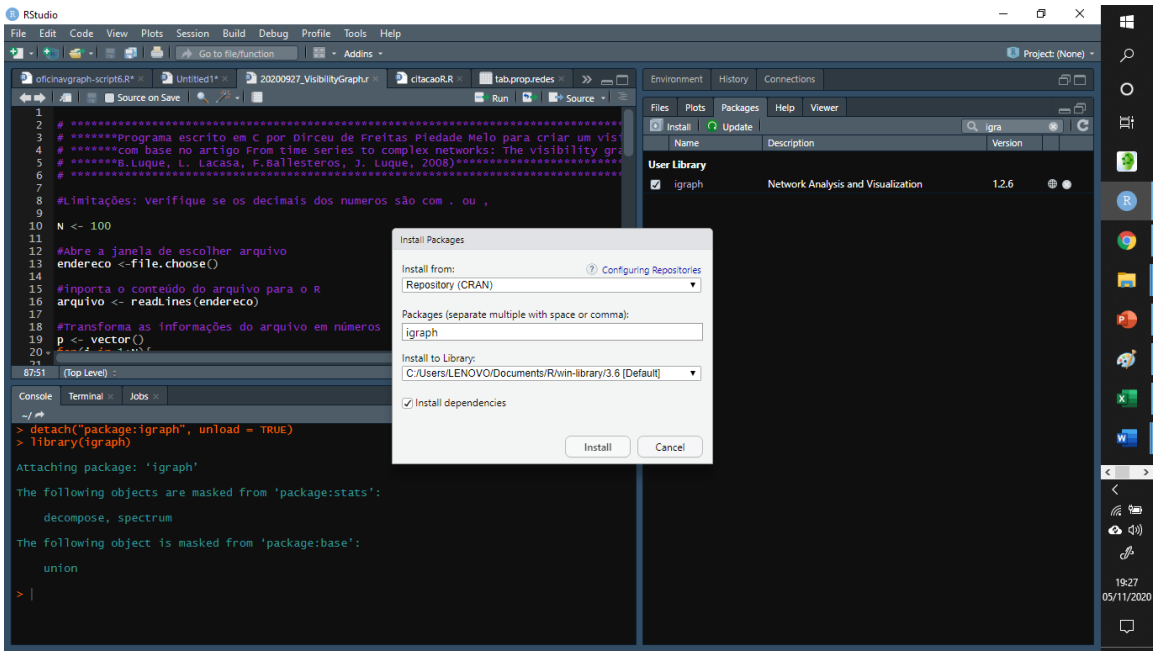


Figura 9: instalação do pacote igraph - passo 3

TECLAS DE ATALHO

Você pode consultar todas as teclas de atalho usando Alt+Shift+k. Uma janela será exibida com os atalhos (Fig. 10).

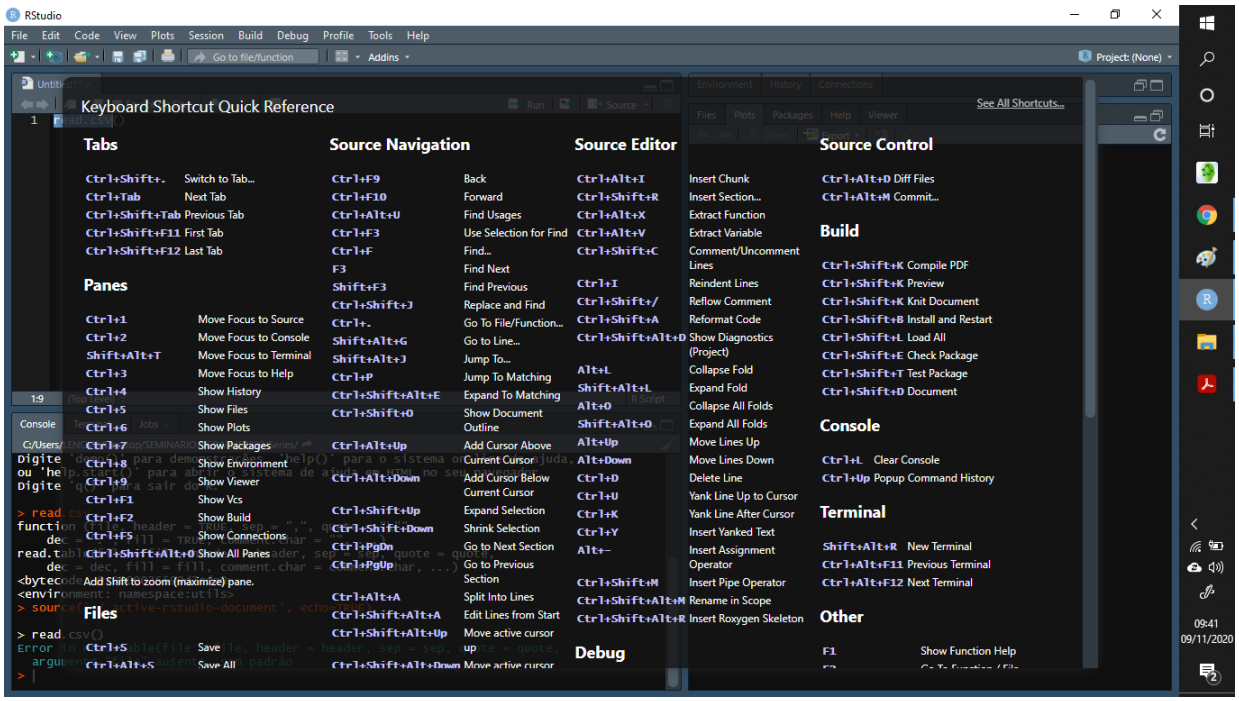


Figura 10: Janela com atalhos do R

A seguir listamos algumas teclas de atalho do RStudio:

- Ctrl+S: salvar
- Ctrl+A: seleciona todo script
- Ctrl+Enter: executa o script selecionado ou executa a linha em que está o prompt
- Ctrl+Shift+Enter: executa o arquivo atual, mostrando comandos na tela;
- Ctrl+Enter: executa a linha selecionada;
- F1: ajuda
- Ctrl+L: limpar console
- Ctrl+6: exibe os plots

ÁREA DE TRABALHO DO RSTUDIO

A área de trabalho do RStudio é formada por quatro campos (Figura 11). Aqui vamos descrever as funcionalidades mais importantes que podem ser acessadas nesses campos:

- Editor de Script - onde se escreve o script do código que será executado.
- Painel do Console R - A aba *Console* apresenta o prompt do R, local onde se visualizam os comandos executados no editor de script. É possivel também executar código diretamente no console, porém, os comandos não serão salvos. Na aba *Terminal* é possível inserir comandos no terminal do windows.
- Painel de Variáveis - Em *Environment* ficam salvas as variáveis executadas no editor no script e o histórico dos comandos executados no RStudio.
- Painel de Plots e Pacotes - local onde se pode acessar os recursos:
  - *Files*: Nessa aba é possível ter acesso aos arquivos e pastas do diretório de trabalho;

- *Plots*: Aqui é possível visualizar gráfico e figuras diversas geradas pelo seu script. Ex: Histogramas, diagramas, grafos.
- *Packages*: Exibe os pacotes instalados e carregados pelo software, também é possível instalar novos pacotes;
- *Help*: Nessa seção é possível adquirir informações sobre comandos, funcões e pacotes;
- *Viewer*: Exibe resultados dinâmicos e interativos.

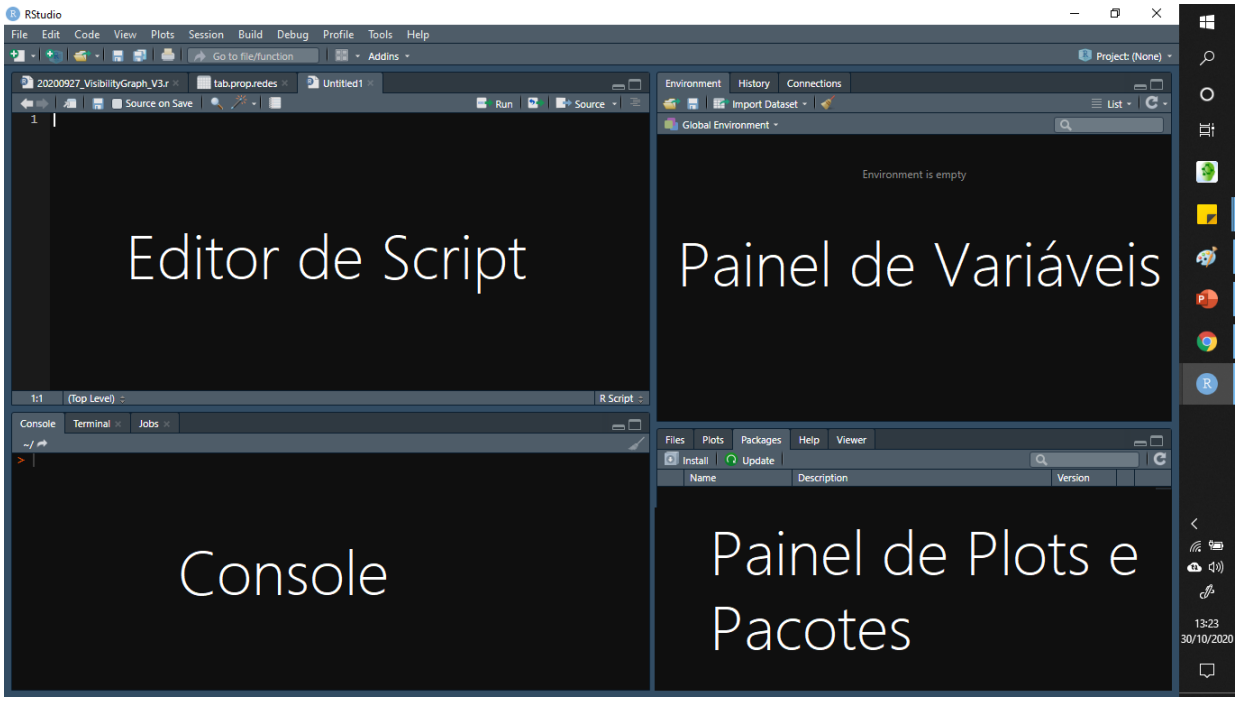


Figura 11: Área de trabalho do RStudio

DIRETÓRIO DE TRABALHO

Uma informação muito útil é saber qual o seu diretório de trabalho. Para saber essa informação use a função $\mathrm{getwd}()$

Três passos para mudar o diretório de trabalho:

PRIMEIRO PASSO: vá no painel de plots e pacotes (Fig.12)

SEGUNDO PASSO: Escolha a sua pasta para ser a área de trabalho

TERCEIRO PASSO: Selecione o novo diretório (Fig.14) (Fig.13).

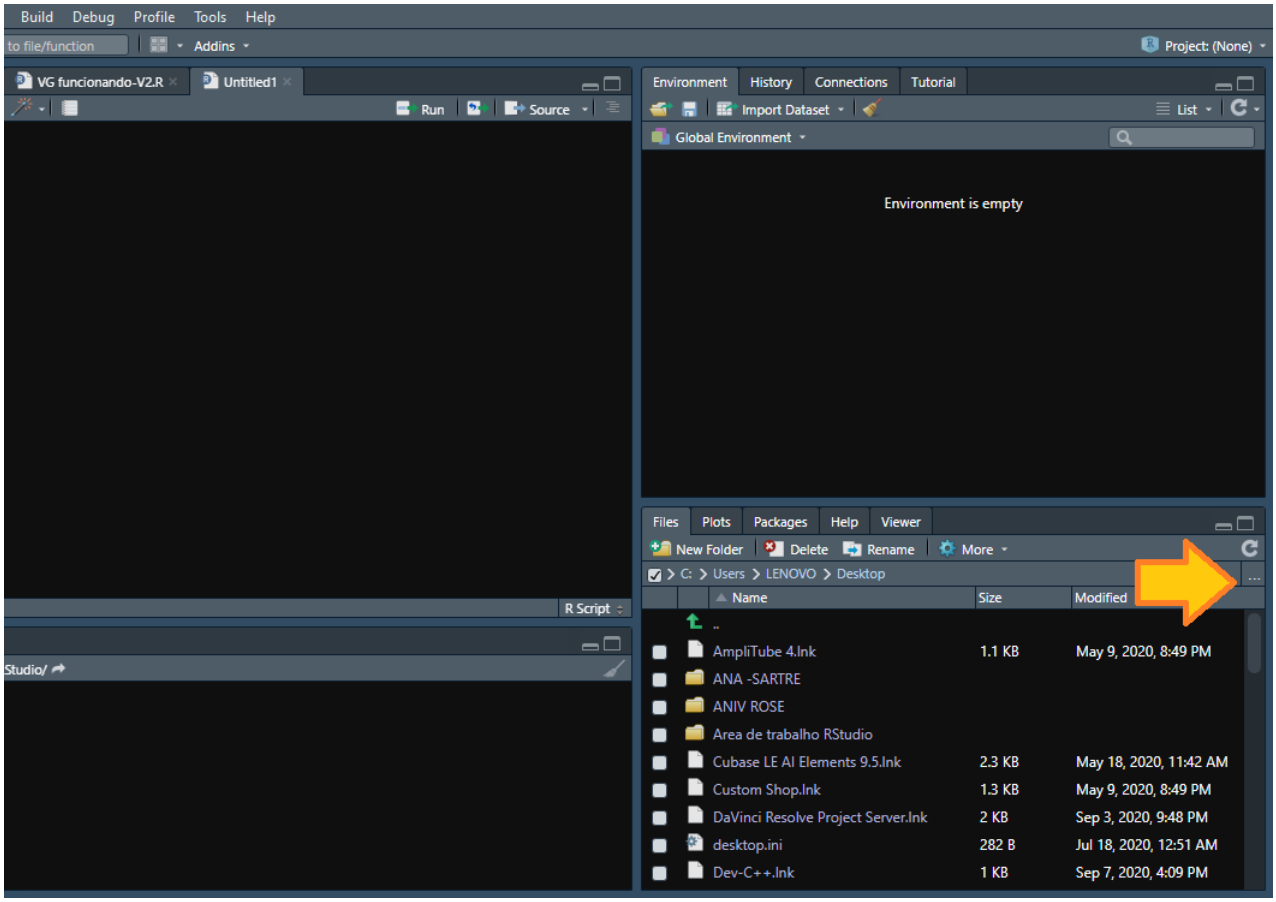


Figura 12: Mudando o diretório: passo 1

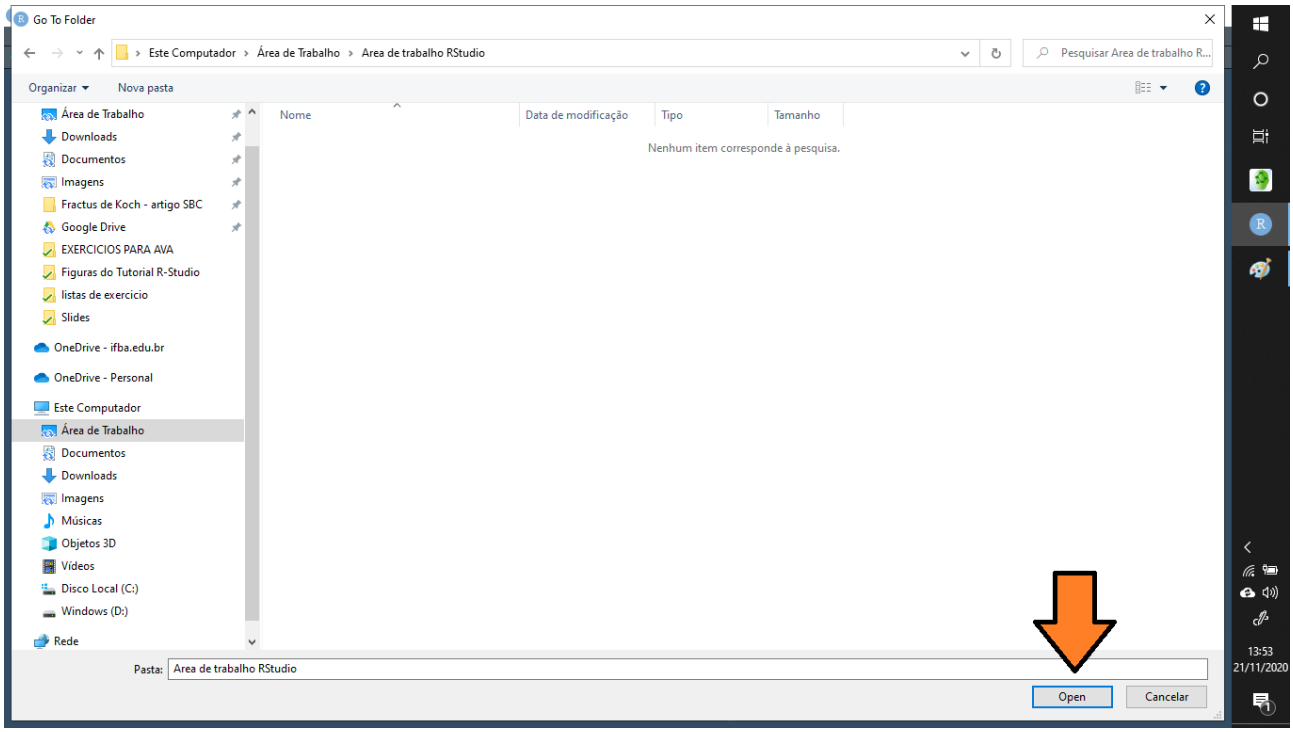


Figura 13: Mudando o diretório: passo 2

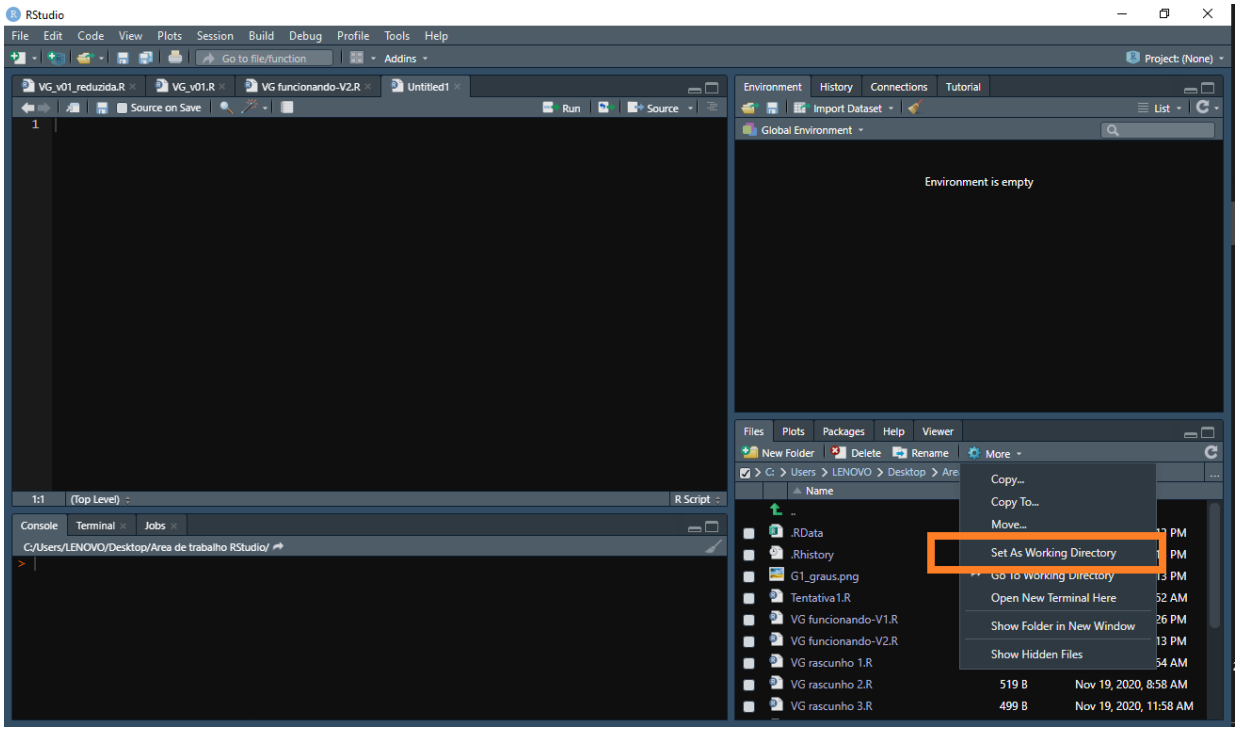


Figura 14: Mudando o diretório: passo 3

Outra maneira é seguir o caminho: Tools >> Global Options >> General >> Basic >> Default working directory e mude o diretório para algum local de sua preferência (Fig:15)

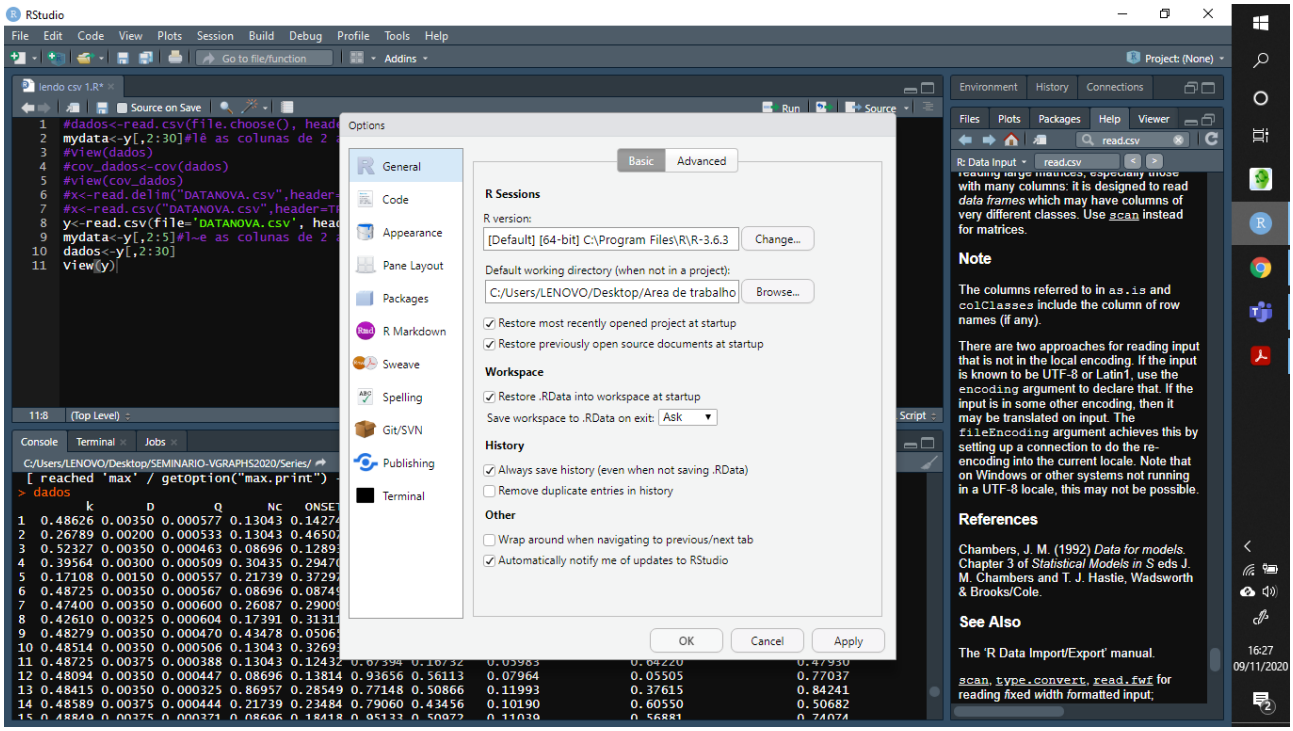


Figura 15: Caption

GERANDO SÉRIES NO R

Em nosssas simulações iremos gerar algumas séries no R e transformá-las em visibility graphs:

- runif(N) - Gera uma série pseudo aleatória de N elementos a partir de uma distribuição uniforme no intervalo $[0, 1]$.
- rnorm(N, media, desv padr) - Gera uma série pseudo aleatória de N elementos a partir de uma distribuição normal no intervalo $[0, 1]$.
- rep(c$(a_1, a_2, \cdots, a_n)$, N) - Cria sequência com N repetições de $a_1, a_2, \cdots, a_n$.
  Ex: rep(c$(5, 11, 2, 1)$,4) repete 5,11,2,1 quatro vezes: 5,11,2,1,5,11,2,1,5,11,2,1,5,11,2,1
- seq(a,b,INC) - Cria uma sequência de a até b incrementando com o valor de INC.
  Ex: seq(3, 4, 0.2) vai resultar em: 3, 3.2, 3.5, 3.8, 4

SIMULAÇÕES NO RStudio

PREPARAÇÃO

- Crie uma pasta na area de trabalho.
- Baixe 3 Scripts R na pasta `https://drive.google.com/drive/folders/1ithaCsWAI7PyXRYBvIoo-8-he4sa8xR5?usp=sharing`
- Baixe as séries na pasta `https://drive.google.com/drive/folders/1fdmKnc-tWGBYQqE7Up3X3tLS_Gb9Al__?usp=sharing`
- Coloque os arquivos que baixou na pasta que você criou na sua área de trabalho.
- Abra o RStudio
- Defina a área de trabalho do RStudio com o endereço da pasta que voce criou e colocou os scripts e as séries.
- Instale os pacotes igraph,somebm, cowplot e poweRlaw

EXPERIMENTO 1 - VISIBILIY GRAPHS A PARTIR DE SÉRIES DO R

Script 1- Algoritmo V-Graphs

Nesse primeiro script iremos:

- criar séries no R usando as funções:
  - runif(N)
  - rnorm(N, media, desv padr)
  - rep$(c(a1, a2, \cdots, a_n)$,N)
  - seq(a,b,INC)
- Observação de algumas métricas
- Gerar séries no R
- Visualização as séries, seus visibility graphs e suas distribuições Calcular algumas métricas

Roteiro 1

1. Observe a função runif(N) na linha 8 do código. Descomente, se necessário.
2. Coloque N=5 (linha 7)
3. Execute o script usando Ctrl+Alt+Enter
4. Observando a janela gráfica compare os gráficos relativos à série com os gráficos relativos ao grafo.
5. veja a tabela gerada com as métricas de rede e estabaleça relações com os gráficos observados.
6. Confira a formação do visibility graph aplicando visualmente o algoritmo na série.
7. Repita o procedimento para N=10, N=100. N=1000

Roteiro 2

1. Observe a função rnorm(N,10,2) na linha 9 do código, e descomente. Agora comente a linha 8.
2. Coloque N=100 (linha 7)
3. Execute o script usando Ctrl+Alt+Enter

4. Faça as observações e comparações
5. Repita o procedimento para N=100 e N=1000

Roteiro 3

1. Observe a função rep(c(1,9,5),N) na linha 10 do código, e descomente. Agora comente a linha 9.
2. Coloque N=1
3. Execute o script usando Ctrl+Alt+Enter
4. Faça as observações e comparações
5. Repita o procedimento para N=2 e N=10

Roteiro 4

1. Observe a função seq(1,N,1) na linha 11 do código, e descomente. Agora comente a linha 10.
2. Coloque N=10
3. Execute o script usando Ctrl+Alt+Enter
4. Faça as observações e comparações
5. Repita o procedimento para N=100

Roteiro 5

1. Observe a série de Fibonacci na linha 14 do código, e descomente. Agora comente a linha 11.
2. Coloque N=11
3. Execute o script usando Ctrl+Alt+Enter
4. Faça as observações e comparações

Agora crie outras séries com o comando da linha 15 ou mudando os parâmetros das funções usadas nos roteitos anteriores.

```
1  #************************** PACOTES ****************************************************
2  library(igraph)
3  library('somebm')
4  library(cowplot)
5  library(poweRlaw)
6  ## SERIES ***************************************************************************
7  N<-11 # informe o numero de elementos
8  #serie<-runif(N) # N numeros aleatorios a.p. dist. unif. em [0,1]
9  #serie<-rnorm(N,10,2) # N numeros aleatorios a.p. dist. norm. em [0,1]
10 #serie<-rep(c(1,9,5),N)# repete c(a,b,c, ... , t), N vezes.
11 #serie<-seq(1,N,1)# sequencia de a ... N com incremento t
12
13 #************************************************************************************
14 serie<-c(1,1,2,3,5,8,13,21,33,54,87) #Fibonacci
15 #serie<-c(7,3,5,1,0,6)#Crie sua propria serie
16 N<-length(serie) #nao comentar
17 #************************************************************************************
18
19 ## VERIFICA CARACTERES INVALIDOS
20 y <- vector()
21 for(i in 1:N){
22   n <-as.numeric(serie[i])
23   if(is.numeric(n) && !(is.na(n))){
24     y <- c(y, as.numeric(serie[i]))
25   }
26 }
27
28 x <- 1:length(y)
29
30 par(pty="s")
31 ## LE A SERIE E APLICA O CRITERIO DE VISIBILIDADE ***************************************
32 tab.pontos <- data.frame()
```

```
tam <- length(x)
for(i in 1:(tam-1)){
  tg.max <- ((y[i+1]-y[i])/(x[i+1]-x[i]))
  tab.pontos <- rbind(tab.pontos, c(x[i],x[i+1]))
  for(k in (i+1):tam){
    tx.var <- ((y[k]-y[i])/(x[k]-x[i]))
    if(tx.var > tg.max){
      tg.max <- tx.var
      tab.pontos <- rbind(tab.pontos, c(x[i],x[k]))
    }
  }
}
colnames(tab.pontos) <- c('x.1','x.2')
#********** VISIBILITY GRAPH GERADO ****************************************************
g <- graph_from_data_frame(tab.pontos, directed = FALSE)
#********** PROPRIEDADES DE REDES ******************************************************
vert <- vcount(g)
arest <- ecount(g)
grau<-degree(g,mode="all")
gr_m<-mean(grau)#grau medio
dens <- edge_density(g, loops = FALSE)#densidade
lc <- cluster_louvain(g)
mod<-modularity(lc)#modularidade
diam<- diameter(g)#diametro
cmm<-average.path.length(g,directed = FALSE)#caminho minimo medio
c.aglom<-transitivity(g, type = "average")#coeficiente de aglomeracao
#*************** Tabela com as propriedades de redes**********************************
tab.prop.redes <- data.frame(vert,arest,gr_m,dens,mod,diam,cmm,c.aglom)
View(tab.prop.redes)
#********************** PLOTS ***********************************************************
#dev.new()
par(mfrow = c(2, 2))## Divide a janela grafica em 2 linhas e 2 colunas
#*****************************************************************************************
## serie
barplot(serie,names.arg =c(1:length(serie)),main="Serie")

## grafo
plot(g,vertex.size=20,main="Visibility Graph")

## histograma da serie
hist(serie, prob=T,main="Distribuicao de Frequencias da Serie")

## distribuicao de graus do grafo ##
d1 = degree.distribution(g, cumulative=FALSE)
len = length(d1)
#png(filename="G1_graus.png", height=500, width=500, bg="white")
barplot(d1, main="Distribuicao de Graus do Grafo",xlab="Graus", ylab="PDF",
        names.arg=c(0:(len-1)))
#*****************************************************************************************
par(mfrow = c(1, 1)) #volta para a janela padrao
```

EXPERIMENTO 2

Script 2 - Estudando do Fractional Brownian Motion, Fractalidade e lei de potência

Com esse script iremos:

- criar séries usando a função fbm(hurst,N).
- visualizar a série, o grafo, o histograma da série e a distribuição de graus do grafo.

Procure observar em cada roteiro a seguir, as mudanças do perfil da série, a variação do índice hurst, e o valor do γ. Tente estabelecer uma relação entre esse valor e a fractalidade aparente da série.

Roteiro 1

1. Observe a função fbm(hurst, N) na linha 8 do código. Descomente, se necessário. Coloque hurst=0.3
2. Insira N=1000 (linha 7)
3. Execute o script usando Ctrl+Alt+Enter
4. Observe os gráficos
5. Veja que na tabela de métricas da rede aparece o valor do expoente γ da lei de potência.
6. Repita o procedimento para hurst=0.5 e 0.8.

Roteiro 2

1. Troque o N para 2000 e repita o procedimento para hurst=0.3, 0.5 e 0.8.

Roteiro 3

1. Troque o N para 5000 e repita o procedimento para hurst=0.3, 0.5 e 0.8.

```
1  #************************* PACOTES ************************************************
2  library(igraph)
3  library('somebm')
4  library(cowplot)
5  library(poweRlaw)
6  ## SERIES ***************************************************************************
7  N<-10000 # informe o numero de elementos
8  serie <- fbm(hurst=0.8, N)
9  #************************************************************************************
10 ## VERIFICA CARACTERES INVALIDOS
11 y <- vector()
12 for(i in 1:N){
13   n <-as.numeric(serie[i])
14   if(is.numeric(n) && !(is.na(n))){
15     y <- c(y, as.numeric(serie[i]))
16   }
17 }
18
19 x <- 1:length(y)
20
21 par(pty="s")
22 #plot(x,y)
23 ## LE A SERIE E APLICA O CRITERIO DE VISIBILIDADE ***************************************
24 tab.pontos <- data.frame()
25 tam <- length(x)
26 for(i in 1:(tam-1)){
27   tg.max <- ((y[i+1]-y[i])/(x[i+1]-x[i]))
28   tab.pontos <- rbind(tab.pontos, c(x[i],x[i+1]))
29   for(k in (i+1):tam){
30     tx.var <- ((y[k]-y[i])/(x[k]-x[i]))
31     if(tx.var > tg.max){
32       tg.max <- tx.var
33       tab.pontos <- rbind(tab.pontos, c(x[i],x[k]))
34     }
35   }
36 }
37 colnames(tab.pontos) <- c('x.1','x.2')
```

```
#********** VISIBILITY GRAPH GERADO ***************************************************************
g <- graph_from_data_frame(tab.pontos, directed = FALSE)
#********** PROPRIEDADES DE REDES *****************************************************************
vert <- vcount(g)
arest <- ecount(g)
grau<-degree(g,mode="all")
gr_m<-mean(grau)
dens<- edge_density(g, loops = FALSE)
lc <- cluster_louvain(g)
mod<-modularity(lc)
diam <- diameter(g)
cmm<-average.path.length(g,directed = FALSE)
c.aglom<-transitivity(g, type = "average")
#************************** Fitting Lei de Potencia ***************************************
fit1 <- fit_power_law(grau+1, 10)
fit2 <- fit_power_law(grau+1, 10, implementation="R.mle")
#fit1$alpha#gama1
gama1<-fit1$alpha
#stats4::coef(fit2)#gama2
gama2<-stats4::coef(fit2)
fit1$logLik
stats4::logLik(fit2)
#*************** Tabela com as propriedades de redes*****************************************
tab.prop.redes <- data.frame(vert,arest,gr_m,dens,mod,diam,cmm,c.aglom,gama1,gama2)
View(tab.prop.redes)
#********************** PLOTS *********************************************************************
#dev.new()
#**************************************************************************************************
## plotagem de quatro graficos na mesma janela
par(mfrow = c(2, 2))## Divide a janela grafica em 2 linha e 2 colunas

## serie
plot.serie<-plot(serie,type="l",main="serie")

## grafo
plot(g, layout=layout_with_kk(g), vertex.label=NA, vertex.size=2, main="Visibility Graph")#layout
    Kamada Kawai
#plot(g, layout=layout_with_fr(g), vertex.label=NA, vertex.size=3,main="Visibility Graph")#layout
    Fruchterman-Reingold

## histograma da serie
hist(serie, prob=T,main="Distribuicao de Frequencias da Serie")
## distribuicao de graus - log-log e semilog ***********************************************
dist.grau <- degree.distribution(g)
d <- 1:max(grau)-1
ind <- grau !=0
plot(d[ind], dist.grau[ind], log="xy", pch=19,xlab=c("Log-Degree"), ylab=c("Log-n"), main="Log-Log
    Degree Distribution")
#plot(d[ind],dist.grau[ind],log="y",pch=19,xlab=c("Degree"), ylab=c("Log-n"), main="SemiLog Degree
    Distribution")
#**************************************************************************************************
par(mfrow = c(1, 1)) #volta para a janela padrao
```

EXPERIMENTO 3

Nesse experimento trabalharemos com as séries que foram baixadas para a pasta de trabalho, conforme vimos no início do tutorial.

Abrindo um arquivo e compilando

Antes de abrir o arquivo, vá no script e defina o número N, ou seja, a quantidade de pontos que você quer na sua série (Figura 16 - seta 1). Depois clique em *Source* (seta 2).

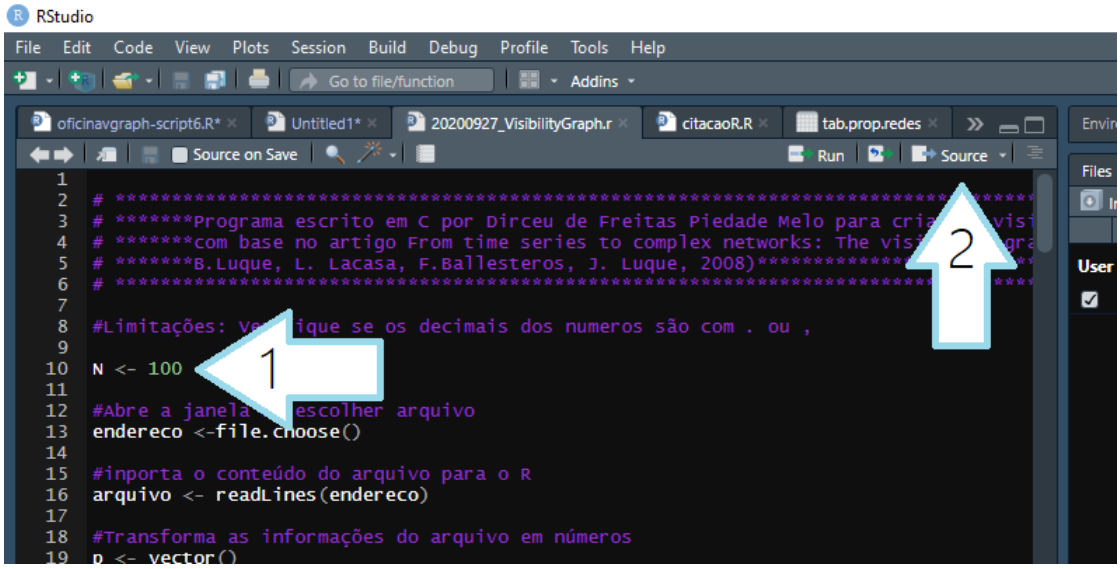


Figura 16: como abrir um arquivo e definir o N

Daí aparece a janela *Selecionar arquivo* (Figura17)

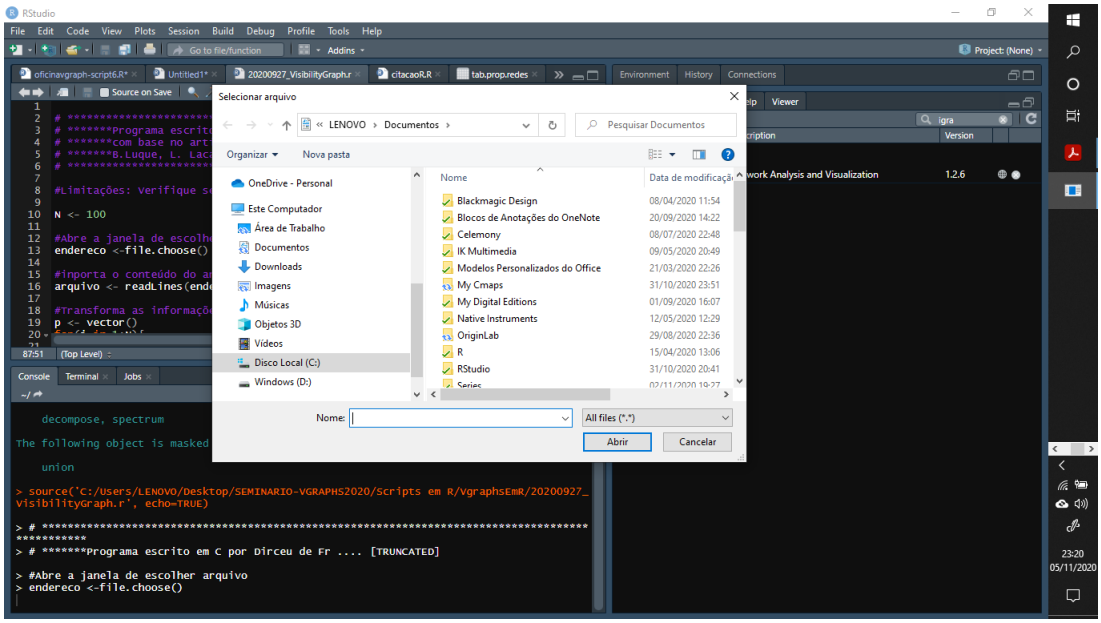


Figura 17: Janela para selecionar um arquivo

Assim que seleciona o arquivo, automaticamente o RStudio compila o programa. Como podemos ver na Figura 18, na saída temos o gráfico da rede (seta 1) , uma tabela com as propriedades de rede (seta2), o registro da criação e gravação da rede em formato .net (seta 3), e o mesmo procedimento para a tabela CSV (seta 4).

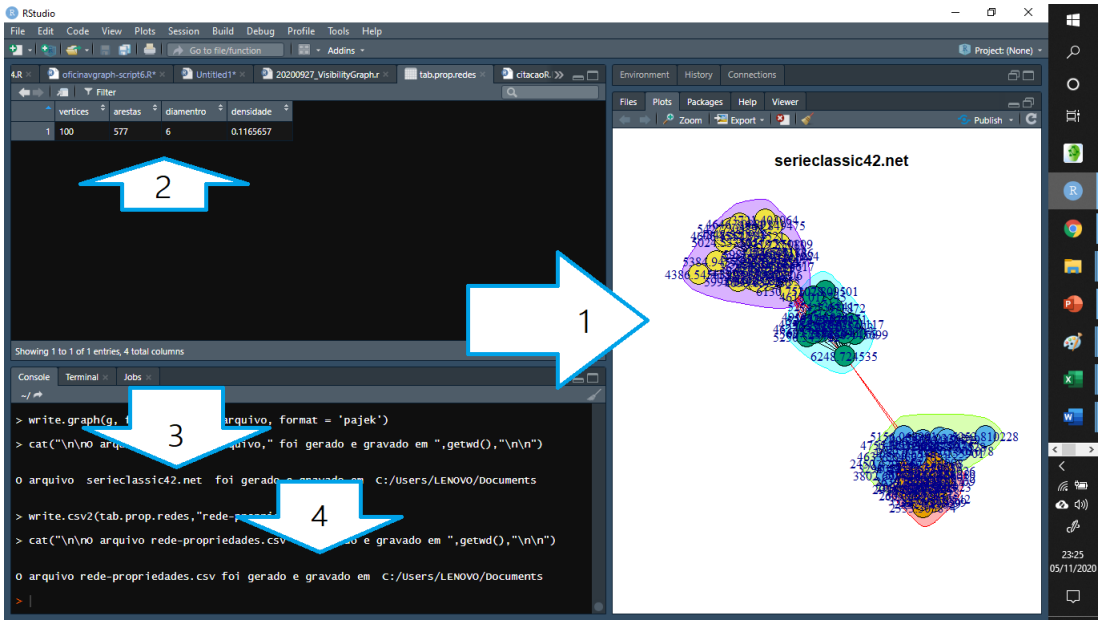


Figura 18: Resultados da compilaçao do programa

Script 4 - importando arquivos, gerando V-Graphs e exportando os resultados

Com esse script iremos:

- importada uma série de um arquivo TXT.
- visualizar a série e a distribuição de graus em um gráfico log-log.
- calcular propriedades e o coeficiente de ajuste da lei de potência da dsitribuição de graus do grafo.

Roteiro

1. Nesse script você não precisa definir inicialmente o N, pois será lido pelo programa.
2. Execute o Ctrl+Alt+Enter
3. Escolha na pasta de trabalho um arquivo com a série que preferir
4. Faça suas observações e modificações com base no que aprendeu nos roteiros anteriores.

```
#************************** PACOTES ************************************************
library(igraph)
library('somebm')
library(cowplot)
library(poweRlaw)
#******************* ENTRADA ARQUIVO EXTERNO ***************************************
endereco <-file.choose()
serie <- readLines(endereco)
N<-length(serie)
#*****************************************************************************************
## Verifica se na serie existem caracteres alfabeticos ou valores invalidos
y <- vector()
for(i in 1:N){
  n <-as.numeric(serie[i])
  if(is.numeric(n) && !(is.na(n))){
    y <- c(y, as.numeric(serie[i]))
  }
}

x <- 1:length(y)

par(pty="s")
#plot(x,y) #nao descomentar
## LE A SERIE E APLICA O CRITERIO DE VISIBILIDADE ***********************************
tab.pontos <- data.frame()
```

```
tam <- length(x)
for(i in 1:(tam-1)){
  tg.max <- ((y[i+1]-y[i])/(x[i+1]-x[i]))
  tab.pontos <- rbind(tab.pontos, c(x[i],x[i+1]))
  for(k in (i+1):tam){
    tx.var <- ((y[k]-y[i])/(x[k]-x[i]))
    if(tx.var > tg.max){
      tg.max <- tx.var
      tab.pontos <- rbind(tab.pontos, c(x[i],x[k]))
    }
  }
}
colnames(tab.pontos) <- c('x.1','x.2')
#********** VISIBILITY GRAPH GERADO ************************************************
g <- graph_from_data_frame(tab.pontos, directed = FALSE)
#********** PROPRIEDADES DE REDES **************************************************
vert <- vcount(g)
arest <- ecount(g)
grau<-degree(g,mode="all")
gr_m<-mean(grau)
dens<- edge_density(g, loops = FALSE)
lc <- cluster_louvain(g)
mod<-modularity(lc)
diam <- diameter(g)
cmm<-average.path.length(g,directed = FALSE)
c.aglom<-transitivity(g, type = "average")
#**************************Fitting Lei de potencia*********************************
fit1 <- fit_power_law(grau+1, 10)
fit2 <- fit_power_law(grau+1, 10, implementation="R.mle")
#fit1$alpha#gama1
gama1<-fit1$alpha
#stats4::coef(fit2)#gama2
gama2<-stats4::coef(fit2)
fit1$logLik
stats4::logLik(fit2)
#*************** Tabela com as propriedades de redes*********************************
tab.prop.redes <- data.frame(vert,arest,gr_m,dens,mod,diam,cmm,c.aglom,gama1,gama2)
View(tab.prop.redes)
#********************** PLOTS ***************************************************
#dev.new()
## plotagem de quatro graficos na mesma janela
par(mfrow = c(2, 2))## Divide a janela grafica em 2 linhas e 2 colunas

## serie
plot.serie<-plot(serie,type="l",main="serie")

## grafo
#plot(g, layout=layout_with_kk(g), vertex.label=NA, vertex.size=2, main="Visibility Graph")#layout
    Kamada Kawai
#plot(g, layout=layout_with_fr(g), vertex.label=NA, vertex.size=3,main="Visibility Graph")#layout
    Fruchterman-Reingold

## histograma da serie
#hist(serie, prob=T,main="Distribuicao de Frequencias da Serie")
#hist(serie)

## distribuicao de graus - log-log e semilog
dist.grau <- degree.distribution(g)
d <- 1:max(grau)-1
ind <- grau !=0
plot(d[ind], dist.grau[ind], log="xy", pch=19,xlab=c("Log-Degree"), ylab=c("Log-n"), main="Log-Log
    Degree Distribution")
#plot(d[ind],dist.grau[ind],log="y",pch=19,xlab=c("Degree"), ylab=c("Log-n"), main="SemiLog Degree
    Distribution")
#*************************************************************************************
par(mfrow = c(1, 1)) #volta para a janela padrao
```

Referências

G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL `https://igraph.org`.

L. Lacasa, B. Luque, F. Ballesteros, J. Luque, and J. C. Nuno. From time series to complex networks: the visibility graph. *Proceedings of the National Academy of Sciences*, 105(13):4972–4975, 2008.

L. Lacasa, B. Luque, J. Luque, and J. C. Nuno. The visibility graph: a new method for estimating the hurst exponent of fractional brownian motion. *EPL (Europhysics Letters)*, 86(3):30001, 2009.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020. URL `https://www.R-project.org/`.