Econometria Espacial com o R

Raphael F. Saldanha, Eduardo Almeida

Novembro de 2016

Script: `https://git.io/vXAsc`

1 Introdução ao R

R é uma linguagem de programação voltada para análises estatísticas, atualmente mantida pela *R Foundation for Statistical Computing*. Sua origem deriva da linguagem *S* e começou a ser desenvolvida em 1992.

Seu código fonte é aberto e pode ser instalado sem custos em diversos sistemas operacionais, incluindo Windows, Mac e Linux. Além das funções básicas, suas possibilidades de uso são expandidas através da utilização de pacotes, que também são gratuitos. Através dos pacotes, que podem ser criados por qualquer usuário, a linguagem *R* vem ganhando grande espaço na área acadêmica, acompanhando rapidamente o desenvolvimento de diversas áreas do conhecimento.

1.1 Instalando o R

Uma copia do *R* pode ser obtida gratuitamente no site `https://cran.r-project.org/`. Atente para o sistema operacional e arquitetura (32 ou 64 bytes)antes de efetuar o download.

1.2 RStudio

Após a instalação do *R*, ele já pode ser usado através de sua interface gráfica padrão. Contudo, é comum a utilização de interfaces de desenvolvimento (IDE - *Integrated Development Environment*) criada por terceiros.

Neste material, iremos utilizar a IDE chamada *RStudio*. Ela pode ser obtida gratuitamente no site `https://www.rstudio.com/products/RStudio/`.

Após instalar e abrir o RStudio, você verá na tela duas áreas principais. A primeira é chamada de *script* e a segunda de *console*.

A área de *script* é onde digitamos os comandos utilizados no *R*. Após preparar um comando, ele é executado pressionando o botão *run* na interface ou pressionando *Ctrl + r* no teclado. Ao executar um comando, ele será reproduzido na área do console e seu resultado virá logo abaixo.

O *script* é um arquivo de texto que pode ser salvo e reaproveitado posteriormente. É comum encontrarmos em *scripts* do *R* linhas precedidas com o caractere #. Este simbolo indica um comentário, ou seja, um texto que não será interpretado como um comando no *R*.

1.3 Comandos básicos no R

Os primeiros comandos no R são funções muito básicas para operações matemáticas. Teste os comandos abaixo:

```
# Soma
2+2

# Subtração
4-2

# Multiplicação
2*3

# Divisão
4/2

# Exponenciação
2^3
```

1.4 Funções

Funções são comandos que apresentam resultados com base em um ou mais argumentos. Teste os exemplos abaixo:

```
# Raiz quadrada
sqrt(x=9)

# Combinação de 10 elementos tomados 2 a 2
choose(n=10, k=2)
```

Ao decorrer deste material, iremos utilizar diversas funções. Para saber mais informações sobre uma função, digite por exemplo:

```
?choose
```

Para descobrir funções através de uma palavra chave:

```
??"ceiling"
```

1.5 Objetos

Os resultados de operações simples e funções no R podem ser armazenados na memória para utilização posterior através de objetos.

Na linguagem R, objetos são abstrações na memória que podem conter desde um simples número até complexos bancos de dados.

Estude os comandos abaixo:

```
# Criar o objeto x
x <- 2

# Imprimir o objeto x
x

# Operações com o objeto x
x^2
x+x

# Apagar o objeto x
rm(x)

# Criar o vetor y
y <- c(2, 5, 10, 11.3, 12)

# Imprimir o vetor y
y

# Elevar o vetor y ao quadrado
y^2

# Apagar o vetor y
rm(y)

# Criar o vetor 'notas'
notas <- c(60, 50, 20, 50, 90)
```

```
# Ordenar o vetor 'notas'
notas <- sort(notas)

# Imprimir o vetor 'notas'
notas
```

1.6 Banco de dados

A interface do *R* não é recomendada para a criação de banco de dados. Como já existem centenas de opções de softwares de qualidade e gratuitos para este fim, os bancos de dados podem ser criados em outros softwares como o Excel e importados para o *R*.

Para ler um arquivo do Excel, a opção mais recomendada é salvar a planilha no formato CSV. Para importar este arquivo CSV no *R*, teste o seguinte:

```
# Importar o arquivo 'notas.csv'
dados <- read.csv2(file.choose(), header = TRUE)
```

Atenção para os nomes das variáveis! Evite a utilização de nomes muito longos, caracteres especiais e acentos.

Após importar um banco de dados, teste os seguintes comandos:

```
# Estrutura do objeto
str(dados)

# Listar os nomes das variáveis
names(dados)

# Imprimir as seis primeiras linhas
head(dados)

# Imprimir as seis últimas linhas
tail(dados)

# Ver todo o banco de dados
View(dados)
```

Para acessar uma variável específica, use:

```
dados$nota
```

1.7 Estatísticas básicas

Teste os comandos abaixo:

```
# Média
mean(dados$nota)

# Mediana
median(dados$nota)

# Variância
var(dados$nota)

# Desvio padrão
sd(dados$nota)

# Valor máximo
max(dados$nota)

# Valor mínimo
min(dados$nota)

# Amplitude
range(dados$nota)

# Coeficiente de variação
sd(dados$nota)/mean(dados$nota)

# Quartis
quantile(dados$nota)

# Sumário
summary(dados$nota)

# Histograma
hist(dados$nota)

# Boxplot
boxplot(dados$nota)
```

2 Dados espaciais no R

2.1 Leitura do *Shapefile*

Para a leitura de arquivos *shapefile* no *R*, precisamos usar alguns pacotes. Após a sua instalação, use:

```
# Pacotes
library(maptools)
library(rgeos)

# Abra o arquivo 'gm10.shp'
gm10.shp <- readShapePoly(file.choose(), IDvar = "CODMUN6")

# Plotar o mapa
plot(gm10.shp)
```

2.2 Matrizes de vizinhos espaciais

Teste o seguinte:

```
# Pacotes
library(spdep)

# Matriz queen
w1 <- nb2listw(poly2nb(gm10.shp, queen = TRUE))
summary(w1)

# Matrix queen padronizada na linha
w1.w <- nb2listw(poly2nb(gm10.shp, queen=TRUE), style="W")
summary(w1.w)

# Matriz rook
w2 <- nb2listw(poly2nb(gm10.shp, queen = FALSE))
summary(w2)

# Matriz rook padronizada globalmente
w2.c <- nb2listw(poly2nb(gm10.shp, queen = FALSE), style = "C")
summary(w2.c)

# Distância inversa
coords <- coordinates(gm10.shp)
```

```
nb <- dnearneigh(coords, 0, 1000)
dlist <- nbdists(nb, coords)
dlist <- lapply(dlist, function(x) 1/x)
w3 <- nb2listw(nb, glist=dlist)
summary(w3)

# Distância inversa padronizada pelo número de vizinhos
w3.u <- nb2listw(nb, glist=dlist, style="U")
summary(w3.u)
```

Para ver mais opções: `?nb2listw`

Matriz de k vizinhos espaciais

A escolha do número ideal de k vizinhos será realizada testando-se vários k e utilizando-se o que retornou o maior valor para a estatística I de Moran significativo.

```
# Número de permutações
per <- 999

# Número máximo de k vizinhos testados
kv <- 20

# Nome dos registros
IDs <- row.names(gm10.shp@data)

# Criação da tabela que irá receber a estatística I de Moran e
   significância para cada k número de vizinhos testado
res.pesos <- data.frame()

# Início do loop
for(k in 1:kv)
{
  # Armazenando número k atual
  res.pesos[k,1] <- k
  # Calculando o I e significância para o k atual
  moran.k <- moran.mc(gm10.shp@data$TCVPA10,
                      listw=nb2listw(knn2nb(
                      knearneigh(coords, k=k),
                      row.names=IDs),style="B"),
                      nsim=per)
```

```
  # Armazenando o valor I para o k atual
  res.pesos[k,2] <- moran.k$statistic
  # Armazenando o p-value para o k atual
  res.pesos[k,3] <- moran.k$p.value
}

# Ver a tabela de k vizinhos, I de Moran e significância
res.pesos

# Sendo todos significativos, iremos usar o k que retornou o
   maior valor I
maxi <- which.max(res.pesos[,2])

# Criação da matriz usando o k escolhido
w5 <- nb2listw(knn2nb(knearneigh(coords, k=maxi),row.names=IDs)
   ,style="B")
```

Importando um arquivo GAL do Geoda

```
w6 <- nb2listw(read.gal(file.choose(), override.id = TRUE))
```

3 Modelagem espacial

3.1 Modelo não espacial

```
esp <- TCVPA10 ~ GM + POLMPC09 + RICPOB10 + RENDA10 + DESOCU10
   + CHEFA10 + DPOP10
mod1 <- lm(formula = esp, data = gm10.shp@data)
summary(mod1)
```

3.2 Testes dos resíduos

Moran's I

```
mod1.moran <- lm.morantest(model = mod1, listw = w5)
mod1.moran
```

Multiplicador de Lagrange

```
mod1.lagrange <- lm.LMtests(model = mod1, listw = w5,
                            test = c("LMerr","RLMerr","LMlag","
                               RLMlag",
                            "SARMA"))
mod1.lagrange
```

3.3 SAR

```
mod1.sar <- lagsarlm(formula = esp, data = gm10.shp@data, listw
    = w5)
summary(mod1.sar)
impacts(mod1.sar, listw = w5)
```

3.4 CAR

```
mod1.car <- spautolm(formula = esp, data = gm10.shp@data, listw
    = w5, family = "CAR")
summary(mod1.car)
```

3.5 SEM

```
mod1.sem <- errorsarlm(formula = esp, data = gm10.shp@data,
   listw = w5)
summary(mod1.sem)
```

3.6 SEM por GMM

```
mod1.semGMM <- GMerrorsar(formula = esp, data = gm10.shp@data,
   listw = w5)
summary(mod1.semGMM)
```

3.7 SAC

```
mod1.sac <- sacsarlm(formula = esp, data = gm10.shp@data, listw
    = w5)
summary(mod1.sac)
impacts(mod1.sac, listw = w5)
```

3.8 SAC por GMM

```
mod1.sacGMM <- gstsls(formula = esp, data = gm10.shp@data,
   listw = w5)
summary(mod1.sacGMM)
impacts(mod1.sacGMM, listw = w5)
```

3.9 SDM

```
mod1.sdm <- lagsarlm(formula = esp, data = gm10.shp@data, listw
    = w5, type = "mixed")
summary(mod1.sdm)
impacts(mod1.sdm, listw = w5)
```

3.10 SDEM

```
mod1.sdem <- errorsarlm(formula = esp, data = gm10.shp@data,
   listw = w5, etype = "emixed")
summary(mod1.sdem)
```

3.11 SDEM

```
mod1.sma <- spautolm(formula = esp, data = gm10.shp@data, listw
    = w5, family = "SMA")
summary(mod1.sma)
```