Econometria Espacial aplicada com R

Raphael Saldanha; Eduardo Almeida

2023-01-01

Índice

Apresentação

Este livro *on-line* apresenta uma introdução a Econometria Espacial aplicada com R. Recomendamos como referência o livro *Econometria Espacial aplicada* (Almeida 2012).

Raphael Saldanha é geógrafo, mestre em saúde pública, doutor em Informação e Comunicação em Saúde, *postdoc researcher* no Inria (*Institut national de recherche en sciences et technologies du numérique, France*).

Eduardo Almeida é economista, mestre e doutor em Economia, postdoc pela Escola Superior de Agricultura "Luiz de Queiroz", professor adjunto da Faculdade de Economia da Universidade Federal de Juiz de Fora (UFJF) desde 2005.

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 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 uma 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*.

1.2.1 Que documento é este?

O **script** é a forma mais tradicional de criar códigos para o *R*, mas ele tem algumas limitações. Por ser apenas um arquivo de texto simples, ele não é capaz de armazenar os resultados produzidos pelo *R*, guardar figuras e ser esteticamente apresentável.

Por estes motivos e na onda da ciência reprodutível, foi criado o **R Notebook**. Um notebook é uma uma espécie de script versão 2.0, onde é possível escrever comentários com formatação, inserir código do *R* e ver o seu resultado logo abaixo. E a melhor parte: um notebook pode ser exportado para HTML ou PDF facilmente, facilitando o envio de pesquisas feitas com o R.

Este arquivo que você está visualizando é um R Notebook. Os asteríscos e alguns símbolos que você está vendo no texto é uma forma de formatação chamada *Markdown*. Por exemplo, dois asteríscos entre uma palavra fazem com que ela fique em *negrito*. Neste link tem um guia desta **linguagem**.

Para ver como está ficando o seu notebook, clique no botão **Preview**. Uma página irá aparecer com a pré-visualização do seu notebook. Para exportar para HTML ou PDF, use a seta para abaixo ao lado do botão (*Knit to…*).

2 Comandos básicos

2.1 Operações matemáticas

Os primeiros comandos no R são funções muito básicas para operações matemáticas. A parte verde abaixo é um *chunk* de código R no notebook. Isto indica ao RStudio que esta parte do texto deve ser intepretada como código R e não como texto. Para executar um chunk, pressione Ctrl + Shift + Enter.

```
# Soma
2+2
```

```
[1] 4
```

```
# Subtração
4-2
```

```
[1] 2
```

```
# Multiplicação
2*3
```

```
[1] 6
```

```
# Divisão
4/2
```

```
[1] 2
```

```
# Exponenciação
2^3
```

```
[1] 8
```

Se tudo correu bem, você deve ver acima o resultado das operações. Use o preview do notebook para ver como está ficando o resultado do documento.

2.1.1 Sua vez

Realize a seguinte operação no R $((2 + 3(5))/10)^2$.

```
((2+3*5)/10)^2
```

```
[1] 2.89
```

Resultado correto: 2,89.

2.2 Funções

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

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

```
[1] 3
```

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

```
[1] 45
```

```
# e^x
exp(2)
```

```
[1] 7.389056
```

```r
# Logarítmo natural
log(1)
```

```
[1] 0
```

2.2.1 Sua vez

Calcule a raiz quadrada do logaritmo de 10 no R.

```r
sqrt(log(x = 10))
```

```
[1] 1.517427
```

Resultado correto: 1,51.

2.3 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.

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

# Imprimir o objeto x
x
```

```
[1] 2
```

```r
# Operações com o objeto x
x^2
```

```
[1] 4
```

```r
x+x
```

```
[1] 4
```

```r
# Apagar o objeto x
rm(x)

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

# Imprimir o vetor y
y
```

```
[1]  2.0  5.0 10.0 11.3 12.0
```

```r
# Elevar o vetor y ao quadrado
y^2
```

```
[1]   4.00  25.00 100.00 127.69 144.00
```

```r
# 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] 20 50 50 60 90
```

2.3.1 Sua vez

Crie um objeto chamado **dados** com os valores {5, 7, 12.7, 13, 15}. Eleve ao quadrado e obtenha o resultado da soma dos valores elevados com a função `sum(x)`.

```r
dados <- c(5, 7, 12.7, 13, 15)
sum(dados^2)
```

```
[1] 629.29
```

Resultado correto: 629,29.

2.4 Operadores lógicos

Estes operados verificam afirmações de lógica, veja os comandos abaixo.

```r
# Igualdade
2 == 2
```

```
[1] TRUE
```

```r
2 == 5
```

```
[1] FALSE
```

```r
"a" == "a"
```

```
[1] TRUE
```

```r
"a" == 'A'
```

```
[1] FALSE
```

```r
# Maior
2 > 3
```

```
[1] FALSE
```

```r
2 >= 3
```

```
[1] FALSE
```

```r
# Menor
2 < 4
```

```
[1] TRUE
```

```r
2 <= 1
```

```
[1] FALSE
```

```r
c(1,2,2,4,7,10) < 7
```

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

3 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 posteriormente para o R.

Existem diversos pacotes para ler aquivos de banco de dados no R, como DBF, SAV, Stata e etc. Mas a forma mais segura e recomendada é exportar os dados no formato CSV para ser importados no R.

3.1 Arquivos CSV

Para importar este arquivo CSV no R, teste o código abaixo. Se não funcionar, tente trocar "/" por "\".

```
# Importar o arquivo 'notas.csv'
dados <- read.csv2(file = "data/notas.csv", 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, verifique se tudo correu bem com estes comandos.

```
# Estrutura do objeto
str(dados)
```

```
'data.frame':   6 obs. of  3 variables:
 $ id  : int  1 2 3 4 5 6
 $ nome: chr  "Maria" "João" "José" "Maria" ...
 $ nota: num  10 20 32.6 15 40.9 ...
```

```
# Listar os nomes das variáveis
names(dados)
```

```
[1] "id"   "nome" "nota"
```

```r
# Imprimir as seis primeiras linhas
head(dados)
```

```
  id    nome  nota
1  1   Maria 10.00
2  2    João 20.00
3  3    José 32.60
4  4   Maria 15.00
5  5 Gustavo 40.90
6  6 Alfredo 32.86
```

```r
# Imprimir as seis últimas linhas
tail(dados)
```

```
  id    nome  nota
1  1   Maria 10.00
2  2    João 20.00
3  3    José 32.60
4  4   Maria 15.00
5  5 Gustavo 40.90
6  6 Alfredo 32.86
```

Para acessar uma variável específica:

```r
dados$nota
```

```
[1] 10.00 20.00 32.60 15.00 40.90 32.86
```

3.1.1 Sua vez

Importe o mesmo arquivo do exemplo mas com o argumento `header = FALSE`. Execute o comando `str(x)` e veja o que mudou.

```r
dados <- read.csv2(file = "data/notas.csv", header = TRUE)
str(dados)
```

```
'data.frame':   6 obs. of  3 variables:
 $ id  : int  1 2 3 4 5 6
 $ nome: chr  "Maria" "João" "José" "Maria" ...
 $ nota: num  10 20 32.6 15 40.9 ...
```

4 Estatísticas básicas

Abaixo temos uma lista comandos para obter estatísticas básicas com o R e alguns gráficos.

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

# Média
mean(dados$nota)
```

```
[1] 25.22667
```

```
# Mediana
median(dados$nota)
```

```
[1] 26.3
```

```
# Variância
var(dados$nota)
```

```
[1] 144.4083
```

```
# Desvio padrão
sd(dados$nota)
```

```
[1] 12.017
```

```
# Valor máximo
max(dados$nota)
```

```
[1] 40.9
```

```r
# Valor mínimo
min(dados$nota)
```

```
[1] 10
```

```r
# Amplitude
range(dados$nota)
```

```
[1] 10.0 40.9
```

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

```
[1] 0.476361
```

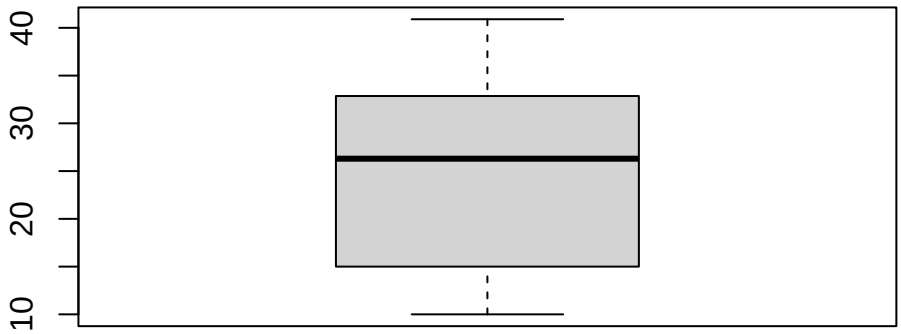
```r
# Quartis
quantile(dados$nota)
```

```
    0%    25%    50%    75%   100%
10.000 16.250 26.300 32.795 40.900
```

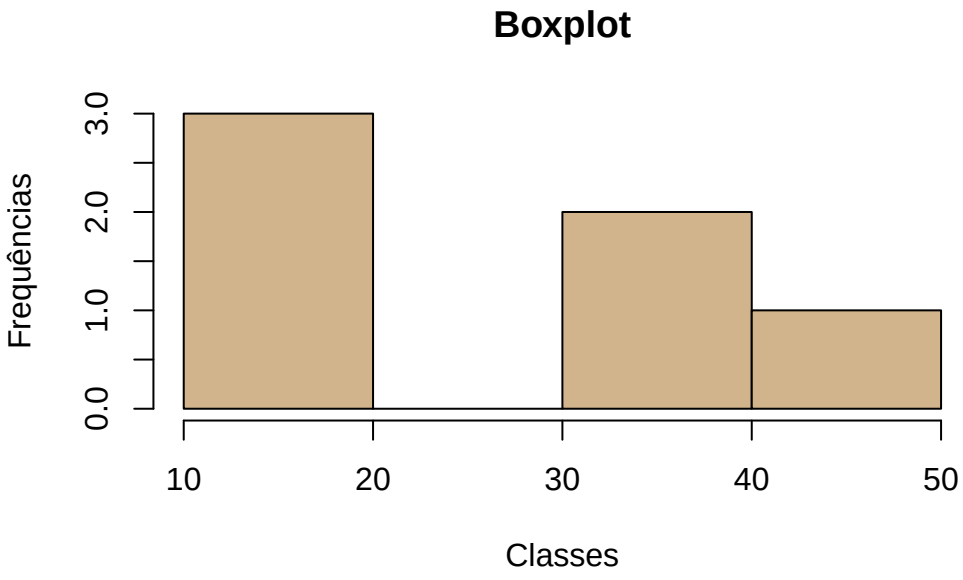
```r
# Sumário
summary(dados$nota)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.00   16.25   26.30   25.23   32.80   40.90
```

```r
# Boxplot
boxplot(dados$nota)
```




```r
# Histograma
hist(dados$nota, main = "Boxplot", xlab = "Classes", ylab = "Frequências", col = "tan")
```



4.0.1 Sua vez

O comando `normal <- rnorm(1000)` cria um objeto chamado **normal** com mil valores aleatórios obtidos à partir de uma distribuição normal $N(0, 1)$. Verifique a média e o desvio padrão deste objeto e crie um histograma.

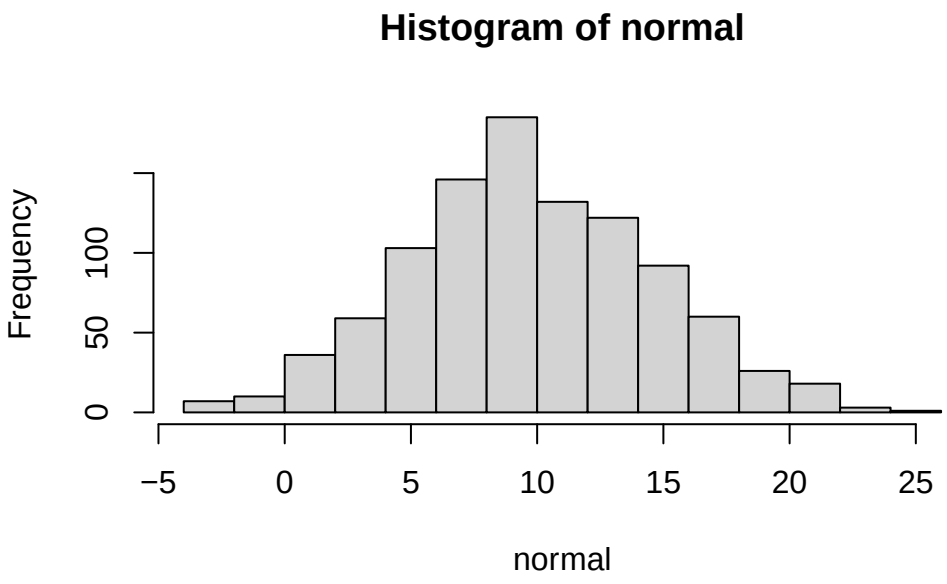
```
normal <- rnorm(1000, mean = 10, sd = 5)
mean(normal)
```

```
[1] 9.786613
```

```
sd(normal)
```

```
[1] 4.846284
```

```
hist(normal)
```



5 Análise Exploratória de Dados Espaciais (AED-E)

5.1 Leitura do shapefile

Para a leitura de arquivos **shapefile** no R, precisamos usar alguns pacotes. Após a instalação dos pacotes, use os seguintes comandos.

```
# Pacotes
library(sf)
```

```
Linking to GEOS 3.10.2, GDAL 3.4.1, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(sp)
```

```
The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
which was just loaded, will retire in October 2023.
Please refer to R-spatial evolution reports for details, especially
https://r-spatial.org/r/2023/05/15/evolution4.html.
It may be desirable to make the sf package available;
package maintainers should consider adding sf to Suggests:.
The sp package is now running under evolution status 2
     (status 2 uses the sf package in place of rgdal)
```

```
# Abra o arquivo 'gm10.shp'
fp_mg.shp <- st_read("data/FP_MG.shp", options = "ENCODING=WINDOWS-1252")
```

```
options:        ENCODING=WINDOWS-1252
Reading layer `FP_MG' from data source
  `/home/rfsaldanha/projects/ecoespacial/data/FP_MG.shp' using driver `ESRI Shapefile'
```

```
Warning in CPL_read_ogr(dsn, layer, query, as.character(options), quiet, : GDAL
Message 1: organizePolygons() received an unexpected geometry.  Either a
polygon with interior rings, or a polygon with less than 4 points, or a
non-Polygon geometry.  Return arguments as a collection.

Simple feature collection with 66 features and 41 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -51.06258 ymin: -22.91696 xmax: -39.85724 ymax: -14.23725
CRS:           NA
```

```r
fp_mg.shp <- st_make_valid(fp_mg.shp)
fp_mg.shp <- as_Spatial(fp_mg.shp)
# encoding = "UTF-8"

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

5.2 Atributos do shapefile

Podemos ver a tabela de atributos do shapefile desta forma.

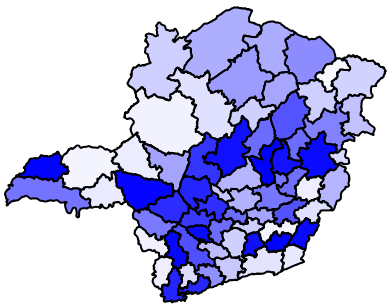
```
head(fp_mg.shp@data)
```

```
  CODMIC      NOMMIC      Q     AC     AP    K   R  RP RNP DRNP   F    AREA
1     41     Aimorés 134310  60132  60132 1833 363 155 208   25  84  8354.1
2     49     Alfenas 564476 113253 113253 7203 312 273  39    8   1  4998.9
3     14    Almenara  84631  20737  20737  783 383 150 233   15   1 15504.5
4     55 Andrelândia  59164  27764  27764 1154 264 200  64   13 227  5047.3
5     12     Araçuaí 159897  36779  36779  552 315 162 153   15   1 10299.4
6     23       Araxá 495102 160580 160643 7582 645 564  81    6 327 14145.6
  ESCTOT POPTOT    LP   ACP    KP   ETOTP   DRNPP       FP      RNPP       RPP
1    347 152658 0.111 0.394 0.012 0.00227 0.00016 0.000550 0.001363 0.001015
2    229 179366 0.111 0.631 0.040 0.00128 0.00004 0.000006 0.000217 0.001522
3    384 213342 0.047 0.097 0.004 0.00180 0.00007 0.000005 0.001092 0.000703
4    191  70783 0.037 0.392 0.016 0.00270 0.00018 0.003207 0.000904 0.002826
5    327 143468 0.143 0.256 0.004 0.00228 0.00010 0.000007 0.001066 0.001129
6    182 158275 0.043 1.015 0.048 0.00115 0.00004 0.002066 0.000512 0.003563
       R_P       QP   X_COORD   Y_COORD CMICRO     VP      VPP    LA   LM     L
1 0.002378 0.879810 -41.38876 -19.53116  31041  78349 0.513232 42737 3997 46734
2 0.001739 3.147062 -46.01852 -21.37551  31049 216064 1.204598 33737 1118 34855
3 0.001795 0.396692 -40.65978 -16.35385  31014  40164 0.188261 30920 4259 35179
4 0.003730 0.835850 -44.44562 -21.95045  31055  42360 0.598449 15135  908 16043
5 0.002196 1.114513 -41.87023 -17.00645  31012  28864 0.201188 45015 7069 52084
6 0.004075 3.128112 -46.95948 -19.59779  31023 197010 1.244732 23442 1488 24930
  CO_RUR  NU_RUR   CO_TOT      EER      K_L VEG TEMP PREC     KL
1   4937 1314045  4402218 0.003757 0.039222   1    4    1 0.0392
2   6736 2957879 12228817 0.002277 0.206656   2    2    3 0.2067
3   1431  282421  3363297 0.005067 0.022258   2    5    1 0.0223
4   1717  498819  2819950 0.003442 0.071932   2    1    4 0.0719
5   1431  365829  2404595 0.003912 0.010598   3    4    1 0.0106
6   4357 1476959 14220985 0.002950 0.304132   1    2    4 0.3041
```

5.3 Mapa

Podemos produzir um mapa colorido com os seguintes comandos.

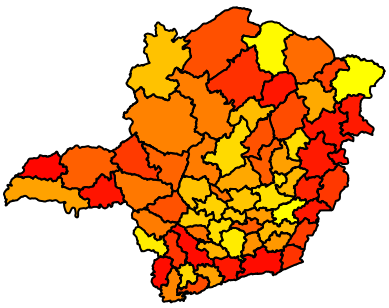
```
p <- colorRampPalette(c("white", "blue"))(128)
palette(p)
plot(fp_mg.shp, col = fp_mg.shp@data$Q)
```



5.3.1 Sua vez

Faça um mapa com a variável AC com a cor vermelha.

```
p <- colorRampPalette(c("yellow", "red"))(128)
palette(p)
plot(fp_mg.shp, col = fp_mg.shp@data$AC)
```



6 Matrizes de vizinhos espaciais

6.1 Leitura do shapefile

Para a leitura de arquivos **shapefile** no R, precisamos usar alguns pacotes. Após a instalação dos pacotes, use os seguintes comandos.

```
# Pacotes
library(sf)
```

```
Linking to GEOS 3.10.2, GDAL 3.4.1, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(sp)
```

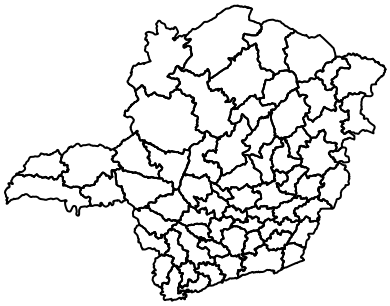
```
The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
which was just loaded, will retire in October 2023.
Please refer to R-spatial evolution reports for details, especially
https://r-spatial.org/r/2023/05/15/evolution4.html.
It may be desirable to make the sf package available;
package maintainers should consider adding sf to Suggests:.
The sp package is now running under evolution status 2
     (status 2 uses the sf package in place of rgdal)
```

```
# Abra o arquivo 'gm10.shp'
fp_mg.shp <- st_read("data/FP_MG.shp", options = "ENCODING=WINDOWS-1252")
```

```
options:        ENCODING=WINDOWS-1252
Reading layer `FP_MG' from data source
  `/home/rfsaldanha/projects/ecoespacial/data/FP_MG.shp' using driver `ESRI Shapefile'

Warning in CPL_read_ogr(dsn, layer, query, as.character(options), quiet, : GDAL
Message 1: organizePolygons() received an unexpected geometry.  Either a
polygon with interior rings, or a polygon with less than 4 points, or a
non-Polygon geometry.  Return arguments as a collection.
```

```
Simple feature collection with 66 features and 41 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: -51.06258 ymin: -22.91696 xmax: -39.85724 ymax: -14.23725
CRS:           NA
```

```r
fp_mg.shp <- st_make_valid(fp_mg.shp)
fp_mg.shp <- as_Spatial(fp_mg.shp)

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



Para a criação de matrizes de vizinhos espaciais, iremos utilizar o pacote `spdep`.

```r
# Pacote
library(spdep)
```

```
Loading required package: spData

To access larger datasets in this package, install the spDataLarge
package with: `install.packages('spDataLarge',
repos='https://nowosad.github.io/drat/', type='source')`
```

6.1.1 Matriz queen e rook

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

```
Characteristics of weights list object:
Neighbour list object:
Number of regions: 66
Number of nonzero links: 336
Percentage nonzero weights: 7.713499
Average number of links: 5.090909
Link number distribution:

 2  3  4  5  6  7  8  9
 2  9 12 16 16  8  2  1
2 least connected regions:
29 38 with 2 links
1 most connected region:
65 with 9 links

Weights style: W
Weights constants summary:
   n   nn S0       S1       S2
W 66 4356 66 27.58858 269.8006
```

```
# Matriz queen 2ª ordem
w1.2 <- nb2listw(nblag_cumul(nblag(poly2nb(fp_mg.shp, queen = TRUE), maxlag = 2)))

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

```
Characteristics of weights list object:
Neighbour list object:
Number of regions: 66
Number of nonzero links: 336
Percentage nonzero weights: 7.713499
Average number of links: 5.090909
Link number distribution:
```

```
 2  3  4  5  6  7  8  9
 2  9 12 16 16  8  2  1
2 least connected regions:
29 38 with 2 links
1 most connected region:
65 with 9 links

Weights style: W
Weights constants summary:
   n   nn S0       S1       S2
W 66 4356 66 27.58858 269.8006
```

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

```
Characteristics of weights list object:
Neighbour list object:
Number of regions: 66
Number of nonzero links: 332
Percentage nonzero weights: 7.621671
Average number of links: 5.030303
Link number distribution:

 2  3  4  5  6  7  8
 2  9 12 18 15  7  3
2 least connected regions:
29 38 with 2 links
3 most connected regions:
12 26 65 with 8 links

Weights style: W
Weights constants summary:
   n   nn S0       S1       S2
W 66 4356 66 27.82221 269.6778
```

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

```
Characteristics of weights list object:
Neighbour list object:
Number of regions: 66
Number of nonzero links: 332
Percentage nonzero weights: 7.621671
Average number of links: 5.030303
Link number distribution:

 2  3  4  5  6  7  8
 2  9 12 18 15  7  3
2 least connected regions:
29 38 with 2 links
3 most connected regions:
12 26 65 with 8 links

Weights style: C
Weights constants summary:
   n   nn S0       S1      S2
C 66 4356 66 26.24096 285.489
```

6.1.2 Distância inversa

```r
coords <- coordinates(fp_mg.shp)
nb <- dnearneigh(coords, 0, 1000)
dlist <- nbdists(nb, coords)
dlist <- lapply(dlist, function(x) 1/x^2)
w3 <- nb2listw(nb, glist=dlist)
summary(w3)
```

```
Characteristics of weights list object:
Neighbour list object:
Number of regions: 66
Number of nonzero links: 4290
Percentage nonzero weights: 98.48485
Average number of links: 65
Link number distribution:

65
66
66 least connected regions:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
```

```
66 most connected regions:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34

Weights style: W
Weights constants summary:
   n   nn S0       S1       S2
W 66 4356 66 8.369751 267.6908
```

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

```
Characteristics of weights list object:
Neighbour list object:
Number of regions: 66
Number of nonzero links: 4290
Percentage nonzero weights: 98.48485
Average number of links: 65
Link number distribution:

65
66
66 least connected regions:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
66 most connected regions:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34

Weights style: U
Weights constants summary:
   n   nn S0          S1         S2
U 66 4356  1 0.002368187 0.07316268
```

Para ver mais opções, veja a ajuda deste comando: `?nb2listw`

6.1.3 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.

```r
# Número de permutações
per <- 999

# Número máximo de k vizinhos testados
kv <- 20

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

# Criação da tabela que irá receber a estatística I de Moran e significância para cada k t
res.pesos <- data.frame(k=numeric(),i=numeric(),valorp=numeric())

# 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(fp_mg.shp@data$Q,
                      listw=nb2listw(knn2nb(
                      knearneigh(coords, k=k),
                      row.names=IDs),style="B"),
                      nsim=per)
  # Armazenando o valor I para o k atual
  res.pesos[k,2] <- moran.k$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
```

```
   k         i valorp
1  1 0.5228074  0.006
2  2 0.3875458  0.005
3  3 0.4531317  0.001
4  4 0.4199339  0.001
5  5 0.3944831  0.001
6  6 0.3595862  0.001
7  7 0.3461349  0.001
8  8 0.3286129  0.001
9  9 0.3064023  0.001
```

```
10 10 0.3157462  0.001
11 11 0.3028398  0.001
12 12 0.2942354  0.001
13 13 0.2791438  0.001
14 14 0.2620697  0.001
15 15 0.2541920  0.001
16 16 0.2429784  0.001
17 17 0.2320723  0.001
18 18 0.2213339  0.001
19 19 0.2117356  0.001
20 20 0.2017898  0.001
```

```r
# 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")
summary(w5)
```

```
Characteristics of weights list object:
Neighbour list object:
Number of regions: 66
Number of nonzero links: 66
Percentage nonzero weights: 1.515152
Average number of links: 1
Non-symmetric neighbours list
Link number distribution:

 1
66
66 least connected regions:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
66 most connected regions:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34

Weights style: B
Weights constants summary:
   n   nn S0 S1  S2
B 66 4356 66 98 308
```

6.2 Autocorrelação espacial global

6.2.1 I de Moran

```
moran.test(fp_mg.shp@data$Q, listw = w5)
```

```
    Moran I test under randomisation

data:  fp_mg.shp@data$Q
weights: w5

Moran I statistic standard deviate = 3.8645, p-value = 5.566e-05
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
       0.52280745       -0.01538462        0.01939499
```

```
moran.mc(fp_mg.shp@data$Q, listw = w5, nsim = 999)
```

```
    Monte-Carlo simulation of Moran I

data:  fp_mg.shp@data$Q
weights: w5
number of simulations + 1: 1000

statistic = 0.52281, observed rank = 996, p-value = 0.004
alternative hypothesis: greater
```

6.2.2 C de Geary

```
geary.test(fp_mg.shp@data$Q, listw = w5)
```

```
    Geary C test under randomisation

data:  fp_mg.shp@data$Q
```

```
weights: w5

Geary C statistic standard deviate = 2.6176, p-value = 0.004428
alternative hypothesis: Expectation greater than statistic
sample estimates:
Geary C statistic       Expectation          Variance
       0.46130049        1.00000000        0.04235442
```

```
geary.mc(fp_mg.shp@data$Q, listw = w5, nsim = 999)
```

```
    Monte-Carlo simulation of Geary C

data:  fp_mg.shp@data$Q
weights: w5
number of simulations + 1: 1000

statistic = 0.4613, observed rank = 3, p-value = 0.003
alternative hypothesis: greater
```

6.2.3 G de Getis-Ord

```
globalG.test(as.vector(scale(fp_mg.shp@data$Q, center = FALSE)), listw = w5, B1correct = T
```

```
    Getis-Ord global G statistic

data:  as.vector(scale(fp_mg.shp@data$Q, center = FALSE))
weights: w5

standard deviate = 3.2113, p-value = 0.0006607
alternative hypothesis: greater
sample estimates:
Global G statistic        Expectation           Variance
      3.071155e-02       1.538462e-02       2.277991e-05
```

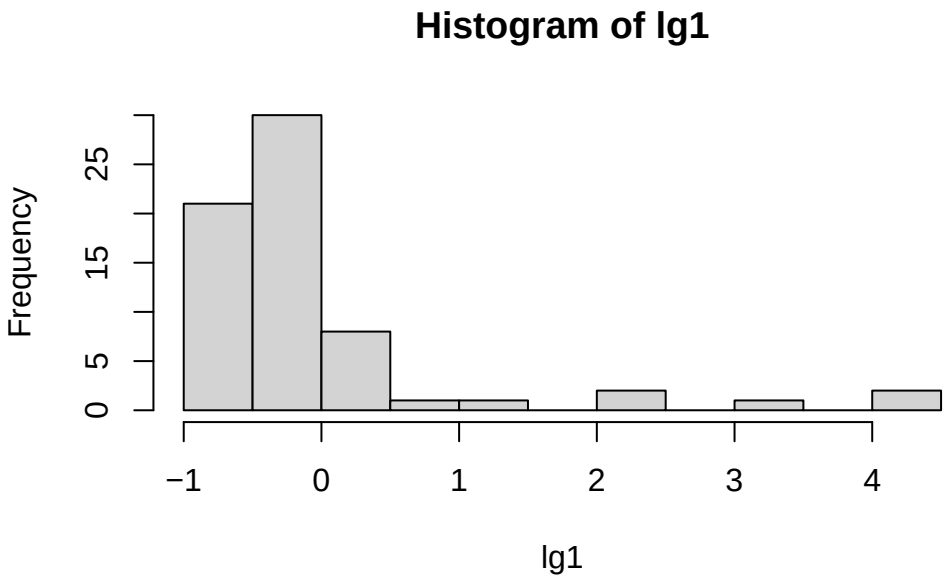
6.3 Autocorrelação espacial local

6.3.1 G de Gettis-Ords

```
lg1 <- localG(fp_mg.shp@data$Q, listw = w5)
summary(lg1)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.69900 -0.53552 -0.44190 -0.04454 -0.06779  4.21616
```

```
hist(lg1)
```



6.3.2 I de Moran

```
# Cálculo
lm1 <- localmoran(fp_mg.shp@data$Q, listw = w5)
summary(lm1)
```

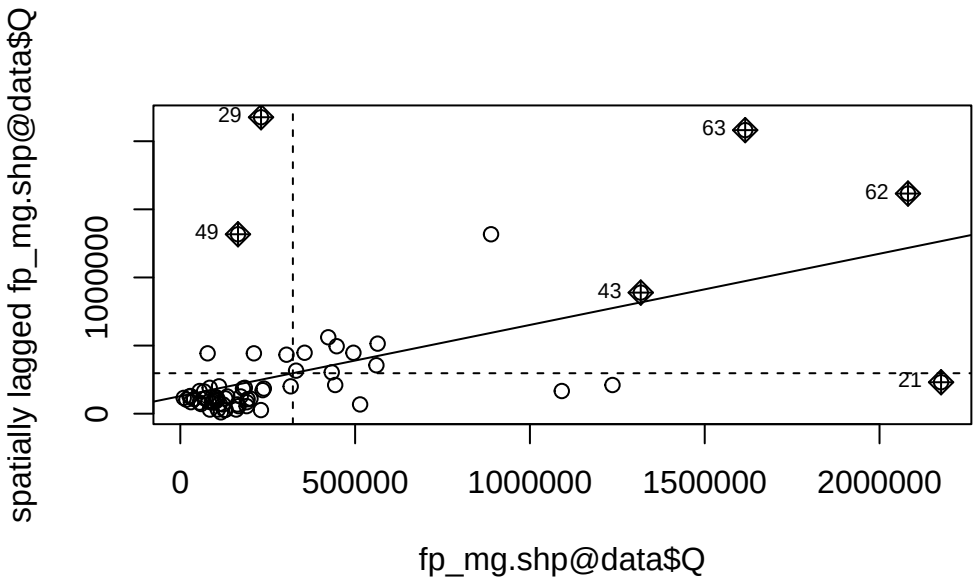
```
      Ii                E.Ii                Var.Ii              Z.Ii
 Min.   :-0.83956   Min.   :-2.629e-01   Min.   : 0.000197   Min.   :-4.1007
 1st Qu.: 0.07066   1st Qu.:-4.908e-03   1st Qu.: 0.089934   1st Qu.: 0.2869
 Median : 0.20075   Median :-3.384e-03   Median : 0.222559   Median : 0.4695
 Mean   : 0.52281   Mean   :-1.538e-02   Mean   : 0.860585   Mean   : 0.3957
 3rd Qu.: 0.27959   3rd Qu.:-1.365e-03   3rd Qu.: 0.322327   3rd Qu.: 0.5559
 Max.   :11.31579   Max.   :-2.980e-06   Max.   :12.789972   Max.   : 4.2162
 Pr(z != E(Ii))
 Min.   :0.0000248
 1st Qu.:0.5703228
 Median :0.6347460
 Mean   :0.6120352
 3rd Qu.:0.7541930
 Max.   :0.9892306
```

```
# Quantos são significativos?
lm1 <- as.data.frame(lm1)
table(lm1$`Pr(z > 0)` < 0.05)
```

```
< table of extent 0 >
```

6.4 Diagrama de dispersão de Moran

```
moran.plot(fp_mg.shp@data$Q, listw = w5)
```



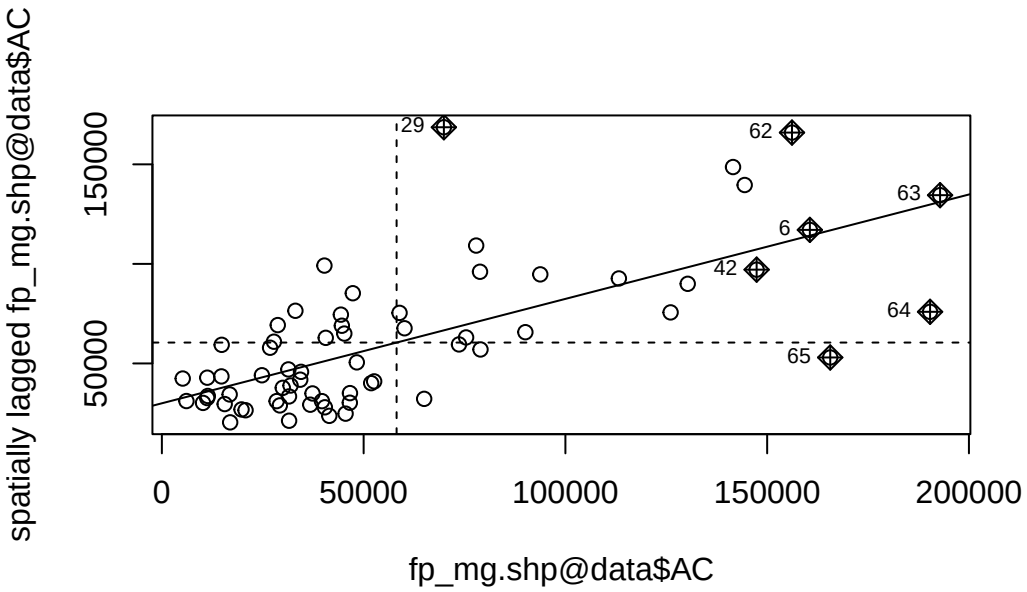
6.4.1 Sua vez

Calcule o I de Moran local usando a matriz de vizinhança `w1` para a variável `AC`e verifique quantas regiões são significativas. Depois, faça o diagrama de dispersão.

```
head(localmoran(fp_mg.shp@data$AC, listw = w1))
```

```
           Ii           E.Ii       Var.Ii       Z.Ii Pr(z != E(Ii))
1  0.008331081 -2.648711e-05 0.0004165402  0.4094977   0.6821744760
2  0.849842734 -2.085884e-02 0.2071096162  1.9132389   0.0557174771
3  0.529839200 -9.632289e-03 0.1500236654  1.3927995   0.1636804207
4 -0.037360854 -6.355270e-03 0.0781466481 -0.1109136   0.9116848748
5  0.276052134 -3.145335e-03 0.0317953356  1.5657765   0.1174009499
6  2.695852306 -7.209967e-02 0.6784210807  3.3605386   0.0007779067
```

```
moran.plot(fp_mg.shp@data$AC, listw = w1)
```



6.5 LISA map

O R não tem uma função pronta para criar um mapa LISA, então nós criamos abaixo nossa própria função: `lisaplot`. Depois de declarada, uma função pode ser usada repetidamente variando seus argumentos.

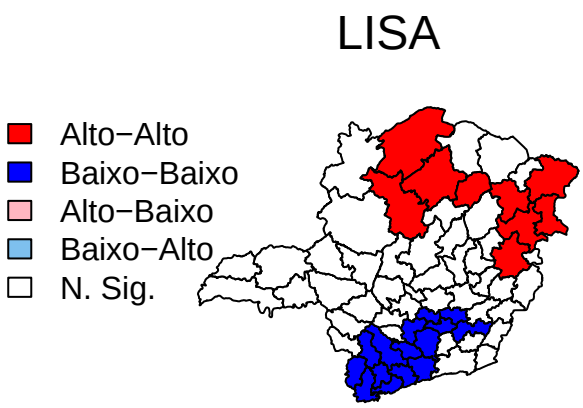
Rode o código abaixo.

```
lisaplot <- function(shapefile, values, listw, pval = 0.05){
  require(spdep)

  svalues <- as.vector(scale(values, scale = FALSE))
  lag_svalues <- spdep::lag.listw(listw, svalues)
  locm <- spdep::localmoran(values, listw)
  sig <- rep(0, length(values))

  sig[(svalues >= 0 & lag_svalues >= 0) & (locm[,5] <= pval)] <- 1
  sig[(svalues <= 0 & lag_svalues <= 0) & (locm[,5] <= pval)] <- 2
  sig[(svalues >= 0 & lag_svalues <= 0) & (locm[,5] <= pval)] <- 3
  sig[(svalues <= 0 & lag_svalues >= 0) & (locm[,5] <= pval)] <- 4
  sig[locm[,5] > pval] <- 5
```

```
  breaks <- seq(1, 5, 1)
  labels <- c("Alto-Alto", "Baixo-Baixo", "Alto-Baixo", "Baixo-Alto", "N. Sig.")
  np <- findInterval(sig, breaks)
  colors <- c("red", "blue", "lightpink", "skyblue2", "white")
  plot(shapefile, col = colors[np])
  mtext("LISA", cex = 1.5, side = 3, line = 1)
  legend("topleft", legend = labels, fill = colors, bty = "n")
}
```

E o LISA para a variável `TEMP`.

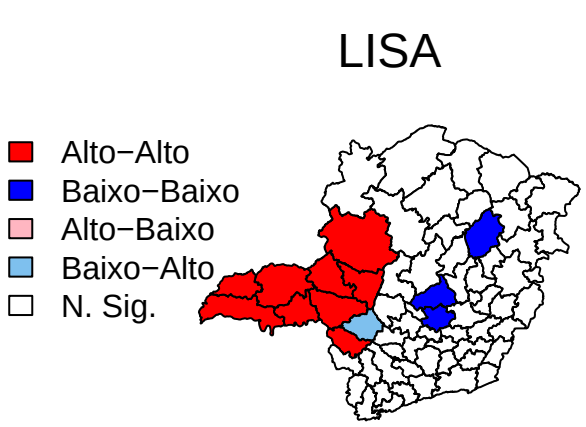
```
lisaplot(fp_mg.shp, fp_mg.shp@data$TEMP, w1)
```



6.5.1 Sua vez

Faça o LISA para a variável `AP` com a matriz `w1`.

```
lisaplot(fp_mg.shp, fp_mg.shp@data$AP, w1)
```


LISA
Alto−Alto
Baixo−Baixo
Alto−Baixo
Baixo−Alto
N. Sig.

Referências

Almeida, Eduardo. 2012. *Econometria espacial aplicada*. Campinas: Alínea.