Equações em diferenças no R

Ivette Luna

23 de abril de 2019

Contents

Data frames

São estruturas de dados em que cada coluna é associada a uma variável (atributo) para cada objeto (agente, indivíduo, elemento) armazenado nas linhas.

Por exemplo, dentro de uma simulação por agentes, precisaremos criar uma população inicial. Suponhamos que seja uma população de firmas com os atributos:

- Tamanho da firma (`tamanho`) do tipo `integer`,
- Valor adicionado (`va`) do tipo `numeric`,
- Se a empresa inova ou não (`inova`) do tipo binário.

Podemos criar as variáveis e produir um dataframe para armazenar as informações. Mas, para isso, precisamos tomar algumas decisões:

- Qual será o tamanho da população?
- Qual será o tamanho das firmas?
- Haverá um tamanho mínimo?

```
library(ggplot2)
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
n = 100
tamanho = 5 + round( 595*runif( n ) )
va = 100000*runif( n )
inova = round(runif(n))

firmas <- data.frame(  tamanho, va, inova  )

View(firmas)

# Para dar uma olhada geral na estrutura da base:
str(firmas)
```

```
## 'data.frame':    100 obs. of  3 variables:
##  $ tamanho: num  173 382 395 406 226 573 217 204 183 545 ...
##  $ va     : num  56120 11822 1622 24797 34499 ...
##  $ inova  : num  1 0 1 0 1 1 0 1 1 1 ...
```

```
# Para ter uma ideia dos seus valores

summary(firmas)
```

```
##     tamanho            va              inova
##  Min.   :  6.0   Min.   :  595.8   Min.   :0.00
##  1st Qu.:179.5   1st Qu.:25547.1   1st Qu.:0.00
##  Median :298.0   Median :51767.0   Median :0.00
##  Mean   :303.6   Mean   :50883.2   Mean   :0.42
```

```
##  3rd Qu.:424.2   3rd Qu.:77599.0   3rd Qu.:1.00
##  Max.   :595.0   Max.   :98383.2   Max.   :1.00
```

```
# Querendo transformar a variavel inova em categorica

firmas$inova <- as.factor(firmas$inova)

summary(firmas)
```

```
##     tamanho            va          inova
##  Min.   :  6.0   Min.   :  595.8   0:58
##  1st Qu.:179.5   1st Qu.:25547.1   1:42
##  Median :298.0   Median :51767.0
##  Mean   :303.6   Mean   :50883.2
##  3rd Qu.:424.2   3rd Qu.:77599.0
##  Max.   :595.0   Max.   :98383.2
```

Manipulação de data frames

O pacote que se destaca para esta finalidade é o pacote `dplyr`, pela sua versatilidade. Ele é baseado em verbos que representam ações a executar sobre determinada base de dados.

As instruções são realizadas por meio de uma sequência de ações definidas com a ajuda do operador `%>%` (o *pimp*).

Por exemplo, podemos criar a variavel `produtividade` com o verbo `mutate`:

```
firmas <- firmas %>% mutate(produtividade = va/tamanho)

head(firmas)
```

```
##   tamanho        va inova produtividade
## 1     173 56120.221     1    324.394340
## 2     382 11821.636     0     30.946692
## 3     395  1622.337     1      4.107181
## 4     406 24796.548     0     61.075242
## 5     226 34499.105     1    152.650908
## 6     573 70630.467     1    123.264340
```

Já se queremos analisar apenas as empresas inovadoras, podemos aplicar um filtro fazendo uso do verbo `filter`:

```
firmas_inovadoras <- firmas %>% filter( inova==1 )

summary(firmas_inovadoras)
```

```
##     tamanho            va        inova  produtividade
##  Min.   :  6.0   Min.   : 1622   0: 0   Min.   :   4.107
##  1st Qu.:174.2   1st Qu.:24778   1:42   1st Qu.: 126.875
##  Median :256.5   Median :53827          Median : 167.941
##  Mean   :293.2   Mean   :52261          Mean   : 383.699
##  3rd Qu.:397.2   3rd Qu.:77389          3rd Qu.: 318.379
##  Max.   :595.0   Max.   :98025          Max.   :3054.847
```

Mas, usando outros verbos, podemos fazer esse filtro para cada categoria de `inova`:

```
descritivas <- firmas %>% group_by(inova) %>%
```

```
                summarise( media_tamanho = mean(tamanho),
                           dp_tamanho = sd(tamanho),
                           media_prod = mean(produtividade),
                           dp_prod = sd(produtividade),
                           n = n())

descritivas
```

```
## # A tibble: 2 x 6
##   inova media_tamanho dp_tamanho media_prod dp_prod     n
##   <fct>         <dbl>      <dbl>      <dbl>   <dbl> <int>
## 1 0              311.       145.       231.    216.    58
## 2 1              293.       172.       384.    586.    42
```

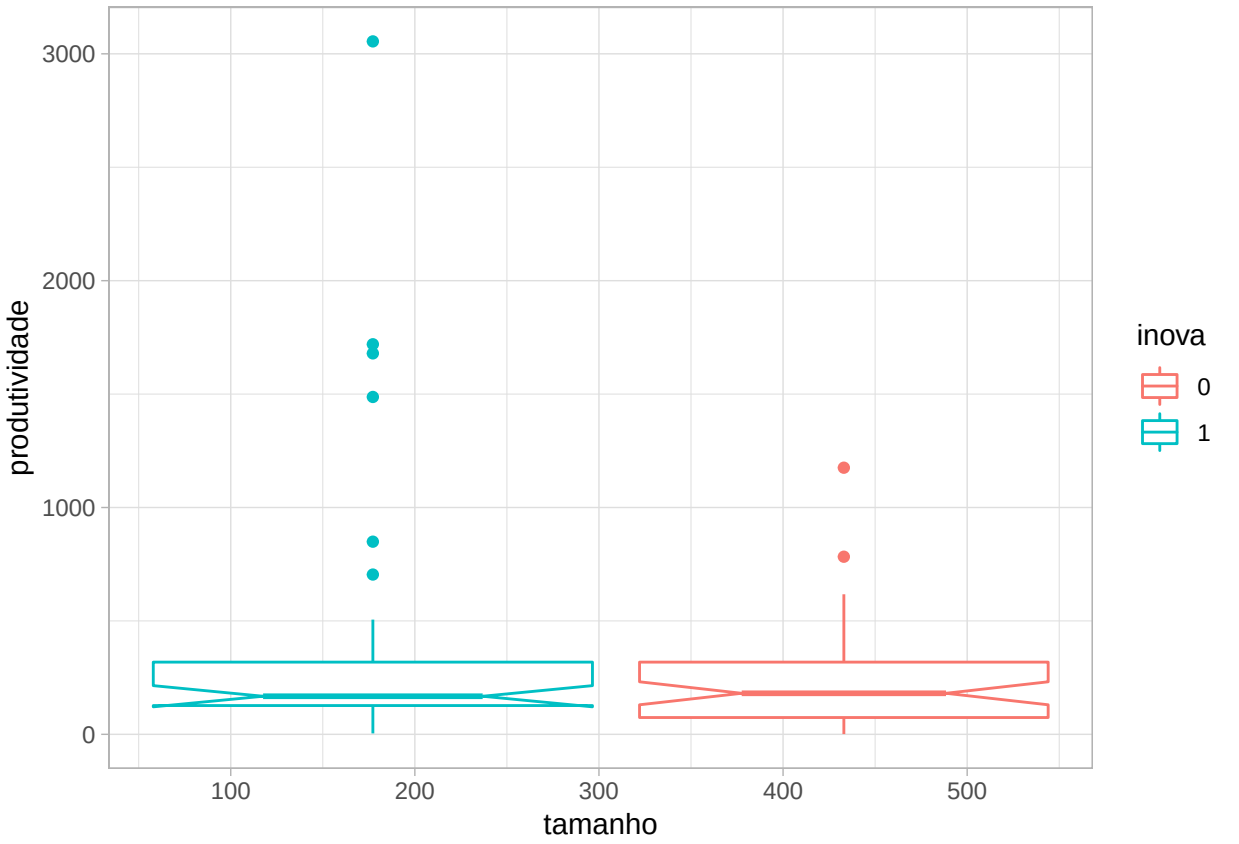
Ainda, o ggplot exige que as informações que desejamos visualizar em um gráfico estejam armazenadas em uma data frame:

```
ggplot(firmas, aes(x = tamanho, y = produtividade, color = inova)) +

  #geom_point(  ) +
  geom_boxplot( notch=TRUE ) +

  theme_light( )
```

```
## notch went outside hinges. Try setting notch=FALSE.
```



O céu é o limite! Verifique os sites sugeridos ao final deste material, além dos cursos no Datacamp.

Para acessar dados de um dataframe usamos a notação matricial (os indexadores de linhas e colunas:

```
firmas[ , "va" ] # pelo nome da variável

firmas$va # usando o operador $

firmas[10, ] # pelo numero de linha

firmas[ , 2 ] # pelo numero de coluna

names( firmas  ) # para ver apenas os nomes das variaveis da base
```

Funções

São objetos (como as matrizes e os vetores) cuja estrutura geral de uma função é dada por

```
myfunction <- function(arg1, arg2, ... ){

  statements

  return(object)
}
```

De Hadley:

A function has three parts:

- *The formals(), the list of arguments that control how you call the function.*
- *The body(), the code inside the function.*
- *The environment(), the data structure that determines how the function finds the values associated with the names.*

Por exemplo, podemos construir uma função que calcule a média geométrica de dois valores `x` e `y`:

```
med.geom <- function(x, y){

  resultado <- (x*y)^(1/2)

  return(resultado)

}

med.geom(10, 3)

med.geom( c(1,2), c(2,4) ) # o que acontece neste caso?
```

Funções e as equações em diferença

Podemos fazer uso de funções para aprimorar as nossas simulações com as equações em diferença. Por exemplo, dada a equação

$$y_{t+1} = a_0 y_t$$

a simulação requer que definamos a condição inicial, o número de iterações e o valor do parâmetro a_0. Além disso, precisamos de inicializar alguns objetos para armazenar os resultados. Se trata da especiicação do cenário de simulação que implica a tomada de decisões:

```
tmax = 20

y0 = 2
a0 = 3/4

y = rep(0, tmax)
y[1] = y0

for (t in 2:tmax){

  y[t] = a0*y[t-1]

}

y
```

```
##  [1] 2.000000000 1.500000000 1.125000000 0.843750000 0.632812500
##  [6] 0.474609375 0.355957031 0.266967773 0.200225830 0.150169373
## [11] 0.112627029 0.084470272 0.063352704 0.047514528 0.035635896
## [16] 0.026726922 0.020045192 0.015033894 0.011275420 0.008456565
```

Para construir uma função que produza a série temporal como resultado, podemos passar à mesma os parâmetros na forma de argumentos. Por exemplo:

```
eq.ordem1 <- function(y0, a0, tmax){

  y = rep(0, tmax)
  y[1] = y0

  for (t in 2:tmax){

    y[t] = a0*y[t-1]

  }

  return(y)

}


serie_temp1 <- eq.ordem1(y0, a0, tmax)

serie_temp1
```

E a função pode assim ser reutilizada para avaliar outros casos:

```
serie_temp2 <- eq.ordem1( y0, -1.1, tmax )

serie_temp2

serie_temp3 <- eq.ordem1( y0, 1.1, tmax )
```

```
serie_temp3
```

Sites interessantes

Jogo de xadrez

Dplyr

Introdução ao dplyr

Data manipulation with R

Functions fundamentals in R

Creating functions