Sistemas de equações em diferenças Cournot e Markov

Henri Makika

Junho 4, 2019

Contents

Simulação: Oligopólio de Cournot

Sejam n as firmas em um mercado oligopolístico de produto homogêneo, em que as firmas são maximizadoras de lucro, com curvas de demanda e custos lineares tal que a elasticidade preço da demanda e os custos marginais são constantes. A curva de demanda:

$$p_t = a - b\sum_{i=1}^{n} x_{it}$$

onde x_{it} é a produção da i ésima firma no período t; $a, b > 0$. A curva linear de custos é tal que :

$$C_{it} = d + c_i x_{it}$$

em que $d, c_i > 0$.

```
library(matlib)
library(ggplot2)
library(tidyr)
```

```
set.seed(10)
n = 2;
a = 10
b = 0.5
d = 10

c = runif(n, 0, 1)
tmax = 20
X = matrix(0,n, tmax)
X[,1] = 10*runif(n, 0, 1)

B = (a-c)/2*b

A = matrix(-0.5, n, n)
A = A + diag(0.5, n, n)
A
```

```
##      [,1] [,2]
## [1,]  0.0 -0.5
## [2,] -0.5  0.0
```

```
# Autovalores
lambda = eigen(A)$values
lambda
```

```
## [1]  0.5 -0.5
```

Os autovalores para $n = 2$ são os esperados, e por tanto, temos um sistema estável, com oscilações amortecidas e convergindo a um steady state dado por

$$x_{it+1} = x_{it} = x_i*$$

```
I = diag(1, n, n)
Xeq = inv(I-A)%*%B
Xeq
```

```
##          [,1]
## [1,] 1.548635
## [2,] 1.648990
```

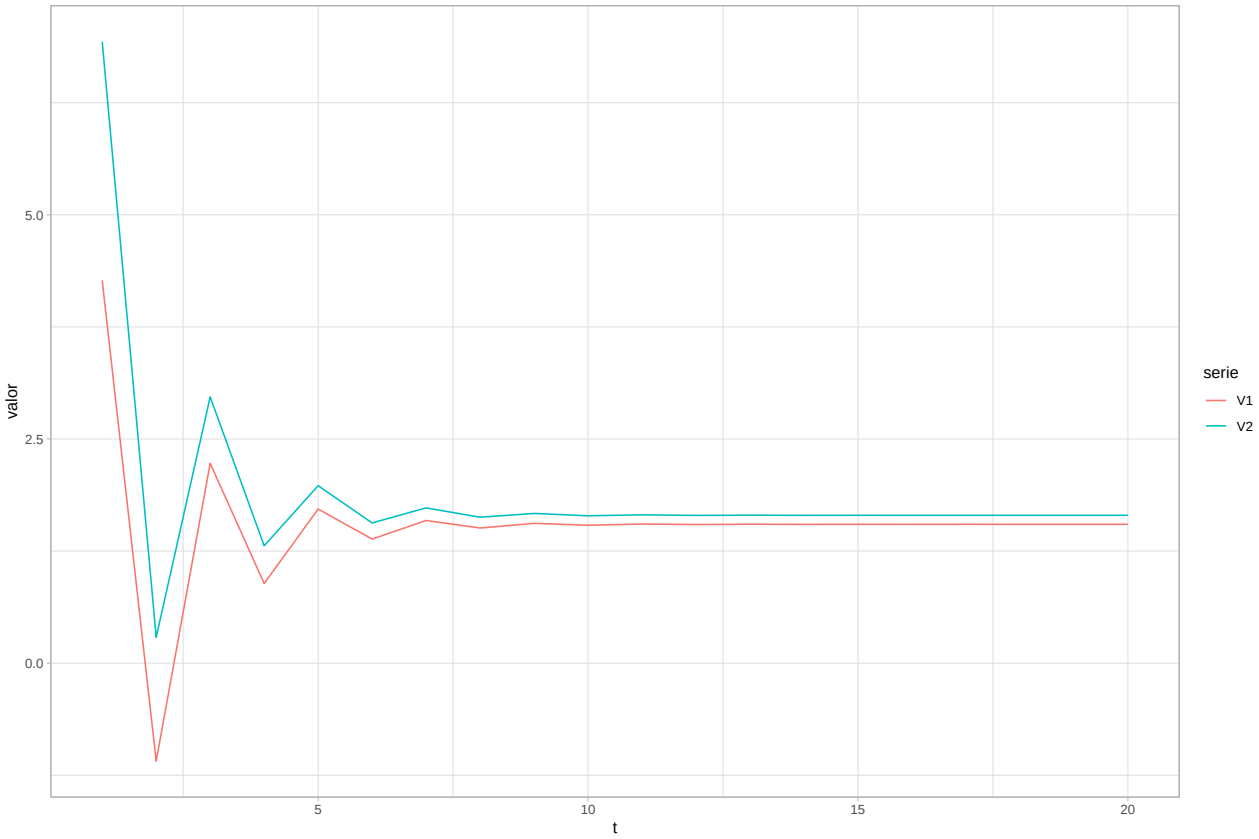
Simulando as trajetórias:

```
for (t in 2:tmax){
  X[ ,t] = A %*% X[ , t-1] + B
}
x = t(X) # transposta
series = as.data.frame(x)
series$t = seq(1, tmax, 1)
head(series)
```

```
##           V1        V2 t
## 1  4.2690767 6.9310208 1
## 2 -1.0923800 0.2887695 2
## 3  2.2287457 2.9694979 3
## 4  0.8883815 1.3089350 4
## 5  1.7186629 1.9791171 5
## 6  1.3835719 1.5639764 6
```

```
series_tidy = gather(series, -t, key="serie", value="valor")

plot_series = ggplot(series_tidy, aes(x = t, y = valor, color = serie)) +
  geom_line() +
  theme_light()
plot_series
```



Pour sauver le graphique

```
#ggsave("series_duopolio.png", plot_series, width = 5, height = 3, units = "in")
#getwd()
```

Cadeias de Markov

Resolvendo o exercício do slide, temos a seguinte matriz de transição:

```
#coefs <-  matrix(c(3, 2, 0,0,0, 1, 5, 5, 0, 0, 0, 1, 8, 7, 5, 0, 0, 4, 16, 32, 0, 0, 0, 11, 100),
                  #nrow = 5, ncol = 5)
#print(coefs)

# On peut aussi écrire
coefs = c(3, 2, 0,0,0, 1, 5, 5, 0, 0, 0, 1, 8, 7, 5, 0, 0, 4, 16, 32, 0, 0, 0, 11, 100)
n_states = 5
S <- matrix(coefs, n_states)
S
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    3    1    0    0    0
## [2,]    2    5    1    0    0
## [3,]    0    5    8    4    0
## [4,]    0    0    7   16   11
## [5,]    0    0    5   32  100
```

```
total_2005 <- rowSums(S)
total_2005
```

```
## [1]   4   8  17  34 137
```

Construindo a matriz de transicao

```
A = S/total_2005
A
```

```
##      [,1]      [,2]       [,3]      [,4]      [,5]
## [1,] 0.75 0.2500000 0.00000000 0.0000000 0.0000000
## [2,] 0.25 0.6250000 0.12500000 0.0000000 0.0000000
## [3,] 0.00 0.2941176 0.47058824 0.2352941 0.0000000
## [4,] 0.00 0.0000000 0.20588235 0.4705882 0.3235294
## [5,] 0.00 0.0000000 0.03649635 0.2335766 0.7299270
```

```
rowSums(A)
```

```
## [1] 1 1 1 1 1
```

Analisando as raízes do sistema dinâmico :

$$X_{t+1} = A.X_t$$

para analisar a dinamica basta olhar para os autovalores

```
eigen(A)$value
```

```
## [1] 1.0000000 0.9229558 0.6010358 0.3750432 0.1470686
```

raizes da matriz de transicão. Temos que todas as raízes são menores o iguais a um em módulo, e que todas asraízes são positivas. Logo, não haverá oscilações, mas a raíz unitária não permite atingir o steady state no longo prazo. Considerando as probabilidades de transição constantes, podemos estimar a nova estrutura produtiva para os anos de 2010 e 2015:

```
total_2010 = colSums(S)
total_2010
```

```
## [1]   5  11  21  52 111
```

```
total_2015 <- round(A%*%total_2010)
total_2015
```

```
##      [,1]
## [1,]    6
## [2,]   11
## [3,]   25
## [4,]   65
## [5,]   94
```

```
total_2020 <- round(A%*%total_2015)
total_2020
```

```
##      [,1]
## [1,]    7
## [2,]   12
## [3,]   30
## [4,]   66
## [5,]   85
```

Simulando para um t longo:

```
tmax = 100
X = matrix(0, n_states, tmax)
X[, 1] = total_2020
for (t in 2:tmax){
  #X[, t] = round(A %*% X[, t-1] )
  X[, t] = A %*% X[, t-1]
}

series = as.data.frame( t(X))
series$t = seq(1, tmax, 1)
series_tidy = gather(series, -t, key="serie", value="valor")

ggplot(series_tidy, aes(x = t, y = valor, color = serie)) +
  geom_line() +
  theme_light()
```