Sistemas de equações em diferenças no R

Ivette Luna

21 de maio de 2019

Contents

Autovalores e autovetores

Para identificar os autovalores e autovetores de uma matriz quadrada, fazemos uso da função `eigen` do pacote base do R:

```
library(Matrix)
library(matlib) # inv - Ginv


v = c(2, 1, 1, 2)
A = matrix(v, nrow=2)


r = eigen(A)

lambda = r$values

P = r$vectors
P
```

```
##           [,1]       [,2]
## [1,] 0.7071068 -0.7071068
## [2,] 0.7071068  0.7071068
```

```
det(P) # diferente de zero
```

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

```
inv(P) %*% A %*% P
```

```
##      [,1] [,2]
## [1,]    3    0
## [2,]    0    1
```

```
# primeiro os valores na diagonal e logo a dimensao
diag(lambda, nrow(A))
```

```
##      [,1] [,2]
## [1,]    3    0
## [2,]    0    1
```

```
#Ginv(P) %*% A %*% P
```

Havendo autovalores reais e diferentes, há garantia para obter n autovetores linearmente independentes, o que garante a existência da matriz P não singular.

Havendo multiplicidade, precisamos verificar se é possível obter a matriz P. Obter a matriz P implica ter uma matriz A *diagonalizável*.

Diagonalização de matrizes

Vejamos o exemplo a seguir:

```
A = matrix(c(1, 0, 0, 2, 1, 0, 3, 2, 1), nrow=3)

r = eigen(A)

lambda = r$values
```

```
P = r$vectors
P
```

```
##      [,1]          [,2]          [,3]
## [1,]    1 -1.000000e+00  1.000000e+00
## [2,]    0  1.110223e-16 -1.110223e-16
## [3,]    0  0.000000e+00  1.232595e-32
```

```
det(P) # diferente de zero?
```

```
## [1] 1.368456e-48
```

```
det(P) != 0 # diferente de zero?
```

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

```
detP = format(det(P), scientific = FALSE, trim = TRUE)
detP
```

```
## [1] "0.000000000000000000000000000000000000000000000001368456"
```

```
tol = 1e-5
detP = ifelse( detP<tol, 0, round(detP,4) )
detP != 0 # diferente de zero?
```

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

```
# verificando os graus de liberdade
Ahat = A - lambda[1]*diag(1, nrow(A) )

rankMatrix(Ahat)[1]
```

```
## [1] 2
```

```
echelon(Ahat)
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    0
## [2,]    0    0    1
## [3,]    0    0    0
```

Simulando um sistema com raízes reais e diferentes

Neste caso, há certeza da diagonalização da matriz de coeficientes.

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:Matrix':
##
##     expand
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
##
```

```
## 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
```

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```r
A = matrix(c(0, 0.125, 1, .25), nrow=2)

r = eigen(A)

lambda = r$values
lambda
```

```
## [1]  0.50 -0.25
```

```r
P = r$vectors
P
```

```
##            [,1]       [,2]
## [1,] -0.8944272 -0.9701425
## [2,] -0.4472136  0.2425356
```

```r
det(P)
```

```
## [1] -0.6507914
```

```r
Z0 = c(1,.1) # condicao inicial

tmax = 20
Z = matrix(0, nrow=2, ncol=tmax)
Z[,1] = Z0

for (t in 2:tmax){

  Z[,t] = A %*% Z[ , t-1]

}

t(Z)
```

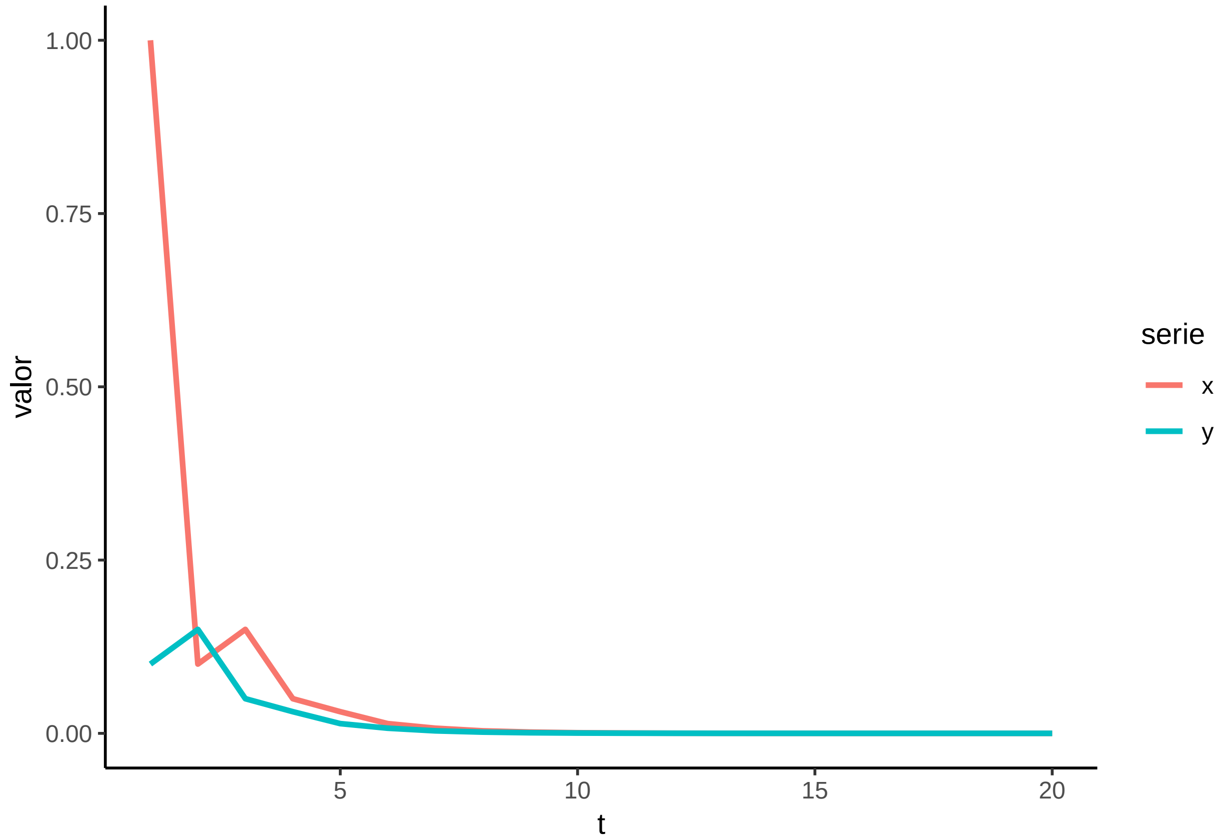
```
##               [,1]         [,2]
##  [1,] 1.000000e+00 1.000000e-01
##  [2,] 1.000000e-01 1.500000e-01
##  [3,] 1.500000e-01 5.000000e-02
##  [4,] 5.000000e-02 3.125000e-02
##  [5,] 3.125000e-02 1.406250e-02
##  [6,] 1.406250e-02 7.421875e-03
##  [7,] 7.421875e-03 3.613281e-03
##  [8,] 3.613281e-03 1.831055e-03
##  [9,] 1.831055e-03 9.094238e-04
## [10,] 9.094238e-04 4.562378e-04
```

```
## [11,] 4.562378e-04 2.277374e-04
## [12,] 2.277374e-04 1.139641e-04
## [13,] 1.139641e-04 5.695820e-05
## [14,] 5.695820e-05 2.848506e-05
## [15,] 2.848506e-05 1.424104e-05
## [16,] 1.424104e-05 7.120892e-06
## [17,] 7.120892e-06 3.560353e-06
## [18,] 3.560353e-06 1.780200e-06
## [19,] 1.780200e-06 8.900941e-07
## [20,] 8.900941e-07 4.450485e-07
```

```r
t = seq(1, tmax, 1)

series = data.frame(t, x = Z[1, ], y = Z[2, ]  )
series
```

```
##     t            x            y
## 1   1 1.000000e+00 1.000000e-01
## 2   2 1.000000e-01 1.500000e-01
## 3   3 1.500000e-01 5.000000e-02
## 4   4 5.000000e-02 3.125000e-02
## 5   5 3.125000e-02 1.406250e-02
## 6   6 1.406250e-02 7.421875e-03
## 7   7 7.421875e-03 3.613281e-03
## 8   8 3.613281e-03 1.831055e-03
## 9   9 1.831055e-03 9.094238e-04
## 10 10 9.094238e-04 4.562378e-04
## 11 11 4.562378e-04 2.277374e-04
## 12 12 2.277374e-04 1.139641e-04
## 13 13 1.139641e-04 5.695820e-05
## 14 14 5.695820e-05 2.848506e-05
## 15 15 2.848506e-05 1.424104e-05
## 16 16 1.424104e-05 7.120892e-06
## 17 17 7.120892e-06 3.560353e-06
## 18 18 3.560353e-06 1.780200e-06
## 19 19 1.780200e-06 8.900941e-07
## 20 20 8.900941e-07 4.450485e-07
```

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

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

  geom_line(size=1) +

  theme_classic()
```



Pela matriz de coeficiente ter um autovalor negativo, espera-se obervar oscilações. Ainda, como os dois autovalores são menores que a unidade em módulo, espera-se uma trajetória amortecida para o conjunto de séries do sistema.

Lembre do efeito da condição inicial e da janela de tempo considerada na simulação. Obseve também que a raíz negativa possui um módulo pequeno (-0.25), que é atenuado rapidamente a medida que t aumenta.

Analisando cada componente podemos ter uma melhor precepção deste caso.

```
tmax = 20
Z1 = matrix(0, nrow=2, ncol=tmax)
Z2 = matrix(0, nrow=2, ncol=tmax)
A1 = .1
A2 = -.1

v1 = P[, 1]
v2 = P[, 2]

for (t in 1:tmax){

  Z1[,t] = A1*v1*lambda[1]^t
  Z2[,t] = A2*v2*lambda[2]^t


}


t = seq(1, tmax, 1)
```

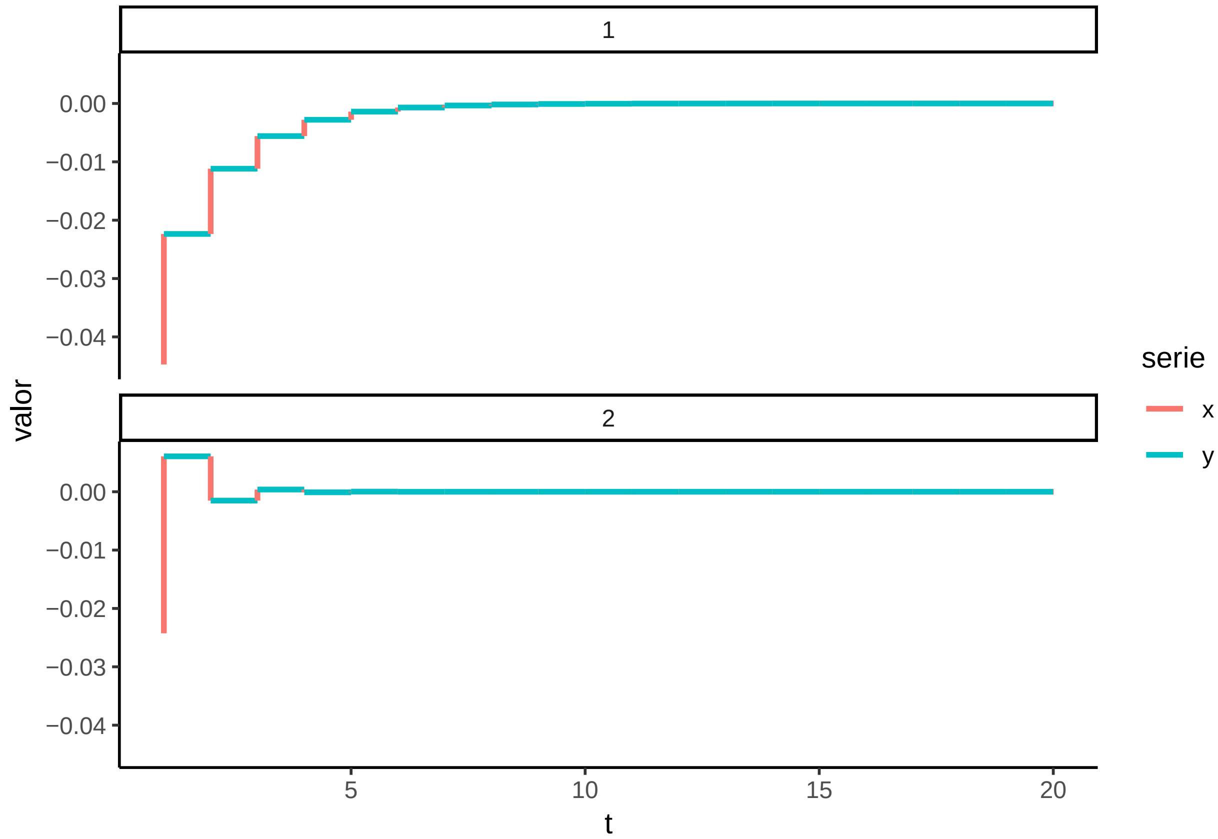
```r
series1 = data.frame(t, x = Z1[1, ], y = Z1[2, ], p = rep(1, length(t)), stringsAsFactors = FALSE  )
head( series1 )
```

```
##   t            x             y p
## 1 1 -0.044721360 -0.0223606798 1
## 2 2 -0.022360680 -0.0111803399 1
## 3 3 -0.011180340 -0.0055901699 1
## 4 4 -0.005590170 -0.0027950850 1
## 5 5 -0.002795085 -0.0013975425 1
## 6 6 -0.001397542 -0.0006987712 1
```

```r
series2 = data.frame(t, x = Z2[1, ], y = Z2[2, ], p = rep(2, length(t)), stringsAsFactors = FALSE   )
head( series2 )
```

```
##   t             x             y p
## 1 1 -2.425356e-02  6.063391e-03 2
## 2 2  6.063391e-03 -1.515848e-03 2
## 3 3 -1.515848e-03  3.789619e-04 2
## 4 4  3.789619e-04 -9.474048e-05 2
## 5 5 -9.474048e-05  2.368512e-05 2
## 6 6  2.368512e-05 -5.921280e-06 2
```

```r
series = bind_rows(series1, series2)


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

ggplot(series_tidy, aes(x=t, y=valor, color=serie, group=p)) +

  geom_line(size=1) +

  facet_wrap(~p, nrow=2) +

  theme_classic()
```



Simulando um sistema com raízes reais e iguais

```
library(limSolve)
```

```
## Warning: package 'limSolve' was built under R version 3.4.4
```

```
##
## Attaching package: 'limSolve'
```

```
## The following object is masked from 'package:ggplot2':
##
##     resolution
```

```
## The following object is masked from 'package:matlib':
##
##     Solve
```

```
A <- matrix(c(4, -1, 1, 2), nrow=2)


r <- eigen(A)

lambda <- r$values
lambda
```

```
## [1] 3 3
```

```r
P <- r$vectors
P
```

```
##            [,1]       [,2]
## [1,]  0.7071068 -0.7071068
## [2,] -0.7071068  0.7071068
```

```r
det(P)
```

```
## [1] 3.140185e-16
```

```r
v1 <- P[,1]

#D <- zapsmall(ginv(P)%*%A%*%P )  # note que P deve ser inversivel
#D

# (A-lambda*I)^2*v2 = v1
# precisamos apenas uma solucao possivel
M <- A - diag(lambda[1], 2)
M
```

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

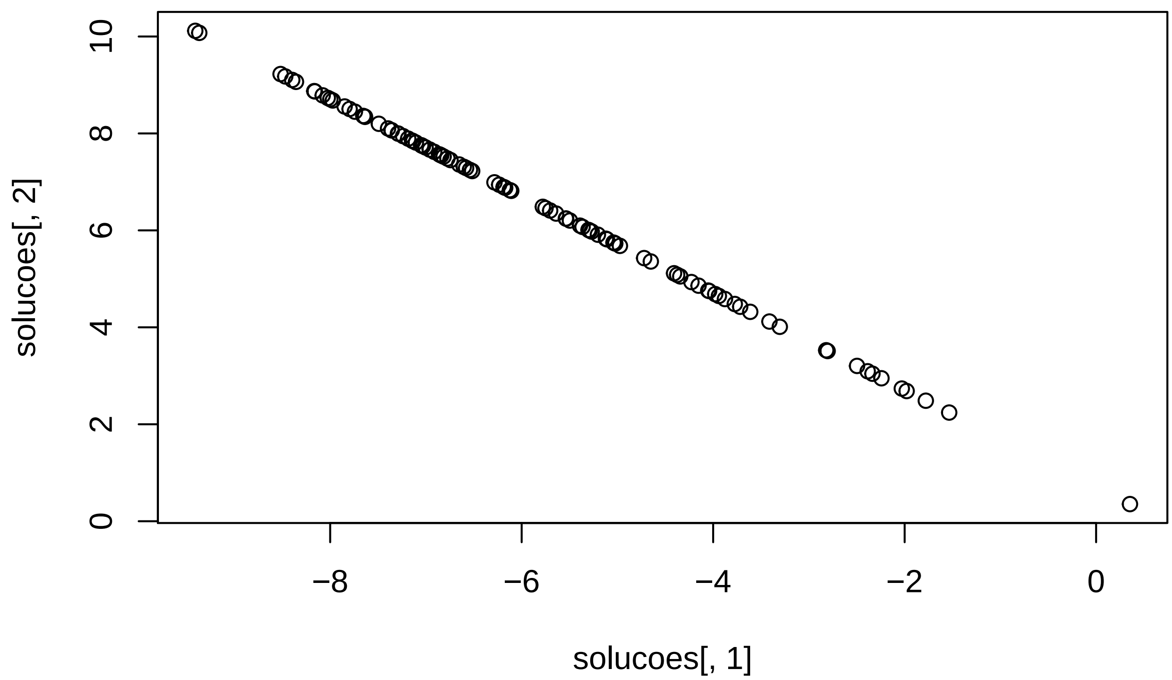
```r
B = matrix( v1, nrow=2)
solucoes = xsample(E=M, F= B, iter=100, jmp=1)$X
solucoes
```

```
##              [,1]      [,2]
##   [1,]  0.3535534 0.3535534
##   [2,] -1.7787690 2.4858757
##   [3,] -1.9790724 2.6861792
##   [4,] -2.8194723 3.5265790
##   [5,] -2.3860098 3.0931166
##   [6,] -1.5346106 2.2417174
##   [7,] -2.0295485 2.7366553
##   [8,] -2.2420472 2.9491539
##   [9,] -4.0419215 4.7490283
##  [10,] -3.7742980 4.4814048
##  [11,] -3.7175039 4.4246106
##  [12,] -5.3639649 6.0710717
##  [13,] -4.9737395 5.6808462
##  [14,] -5.7023758 6.4094826
##  [15,] -5.0222642 5.7293710
##  [16,] -5.0332901 5.7403968
##  [17,] -5.2885526 5.9956593
##  [18,] -5.3903416 6.0974483
##  [19,] -4.3436087 5.0507155
##  [20,] -3.4123638 4.1194706
##  [21,] -3.9403041 4.6474108
##  [22,] -5.6400031 6.3471099
##  [23,] -5.1091437 5.8162505
##  [24,] -5.1205241 5.8276309
##  [25,] -5.7507858 6.4578926
##  [26,] -6.1087231 6.8158299
```

```
##   [27,] -7.0306952  7.7378020
##   [28,] -9.3673860 10.0744928
##   [29,] -9.4091714 10.1162782
##   [30,] -8.4690604  9.1761671
##   [31,] -8.3962383  9.1033451
##   [32,] -8.1662292  8.8733360
##   [33,] -7.3604776  8.0675843
##   [34,] -8.0780281  8.7851349
##   [35,] -7.2868584  7.9939652
##   [36,] -7.6387368  8.3458436
##   [37,] -7.3953199  8.1024267
##   [38,] -7.4917476  8.1988544
##   [39,] -8.0253222  8.7324290
##   [40,] -7.8492561  8.5563629
##   [41,] -6.8151282  7.5222350
##   [42,] -7.1047342  7.8118410
##   [43,] -6.2349494  6.9420561
##   [44,] -5.2039082  5.9110150
##   [45,] -5.5383250  6.2454318
##   [46,] -5.4963944  6.2035012
##   [47,] -6.8409049  7.5480117
##   [48,] -7.1449246  7.8520314
##   [49,] -6.7757545  7.4828613
##   [50,] -5.7792701  6.4863769
##   [51,] -6.1234392  6.8305459
##   [52,] -7.0495902  7.7566970
##   [53,] -6.5404329  7.2475397
##   [54,] -6.9134106  7.6205174
##   [55,] -6.5802032  7.2873100
##   [56,] -6.6051075  7.3122143
##   [57,] -6.6525246  7.3596314
##   [58,] -6.8396792  7.5467860
##   [59,] -6.7433165  7.4504233
##   [60,] -7.6538430  8.3609498
##   [61,] -7.9983607  8.7054675
##   [62,] -7.3566768  8.0637836
##   [63,] -7.1328981  7.8400048
##   [64,] -7.2922828  7.9993896
##   [65,] -6.8620898  7.5691966
##   [66,] -7.1855342  7.8926410
##   [67,] -7.2377329  7.9448396
##   [68,] -6.1721327  6.8792395
##   [69,] -6.1887423  6.8958491
##   [70,] -5.0409806  5.7480873
##   [71,] -4.2268951  4.9340018
##   [72,] -4.1526020  4.8597087
##   [73,] -3.6127319  4.3198387
##   [74,] -4.6510511  5.3581579
##   [75,] -3.8766422  4.5837490
##   [76,] -3.9784107  4.6855175
##   [77,] -2.4970494  3.2041562
##   [78,] -2.8043338  3.5114406
##   [79,] -2.3362612  3.0433680
##   [80,] -3.3039462  4.0110529
```

```
##  [81,] -4.0512125  4.7583193
##  [82,] -4.4082542  5.1153610
##  [83,] -4.3759919  5.0830986
##  [84,] -4.7208347  5.4279415
##  [85,] -5.3021643  6.0092711
##  [86,] -6.5167660  7.2238728
##  [87,] -7.7415933  8.4487001
##  [88,] -7.6360095  8.3431163
##  [89,] -7.0034310  7.7105378
##  [90,] -8.1603688  8.8674756
##  [91,] -7.7972176  8.5043244
##  [92,] -8.5182396  9.2253464
##  [93,] -8.3573985  9.0645053
##  [94,] -7.9727926  8.6798994
##  [95,] -6.8539220  7.5610287
##  [96,] -7.0437909  7.7508977
##  [97,] -6.9599700  7.6670768
##  [98,] -6.2842079  6.9913147
##  [99,] -5.5347707  6.2418775
## [100,] -5.2684297  5.9755365
```

```
plot(solucoes[,1], solucoes[,2])
```




```
c2 = matrix( solucoes[1,], nrow=2 )

tmax = 10
Z = matrix(0,2, tmax)
```

```r
Z[, 1] = c(2,-1)

for (t in 2:tmax){

        Z[,t] = A %*% Z[, t-1]

}

t(Z)
```

```
##        [,1]   [,2]
##  [1,]     2     -1
##  [2,]     7     -4
##  [3,]    24    -15
##  [4,]    81    -54
##  [5,]   270   -189
##  [6,]   891   -648
##  [7,]  2916  -2187
##  [8,]  9477  -7290
##  [9,] 30618 -24057
## [10,] 98415 -78732
```

Modelo de Cournot

```r
set.seed(1)

n = 2
a = 10
b = .5
d = 10
c = runif(n)

A = matrix(-1/2, nrow = n, ncol=n) + diag(1/2, n)
A
```

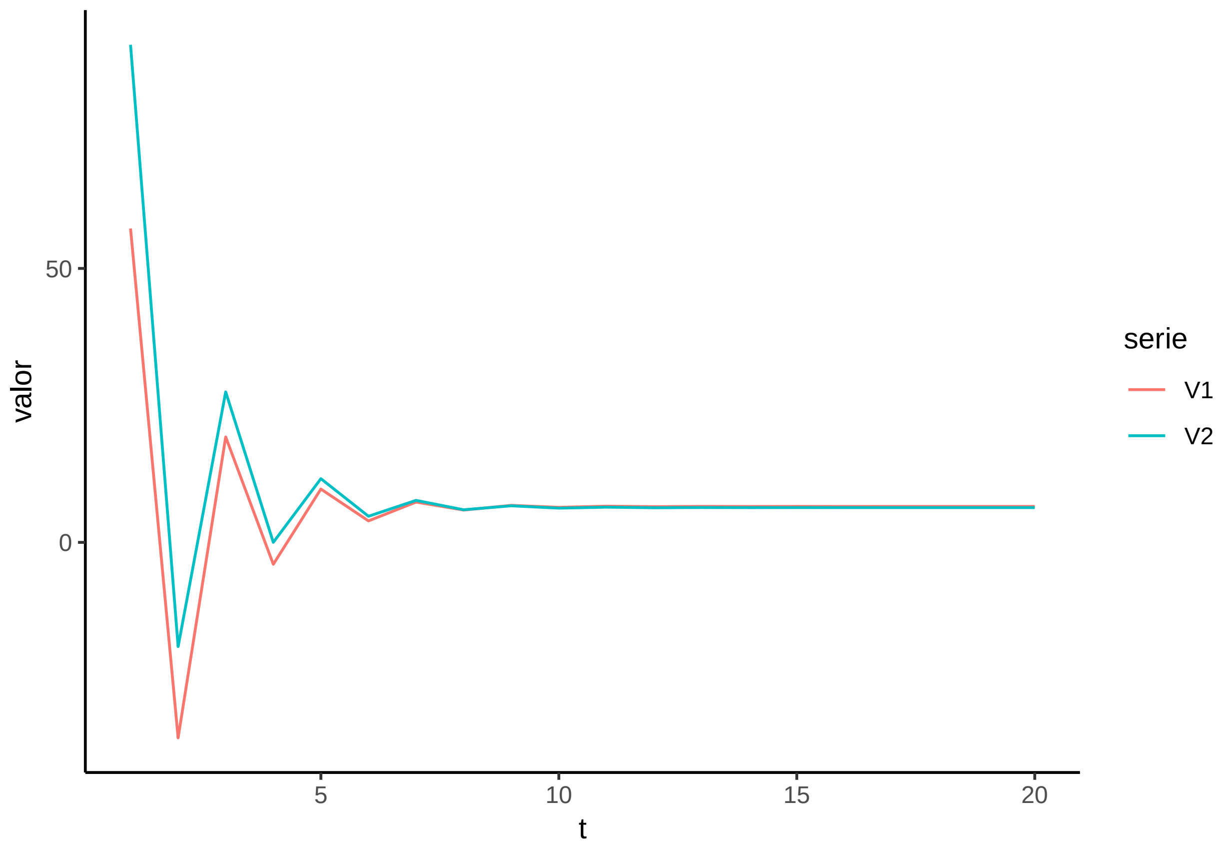
```
##      [,1] [,2]
## [1,]  0.0 -0.5
## [2,] -0.5  0.0
```

```r
B = (a-c)/(2*b)

tmax = 20

X = matrix(0, nrow = n, ncol = tmax)
X[, 1] = 100*runif(n)  # condicoes iniciais para cada uma das n firmas

# simulando

for (t in 2:tmax){
  X[, t] = A %*% X[ , t-1] + B

}
```

```r
t(X)
```

```
##             [,1]         [,2]
##  [1,]  57.285336  90.820778999
##  [2,] -35.675898 -19.014792067
##  [3,]  19.241887  27.465825182
##  [4,]  -3.998421   0.006932415
##  [5,]   9.731025  11.627086727
##  [6,]   3.920948   4.762363536
##  [7,]   7.353310   7.667402114
##  [8,]   5.900790   5.951221316
##  [9,]   6.758881   6.677480960
## [10,]   6.395751   6.248435761
## [11,]   6.610273   6.430000672
## [12,]   6.519491   6.322739372
## [13,]   6.573122   6.368130600
## [14,]   6.550426   6.341315275
## [15,]   6.563834   6.352663082
## [16,]   6.558160   6.345959251
## [17,]   6.561512   6.348796202
## [18,]   6.560093   6.347120245
## [19,]   6.560931   6.347829483
## [20,]   6.560577   6.347410493
```

```r
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_classic()
```




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

```
##          [,1]
## [1,] 6.560738
## [2,] 6.347507
```

```
# verificando os autovalores

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

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

Sites interessantes

- Ginv