Programa de Doutorado em Economia - EPGE/FGV *MDPTEC070 - Microeconomia Empírica*

Lista Empírica #1 - Q1

2 de maio de 2022

Aluno: Rafael Vetromille *Professor: Luiz Braido / TA: Dante Pimentel*

Observação

Para a primeira lista o professor pediu que fizessemos o código em duas linguagens de programação diferentes. A linguagem de programação principal escolhida será o Matlab, de modo a atender ao requisito do professor segue todos os códigos em `R` também. Os valores não serão exatamente os mesmos pois a função de simulação do AR(1) envolve a geração de números aleatórios.

Questão #1

Discretize o processo acima usando o método de Tauchen (1986). Use 9 pontos.

Resposta. O código a seguir propõe uma solução para o exercício.

```r
library(magrittr)

# Exercise 1 --------------------------------------------------------------

## Calibration
rho = 0.95
sig = 0.007

N = 9         # Number of points (states)
m = 3         # Scaling parameter (I don't know why!)

t = 10000

## Grid and Transition matrix function of Tauchen's Method

tauchen = function(N, sig, rho, m) {

  # Grid
  grid = rep(0, N)
  grid[1] = - m * sqrt(sig^2/(1-rho^2))
  grid[N] = + m * sqrt(sig^2/(1-rho^2))
  step = (grid[N] - grid[1]) / (N - 1)
  for (i in 2:(N-1)) {
    grid[i] = grid[i-1] + step
  }

  # Transition matrix
  if (N > 1) {
    step = grid[2] - grid[1]
    P = array(0, dim = c(N, N))
    for (j in 1:N) {
      for (k in 1:N) {
        if (k == 1) {
```

```
        P[j, k] = pnorm((grid[k] - rho * grid[j] + (step / 2)) / sig)
      }
      else if (k == N) {
        P[j, k] = 1 - pnorm((grid[k] - rho * grid[j] - (step / 2)) / sig)
      }
      else {
        P[j, k] = pnorm((grid[k] - rho * grid[j] + (step / 2)) / sig) -
          pnorm((grid[k] - rho * grid[j] - (step / 2)) / sig)
      }
    }
  }
} else {
  P = 1
}
return(list(zgrid = round(grid, 4), P = round(P, 4)))
}

## Grid and Transition matrix (Tauchen's Method)
Tauchen95 = tauchen(N, sig, rho, m)
Tauchen95
```

```
## $zgrid
## [1] -0.0673 -0.0504 -0.0336 -0.0168  0.0000  0.0168  0.0336  0.0504  0.0673
##
## $P
##         [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]
##  [1,] 0.7644 0.2347 0.0009 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
##  [2,] 0.0592 0.7405 0.1997 0.0006 0.0000 0.0000 0.0000 0.0000 0.0000
##  [3,] 0.0001 0.0747 0.7569 0.1679 0.0004 0.0000 0.0000 0.0000 0.0000
##  [4,] 0.0000 0.0001 0.0931 0.7669 0.1396 0.0002 0.0000 0.0000 0.0000
##  [5,] 0.0000 0.0000 0.0002 0.1147 0.7702 0.1147 0.0002 0.0000 0.0000
##  [6,] 0.0000 0.0000 0.0000 0.0002 0.1396 0.7669 0.0931 0.0001 0.0000
##  [7,] 0.0000 0.0000 0.0000 0.0000 0.0004 0.1679 0.7569 0.0747 0.0001
##  [8,] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0006 0.1997 0.7405 0.0592
##  [9,] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0009 0.2347 0.7644
```

Questão #2

Discretize o processo acima usando o método de Rouwenhorst. Use 9 pontos.

Resposta. O código a seguir propõe uma solução para o exercício.

```
## Grid and Transition matrix function of Rouwenhorst's Method
rouwen <- function(rho, sigma, n){

  zgrid <- seq(
```

```r
    from = - (sig / sqrt(1-rho^2)) * sqrt(N-1),
    to = + (sig / sqrt(1-rho^2)) * sqrt(N-1),
    length = n
  )

  p  <- (rho+1)/2
  nu <- ((n-1)/(1-rho^2))^(1/2) * sigma
  P  <- matrix(c(p, 1 - p, 1 - p, p), nrow = 2, ncol = 2)

  for (i in 3:n){
    zeros <- matrix(0, nrow = i-1, ncol = 1)
    zzeros <- matrix(0, nrow = 1, ncol = i-1)

    P        <- p * rbind(cbind(P, zeros), cbind(zzeros, 0)) +
              (1-p) * rbind(cbind(zeros, P), cbind(0, zzeros)) +
              (1-p) * rbind(cbind(zzeros, 0), cbind(P, zeros)) +
               p * rbind(cbind(0, zzeros), cbind(zeros, P))
  }

  for (j in 1:N){
    P[j,] = P[j,]/sum(P[j,])
  }

  return(list(zgrid = round(zgrid, 4), P = round(P, 4)))
}

## Grid and Transition matrix (Rouwenhorst's Method)
Rouwen95 <- rouwen(rho, sig, N)
Rouwen95
```

```
## $zgrid
## [1] -0.0634 -0.0476 -0.0317 -0.0159  0.0000  0.0159  0.0317  0.0476  0.0634
##
## $P
##          [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]
##  [1,] 0.8167 0.1675 0.0150 0.0008 0.0000 0.0000 0.0000 0.0000 0.0000
##  [2,] 0.0209 0.8204 0.1469 0.0113 0.0005 0.0000 0.0000 0.0000 0.0000
##  [3,] 0.0005 0.0420 0.8231 0.1261 0.0081 0.0003 0.0000 0.0000 0.0000
##  [4,] 0.0000 0.0016 0.0630 0.8247 0.1051 0.0054 0.0001 0.0000 0.0000
##  [5,] 0.0000 0.0001 0.0032 0.0841 0.8253 0.0841 0.0032 0.0001 0.0000
##  [6,] 0.0000 0.0000 0.0001 0.0054 0.1051 0.8247 0.0630 0.0016 0.0000
##  [7,] 0.0000 0.0000 0.0000 0.0003 0.0081 0.1261 0.8231 0.0420 0.0005
##  [8,] 0.0000 0.0000 0.0000 0.0000 0.0005 0.0113 0.1469 0.8204 0.0209
##  [9,] 0.0000 0.0000 0.0000 0.0000 0.0000 0.0008 0.0150 0.1675 0.8167
```

Questão #3

Simule o processo contínuo para 10000 períodos. Faça o mesmo para os processos discretizados (lembre-se de usar as mesmas realizações para os choques). Compare os caminhos para cada processo (gráficos serão úteis aqui). Se eles não estiverem muito próximos, utilize mais pontos.

Resposta. O código a seguir propõe uma solução para o exercício.

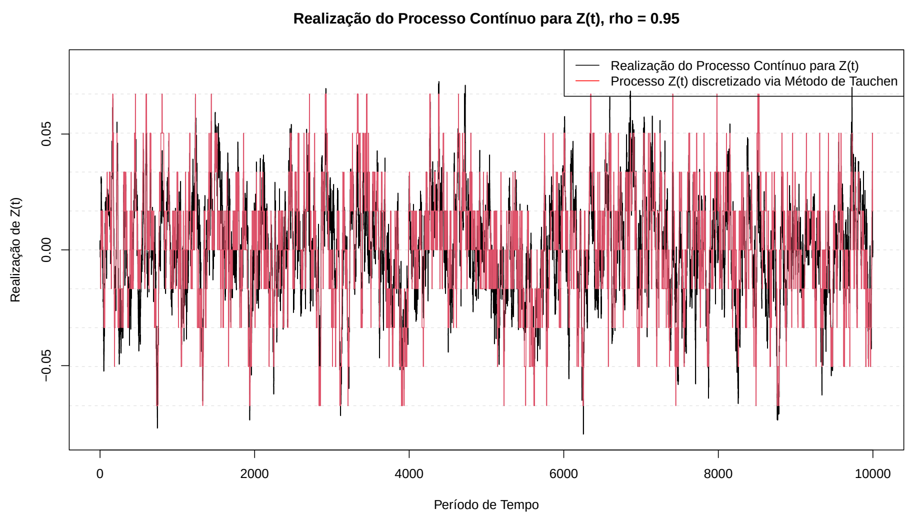
```r
## Function of a AR(1) continuous process
ar1_simulation <- function(rho, sig, t){

  Z = rep(0, t); Z[1] = 0

  eps = rnorm(t, mean = 0, sd = sig)

  for(i in 2:t){
    Z[i] = rho*Z[i-1] + eps[i]
  }

  return(list(Z = round(Z, 4), eps = round(eps, 4)))
}

## Simulation of the process
## (rho = 0.95, mu = 0, sig = 0.007, t = 10000, Z[1] = 0)
set.seed(1347)
ar1_95 <- ar1_simulation(rho, sig, t)

## Show the first 10 numbers
ar1_95 %>% purrr::map(.f = ~ head(.x, 10))
```

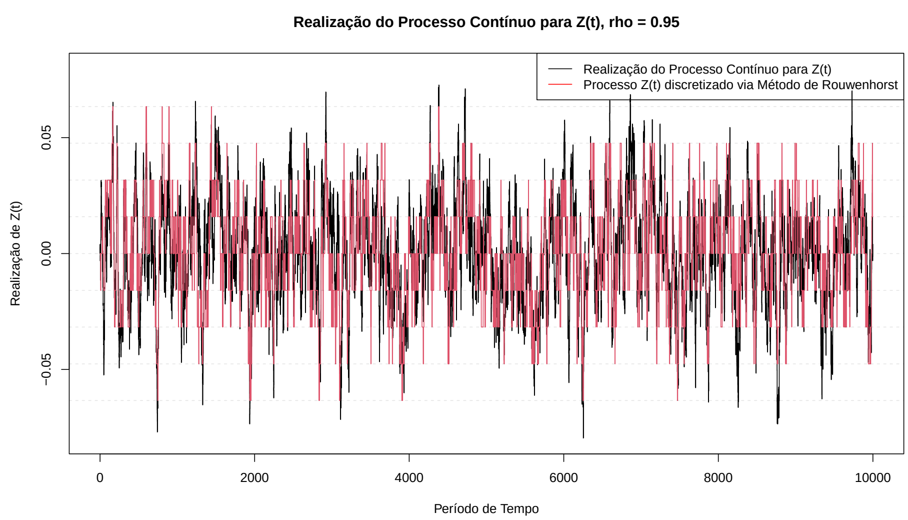
```
## $Z
##  [1]  0.0000  0.0040 -0.0032 -0.0009  0.0027 -0.0010 -0.0115 -0.0076 -0.0023
## [10]  0.0061
##
## $eps
##  [1]  0.0036  0.0040 -0.0070  0.0022  0.0035 -0.0035 -0.0106  0.0034  0.0049
## [10]  0.0083
```

```r
## Function for discretize the process
discret <- function(th0, sig, eps, P, t){

  idx = rep(1, t)
  idx[1] = th0
  cum = t(apply(P, 1, cumsum))

  for(i in 2:t){
    x = which(pnorm(eps[i], mean = 0, sd = sig) <= cum[idx[i-1],])
```

```
    idx[i] = x[1]
  }

  return(idx)
}

## Tauchen's Method, rho = 0.95 ####

## Defining the initial state
th0 <- which(Tauchen95$zgrid == median(Tauchen95$zgrid))

## Returning the indices of the grid
idx = discret(th0, sig, ar1_95$eps, Tauchen95$P, t)

## Simulation the discretized process
ztauchen95 <- Tauchen95$zgrid[idx]

## Plotting
par(mfrow=c(2,1))

plot(ar1_95$Z, type = 'S', col = 1,
     main = "Realização do Processo Contínuo para Z(t), rho = 0.95",
     xlab = "Período de Tempo", ylab = "Realização de Z(t)",
     ylim = c(-0.08, 0.08))
lines(ztauchen95, col = 2)
abline(h = Tauchen95$zgrid, col = scales::alpha('black', 0.1), lty = 2)
legend("topright",
  legend = c("Realização do Processo Contínuo para Z(t)",
             "Processo Z(t) discretizado via Método de Tauchen"),
  col = c("black", "red"),
  lty = c(1, 1)
)

## Rouwenhorst's Method, rho = 0.95 ####

## Defining the initial state
th0 <- which(Rouwen95$zgrid == median(Rouwen95$zgrid))

## Returning the indices of the grid
idx = discret(th0, sig, ar1_95$eps, Rouwen95$P, t)

## Simulation the discretized process
zrouwen95 <- Rouwen95$zgrid[idx]

## Plotting
```

```r
plot(ar1_95$Z, type = 'S', col = 1,
     main = "Realização do Processo Contínuo para Z(t), rho = 0.95",
     xlab = "Período de Tempo", ylab = "Realização de Z(t)",
     ylim = c(-0.08, 0.08))
lines(zrouwen95, col = 2)
abline(h = Rouwen95$zgrid, col = scales::alpha('black', 0.1), lty = 2)
legend("topright",
       legend = c("Realização do Processo Contínuo para Z(t)",
                  "Processo Z(t) discretizado via Método de Rouwenhorst"),
       col = c("black", "red"),
       lty = c(1, 1)
)
```





Questão #4

Estime processos AR(1) com base nos dados simulados, tanto a partir do Tauchen quanto o de Rouwenhorst. Quão próximo eles estão do processo gerador de dados real? Se eles não estiverem muito próximos, utilize mais pontos.

Resposta. O código a seguir propõe uma solução para o exercício.

```r
## Compute the lag (Tauchen's method)
ztauchen95_lag1 <- dplyr::lag(ztauchen95, 1)

## Run the regression and show the results
lm_tauchen95 <- lm(ztauchen95 ~ 0 + ztauchen95_lag1); broom::tidy(lm_tauchen95)
```

```
## # A tibble: 1 x 5
##   term            estimate std.error statistic p.value
##   <chr>              <dbl>     <dbl>     <dbl>   <dbl>
## 1 ztauchen95_lag1    0.946   0.00324      292.       0
```

```r
## Hyphotesis test (H0: rho = 0.95 vs H1: rho \neq 0.95)
tauchen95_tstat <- (lm_tauchen95$coefficients - 0.95)/sqrt(diag(vcov(lm_tauchen95)))
tauchen95_tstat
```

```
## ztauchen95_lag1
##       -1.191776
```

```r
## Confidence interval (if inside, we do not reject the null)
qt(c(.025, .975), df = lm_tauchen95$df.residual)
```

```
## [1] -1.960201  1.960201
```

```r
## Compute the lag (Tauchen's method)
zrouwen95_lag1 <- dplyr::lag(zrouwen95, 1)

## Run the regression and show the results
lm_rouwen95 <- lm(zrouwen95 ~ 0 + zrouwen95_lag1); broom::tidy(lm_rouwen95)
```

```
## # A tibble: 1 x 5
##   term           estimate std.error statistic p.value
##   <chr>             <dbl>     <dbl>     <dbl>   <dbl>
## 1 zrouwen95_lag1    0.951   0.00310      307.       0
```

```r
## Hyphotesis test (H0: rho = 0.95 vs H1: rho \neq 0.95)
rouwen95_tstat <- (lm_rouwen95$coefficients - 0.95)/sqrt(diag(vcov(lm_rouwen95)))
rouwen95_tstat
```

```
## zrouwen95_lag1
##      0.2725319
```

```r
## Confidence interval (if inside, we do not reject the null)
qt(c(.025, .975), df = lm_rouwen95$df.residual)
```

```
## [1] -1.960201  1.960201
```

Questão #5

Refaça os exercícios acima quando $\rho = 0.99$.

Resposta. O código a seguir propõe uma solução para o exercício.

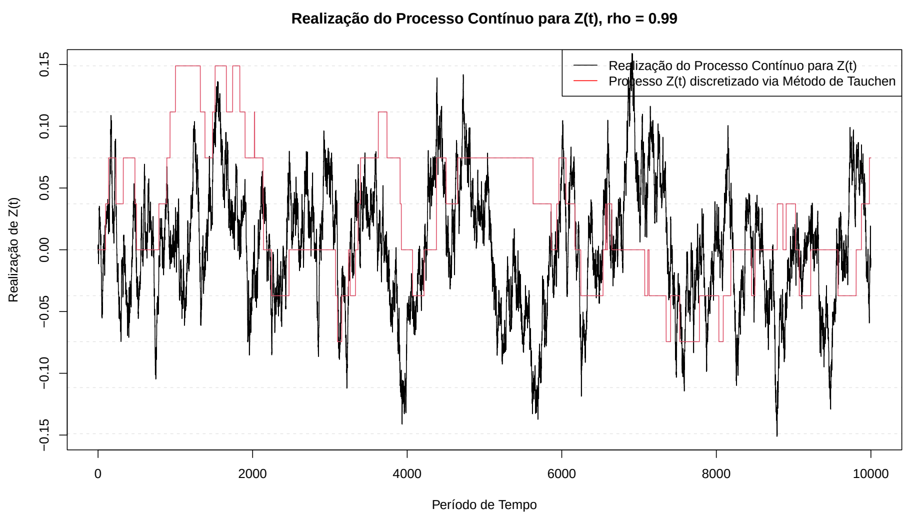
```
## Recalibration
rho = 0.99
sig = 0.007

N = 9         # Number of points (states);
m = 3         # Scaling parameter (I don't know why!);

T = 10000

## 5.1 Tauchen's Method ####
Tauchen99 <- tauchen(N, sig, rho, m)

## 5.2 Rouwenhorst's Method ####
Rouwen99 <- rouwen(rho, sig, N)

## 5.3 Process simulation and discretize ####

## Simulation of the process
## (rho = 0.99, mu = 0, sig = 0.007, T = 10000, Z[1] = 0)
set.seed(1347)
ar1_99 <- ar1_simulation(rho, sig, T)

## Show the first 10 numbers
ar1_99 %>% purrr::map(.f = ~ head(.x, 10))
```

```
## $Z
##  [1]  0.0000  0.0040 -0.0031 -0.0009  0.0027 -0.0009 -0.0114 -0.0080 -0.0030
## [10]  0.0053
##
## $eps
##  [1]  0.0036  0.0040 -0.0070  0.0022  0.0035 -0.0035 -0.0106  0.0034  0.0049
## [10]  0.0083
```

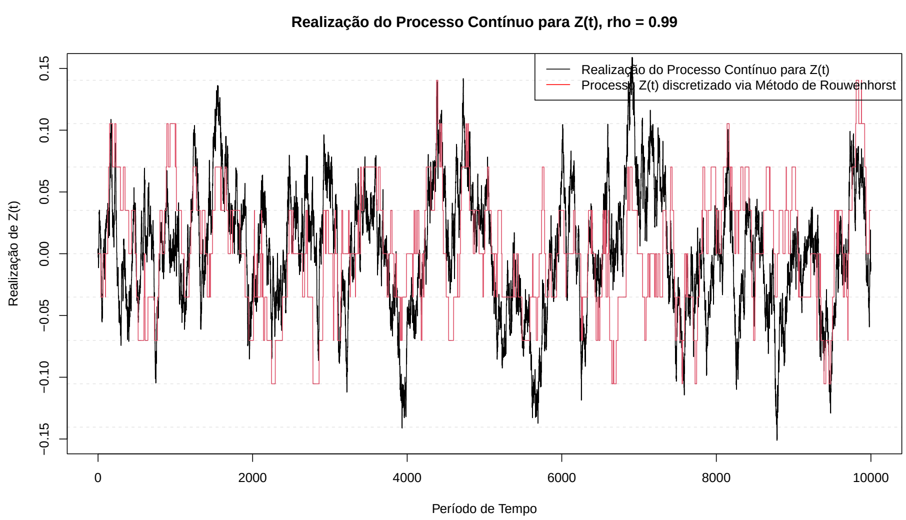
```
## Tauchen's Method, rho = 0.99 ####

## Defining the initial state
th0 <- which(Tauchen99$zgrid == median(Tauchen99$zgrid))

## Returning the indices of the grid
idx = discret(th0, sig, ar1_99$eps, Tauchen99$P, T)
```

```r
## Simulation the discretized process
ztauchen99 <- Tauchen99$zgrid[idx]

## Plotting
par(mfrow = c(2, 1))

plot(ar1_99$Z, type = 'S', col = 1,
     main = "Realização do Processo Contínuo para Z(t), rho = 0.99",
     xlab = "Período de Tempo", ylab = "Realização de Z(t)",
     ylim = c(-0.15, 0.15))
lines(ztauchen99, col = 2)
abline(h = Tauchen99$zgrid, col = scales::alpha('black', 0.1), lty = 2)
legend("topright",
       legend = c("Realização do Processo Contínuo para Z(t)",
                  "Processo Z(t) discretizado via Método de Tauchen"),
       col = c("black", "red"),
       lty = c(1, 1)
)

## Rouwenhorst's Method, rho = 0.99 ####

## Defining the initial state
th0 <- which(Rouwen99$zgrid == median(Rouwen99$zgrid))

## Returning the indices of the grid
idx = discret(th0, sig, ar1_99$eps, Rouwen99$P, T)

## Simulation the discretized process
zrouwen99 <- Rouwen99$zgrid[idx]

## Plotting
plot(ar1_99$Z, type = 'S', col = 1,
     main = "Realização do Processo Contínuo para Z(t), rho = 0.99",
     xlab = "Período de Tempo", ylab = "Realização de Z(t)",
     ylim = c(-0.15, 0.15))
lines(zrouwen99, col = 2)
abline(h = Rouwen99$zgrid, col = scales::alpha('black', 0.1), lty = 2)
legend("topright",
       legend = c("Realização do Processo Contínuo para Z(t)",
                  "Processo Z(t) discretizado via Método de Rouwenhorst"),
       col = c("black", "red"),
       lty = c(1, 1), text.font = 1)
```

```r
## 5.4 Regressions ####
```



Realização do Processo Contínuo para Z(t), rho = 0.99
Realização do Processo Contínuo para Z(t)
Processo Z(t) discretizado via Método de Tauchen
Realização de Z(t)
-0.15
-0.10
-0.05
0.00
0.05
0.10
0.15
0
2000
4000
6000
8000
10000
Período de Tempo



Realização do Processo Contínuo para Z(t), rho = 0.99
Realização do Processo Contínuo para Z(t)
Processo Z(t) discretizado via Método de Rouwenhorst
Realização de Z(t)
-0.15
-0.10
-0.05
0.00
0.05
0.10
0.15
0
2000
4000
6000
8000
10000
Período de Tempo

```r
## Compute the lag (Tauchen's method)
ztauchen99_lag1 <- dplyr::lag(ztauchen99, 1)

## Run the regression and show the results
lm_tauchen99 <- lm(ztauchen99 ~ 0 + ztauchen99_lag1)
broom::tidy(lm_tauchen99)
```

```
## # A tibble: 1 x 5
##   term            estimate std.error statistic p.value
##   <chr>              <dbl>     <dbl>     <dbl>   <dbl>
## 1 ztauchen99_lag1    0.999  0.000534     1870.       0
```

```r
## Hyphotesis test (H0: rho = 0.99 vs H1: rho \neq 0.99)
tauchen99_tstat <- (lm_tauchen99$coefficients - 0.99)/sqrt(diag(vcov(lm_tauchen99)))
tauchen99_tstat %>% as.vector()
```

```
## [1] 16.19571
```

```r
## Confidence interval (if inside, we do not reject the null)
qt(c(.025, .975), df = lm_tauchen99$df.residual)
```

```
## [1] -1.960201  1.960201
```

```r
## Compute the lag (Tauchen's method)
zrouwen99_lag1 <- dplyr::lag(zrouwen99, 1)

## Run the regression and show the results
lm_rouwen99 <- lm(zrouwen99 ~ 0 + zrouwen99_lag1)
broom::tidy(lm_rouwen99)
```

```
## # A tibble: 1 x 5
##   term           estimate std.error statistic p.value
##   <chr>             <dbl>     <dbl>     <dbl>   <dbl>
## 1 zrouwen99_lag1    0.990   0.00142      695.       0
```

```r
## Hyphotesis test (H0: rho = 0.99 vs H1: rho \neq 0.99)
rouwen99_tstat <- (lm_rouwen99$coefficients - 0.99)/sqrt(diag(vcov(lm_rouwen99)))
rouwen99_tstat %>% as.vector()
```

```
## [1] -0.1108278
```

```r
## Confidence interval (if inside, we do not reject the null)
qt(c(.025, .975), df = lm_rouwen99$df.residual)
```

```
## [1] -1.960201  1.960201
```