Finanças Quantitativas: Lista 6

Lucas Moschen

07 de junho de 2020

Exercício 6.6

(i) $X_t - X_{t-1} = W_t - 1.5W_{t-1}$

(ii) $X_t - 0.8X_{t-1} = W_t - 0.5W_{t-1}$

$\{W_t\}_t \sim N(0, \sigma^2)$

1.

i) Temos que $\phi_1 = 1, \theta_1 = -1.5$

Modelo: $(1 - B)X_t = (1 - 1.5B)W_t$

$\phi(B) = 1 - B$

$\theta(B) = 1 - 1.5B$

ii) Temos que $\phi_1 = 0.8, \theta_1 = -0.5$

Modelo: $(1 - 0.8B)X_t = (1 - 0.5B)W_t$

$\phi(B) = 1 - 0.8B$

$\theta(B) = 1 - 0.5B$

2. Sabemos que em um modelo ARMA, a estacionaridade depende apenas do modelo AR e a invertibilidade depende apenas do modelo MA. Assim, se as raízes de $\phi(z)$ tem módulo maior do que 1, a série temporal é dita estacionária e se as raízes de $\theta(z)$ tem módulo maior do que 1, X é invertível. Consideremos cada modelo:

i) Temos que $\phi(B) = 0 \implies B = 1$ e $\theta(B) = 0 \implies B = 2/3$. Desta maneira, dada a explicação acima, vemos que o modelo não é estacionário, mas é invertível.

ii) Temos que $\phi(B) = 0 \implies B = 5/4$ e $\theta(B) = 0 \implies B = 2$. Desta maneira, dada a explicação acima, vemos que o modelo não é estacionário e não é invertível.

Exercício 6.11

Assumimos que $\{X_t\}_t$ é da forma $X_t = \phi_1 X_{t-1} + W_t$, para algum ruído branco forte de variância desconhecida σ^2.

1. Amostra da distribuição normal padrão e amostra para a série temporal.

```
set.seed(200)
N <- 5000
x <- w <- rnorm(n = N)
for (t in 2:N){
```

```
  x[t] <- x[t - 1] + w[t]
}
```

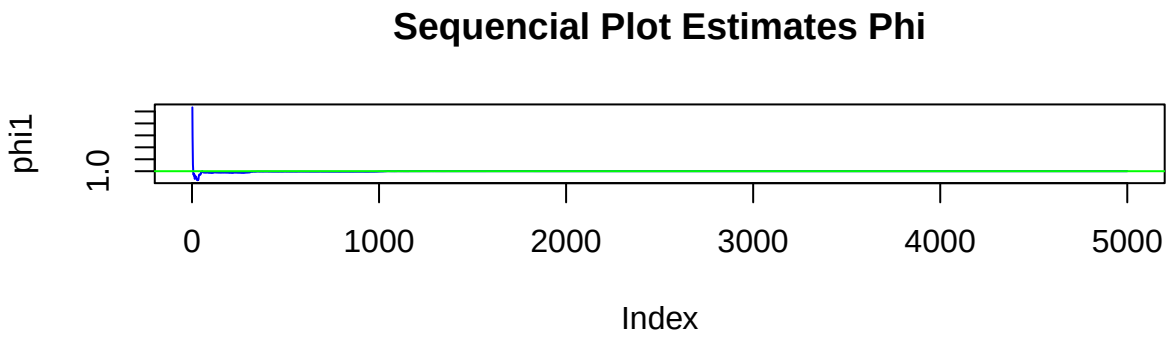
2.

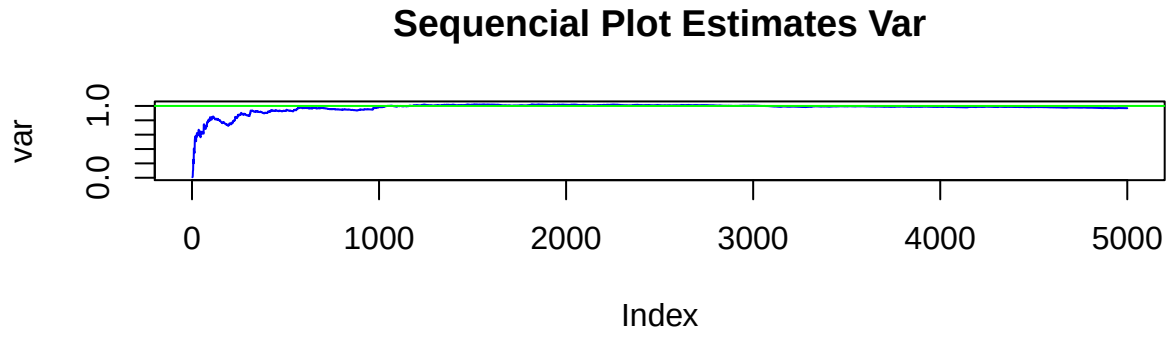
Primeiro declaremos as funções:

```
phi1_hat <- function(x){
  num <- sum(x[2:length(x)]*x[1:length(x)-1])
  den <- sum(x[1:length(x)-1]^2)
  return(num/den)
}
var_hat <- function(x,phi){
  num <- x[1]**2
  num <- num + sum((x[2:length(x)] - phi*x[1:length(x) - 1])^2)
  return(num/(length(x) - 1))
}
DF_test <- function(phi, var){
  return((phi - 1)/sqrt(var))
}
```

Agora, calculemos para cada n essas estatísticas:

```
phi1 <- array(NA, dim = N)
var <- array(NA, dim = N)
DF <- array(NA, dim = N)
for(n in 1:N){
  phi1[n] <- phi1_hat(x[1:n])
  var[n] <- var_hat(x[1:n], phi1[n])
  DF[n] <- DF_test(phi1[n], var[n])
}

par(mfrow = c(2,1))
plot(phi1, main = "Sequencial Plot Estimates Phi", type = 'l', col = 'blue')
abline(h = 1, col = 'green')
plot(var, main = "Sequencial Plot Estimates Var", type = 'l', col = 'blue')
abline(h = 1, col = 'green')
```





Observe de fato, a convergência para os valores que damos para cada parâmetro. Indico pela cor verde essa convergência.

3.

Vamos repetir o processo acima NS vezes para produzir amostras independentes.

```
N <- 1000
NS <- 500
XX <- matrix(NA, nrow = N-1, ncol = NS)

set.seed(100)
w <- matrix(rnorm(N*NS), nrow = N, byrow = TRUE)
for(n in 2:N){
  w[n,] <- w[n-1,] + w[n,]
}
for(i in 2:N){
  for(j in 1:NS){
    phi1 <- phi1_hat(w[1:i,j])
    var <- var_hat(w[1:i,j], phi1)
    DF <- DF_test(phi1, var)
    XX[i-1,j] <- DF
  }
}
```

Agora, para cada $n = 100, 110, 120, ..., 1000$, vamos computar os valores dos quantis da amostra de tamanho NS dada pela $n - th$ linha da matriz.

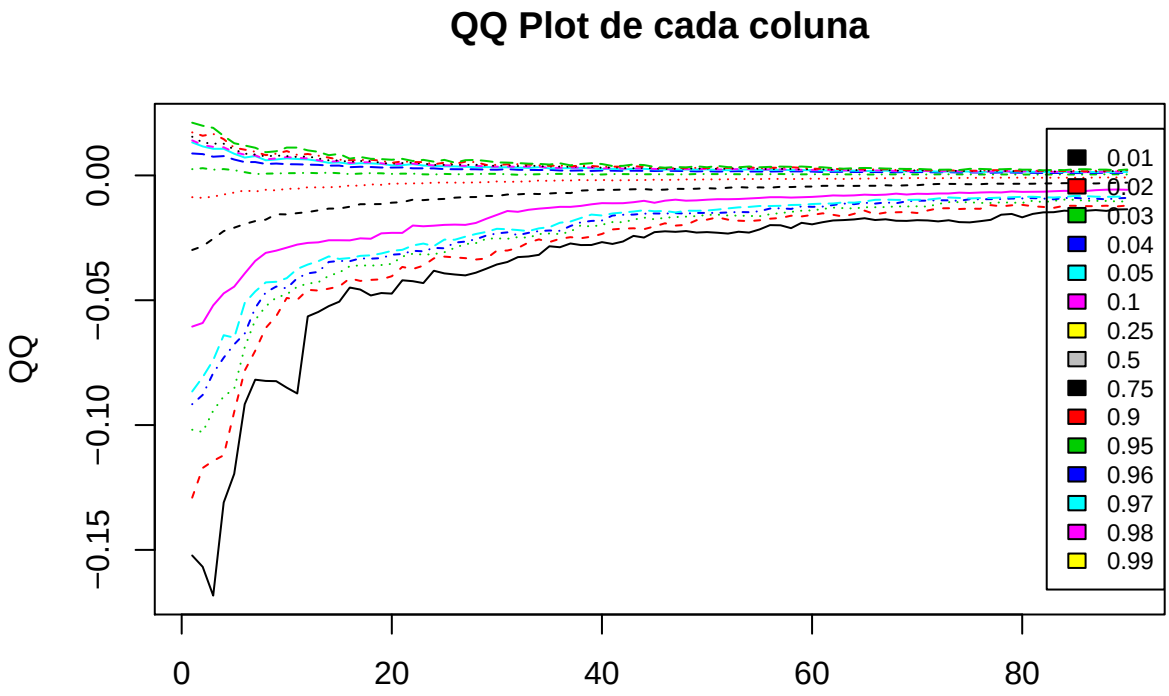
```
q <- c(0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.96, 0.97, 0.98, 0.99)
n_list <- seq(100, N-1,by = 10)

QQ <- matrix(NA, nrow = length(n_list), ncol = length(q))
```

```
for(i in 1:length(n_list)){
  QQ[i,] <- quantile(XX[n_list[i],], probs = q)
}
colnames(QQ) <- q
```

4.

Considere o plot das colunas de QQ. Observe as convergências dos valores.

```
matplot(QQ, type = 'l', main = "QQ Plot de cada coluna")
legend("right", legend = colnames(QQ), col = seq_len(ncol(QQ)), fill = seq_len(ncol(QQ)), cex = 0.75)
```



5.

Consideraremos o data set com os preços diários das ações Calpine para investigar estacionaridade. Os dados são referentes aos anos de 2017 e 2018. Aparentemente neste ano não há mais ações dessa empresa (pelo menos onde eu baixei os dados). Vou primeiro calcular tudo o necessário e depois comentar os resultados.

```
library(tseries)
CPN <- read.csv("../data/CPN.csv")
Price <- rev(CPN$Price)
LRet <- diff(log(Price))

phi1_price <- phi1_hat(Price)
phi1_lret <- phi1_hat(LRet)

var_price <- var_hat(Price, phi1_price)
var_lret <- var_hat(Price, phi1_lret)

DF_price <- DF_test(phi1_price, var_price)
DF_lret <- DF_test(phi1_lret, var_lret)

ADF_price <- adf.test(Price)
ADF_lret <- adf.test(LRet)
```

Agora vejamos a última linha da matriz QQ que será usada como referencial para o p-valor do teste coonstruído no texto.

```
p.value_price <- ecdf(QQ[90,])(DF_price)
p.value_lret <- ecdf(QQ[90,])(DF_lret)
print("------ Prices -------")
```

```
## [1] "------ Prices -------"
```

```
print(DF_price)
```

```
## [1] 0.001139853
```

```
print(p.value_price)
```

```
## [1] 0.6666667
```

```
print("------ Log Return -------")
```

```
## [1] "------ Log Return -------"
```

```
print(DF_lret)
```

```
## [1] -0.07341184
```

```
print(p.value_lret)
```

```
## [1] 0
```

Observe que o p-valor dos preços é bem alto, em torno de 2/3. Enquanto isso, o p-valor do log retorno é bem baixo, na verdade o cálculo do Teste DF é menor do que o menor dos valores calculados pela matriz, o que é um tanto interessante. Desta forma, não é possível rejeitar a hipótese nula de que a série temporal é *random walk* no caso dos Preços, mas a hipótese é rejeitada para 1% no caso do retorno log.

Da mesma forma, usando a versão do teste aumentada:

```
print("------ Prices -------")
```

```
## [1] "------ Prices -------"
```

```
print(ADF_price)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Price
## Dickey-Fuller = -2.4969, Lag order = 6, p-value = 0.3668
## alternative hypothesis: stationary
```

```
print("------ Log Return -------")
```

```
## [1] "------ Log Return -------"
```

```
print(ADF_lret)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  LRet
## Dickey-Fuller = -6.2313, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

Observe que nesse caso, o Log Retorno é dado como estacionário, isto é, a hipótese de que a série de log retornos é um random walk foi rejeitada.

Exercício 6.13

Assumimos que $X_t = \mu + X_{t-1} + W_t, t = 1, 2, ...$, onde μ é chamada de *drift* E $X_0 = x_0$ é conhecida.

1. Mostrarei por Indução.

Caso base $t = 1 : X_1 = \mu + X_0 + W_1 = x_0 + \mu * 1 + W_1$, como queríamos.

Hipótese de Indução: Suponha que valha para t. Assim:

$$X_{t+1} = \mu + X_t + W_{t+1} = \mu + x_0 + \mu t + \sum_{i=1}^{t} W_i + W_{t+1} = x_0 + \mu(t+1) + \sum_{i=1}^{t+1} W_i$$

Logo a identidade é válida por indução, commo queríamos demonstrar.

2. Computemos média, variância e auto-covariância:

$$\mu_X(t) = E[X_t] = x_0 + \mu t + \sum_{i=1}^{t+1} E[W_i] = x_0 + \mu t$$

$$\sigma_X^2(t) = Var(X_t) = \sum_{i=1}^{t} Var[W_i] = t\sigma^2$$

$$\begin{aligned}\gamma_X(s,t) = Cov(X_s, X_t) &= E[X_s X_t] - (x_0 + \mu s)(x_0 + \mu t) \\ &= x_0^2 + x_0\mu(t+s) + x_0 E[\sum_{i=1}^{t} W_i + \sum_{i=1}^{s} W_i] \\ &+ \mu^2 st + \mu(sE[\sum_{i=1}^{t} W_i] + tE[\sum_{i=1}^{s} W_i]) + E[\sum_{i=1}^{t}\sum_{j=1}^{s} W_i W_j] - x_0^2 - x_0\mu(t+s) - \mu^2 st \\ &= \sum_{i=i}^{t}\sum_{j=1}^{s} E[W_i W_j] = \min(s,t)\sigma^2\end{aligned}$$

3. Não. $\{X_t\}_t$ são é estacionário porque as estatísticas variam com o tempo, inclusive a média, o que já não permite nem uma estacionaridade fraca.

4. $\nabla X_t = X_t - X_{t-1} = \mu + W_t$. Logo:

$$\mu_X(t) = E[\nabla X_t] = \mu$$

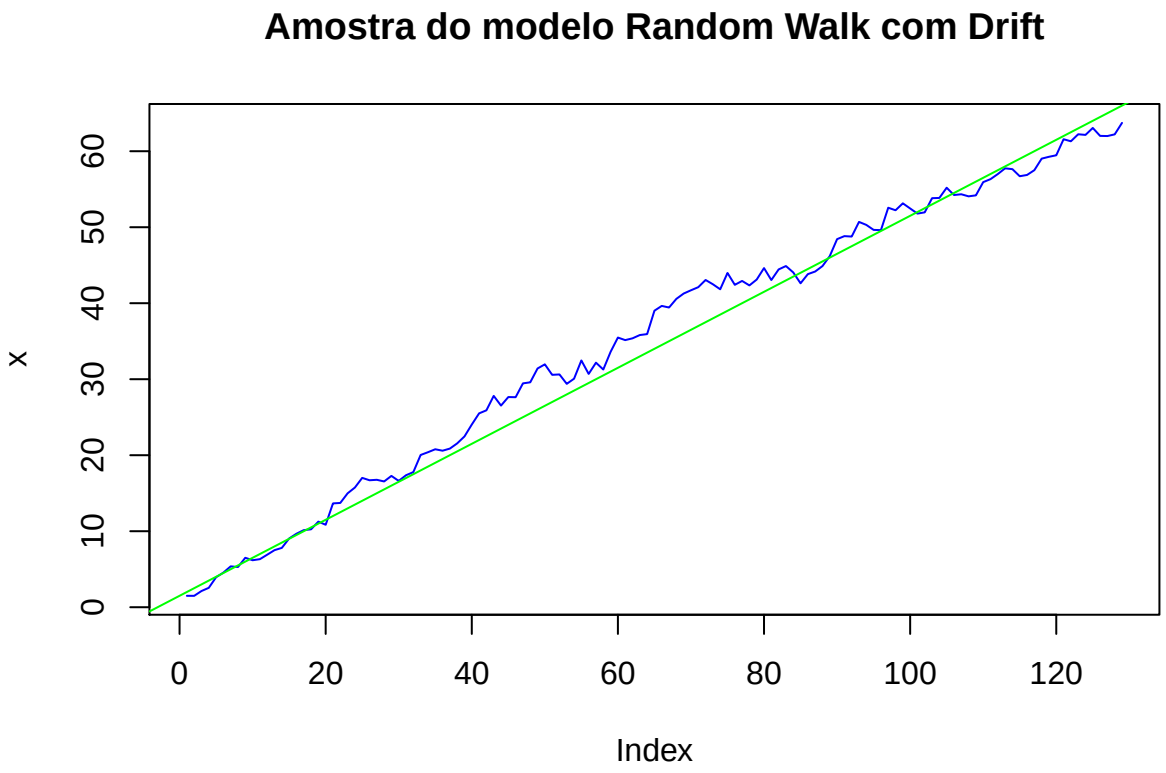
$$\sigma_X^2(t) = Var[\nabla X_t] = Var[W_t] = \sigma^2$$

$$\gamma_X(s,t) = Cov(X_s, X_t) = E[X_s X_t] - \mu^2 = \mu^2 + \mu E[W_t + W_s] + E[W_t W_s] - \mu^2 = \sigma^2 \not\perp_{s=t}$$

5. Ainda não posso garantir que o processo é estacionário, mas posso garantir, baseado na questão 4, que esse processo é, pelo menos, fracamente estacionário.

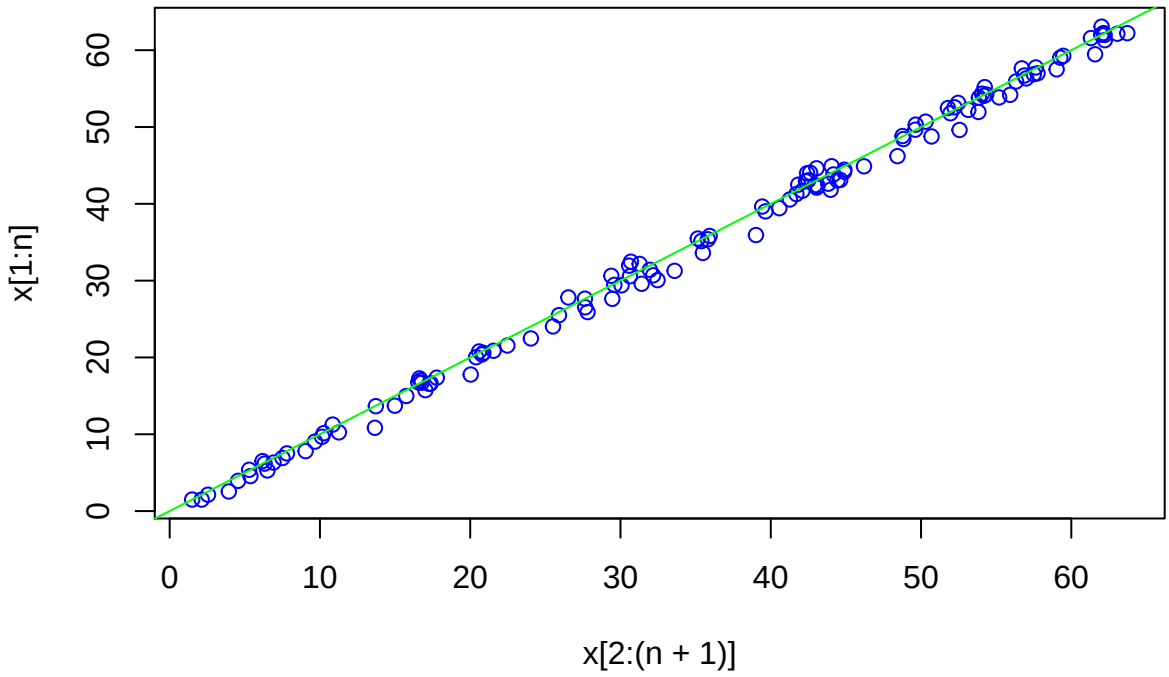
6. Seja $x_0 = 1.5, \mu = 0.5, n = 128$. Assuma o ruído branco como Gaussiano.

```r
n <- 128
mu <- 0.5
x0 <- 1.5
set.seed(100)
w <- rnorm(n = n, mean = 0, sd = 1)
x <- array(NA, dim = n+1)
x[1] <- x0
for(i in 2:(n+1)){
  # note que tomo w[i-1], pois a sequência de x começa em x0
  x[i] <- mu + x[i-1] + w[i-1]
}
plot(x, main = 'Amostra do modelo Random Walk com Drift', col = 'blue', type = 'l')
abline(a = x0, b = mu, col = 'green')
```



Observamos que, como mostrado na reta verde, que tem intercepto $a = x_0$ e inclinação $b = \mu$, esses parâmetros indicam a reta que melhor se encaixa nos dados de random walk com drift. Isto é, lemos μ como a inclinação da melhor reta para os dados (claro que eles tem uma variação natural do processo) e x_0 como o intercepto, o que também já era esperado.

```r
plot(x[2:(n+1)], x[1:n], col = 'blue')
abline(a = 0, b = 1, col = 'green')
```



Nesse plot, podemos ver uma reta $x = y$. Isso acontece pela construção de nosso modelo, onde $y = x + \mu + e$, onde $\mu + e$ é basicamente uma constante que, mesque mude com o tempo, varia como uma normal.

Ajuste de Modelo ARIMA

Considerarei este **tutorial** para fazer esse exercício.

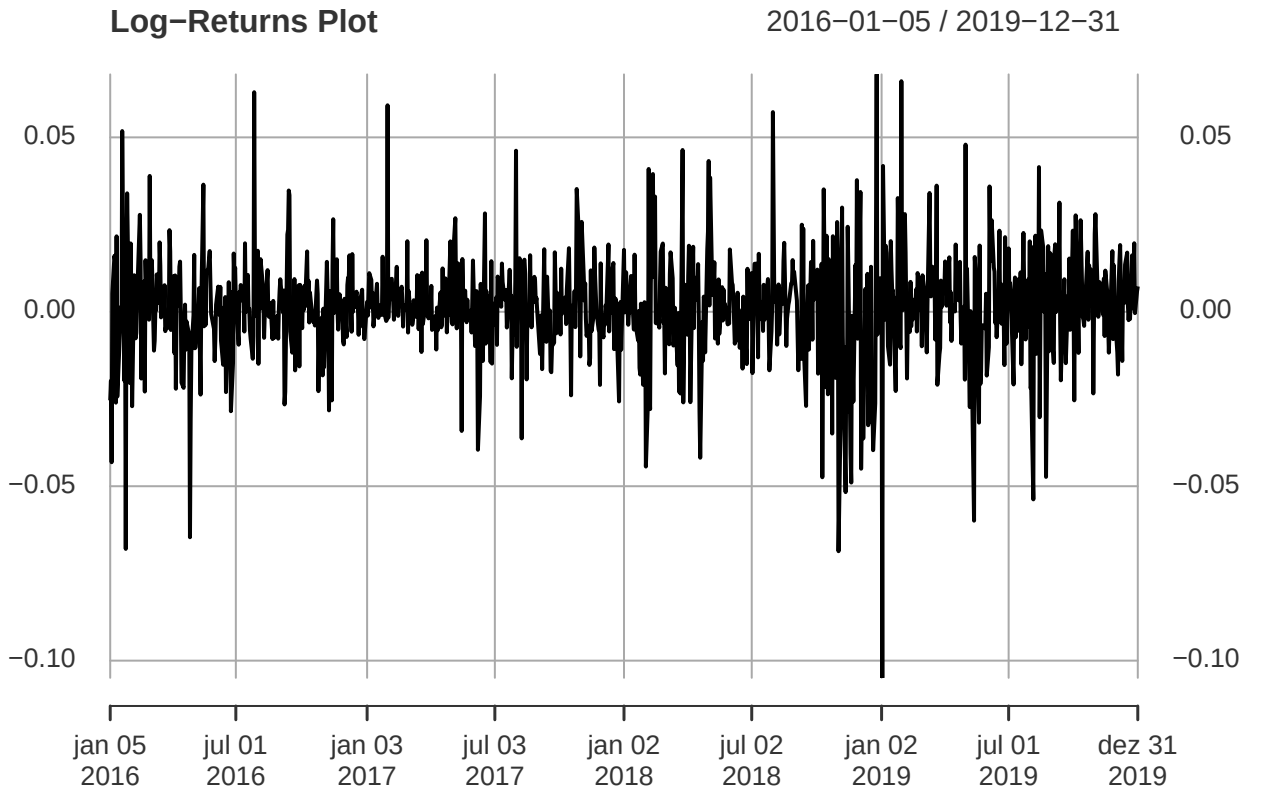
```
library(quantmod)
library(tseries)
library(timeSeries)
library(forecast)
```

Considerarei os dados dos preços de ações da Apple.

```
getSymbols('AAPL', from = '2016-01-01', to = '2020-01-01')
```

```
## [1] "AAPL"
```

```
# Tomo os valores de fechamento
stock_prices <- AAPL[,4]
stock <- diff(log(stock_prices))
stock <- stock[!is.na(stock)]

plot(stock, type = 'l', main = 'Log-Returns Plot')
```



Esses dados não tem periodicidade, dado que a função `stl` não identificou. Então, vamos prossequir dessa forma. Vamos fazer o teste ADF para checar a estacionaridade. Nesse caso:

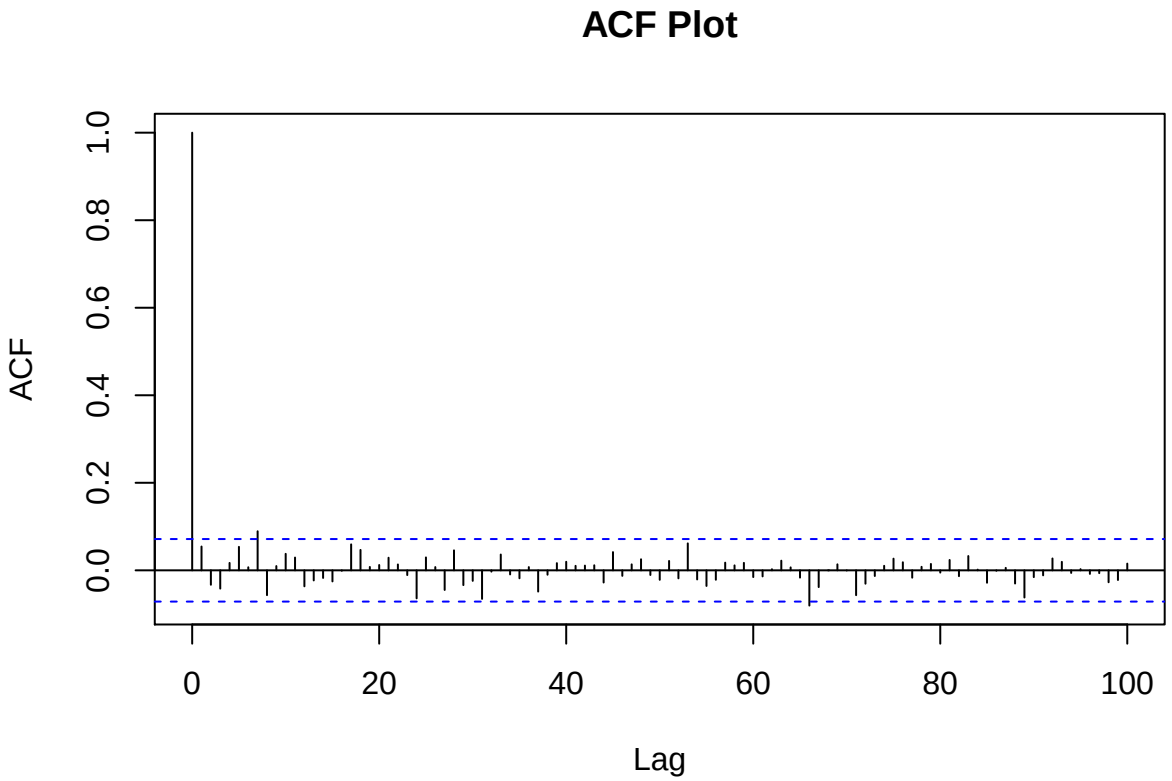
```
print(adf.test(stock))
```

```
## Warning in adf.test(stock): p-value smaller than printed p-value
```

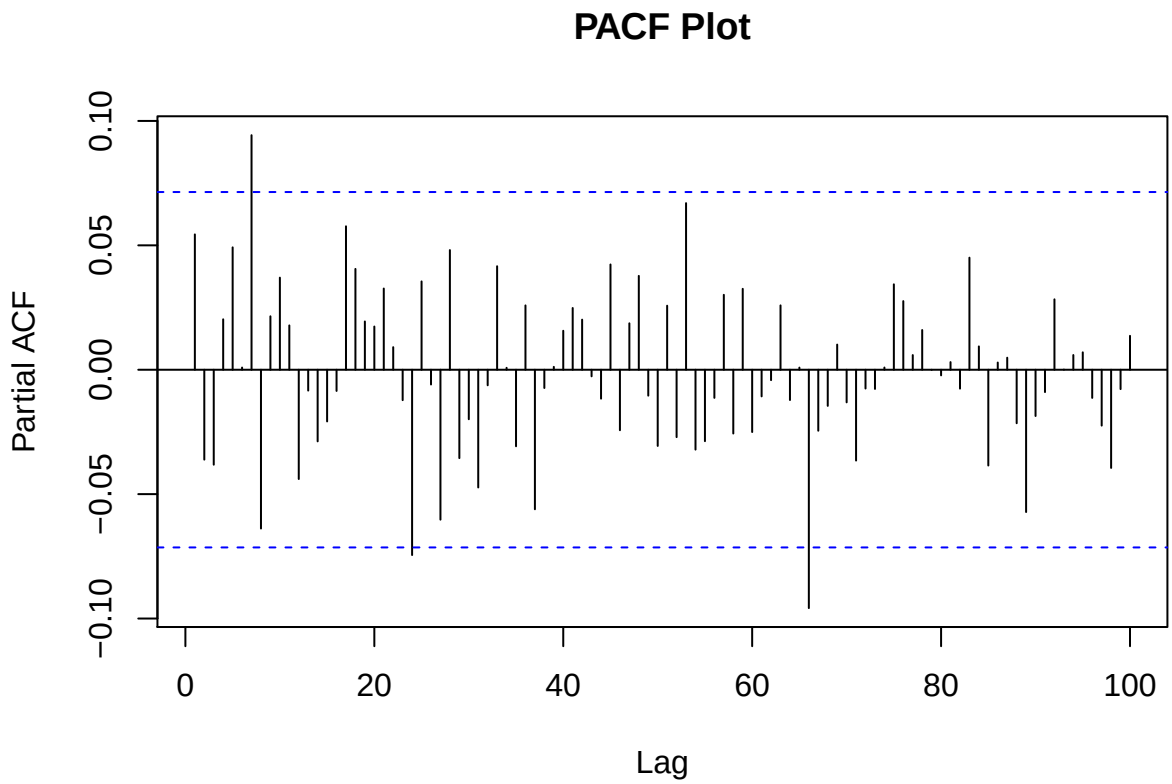
```
##
##  Augmented Dickey-Fuller Test
##
## data:  stock
## Dickey-Fuller = -8.6773, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

Como o *p-value* é menor do que 0.01, rejeitamos a hipótese nula e obtemos que a série é estacionária. Agora, vamos dividir o a lista em a lista em conjunto de treino e conjunto de teste. Vamos calcular nos pontos de treino as funções de Auto Covariância e Auto Correlação Parcial.

```
breakpoint <- floor(nrow(stock)*(3/4))
par(mfrow = c(1,1))
acf.stock <- acf(stock[c(1:breakpoint),], main = 'ACF Plot', lag.max = 100)
```




```
pacf.stock <- pacf(stock[c(1:breakpoint),], main = 'PACF Plot', lag.max = 100)
```



A partir desses plots, podemos obter as ordens do molelo AR e do modelo MA. A ACF do modelo AR decaí exponencialmente, enquando a PACF indica a ordem do modelo. Isso ocorre de forma inversa para o modelo MA. Analizando os picos, podemos selecionar a ordem AR = 3 e a ordem de MA = 2. Logo, teremos

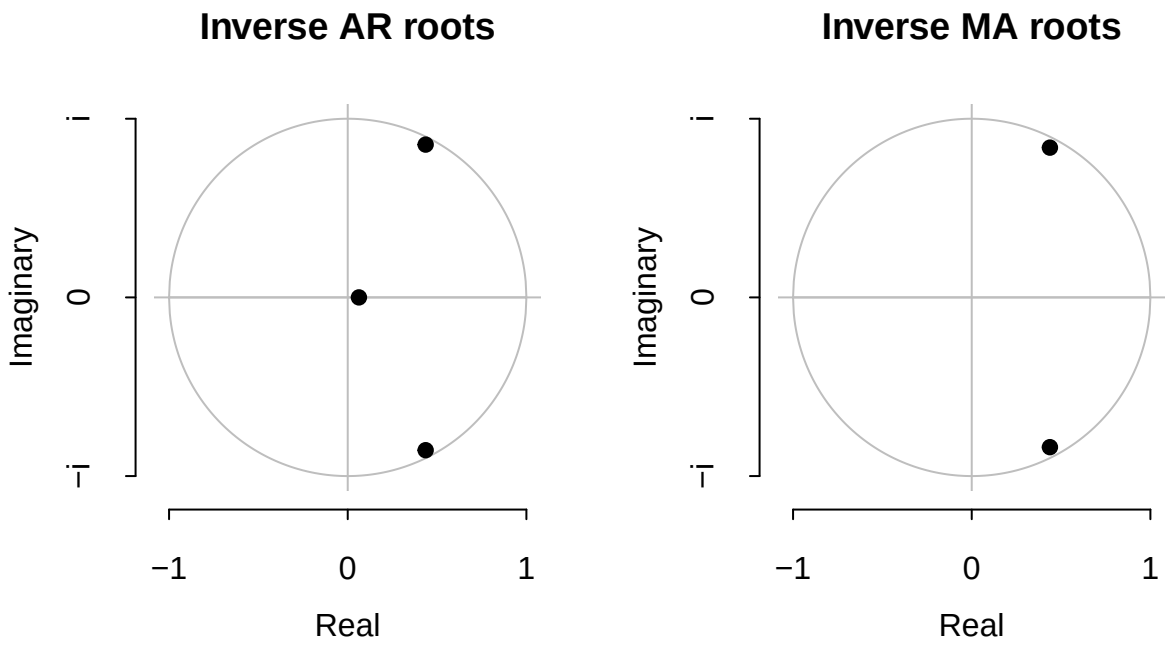
ARIMA(3,1,2), dado que realizei uma diferença. Além disso, fiz alguns testes com outros valores, mas valores maiores tiveram problemas de convergência e menores reduziram a acurácia.

Podemos ver que a função de auto correlação dos resíduos é bem interessante, com os valores em grande maioria dentro de uma margem razoável. Também podemos visualizar que o resumo do `fit` é bem interessante, com os erros bem baixos.

```
stock_train <- stock[1:breakpoint,]
stock_test <- stock[(breakpoint+1):nrow(stock),]
fit <- arima(stock_train, order = c(3,0,2), include.mean = FALSE)
```

```
## Warning in arima(stock_train, order = c(3, 0, 2), include.mean = FALSE):
## possible convergence problem: optim gave code = 1
```

```
plot(fit)
```



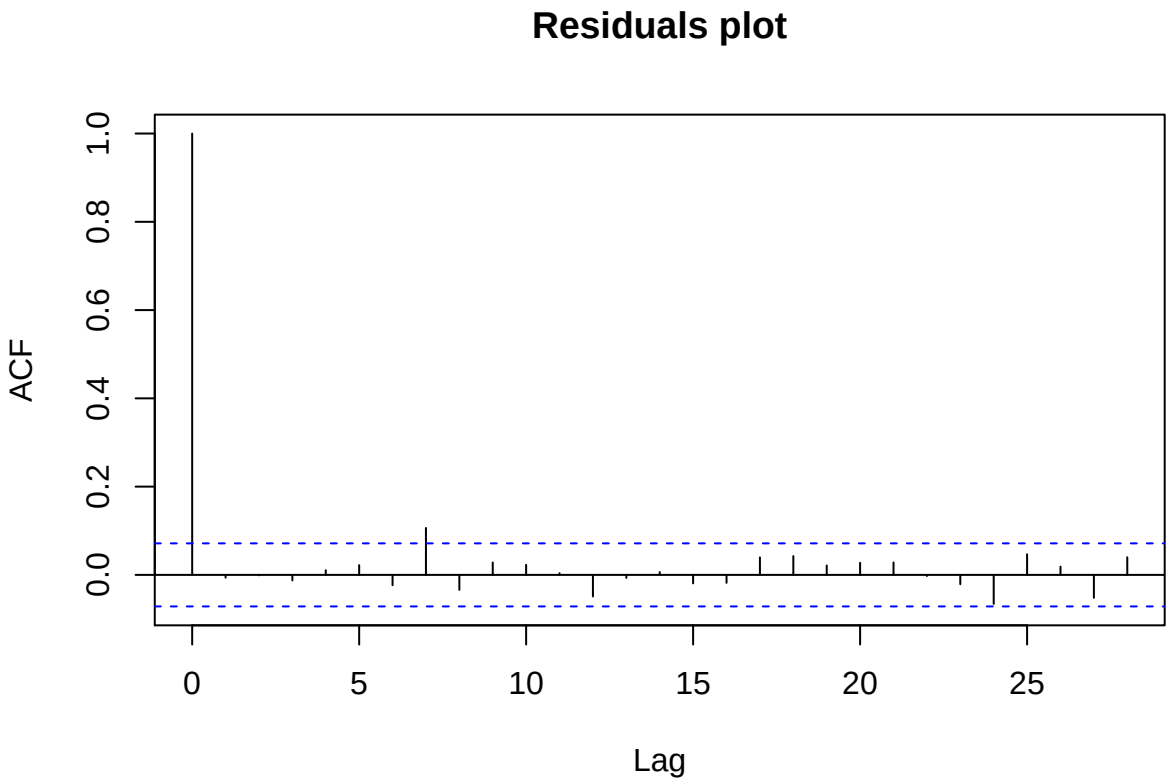

```
summary(fit)
```

```
##
## Call:
## arima(x = stock_train, order = c(3, 0, 2), include.mean = FALSE)
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##          ar1      ar2     ar3      ma1     ma2
##       0.9348  -0.9761  0.0578  -0.8753  0.8941
## s.e.     NaN      NaN  0.0363      NaN     NaN
##
## sigma^2 estimated as 0.000221:  log likelihood = 2100.59,  aic = -4189.18
##
## Training set error measures:
##                       ME       RMSE        MAE  MPE MAPE      MASE         ACF1
## Training set 0.000518415 0.01486698 0.01027722 -Inf  Inf 0.7035417 -0.006280291
```

```r
acf(fit$residuals, main = "Residuals plot")
```



Agora vejamos o forecast.

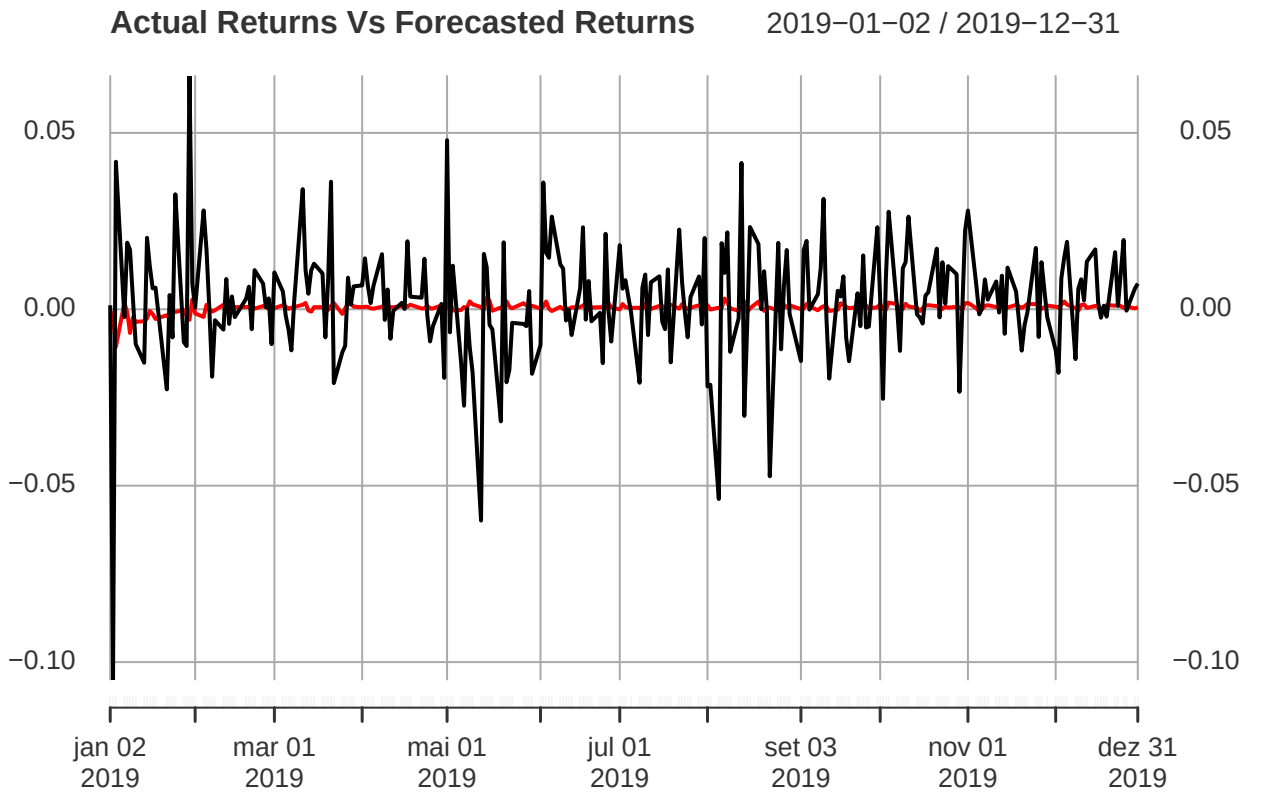
```r
forecasted_series <- data.frame(Forecasted = numeric())

for(b in breakpoint:(nrow(stock)-1)){
  stock_train <- stock[1:b,]

  fit <- arima(stock_train, order = c(3,1,2), include.mean = FALSE)
  arima.forecast = forecast(fit, h = 1, level = 99)
  if(b == breakpoint){
    summary(arima.forecast)
  }
  forecasted_series <- rbind(forecasted_series, arima.forecast$mean[1])
  colnames(forecasted_series) <- c("Forecasted")
}
```

```
##
## Forecast method: ARIMA(3,1,2)
##
## Model Information:
##
## Call:
## arima(x = stock_train, order = c(3, 1, 2), include.mean = FALSE)
##
## Coefficients:
##           ar1      ar2      ar3      ma1      ma2
##       -0.0586  -0.0329  -0.0482  -0.8822  -0.1078
## s.e.   0.4528   0.0451   0.0405   0.4522   0.4496
```

```
##
## sigma^2 estimated as 0.0002226:  log likelihood = 2093.21,  aic = -4174.42
##
## Error measures:
##                         ME       RMSE        MAE  MPE MAPE      MASE        ACF1
## Training set 0.0001421427 0.01490823 0.01035896 -Inf  Inf 0.7091368 0.001785664
##
## Forecasts:
##     Point Forecast       Lo 99      Hi 99
## 754  -0.0005044525 -0.03893103 0.03792212
```

Observe que as estimativas são próximas a média da disctruibição. Assim, vamos checar a acurácia do modelo ARIMA. EM vermelho vemos a previsão do modelo. Por fim, computamos o acerto em relação ao sinal que o modelo propôs. O interessante é que ficou só um pouco melhor do que jogar uma moeda.

```
stock_test <- stock[(breakpoint+1):nrow(stock),]
stock_test$Forecast <- forecasted_series$Forecasted
plot(stock_test, type = 'l', main = 'Actual Returns Vs Forecasted Returns')
```




```
comparsion = merge(stock_test$AAPL.Close,stock_test$Forecast)
comparsion$Accuracy = sign(comparsion$AAPL.Close)==sign(comparsion$Forecast)

Accuracy_percentage = sum(comparsion$Accuracy == 1)*100/length(comparsion$Accuracy)
print(Accuracy_percentage)
```

```
## [1] 54.36508
```