AC Econometria II

José Pedro e Lauro Aguiar

16/11/2022

Contents

1 Introdução:

Trabalho elaborado para a disciplina de Econometria de Séries Temporais da Faculdade Ibmec, sob orientação do professor Frank Pinho. Neste trabalho, iremos realizar tentar encontrar o processo determinístico de uma série temporal. Nesse contexto, escolhemos a receita bruta da Natura & Co.

```
source("/cloud/project/frankfiles/install_and_load_packages.R")
```

```
## Carregando pacotes default para a disciplina...
```

```
## Warning in fun(libname, pkgname): couldn't connect to display ":0"
```

```
## Lista de pacotes carregados....
```

```
##  [1] "tidyquant"            "tidyverse"            "TeachingDemos"
##  [4] "EnvStats"             "readr"                "readxl"
##  [7] "xtable"               "stargazer"            "highcharter"
## [10] "quantmod"             "dygraphs"             "tseries"
## [13] "htmltools"            "Quandl"               "nycflights13"
## [16] "magrittr"             "discreteRV"           "aTSA"
## [19] "fGarch"               "fUnitRoots"           "vars"
## [22] "MTS"                  "seasonal"             "urca"
## [25] "dynlm"                "tbl2xts"              "dlm"
## [28] "stats"                "tcltk"                "tcltk2"
## [31] "forecast"             "Quandl"               "dygraphs"
## [34] "magrittr"             "PerformanceAnalytics" "quantmod"
```

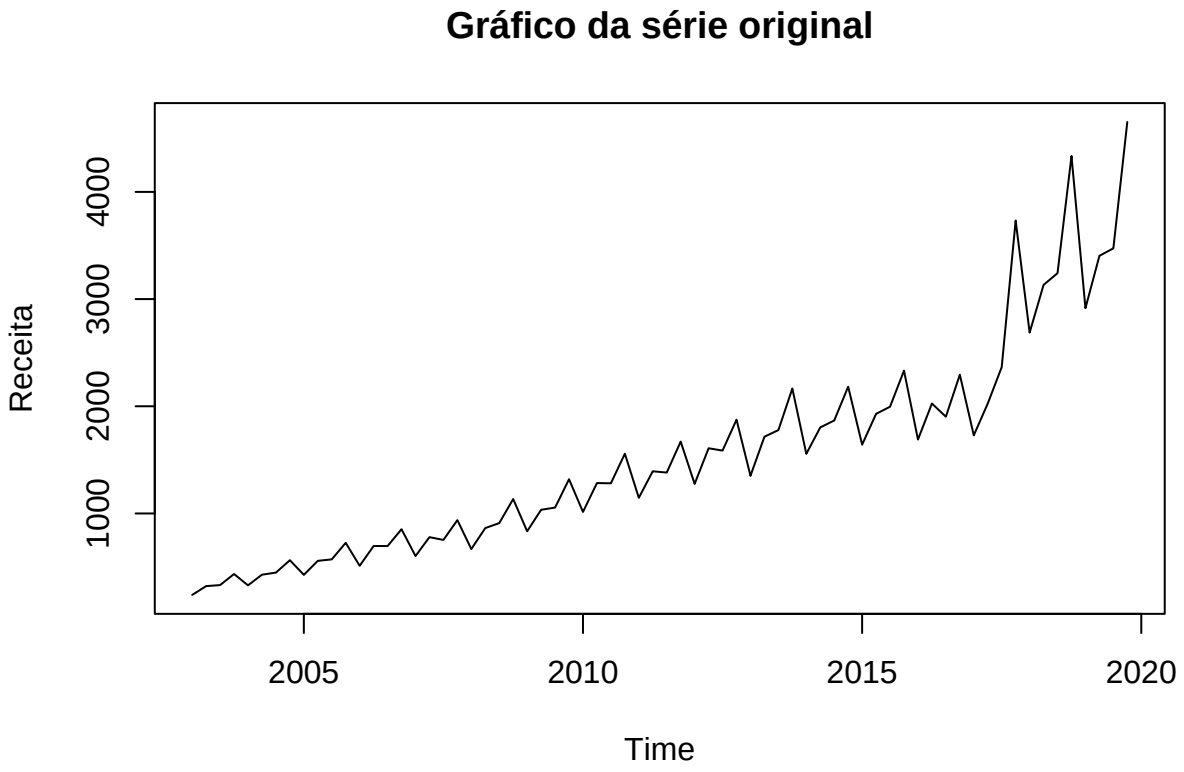
2 Carregando a base de dados:

```
dados <- read_xlsx("/cloud/project/frankfiles/ACEcoII/Natura_Revenue.xlsx")

# Convertendo os vetores números em um objeto de série temporal
dados <- ts(dados, frequency = 4, start = c(2003,1))
```

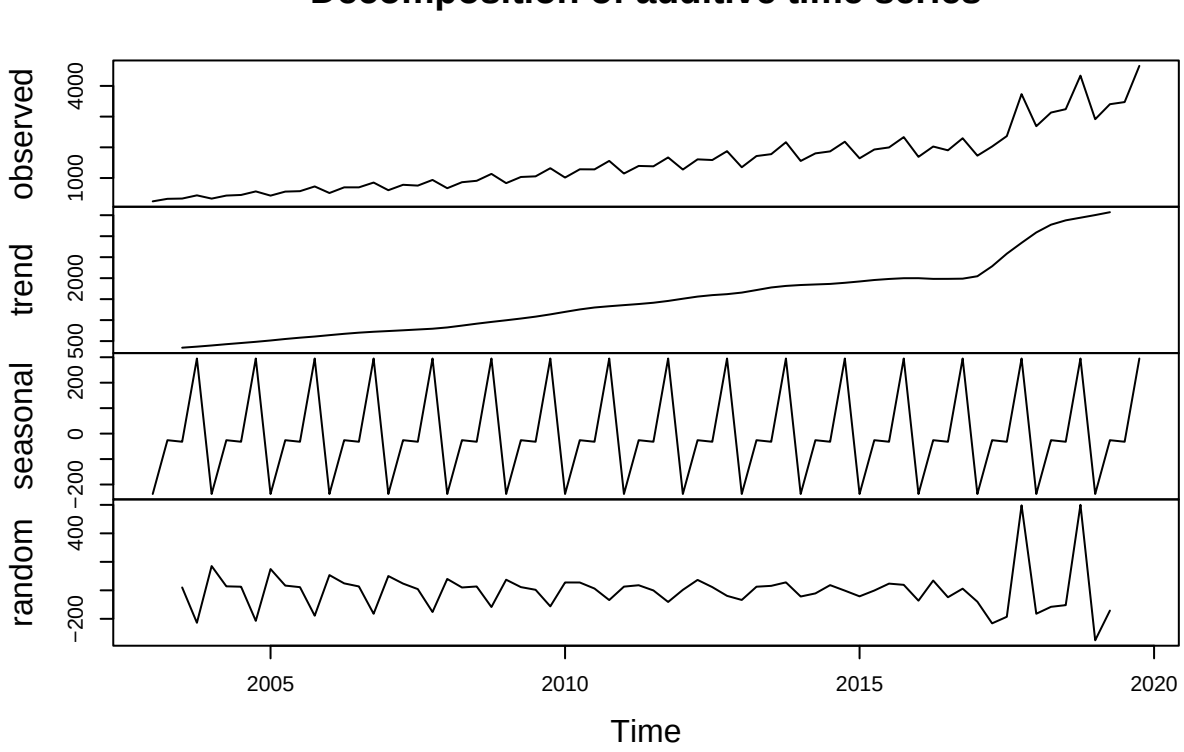
3 Visualizando os dados:

```
plot(dados, main = "Gráfico da série original")
```




```
dygraph(dados)
```

```
plot(decompose(dados))
```



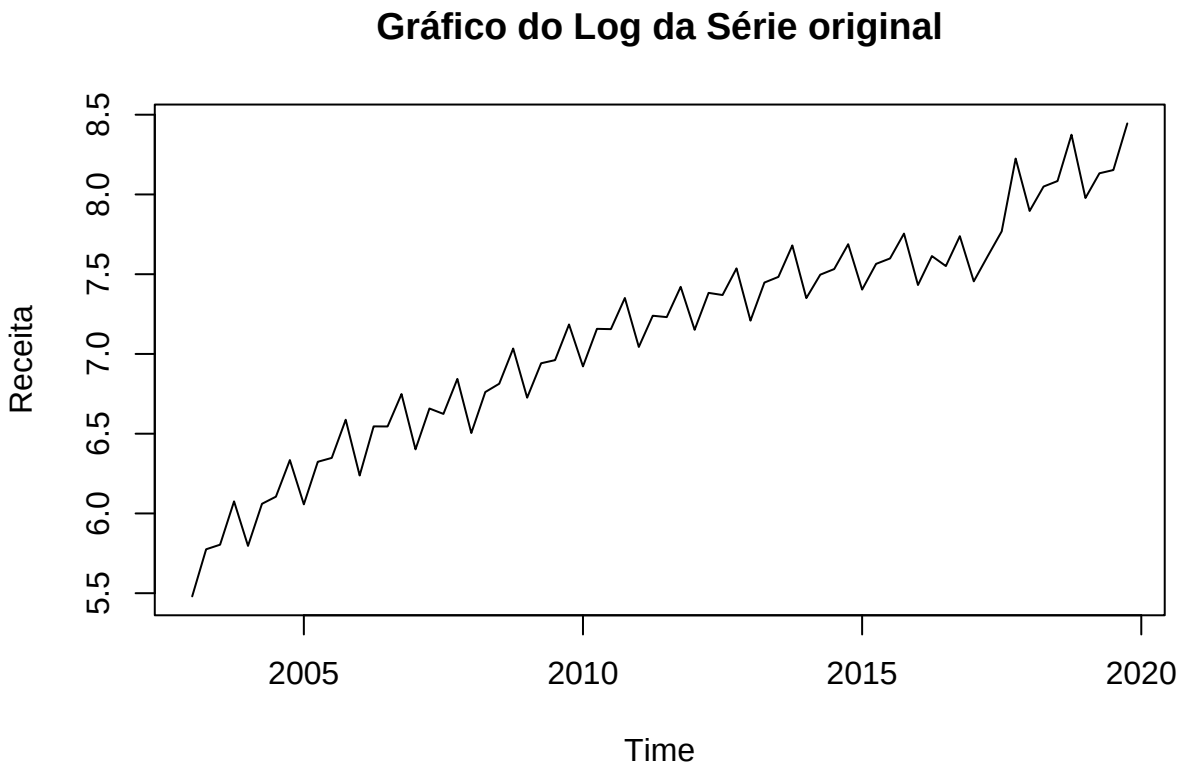
Interpretação: Desde 2003 (início da série) podemos observar uma variação positiva da receita no quarto

trimestre do ano até os anos atuais, indicando uma forte sazonalidade que se estende por mais de uma década. Evidentemente, esse processo é explicado pela configuração do setor varejista que é impactado pelas festividades do final do ano, como o dia das crianças (Outubro), Black Friday (Novembro), Natal e Ano Novo (Dezembro).

Além disso, é nítido uma tendência de crescimento da receita bruta, especialmente de 2019 adiante.

Ajuste no modelo: Como a receita da Natura apresenta grande variabilidade (240k - 12bi), iremos trabalhar com o logaritmo dos dados:

```
dados_log <- log(dados)
plot.ts(dados_log, main = "Gráfico do Log da Série original")
```



4 Indicação de um possível modelo da série - a ser validado no nosso trabalho:

A função auto.arima() em R utiliza uma variação do algoritmo Hyndman-Khandakar (Hyndman & Khandakar, 2008), que combina testes de raiz unitária, minimização do AICc e MLE para obter um modelo ARIMA. Os argumentos para auto.arima() fornecem muitas variações no algoritmo. O que é descrito aqui é o comportamento padrão.

Hyndman-Khandakar algorithm for automatic ARIMA modelling

1. The number of differences $0 \leq d \leq 2$ is determined using repeated KPSS tests.
2. The values of p and q are then chosen by minimising the AICc after differencing the data d times. Rather than considering every possible combination of p and q, the algorithm uses a stepwise search to traverse the model space.

    a. Four initial models are fitted:
    - ARIMA$(0, d, 0)$,
    - ARIMA$(2, d, 2)$,
    - ARIMA$(1, d, 0)$,
    - ARIMA$(0, d, 1)$.

    A constant is included unless $d = 2$. If $d \leq 1$, an additional model is also fitted:
    - ARIMA$(0, d, 0)$ without a constant.

    b. The best model (with the smallest AICc value) fitted in step (a) is set to be the "current model".

    c. Variations on the current model are considered:
    - vary p and/or q from the current model by ± 1;
    - include/exclude c from the current model.

    The best model considered so far (either the current model or one of these variations) becomes the new current model.

    d. Repeat Step 2(c) until no lower AICc can be found.

Referência: Hyndman, R. J., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for R. Journal of Statistical Software, 27(1), 1–22. [DOI].

```
auto.arima(dados_log)
```

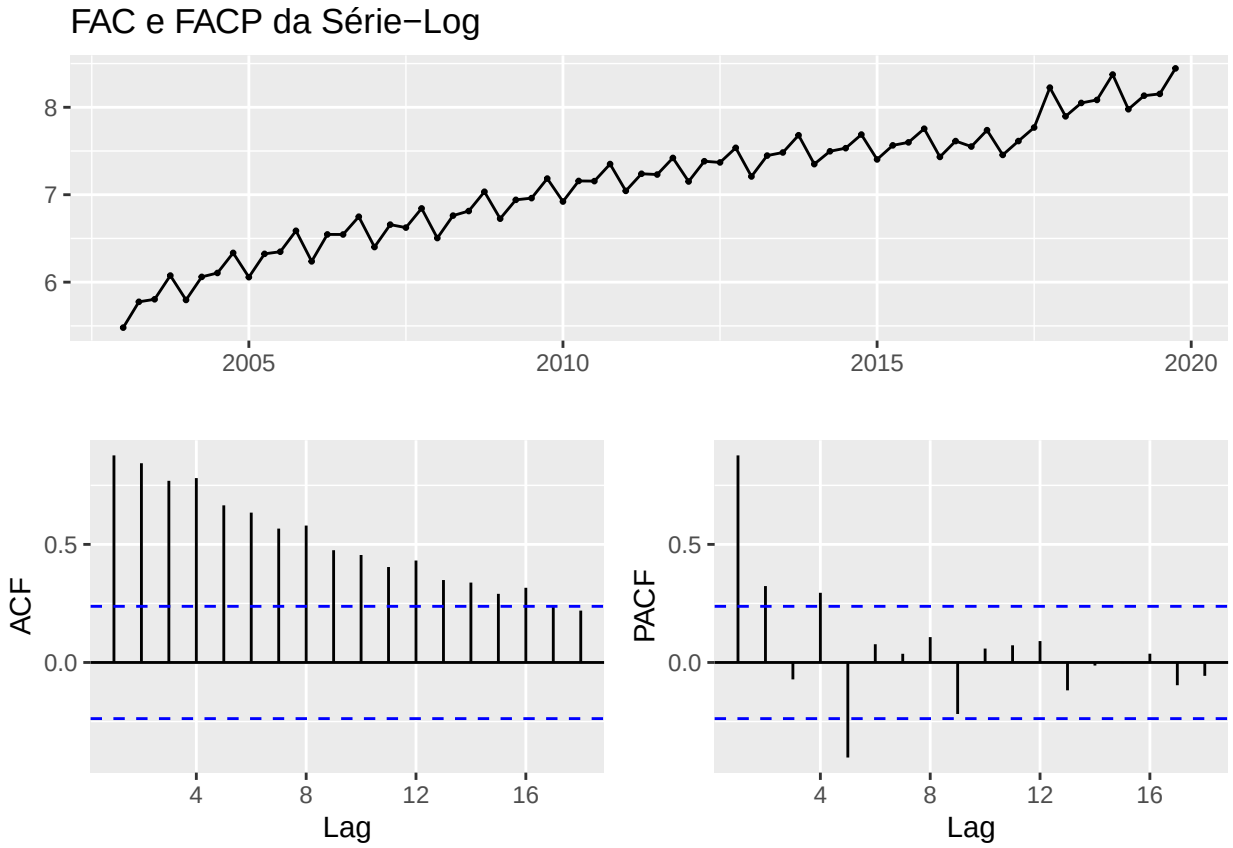
```
## Series: dados_log
## ARIMA(0,1,1)(0,1,1)[4]
##
## Coefficients:
##          ma1     sma1
##       0.2002  -0.5760
## s.e.  0.1210   0.1221
##
## sigma^2 = 0.002755:  log likelihood = 96.5
## AIC=-186.99   AICc=-186.58   BIC=-180.56
```

Utilizamos o algoritmo Hyndman-Khandakar apenas para tentar comparar com nosso resultado final.

5 Verificando estacionariedade por meio da FAC e do teste Dickey-Fuller:

Função de Autocorrelação da série transformada:

```
dados_log %>% ggtsdisplay(main = "FAC e FACP da Série-Log")
```



Interpretação: Ao observar a FAC é perceptível um decaimento lento nos lags, indicando que o modelo não é estacionário. Para confirmar essa tese, devemos realizar o teste Dickey Fuller abaixo onde será possível perceber a presença de raiz unitária.

Teste Dickey-Fuller:

- H0: Processo não estacionário - uma raíz unitária.
- H1: Processo estacionário - sem raízes unitárias.

```
adf_dados_log <- fUnitRoots::adfTest(dados_log, lags = 4, type = c("nc"))
adf_dados_log
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 4
##   STATISTIC:
##     Dickey-Fuller: 1.6218
##   P VALUE:
##     0.9728
##
## Description:
##  Tue Nov 15 00:20:22 2022 by user:
```

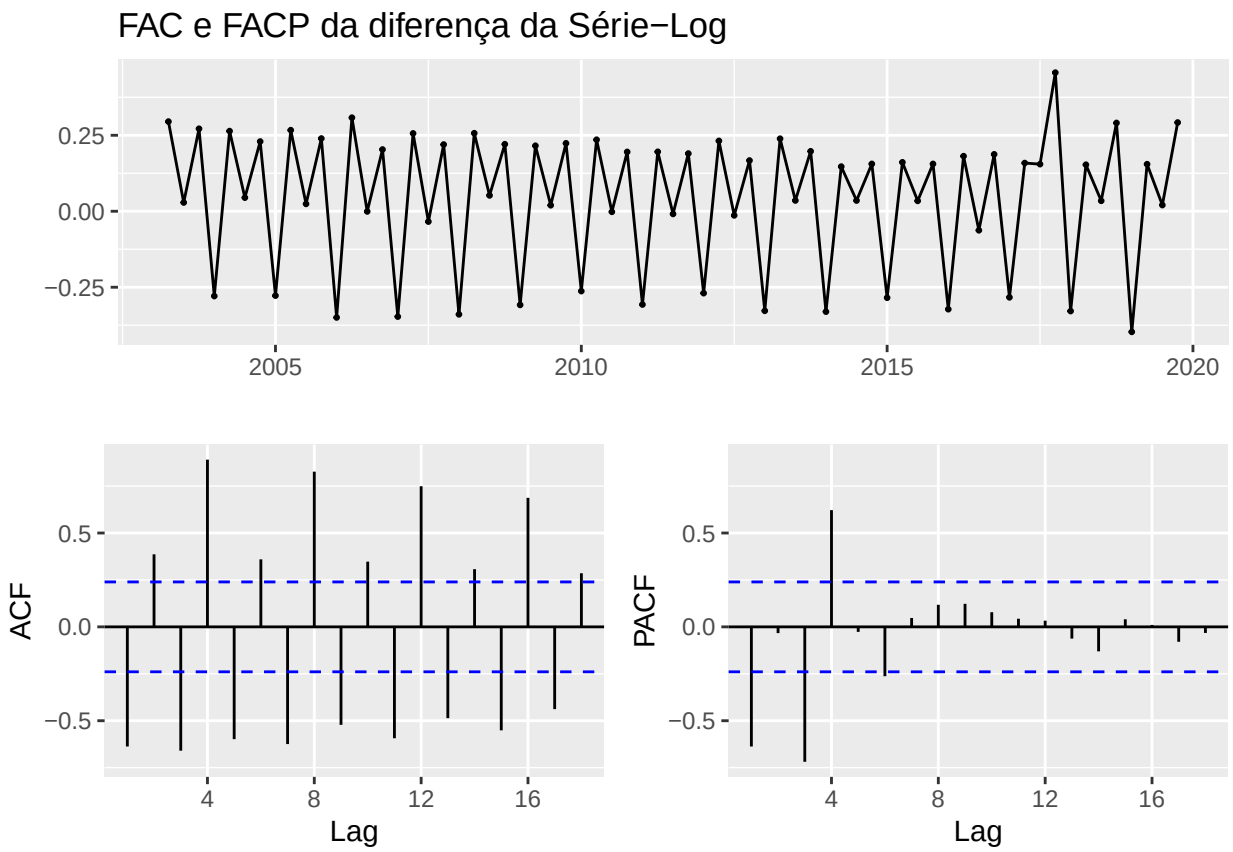
Interpretação: Com um p-valor de 0.9728 o teste não rejeitou a hipotese nula de que o processo é não estacionário. Portanto, devemos fazer a diferenciação dos dados até se tornarem estacionários. Como os

dados possuem uma tendência de crescimento, devemos fazer a diferenciação primeiro da série e depois uma diferenciação da sazonalidade.

6 Realizando a diferenciação dos dados não sazonais da série e analisando a FAC correspondente:

```
# Diferenciação dos dados - lag 1
dados_log_diff <- timeSeries::diff(dados_log, lag = 1, differences = 1)

# FAC dos dados resultantes
dados_log_diff %>% ggtsdisplay(main = "FAC e FACP da diferença da Série-Log")
```

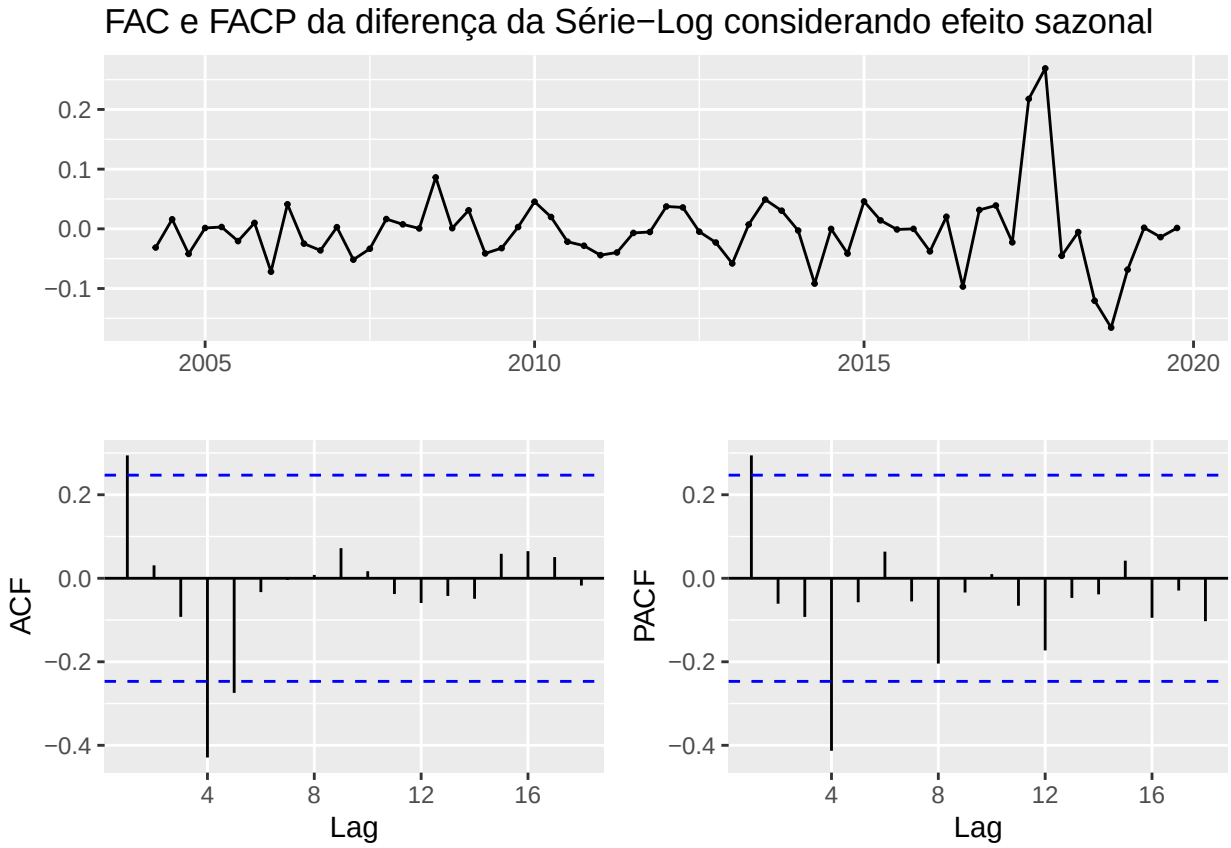


Interpretação: Dessa forma, fazendo a diferenciação dos dados não sazonais já eliminamos o decaimento lento na FAC. Contudo, como há ainda um decaimento lento nos lags sazonais, será necessário mais uma diferenciação: agora com os lags sazonais.

7 Realizando a diferenciação dos dados sazonais e analisando a FAC correspondente:

```
# Diferenciação dos dados - lag 1
dados_log_diff_saz <- timeSeries::diff(dados_log_diff, lag = 4, differences = 1, lag.max = 48)

# FAC e FACP dos dados resultantes
dados_log_diff_saz %>% ggtsdisplay(main = "FAC e FACP da diferença da Série-Log considerando efeito saz
```



Interpretação: Com a diferenciação tanto dos dados não sazonais quanto dos dados sazonais, podemos inferir que o modelo é um ARIMA (1,1,1)(1,1,1)_[4].

8 Testando a estacionaridade da série diferenciada:

Teste Dickey-Fuller - H0: Processo não estacionário - uma raíz unitária. - H1: Processo estacionário - sem raízes unitárias.

```
fUnitRoots::adfTest(dados_log_diff_saz, lags = 4, type = c("nc"))
```

```
## Warning in fUnitRoots::adfTest(dados_log_diff_saz, lags = 4, type = c("nc")): p-
## value smaller than printed p-value
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 4
##   STATISTIC:
##     Dickey-Fuller: -4.7248
##   P VALUE:
##     0.01
##
## Description:
##  Tue Nov 15 00:20:23 2022 by user:
```

Interpretação: Como o p-valor foi inferior a 0.05 concluímos que a série agora é estacionária, já que o teste rejeitou a hipótese nula de não estacionariedade do processo.

9 Estimando todas as combinações de modelos possíveis do modelo SARIMA (p,d,q)(P,D,Q)_S.

```
#Estimando as combinações possíveis de modelos:
modelos_possiveis <- expand.grid (ar = 1, diff = 1, ma = 1, ars = 0:1, diffs = 0:1, mas = 0:1)
#Armazenando os resultados de cada modelo
modelos <- list()
```

Interpretação: Extraímos as informações com base na análise da FAC e FACP para a construção dos modelos possíveis.

10 Estimando os parâmetros dos modelos utilizando a máxima verossimilhança (ML):

- Por default, a função ARIMA utiliza a hipótese de que o termo de erro dos modelos ARIMA seguem uma distribuição normal.

```
for (i in 1:nrow(modelos_possiveis)) {
  modelos[[i]] <- arima(x = dados, order = unlist(modelos_possiveis[i, 1:3]),
                        seasonal = list(order = unlist(modelos_possiveis[i,4:6]), period = 4), method =
}
```

- Obtendo o logaritmo da verossimilhança (valor máximo da função):

```
log_verossimilhanca <- list()

for (i in 1:length(modelos)){
  log_verossimilhanca[[i]] <- modelos[[i]]$loglik
}
```

- Calculando o AIC:

```
aic_sarima <- list()

for (i in 1:length(modelos)) {
  aic_sarima[[i]] <- stats::AIC(modelos[[i]])
}
```

- Calculando o BIC:

```
bic_sarima <- list()

for (i in 1:length(modelos)) {
  bic_sarima[[i]] <- stats::BIC(modelos[[i]])
}
```

- Quantidade de parâmetros estimados por modelo:

```
#
quant_parametros <- list()

for (i in 1:length(modelos)) {
```

```
  quant_parametros[[i]] <- length(modelos[[i]]$coef)+1
}
```

- Montando a tabela com os resultados:

```
especificacao <- paste0("SARIMA",modelos_possiveis$ar, modelos_possiveis$diff, modelos_possiveis$ma, mo
                        modelos_possiveis$diffs, modelos_possiveis$mas)

resultado <- data.frame(especificacao,
                        ln_verossimilhanca = unlist(log_verossimilhanca),
                        quant_parametros = unlist(quant_parametros),
                        aic = unlist(aic_sarima),
                        bic = unlist(bic_sarima))
print(resultado)
```

```
##   especificacao ln_verossimilhanca quant_parametros      aic      bic
## 1  SARIMA111000          -489.9488                3 985.8975 992.5116
## 2  SARIMA111100          -442.9291                4 893.8582 902.6770
## 3  SARIMA111010          -413.0036                3 832.0073 838.4367
## 4  SARIMA111110          -411.2383                4 830.4767 839.0492
## 5  SARIMA111001          -473.6942                4 955.3883 964.2071
## 6  SARIMA111101          -441.6713                5 893.3426 904.3661
## 7  SARIMA111011          -411.1456                4 830.2912 838.8637
## 8  SARIMA111111          -411.1438                5 832.2876 843.0033
```

11 Escolhendo dentre todos os modelos estimados no passo anterior, o modelo com menor AIC e/ou BIC.

```
best1 <- which.min(resultado$aic)
print(best1)
```

```
## [1] 7
```

```
best2 <- which.min(resultado$bic)
print(best2)
```

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

Interpretação: Dessa forma, conseguimos encontrar dois modelos com base no Akaike (AIC) e o Critério Bayesiano de Schwarz (BIC), sendo que pelo AIC o modelo com maior verossimilhança é o modelo 7: SARIMA111011, com 4 parâmetros. Já o BIC selecionou o modelo 3: SARIMA111010, com 3 parâmetros.

12 Validação do modelo:

Teste de autocorrelação dos resíduos: - H0: resíduos não são autocorrelacionados. - H1: resíduos são autocorrelacionados.

```
Box.test(modelos[[best1]]$residuals,type="Ljung",lag = 8)
```

```
##
##  Box-Ljung test
##
## data:  modelos[[best1]]$residuals
## X-squared = 10.855, df = 8, p-value = 0.21
```

Teste de heterocedasticidade condicional: - H0: quadrado dos resíduos são não autocorrelacionados. Homocedasticidade, logo: variância é constante. - H1: quadrado dos resíduos são autocorrelacionados. Heterocedasticidade, logo: variância não é constante.

```
Box.test(modelos[[best1]]$residuals^2,type="Ljung",lag = 8)
```

```
##
##  Box-Ljung test
##
## data:  modelos[[best1]]$residuals^2
## X-squared = 8.443, df = 8, p-value = 0.3914
```

Teste de normalidade dos resíduos | Jarque Bera: - H0: resíduos são normalmente distribuídos. - H1: resíduos não são normalmente distribuídos.

```
normalTest(modelos[[best1]]$residuals, method = "jb")
```

```
##
## Title:
##  Jarque - Bera Normalality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 1538.879
##   P VALUE:
##     Asymptotic p Value: < 2.2e-16
##
## Description:
##  Tue Nov 15 00:20:24 2022 by user:
```

Interpretação geral da validação: No primeiro momento o teste de autocorrelação aponta que o modelo possui resíduos que não são autocorrelacionados, com um p-valor de 0.21. Dentro disso, o teste de heterocedasticidade condicional indicou que o modelo é homocedástico.

Contudo, mesmo realizar diversas diferenciações, alterar os parâmetros e utilizar o algortimo Hyndman-Khandakar o teste de normalidade dos resíduos do modelo não foi aceito, sendo que o p-valor de 2.2e-16 indica que a distribuição dos resíduos não é normal.

Dessa forma, faremos as previsões com a finalidade de conclusão do trabalho para fins didáticos, mesmo compreendendo que os resíduos ao não seguirem uma distribuição normal estarão impactando negativamente nossa previsibilidade.

13 Realizando previsões utilizando o modelo ARIMA(1,1,1)(1,1,1)[4].

```
modelo_prev <- arima(dados, order = c(1,1,1), seasonal = list(order=c(1,1,1), period = 4))
prev <- predict(modelo_prev, n.ahead = 4)
print(prev)
```

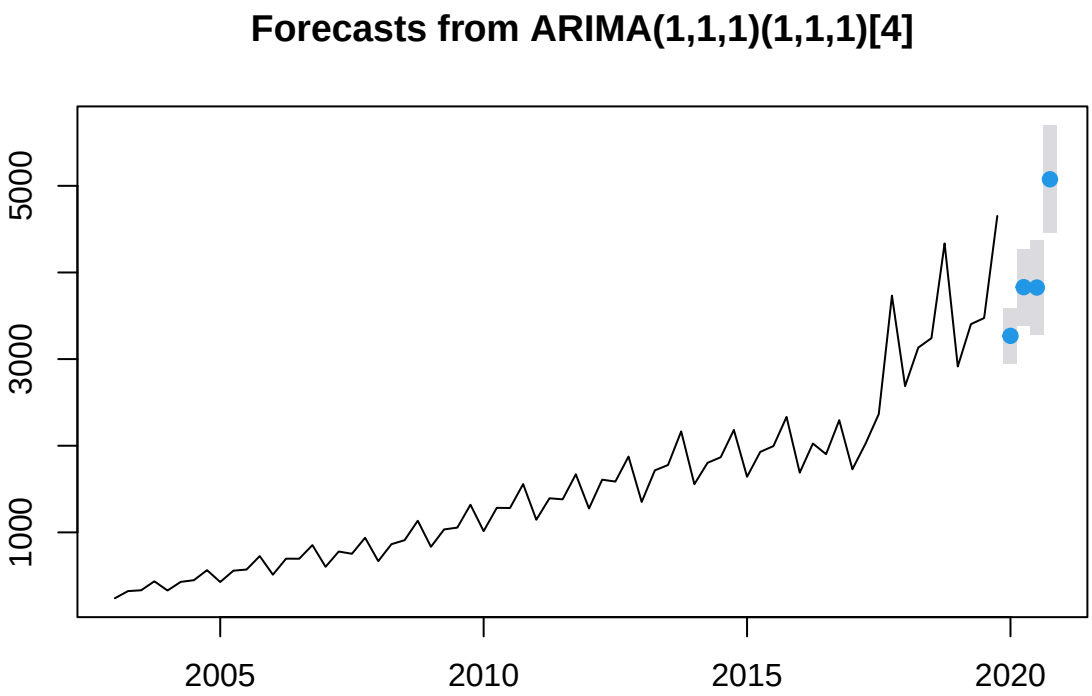
```
## $pred
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2020 3268.160 3830.394 3826.253 5075.652
##
## $se
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2020 164.1587 224.6152 278.1094 317.6469
```

#14 Plotando o gráfico da previsão:

Previsão do valor médio condicional esperado e respectivo desvio:

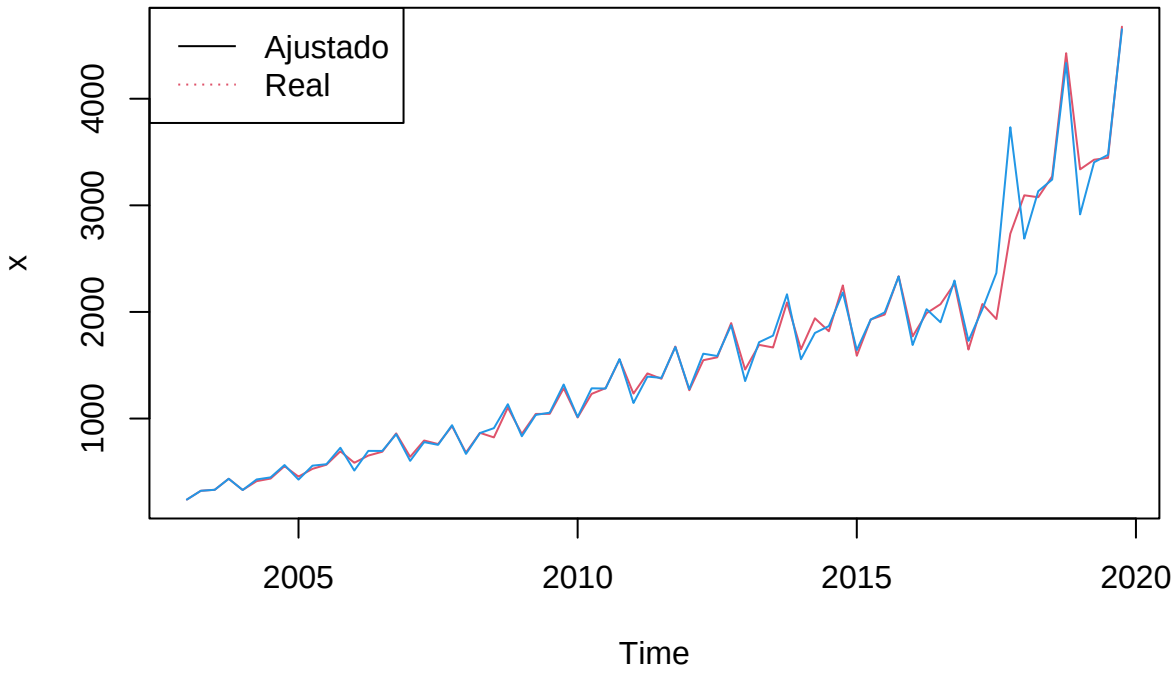
- *Object:* O modelo escolhido anteriormente.
- *Level:* O intervalo de confiança (abaixo, 80%).
- *h:* O horizonte de previsão.

```
plot(forecast::forecast(object = modelo_prev, h = 4, level = 0.95))
```



#15 Gráfico Real x Previsto:

```
par(mfrow=c(1,1))
fitted_modelo <- stats::fitted(modelo_prev)
plot(fitted_modelo, type = "l", lty = 1, col = 2)
lines(dados, type = "l", lty = 1, col = 4)
legend("topleft", legend = c("Ajustado", "Real"), col = c(1,2), lty = c(1,3))
```


Ajustado
Real
x
1000
2000
3000
4000
2005
2010
2015
2020
Time