Séries de Tempo

Aula 5 - Modelos ARIMA

Regis A. Ely

Departamento de Economia
Universidade Federal de Pelotas

13 de agosto de 2020

Conteúdo

Metodologia Box-Jenkins

Box e Jenkins (1970) sugeriram uma metodologia para modelar processos ARIMA baseada em quatro etapas[1]:

1. **Identificação**: determinação das ordens p, d e q$ do modelo ARIMA a ser estimado através das funções de autocorrelação e autocorrelação parcial, ou critérios de identificação
2. **Estimação**: estimação dos parâmetros do modelo $ARIMA(p, d, q)$, normalmente por máxima verossimilhança
3. **Diagnóstico**: inspeção dos resíduos do modelo estimado para verificar se ainda há alguma autocorrelação a ser modelada
4. **Previsão**: utilização dos coeficientes estimados para realizar previsões dos valores futuros da série de tempo

[1] Pode-se incluir uma etapa inicial que envolve a preparação e transformação dos dados originais.

Metodologia Box-Jenkins

- Já vimos as etapas que envolvem a preparação dos dados e a identificação do modelo a ser estimado
- Nessas etapas iniciais, vimos tópicos como:
    1. Sazonalidade e tendência
    2. Transformação logarítmica
    3. Diferenciação e raíz unitária
    4. Critérios de identificação
- Após estas etapas iniciais, devemos estimar o modelo ARIMA de interesse

Estimação

- Modelos ARIMA são usualmente estimados através de funções de máxima verossimilhança
- Há duas opções para estimação destes modelos:

1. **Máxima verossimilhança condicional**: supõe que os choques iniciais são iguais a zero, utilizando a função densidade de distribuição condicional como aproximação da função densidade de distribuição conjunta
2. **Máxima verossimilhança exata**: trata os choques iniciais como parâmetros adicionais do modelo e estima eles conjuntamente com os outros parâmetros

Diagnóstico

- Se um modelo $ARIMA(p, d, q)$ representa bem os dados, então os resíduos serão próximos de um ruído branco
- Isto pode ser testado através de um teste de hipótese estatístico com a hipótese nula de resíduos não autocorrelacionados
- Para construir este teste, primeiro devemos estimar a autocorrelação dos resíduos:

$$\hat{r}_k = \frac{\sum_{t=k+1}^{n} \hat{\varepsilon}_t \hat{\varepsilon}_{t-k}}{\sum_{t=1}^{n} \hat{\varepsilon}_t^2}$$

Diagnóstico

- A partir da autocorrelação residual de uma modelo $ARIMA(p, d, q)$ calculada no slide anterior, podemos construir um teste de hipótese conhecido como o *teste de Ljung-Box*:

$$Q(K) = n(n+2)\sum_{k=1}^{K} \frac{\hat{r}_k^2}{(n-k)} \sim \chi^2(K-p-q)$$

- Sendo n o número de observações da série de tempo, K a defasagem máxima considerada no teste, e χ a distribuição qui-quadrada
- A hipótese nula do teste de *Ljung-Box* é a de resíduos não correlacionados
- Usualmente utilizamos valores de K iguais a 5, 10 e 15

Previsão

- A previsão de erro quadrático médio mínimo com origem T e horizonte h de um modelo $ARIMA(p, d, q)$ é dada por:

$$\hat{Y}_t(h) = E(c + \phi_1 Y_{T+h-1} + \ldots + \phi_p Y_{T+h-p} + \varepsilon_{T+h} + \theta_1 \varepsilon_{T+h-1} + \ldots + \theta_q \varepsilon_{T+h-q} | Y_T, Y_{T-1}, \ldots)$$

- Ou seja, é o valor esperado condicional de Y_{T+h} dado o passado $X_T, X_{T-1}, \ldots$

Previsão

Para calcular estas previsões, substituímos esperanças passadas por valores conhecidos e esperanças futuras por previsões:

1. $E(Y_{T+j}|Y_T, Y_{T-1}, \ldots) = \begin{cases} Y_{T+j}, & \text{se } j \leq 0 \\ \hat{Y}_T(j), & \text{se } j > 0 \end{cases}$
2. $E(\varepsilon_{T+j}|Y_T, Y_{T-1}, \ldots) = \begin{cases} \varepsilon_{T+j}, & \text{se } j \leq 0 \\ 0, & \text{se } j > 0 \end{cases}$

Exemplo no R: turismo na Austrália

Para o nosso exemplo, além dos pacotes utilizados na Aula 2 vamos também precisar do pacote `fable` para estimação dos modelos ARIMA:

```
library(tidyverse)
library(lubridate)
library(tsibble)
library(feasts)
library(fable)
```

Exemplo no R: turismo na Austrália

Vamos utilizar a base de dados de viagens domésticas na Austrália, agrupando o número de viagens em cada região por propósito:

```
data <- tourism %>%
  group_by(Purpose) %>%
  summarise(Trips = sum(Trips))
```

Autocorrelação

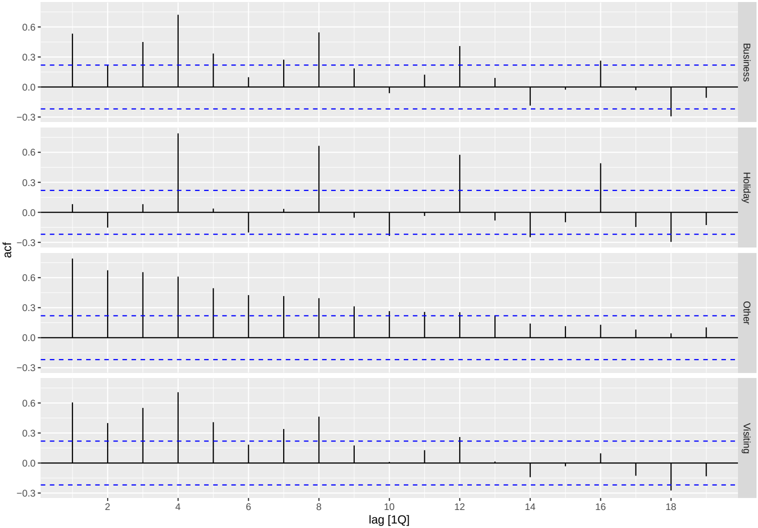
Na etapa de identificação, primeiro vamos plotar as funções de autocorrelação para cada série de tempo através da função `ACF`[2]:

```
data %>%
  ACF(log(Trips)) %>%
  autoplot()
```

Podemos observar uma persistência na autocorrelação das séries, além de correlações altas nos períodos sazonais

[2]Note que estamos utilizando o logaritmo natural da série de tempo antes para estabilizar a variância.

Autocorrelação



Teste de estacionariedade

Para verificar se as séries de tempo são estacionárias obtemos o número de diferenças necessárias para torná-las estacionárias de acordo com o teste KPSS:

```
data %>%
  features(log(Trips), unitroot_ndiffs)
```

```
## # A tibble: 4 x 2
##   Purpose  ndiffs
##   <chr>     <int>
## 1 Business      1
## 2 Holiday       1
## 3 Other         1
## 4 Visiting      1
```

Teste de estacionariedade

O mesmo teste pode ser aplicado para o período sazonal, verificando se existem raízes unitárias sazonais:

```
data %>%
  features(log(Trips), unitroot_ndiffs)
```

```
## # A tibble: 4 x 2
##   Purpose  ndiffs
##   <chr>     <int>
## 1 Business      1
## 2 Holiday       1
## 3 Other         1
## 4 Visiting      1
```

Sazonalidade e diferenciação

Primeiro vamos remover a sazonalidade dos dados através da função STL e depois vamos tirar a primeira diferença dos logaritmos para remover a raíz unitária:

```
data <- data %>%
  model(STL = STL(Trips)) %>%
  components() %>%
  select(Purpose, Quarter, Trips, season_adjust) %>%
  group_by(Purpose) %>%
  mutate(Var_Trips = difference(log(season_adjust)))
```

Sazonalidade e diferenciação

Agora vamos testar novamente a raíz unitária dos dados dessazonalizados e diferenciados:

```
data %>%
  features(Var_Trips, unitroot_ndiffs)
```

```
## # A tibble: 4 x 2
##   Purpose  ndiffs
##   <chr>     <int>
## 1 Business      0
## 2 Holiday       0
## 3 Other         0
## 4 Visiting      0
```

Sazonalidade e diferenciação

Também vamos testar novamente a raíz unitária sazonal dos dados dessazonalizados e diferenciados:

```
data %>%
  features(Var_Trips, unitroot_nsdiffs)
```

```
## # A tibble: 4 x 2
##   Purpose  nsdiffs
##   <chr>      <int>
## 1 Business       0
## 2 Holiday        0
## 3 Other          0
## 4 Visiting       0
```

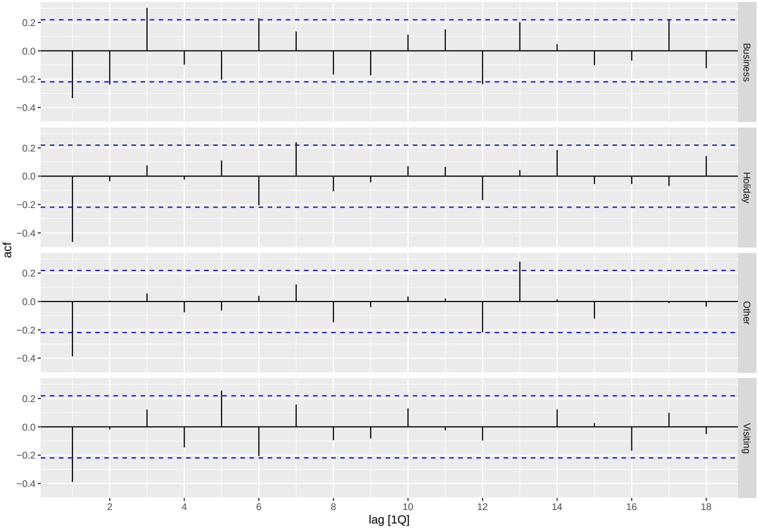
Autocorrelação

Depois de estacionarizar e dessazonalizar a série de tempo, podemos agora plotar novamente a autocorrelação:

```
data %>%
  ACF(Var_Trips) %>%
  autoplot()
```

A persistência na autocorrelação das séries desapareceu, bem como as correlações altas nos períodos sazonais

Autocorrelação

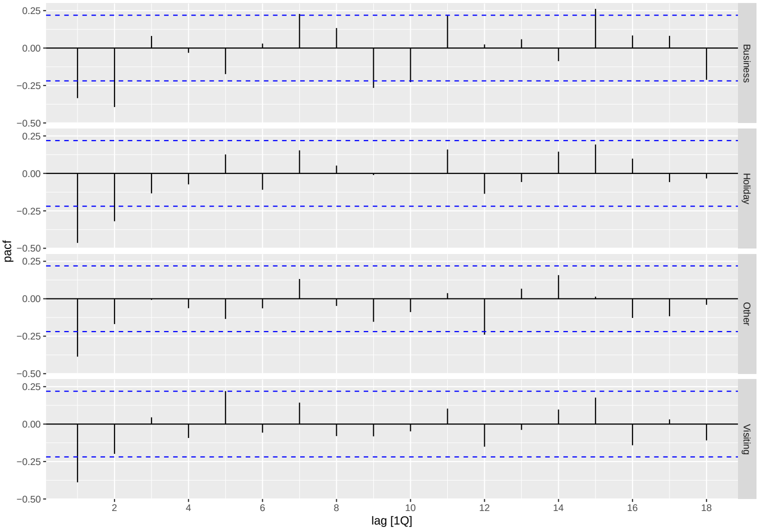


Autocorrelação parcial

Para conseguirmos identificar quais modelos estimar precisamos também observar as autocorrelações parciais para cada um das séries de tempo através da função `PACF`:

```
data %>%
  PACF(Var_Trips) %>%
  autoplot()
```

Autocorrelação parcial



Estimação

Agora vamos estimar quatro modelos possíveis com base na inspeção das funções `ACF` e `PACF` para cada uma das 4 séries de tempo:

```
my_arima <- data %>%
  model(
    MA_1 = ARIMA(Var_Trips ~ 1 + pdq(0,0,1)),
    MA_2 = ARIMA(Var_Trips ~ 1 + pdq(0,0,2)),
    AR_1 = ARIMA(Var_Trips ~ 1 + pdq(1,0,0)),
    AR_2 = ARIMA(Var_Trips ~ 1 + pdq(2,0,0))
  )
```

Podemos acessar os coeficientes de todos os modelos estimados através da função `tidy(my_arima)`

Estimação automática

A função `ARIMA` possui um algoritmo automatizado (Hyndman e Khandakar, 2008) para fazer todas as etapas de testes de estacionariedade utilizando KPSS, seleção das defasagens com base no critério de Akaike, e estimação dos modelos ARIMA com ou sem componentes sazonais:

```
arima_fit <- data %>%
  model(ARIMA = ARIMA(log(Trips)))
```

Para estimar um modelo ARIMA de maneira automática, basta omitir os argumentos após a variável dependente

Estimação automática

Novamente, podemos acessar os coeficientes através da função `tidy`:

```r
tidy(arima_fit)
```

```
## # A tibble: 11 x 7
##    Purpose  .model term  estimate std.error statistic  p.value
##    <chr>    <chr>  <chr>    <dbl>     <dbl>     <dbl>    <dbl>
##  1 Business ARIMA  ar1     -0.441    0.108      -4.10 1.05e- 4
##  2 Business ARIMA  ar2     -0.473    0.105      -4.49 2.51e- 5
##  3 Business ARIMA  sma1    -0.924    0.149      -6.21 2.74e- 8
##  4 Holiday  ARIMA  ma1     -0.686    0.0928     -7.39 1.71e-10
##  5 Holiday  ARIMA  sma1    -0.831    0.126      -6.59 5.45e- 9
##  6 Other    ARIMA  ma1     -0.532    0.105      -5.05 2.83e- 6
##  7 Other    ARIMA  sar1     0.209    0.113       1.85 6.79e- 2
##  8 Visiting ARIMA  ar1      0.899    0.0635     14.2  5.10e-23
##  9 Visiting ARIMA  ma1     -0.429    0.118      -3.63 5.19e- 4
## 10 Visiting ARIMA  sar1    -0.542    0.115      -4.70 1.14e- 5
## 11 Visiting ARIMA  sar2    -0.268    0.114      -2.36 2.11e- 2
```

Raízes do modelo ARIMA

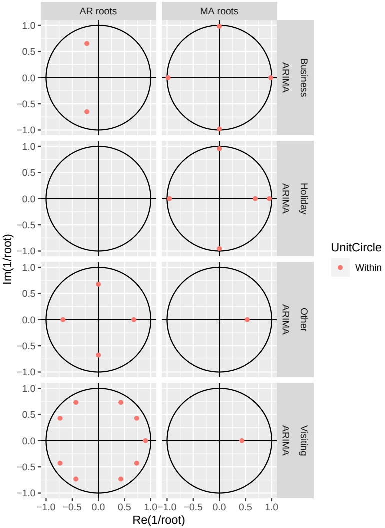
Após a estimação do modelo, podemos checar se as raízes do modelo estimado são menores do que o círculo unitário com o comando `gg_arma`:

```
gg_arma(arima_fit)
```

Quanto mais próximo do círculo unitário, mais instável é a estimação dos parâmetros[3]

[3] O algoritmo automatizado de Hyndman e Khandakar (2008) exclui modelos que contém coeficientes próximos a raíz unitária.

Raízes do modelo ARIMA



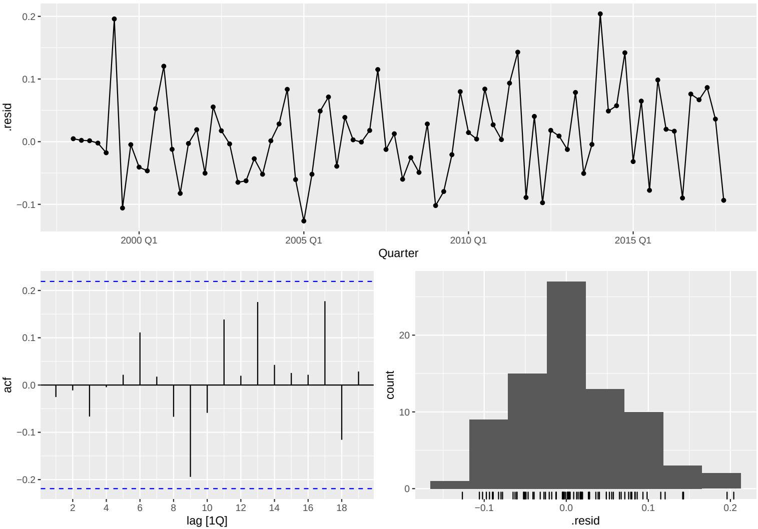
Gráficos dos resíduos

Podemos observar o comportamento dos resíduos para cada uma das séries de tempo com a função `gg_tsresiduals`

- Vamos utilizar como exemplo as viagens por negócios:

```
gg_tsresiduals(filter(arima_fit, Purpose == "Business"))
```

Gráficos dos resíduos



Teste de Ljung-Box

A outra etapa de diagnóstico é o teste de autocorrelação residual de `Ljung-Box`, que pode ser feito para todas as 4 séries de tempo através do comando `features` e da função `ljung_box`:

```
augment(arima_fit) %>%
  features(.resid, ljung_box, lag = 10, dof = c(2,3,4))
```

```
## # A tibble: 4 x 6
##   Purpose  .model lb_stat lb_pvalue1 lb_pvalue2 lb_pvalue3
##   <chr>    <chr>    <dbl>      <dbl>      <dbl>      <dbl>
## 1 Business ARIMA     5.84      0.666      0.559      0.442
## 2 Holiday  ARIMA     6.43      0.599      0.490      0.377
## 3 Other    ARIMA     7.23      0.512      0.405      0.300
## 4 Visiting ARIMA     6.49      0.592      0.484      0.370
```

Teste de Ljung-Box

Para calcular o teste de Ljung-Box, devemos especificar os argumentos:

- `lag`: número de defasagens (K) utilizado no teste;
- `dof`: número de graus de liberdade a serem reduzidos da distribuição devido ao teste ser aplicado nos resíduos (usualmente utiliza-se $dof = p + q$)

Note que no nosso exemplo utilizamos `dof = c(2,3,4)`, pois os modelos estimados contém 2, 3 ou 4 coeficientes

- Mais especificamente, devemos olhar para `lb_pvalue2` na série `Business`, `lb_pvalue1` na série `Holiday` e `Other`, e `lb_pvalue3` na série `Visiting`

Previsão

Depois de estimados os modelos, realizamos as previsões para horizonte `h` com a função `forecast`. O comando `hilo` nos mostra os intervalos de confiança das previsões:

```
arima_fc <- forecast(arima_fit, h = 10)
hilo(arima_fc, level = 95)
```

```
...
## # A tsibble: 40 x 6 [1Q]
## # Key:       Purpose, .model [4]
##    Purpose  .model Quarter              Trips .mean                  `95%`
##    <chr>    <chr>    <qtr>             <dist> <dbl>                 <hilo>
##  1 Business ARIMA  2018 Q1 t(N(8.5, 0.0052)) 4728. [4093.826, 5431.570]95
##  2 Business ARIMA  2018 Q2 t(N(8.7, 0.0068)) 5894. [4995.532, 6906.867]95
##  3 Business ARIMA  2018 Q3 t(N(8.7, 0.0072)) 6119. [5160.771, 7203.679]95
##  4 Business ARIMA  2018 Q4 t(N(8.6, 0.0092)) 5736. [4732.112, 6889.766]95
##  5 Business ARIMA  2019 Q1  t(N(8.5, 0.012)) 4929. [3968.256, 6052.213]95
##  6 Business ARIMA  2019 Q2  t(N(8.7, 0.013)) 5934. [4720.994, 7363.519]95
...
```

Previsão

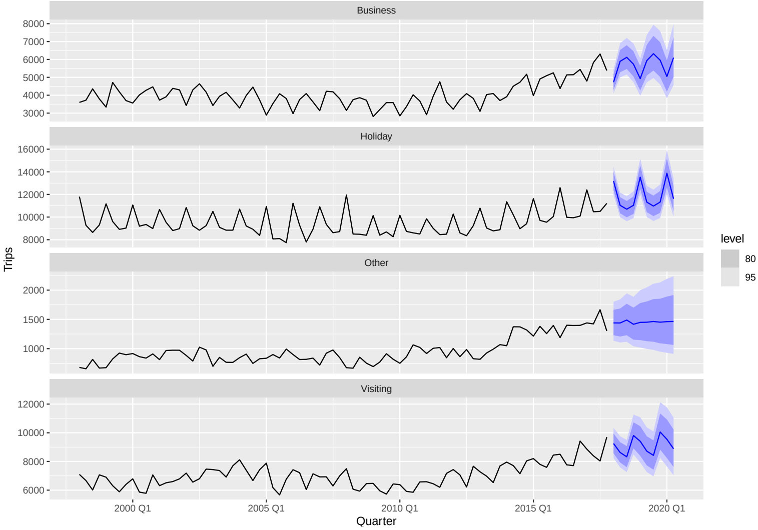
Por fim, podemos plotar as séries de tempo com as previsões e intervalos de confiança automaticamente no `R`:

```
arima_fc %>% autoplot(data)
```

Temos assim as previsões para o número de viagens por cada propósito para os próximos 10 trimestres[4]

[4] Note que a função `forecast` já efetua a chamada *back-transformation* ao fazer as previsões, aplicando o exponencial no logaritmo dos dados.

Previsão


Business
8000
7000
6000
5000
4000
3000
Holiday
16000
14000
12000
10000
8000
Other
2000
1500
1000
Visiting
12000
10000
8000
6000
Trips
2000 Q1
2005 Q1
2010 Q1
2015 Q1
2020 Q1
Quarter
level
80
95

Referências

Box, G. E. P. e Jenkins, G. M. (1970) Time series analysis: Forecasting and control, San Francisco: Holden-Day.

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