Séries de Tempo

Aula 6 - Outros modelos univariados

Regis A. Ely

Departamento de Economia
Universidade Federal de Pelotas

21 de agosto de 2020

Conteúdo

Outros modelos univariados

Alguns outros modelos univariados que podemos utilizar para previsão incluem:

- **Média**: a previsão de um valor futuro é sempre igual a média dos valores passados
- **Naive**: a previsão de um valor futuro é o valor imediatamente anterior (passeio aleatório)
- **Seasonal Naive**: a previsão de um valor futuro é o valor sazonal imediatamente anterior
- **Modelos dinâmicos aditivos**: modelos que incluem componentes sazonais, tendências e regressores exógenos

Identificação e seleção de modelos

- Em modelos ARIMA, usamos os critérios de identificação para selecionar entre modelos distintos
- Como estes critérios dependem do valor da função de verossimilhança, não podemos utilizá-los para comparar modelos com estruturas distintas[1]
- Nesse caso, devemos construir uma estratégia de validação das previsões e utilizar medidas de acurácia baseadas nos erros de previsão
- Pode-se demonstrar que o critério de Akaike corrigido é equivalente a uma validação do tipo *leave-one-out* (LOOCV)

[1] Também deve-se utilizar com cautela o critério AICc quando o horizonte de previsão é longo.

Validação

- Para avaliarmos a qualidade da previsão de um modelo, devemos construir uma estratégia de validação em treinamos o modelo em um pedaço dos dados (*conjunto treino*) e testamos a previsão em outro pedaço (*conjunto teste*)
- Existem diversas estratégias de validação possíveis, sendo que a escolha da estratégia depende do problema analisado
- As estratégia mais comum de validação para séries de tempo é a *rolling cross validation*

Rolling cross validation

- Antes de construirmos a estratégia de validação, podemos separar os dados em dois grupos (treino e teste), utilizando, por exemplo, 75% dos dados para o grupo treino e 25% para o grupo teste
- A estratégia *rolling cross validation* se aplica no conjunto treino, e define um conjunto inicial de observações (n), uma janela móvel (j) e um horizonte de previsão (h)
  - Primeiro estimamos o modelo nas n observações iniciais,
  - Depois avaliamos a previsão de horizonte h,
  - Depois estimamos o modelo novamente adicionando as j próximas observações,
  - Seguimos estas etapas até o fim da série de tempo

Medidas de performance

As medidas mais comuns para calcular a performance de modelos de previsão incluem:

- **Erro absoluto médio**: $MAE = \frac{\sum_{t=1}^{T} |\hat{Y}_t - Y_t|}{T}$
- **Raiz do erro quadrático médio**: $RMSE = \sqrt{\frac{\sum_{t=1}^{T} (\hat{Y}_t - Y_t)^2}{T}}$
- **Erro percentual médio absoluto**: $MAPE = \frac{1}{T} \sum_{t=1}^{T} \left| \frac{Y_t - \hat{Y}_t}{Y_t} \right|$

Exemplo no R

Para este exemplo iremos utilizar os pacotes e os dados da aula passada:

```
library(tidyverse)
library(lubridate)
library(tsibble)
library(feasts)
library(fable)
library(fasster)
data <- tourism %>%
  group_by(Purpose) %>%
  summarise(Trips = sum(Trips))
```

Conjunto treino

Para construir um procedimento de validação, vamos primeiro definir o horizonte de previsão h e então criar um conjunto treino excluindo as últimas h observações para medirmos a performance da previsão (conjunto teste):

```
h <- 10
train <- data %>%
  group_by(Purpose) %>%
  slice(1:(n()-h)) %>%
  ungroup()
```

Conjunto validação

A função `stretch_tsibble` possibilita criar conjuntos de validação utilizando janelas móveis de 10 em 10, começando com 20 observações:

```
train_cv <- train %>%
  group_by(Purpose) %>%
  slice(1:(n() - 10)) %>%
  stretch_tsibble(.init = 20, .step = h) %>%
  ungroup()
```

Uma nova coluna `.id` será criada com o identificador do conjunto validação. Ao todo são 60 observações por série de tempo (10 foram inicialmente excluídas), o que nos dá 5 grupos distintos (o primeiro grupo contém 20, o segundo 30, etc.)

Estimação

Vamos criar uma função para estimar vários modelos em apenas um comando:

```
models <- function(data) {
  data %>%
    model(
      MEAN = MEAN(log(Trips)),
      NAIVE = NAIVE(log(Trips)),
      SNAIVE = SNAIVE(log(Trips)),
      ETS = ETS(log(Trips)),
      ARIMA = ARIMA(log(Trips)),
      DLM = FASSTER(log(Trips) ~ poly(1) + season(4))
    )
}
```

Estimação

Agora vamos estimar todos os modelos no conjunto treino e nos grupos do conjunto validação, realizando as previsões com a função `forecast`:

```
train_fit <- models(train)
train_cv_fit <- models(train_cv)
train_fc <- forecast(train_fit, h = 10)
cv_fc <- forecast(train_cv_fit, h = 10)
```

Performance das previsões

Com a função `accuracy` podemos calcular a performance das previsões no conjunto treino e validação:

```
acc_train <- accuracy(train_fit)
acc_cv <- cv_fc %>% accuracy(train)
```

Minimizar o erro de previsão no conjunto treino pode levar a `overfitting`, por isso precisamos olhar para os erros de previsão no conjunto validação e no conjunto teste

Performance das previsões

```
acc_train %>% print(n = 24)
```

```
## # A tibble: 24 x 10
##    Purpose  .model .type        ME   RMSE    MAE     MPE  MAPE  MASE     ACF1
##    <chr>    <chr>  <chr>     <dbl>  <dbl>  <dbl>   <dbl> <dbl> <dbl>    <dbl>
##  1 Business MEAN   Training  33.8   508.   396.  -0.909  10.6  1.34   0.255
##  2 Business NAIVE  Training  19.0   610.   532.  -0.881  14.2  1.80  -0.187
##  3 Business SNAIVE Training  50.2   398.   295.   0.688   7.41 1      0.276
##  4 Business ETS    Training  29.2   281.   210.   0.304   5.35 0.712  0.0173
##  5 Business ARIMA  Training  29.1   328.   236.   0.204   5.95 0.799 -0.0485
##  6 Business DLM    Training  37.6   330.   238.   0.452   5.98 0.806  0.0776
##  7 Holiday  MEAN   Training  52.5  1019.   804.  -0.537   8.34 2.02  -0.0805
##  8 Holiday  NAIVE  Training -30.4  1478.  1171.  -1.42   12.1  2.94  -0.354
##  9 Holiday  SNAIVE Training  10.6   528.   398.  -0.0379  4.26 1     -0.00209
## 10 Holiday  ETS    Training  14.0   417.   315.  -0.0179  3.34 0.792 -0.0624
## 11 Holiday  ARIMA  Training  11.2   483.   371.  -0.0506  3.97 0.934  0.154
## 12 Holiday  DLM    Training   7.47  472.   363.  -0.0696  3.86 0.912 -0.0403
## 13 Other    MEAN   Training  13.1   161.   117.  -1.38   12.8  0.998  0.659
## 14 Other    NAIVE  Training  10.2   117.    95.2  0.146  10.8  0.813 -0.271
## 15 Other    SNAIVE Training  37.4   148.   117.   2.55   12.8  1      0.494
## 16 Other    ETS    Training  18.5    97.0   78.1  1.04    8.81 0.667  0.0404
## 17 Other    ARIMA  Training  17.6   106.    86.9  0.867   9.77 0.742  0.0834
## 18 Other    DLM    Training  25.2   113.    91.3  1.62   10.1  0.779  0.255
## 19 Visiting MEAN   Training  30.8   650.   546.  -0.454   8.04 1.34   0.481
## 20 Visiting NAIVE  Training  10.2   655.   531.  -0.335   7.84 1.30  -0.192
## 21 Visiting SNAIVE Training  65.8   504.   408.   0.636   5.99 1      0.374
## 22 Visiting ETS    Training  29.1   330.   259.   0.226   3.85 0.636 -0.0674
## 23 Visiting ARIMA  Training  33.8   379.   295.   0.299   4.38 0.724 -0.0401
## 24 Visiting DLM    Training  40.9   388.   295.   0.354   4.37 0.724  0.127
```

Performance das previsões

```
acc_cv %>% arrange(Purpose) %>% print(n = 24)
```

```
## # A tibble: 24 x 10
##    .model Purpose  .type      ME  RMSE   MAE      MPE  MAPE  MASE    ACF1
##    <chr>  <chr>    <chr>   <dbl> <dbl> <dbl>    <dbl> <dbl> <dbl>   <dbl>
##  1 ARIMA  Business Test    11.5   498.  392. -0.790   10.2   1.45  0.693
##  2 DLM    Business Test    40.3   485.  367.  0.0535   9.51  1.36  0.632
##  3 ETS    Business Test    69.1   458.  346.  0.728    8.97  1.28  0.646
##  4 MEAN   Business Test  -111.    567.  438. -5.06    12.2   1.63  0.329
##  5 NAIVE  Business Test  -214.    636.  506. -7.73    14.4   1.88  0.339
##  6 SNAIVE Business Test     7.19  516.  406. -0.791   10.5   1.51  0.569
##  7 ARIMA  Holiday  Test    36.1   649.  508.  0.00848  5.47  1.34  0.266
##  8 DLM    Holiday  Test     9.49  635.  502. -0.273    5.41  1.32  0.279
##  9 ETS    Holiday  Test    80.9   611.  493.  0.556    5.27  1.30  0.445
## 10 MEAN   Holiday  Test  -169.   1074.  929. -3.03     9.97  2.44 -0.0566
## 11 NAIVE  Holiday  Test   -44.0  1090.  893. -1.54     9.39  2.35 -0.0636
## 12 SNAIVE Holiday  Test    14.4   670.  529. -0.249    5.73  1.39  0.204
## 13 ARIMA  Other    Test    42.6   197.  146.  1.40    15.1   1.42  0.723
## 14 DLM    Other    Test    28.0   180.  131.  0.0664  13.8   1.28  0.622
## 15 ETS    Other    Test    31.8   189.  145.  0.252   15.2   1.41  0.726
## 16 MEAN   Other    Test    66.8   186.  128.  4.35    12.7   1.24  0.687
## 17 NAIVE  Other    Test   -12.3   199.  159. -4.85    17.2   1.55  0.749
## 18 SNAIVE Other    Test    26.4   185.  138.  0.0353  14.6   1.34  0.457
## 19 ARIMA  Visiting Test    41.1   742.  638. -0.0338   9.28  1.51  0.587
## 20 DLM    Visiting Test    93.2   610.  523.  0.758    7.66  1.24  0.544
## 21 ETS    Visiting Test   196.    655.  560.  2.21     8.05  1.33  0.610
## 22 MEAN   Visiting Test   181.    754.  644.  1.57     9.23  1.53  0.548
## 23 NAIVE  Visiting Test  -108.    774.  632. -2.19     9.48  1.50  0.513
## 24 SNAIVE Visiting Test    67.5   642.  561.  0.379    8.21  1.33  0.399
```

Seleção do melhor modelo

Os melhores modelos devem ser aqueles que minimizam o erro de previsão no conjunto validação. Iremos utilizar o RMSE para escolher:

```
best_model <- acc_cv %>%
  group_by(Purpose) %>%
  filter(RMSE == min(RMSE))
best_model
```

```
## # A tibble: 4 x 10
## # Groups:   Purpose [4]
##   .model Purpose  .type    ME  RMSE   MAE    MPE  MAPE  MASE  ACF1
##   <chr>  <chr>    <chr> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 DLM    Other    Test   28.0  180.  131. 0.0664 13.8   1.28 0.622
## 2 DLM    Visiting Test   93.2  610.  523. 0.758   7.66  1.24 0.544
## 3 ETS    Business Test   69.1  458.  346. 0.728   8.97  1.28 0.646
## 4 ETS    Holiday  Test   80.9  611.  493. 0.556   5.27  1.30 0.445
```

Seleção do melhor modelo

Agora podemos estimar os melhores modelos no conjunto treino completo e calcular as previsões:

```
good_models <- train %>%
  model(
    ETS = ETS(log(Trips)),
    DLM = FASSTER(log(Trips) ~ poly(1) + season(4))
  )
train_fc <- forecast(good_models, h = h)
```

Seleção do melhor modelo

Por fim, podemos avaliar a performance do melhor modelo no conjunto teste que contém 10 observações não utilizadas durante todo o procedimento:

```
train_fc %>% accuracy(data) %>% arrange(Purpose)
```

```
## # A tibble: 8 x 10
##   .model Purpose  .type    ME  RMSE   MAE   MPE  MAPE  MASE   ACF1
##   <chr>  <chr>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 DLM    Business Test  536.   643. 536.  9.86   9.86 1.82   0.473
## 2 ETS    Business Test  493.   621. 533.  9.07   9.85 1.81   0.396
## 3 DLM    Holiday  Test  891.   996. 891.  8.24   8.24 2.24   0.164
## 4 ETS    Holiday  Test  906.  1003. 906.  8.37   8.37 2.28   0.241
## 5 DLM    Other    Test   50.7  122.  86.3 3.12   5.97 0.737 -0.217
## 6 ETS    Other    Test   11.3  110.  90.2 0.245  6.46 0.770 -0.123
## 7 DLM    Visiting Test  656.   792. 661.  7.52   7.59 1.62   0.323
## 8 ETS    Visiting Test  582.   721. 582.  6.65   6.65 1.43   0.323
```

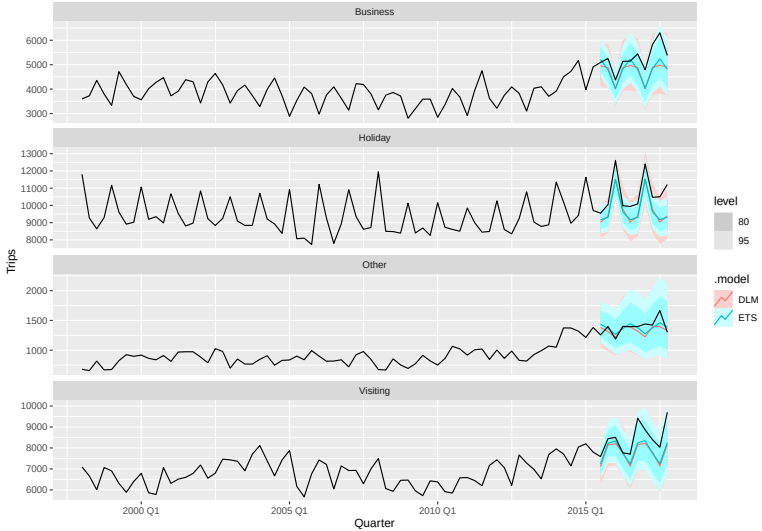
Seleção do melhor modelo

Podemos comparar graficamente as previsões para estas 10 observações com os dados verdadeiros através da função `autoplot`:

```
train_fc %>% autoplot(data)
```

Observa-se que alguns modelos parecem não estar captando bem a tendência de crescimento mais recente nas séries de tempo. Como você corrigiria isso?

Seleção do melhor modelo



Previsões finais

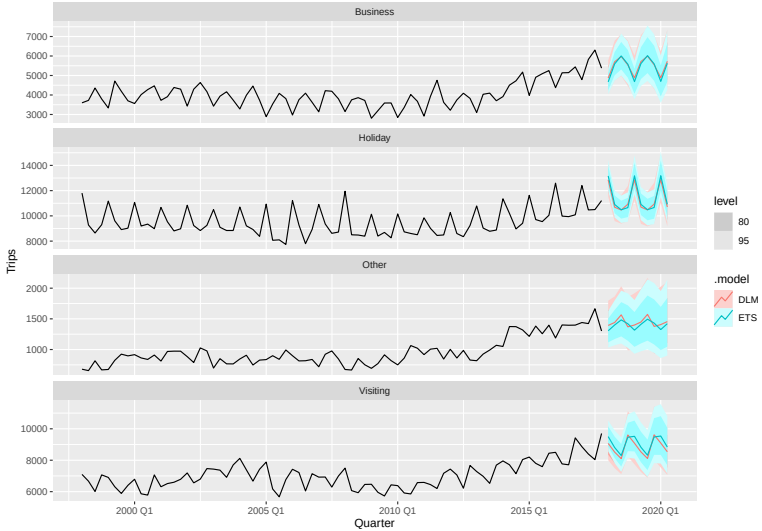
Por fim, podemos estimar os melhores modelos no conjunto total de dados para fazer previsões para períodos desconhecidos:

```
final_models <- data %>%
  model(
    ETS = ETS(log(Trips)),
    DLM = FASSTER(log(Trips) ~ poly(1) + season(4))
  )
final_fc <- forecast(final_models, h = h)
```

E plotar as previsões finais:

```
final_fc %>% autoplot(data)
```

Previsões finais



Previsões finais

Como podemos melhorar as previsões destes modelos?

- Considerar diferentes estratégias de validação (divisão entre conjuntos treino, teste e validação)
- Utilizar parâmetros diferentes nas estimações dos modelos
- Estimar outros modelos e tirar a média de previsões de modelos distintos
- Utilizar outras medidas de performance