Prevendo a Produção de Leite no Paraná

Everton Artuso e Hevans Vinícius Pereira

06/06/2022

Introdução

O Brasil ocupa a quarta colocação em produção mundial de leite, participando com aproximadamente 5,1% da produção mundial, considerando os dados de 2016 (FAO, 2018; IBGE, 2018), conforme https://sindileiteparana.com.br/dados-do-setor/.

A produção de leite do estado do Paraná alcançou recentemente o segundo lugar no ranking nacional, atrás apenas do estado de Minas Gerais e pouco a frente do Rio Grande do Sul, com aproximadamente 4,4 bilhões de litros por ano, e representa a cadeia produtiva mais importante no contexto da agricultura familiar no estado. Em 10 anos, de 2008 a 2018, a produção no estado se elevou em 55% e alcançou tal marca com quase 1,4 milhões de vacas ordenhadas, de acordo com https://www.idrparana.pr.gov.br/Pagina/Bovinocultura-de-Leite.

A produtividade e a renda dos produtores tem sido continuamente aumentadas, o que é muito importante uma vez que o leite é a principal fonte de renda de uma parcela significativa das famílias que atuam no ramo. Tais feitos tem sido alcançados graças a investimentos contínuos em equipamentos, melhoramento genético, fertilização do solo e otimização de processos, além da capacitação profissional oferecida pelo extensionismo rural através da Emater e prefeituras municipais.

Optamos por utilizar dados coletados no banco do IPEADATA, e para diferenciarmos um pouco nossa análise das mais frequentemente realizadas, escolhemos trabalhar com o agrupamento da produção por mesorregiões (ver Figura 1). O Paraná conta com dez mesorregiões geográficas, cada uma com suas particularidades e desafios específicos. Entre as mesorregiões com maior crescimento nos últimos anos presentes nos dados apresentados, destacam-se as mesorregiões Sudoeste e Oeste do estado (ver Figura 2) e, por conta disso, nos concentraremos na análise de modelos de séries temporais para essas mesorregiões específicas.

O banco de dados disponível no site do IPEADATA conta com dados de produção leiteira anuais de 1974 a 2016 e pode ser acessado em http://www.ipeadata.gov.br/Default.aspx -> Regional -> Temas -> Agropecuária -> Produção - leite - quantidade (anual) podem ser escolhidas as mesorregiões do estado do Paraná, além da janela de tempo de interesse.

Modelagem com Séries Temporais

Análise Preliminar

Carregando os pacotes que serão usados ao longo do trabalho:

```
library(tidyverse)
library(readxl)
library(ggfortify)
library(cowplot)
```

Figure 1: Mesorregiões do Paraná. Fonte: http://www.baixarmapas.com.br/mapa-do-parana-mesorregioes/

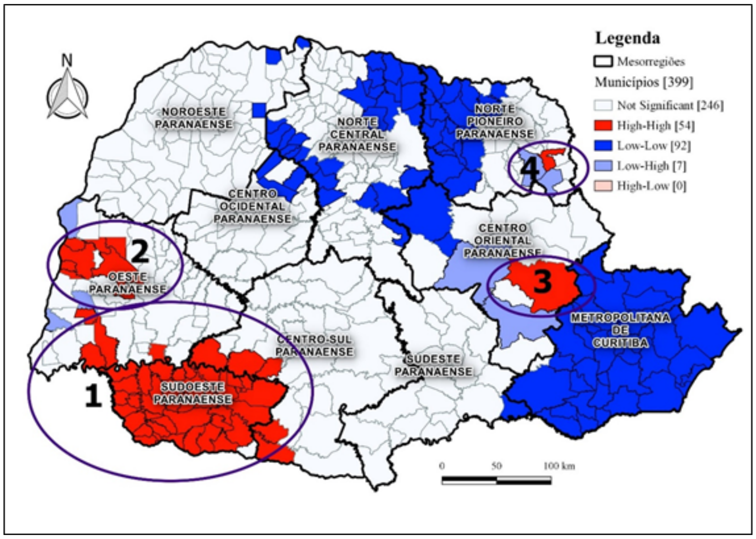


Figure 2: Produção por Mesorregião. Fonte: https://www.redalyc.org/journal/5520/552068861021/html/

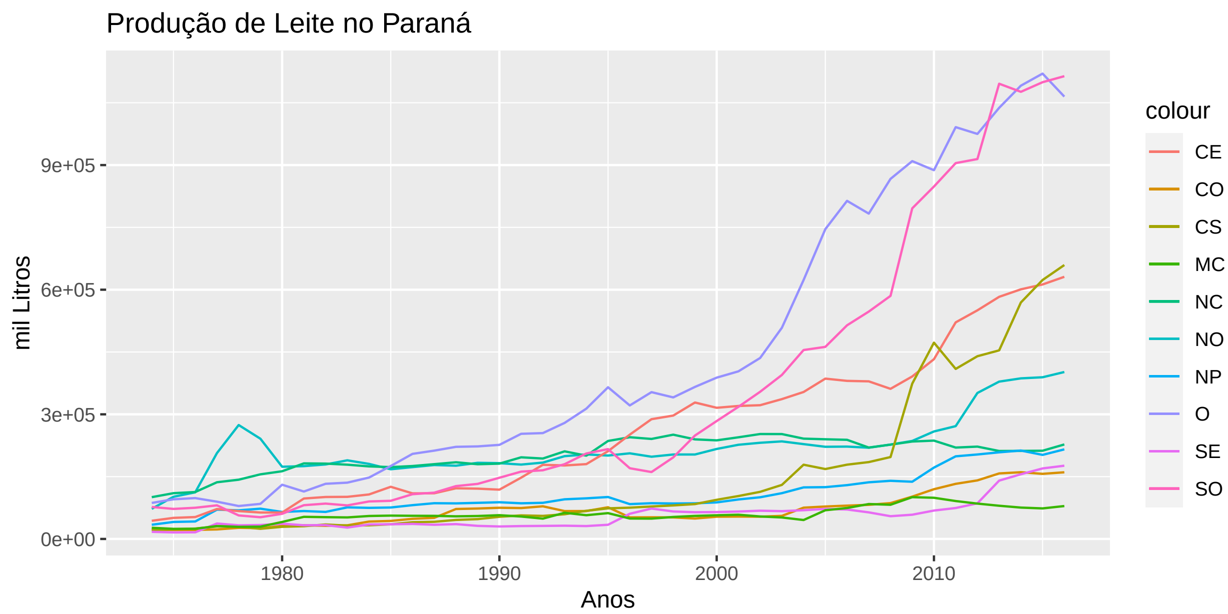
```
library(patchwork)
library(forecast)
library(TSstudio)
library(plotly)
library(h2o)
library(reshape2)
```

Vamos importar os dados e fazer alguns pequenos tratamentos para que possamos utilizá-los depois.

```
leite_mesorregioes <- read_excel('F:/Dropbox/SeriesTemporais/ipeadata_leite.xls')

# eliminando colunas irrelevantes
leite_meso <- leite_mesorregioes[,-c(1:3)]

# transpondoo dataframe
leite_meso <- t(leite_meso)

# arrumando o indice
rownames(leite_meso) <- NULL

# nomeando as colunas
colnames(leite_meso) <- c('NO','CO','NC','NP','CE','O','SO','CS','SE','MC')

# convertendo para dataframe
leite_meso <- as.data.frame(cbind(Ano = c(1974:2016), leite_meso))

# visualizando o formato final dos dados
head(leite_meso)
```

```
##    Ano     NO    CO     NC    NP    CE     O    SO    CS    SE    MC
## 1 1974  72551 22939 100591 34187 43736 86731 76945 21557 17158 26914
## 2 1975 101507 21536 110175 40920 50737 95453 72258 23339 15723 24271
## 3 1976 112234 21821 112926 42046 52725 98353 75100 24976 16032 24511
## 4 1977 206671 22945 136449 70468 72033 89696 80895 30149 37160 31057
## 5 1978 274231 26776 142650 68879 67057 79276 56608 29374 32826 29120
## 6 1979 241260 26068 155549 72899 63601 84474 52237 24430 33509 29651
```

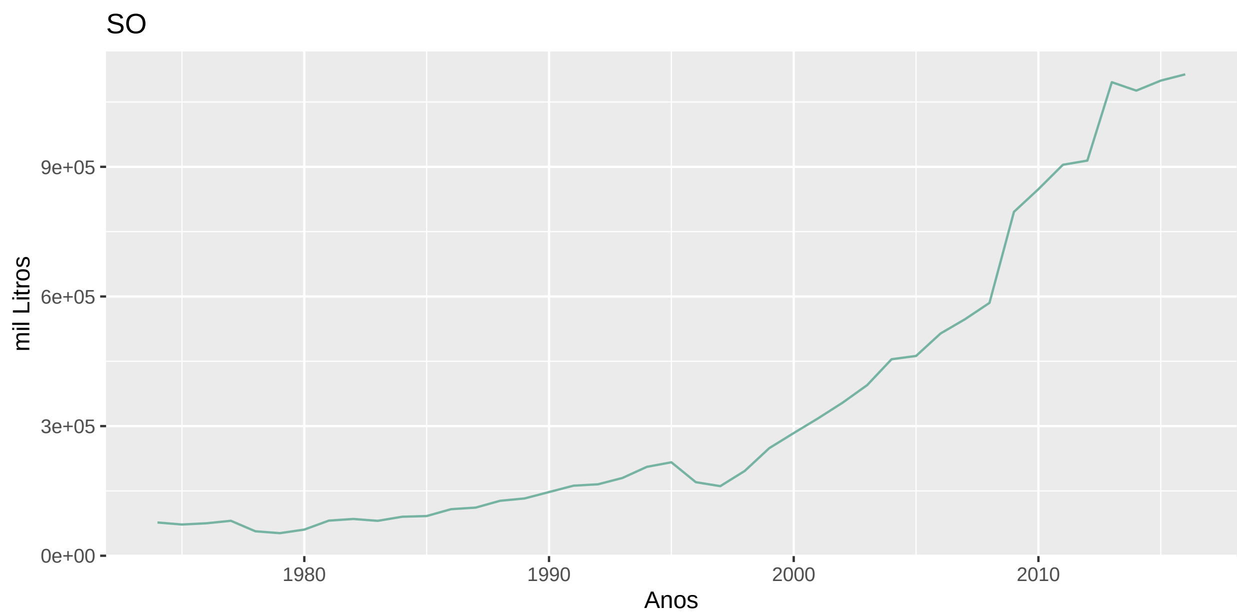
Agora que os dados estão prontos, podemos visualizar a produção de leite por mesorregião.



Claramente, as regiões Oeste e Sudoeste do estado concentram boa parte da produção total. Vamos nos ater a produção e a série histórica destas duas principais regiões.

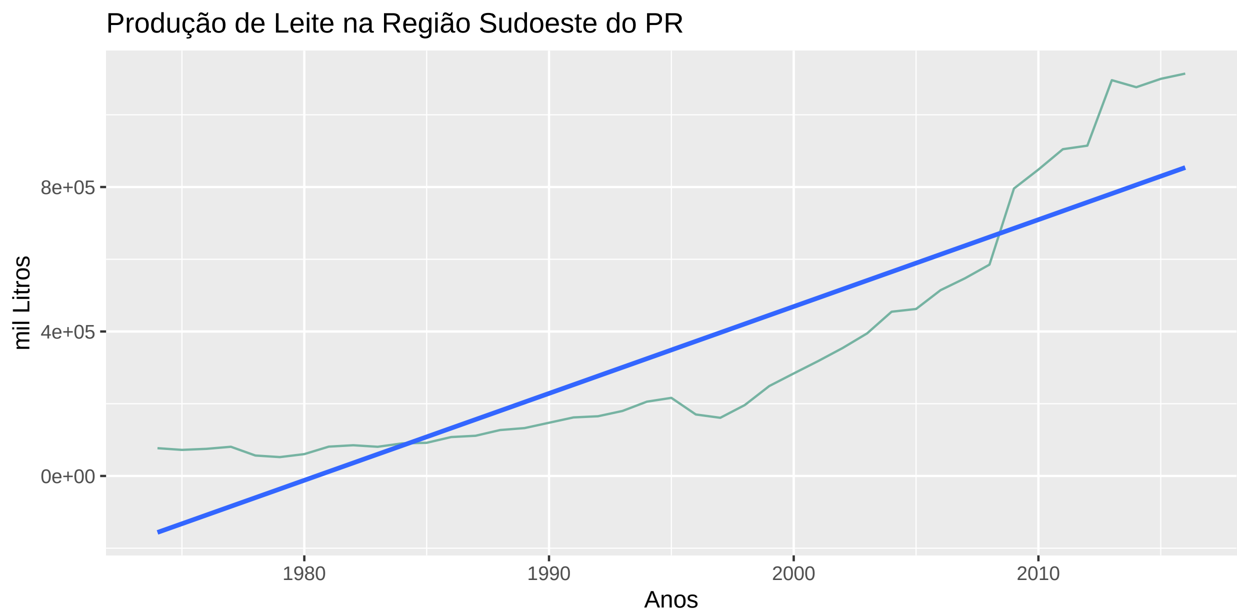
Região Sudoeste

Começando pela região sudoeste, à qual corresponde o gráfico a seguir:



Podemos observar que até meados dos anos 90 a produção de leite na região sudoeste do Paraná tinha um comportamento e que após 1997 a produção teve uma elevação bastante considerável. Não sabemos exatamente quais variáveis influenciaram no aumento da produção, mas é provável que o aumento do poder de compra do brasileiro nos anos 2000 tenha tido alguma influência.

Podemos ajustar um modelo linear para ter uma noção da tendência apresentada pela série.



Executando o comando **auto.arima**, podemos investigar um primeiro modelo, ARIMA não sazonal, candidato a modelar o comportamento da série histórica.

```
auto.arima(sudoeste, seasonal = FALSE)
```

```
## Series: sudoeste
## ARIMA(0,2,1)
##
## Coefficients:
##           ma1
##       -0.8396
## s.e.   0.0782
##
## sigma^2 estimated as 1.876e+09:  log likelihood=-496.01
## AIC=996.01   AICc=996.33   BIC=999.44
```

Obtemos um modelo ARIMA(0,2,1), ou seja, o modelo sugere que tomemos duas diferenças para eliminar a tendência e o que nos restará será um modelo de médias móveis de primeira ordem estacionário, o qual pode ser escrito como

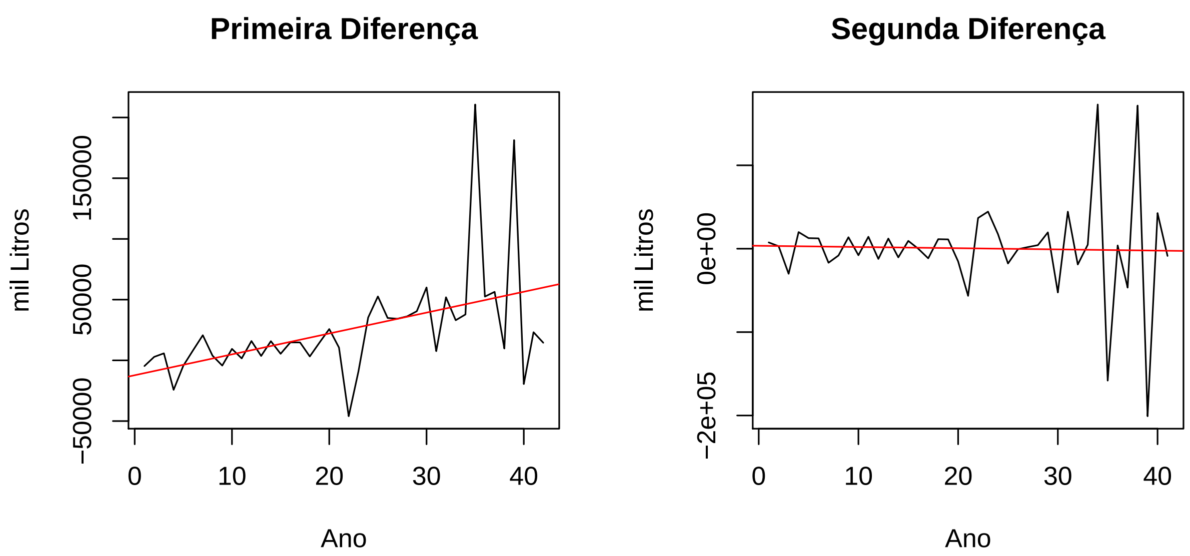
$$(1-B)^2 x_t = (1-0,84B)\varepsilon_t,$$

em que B é o operador *lag* e ε_t é o erro.

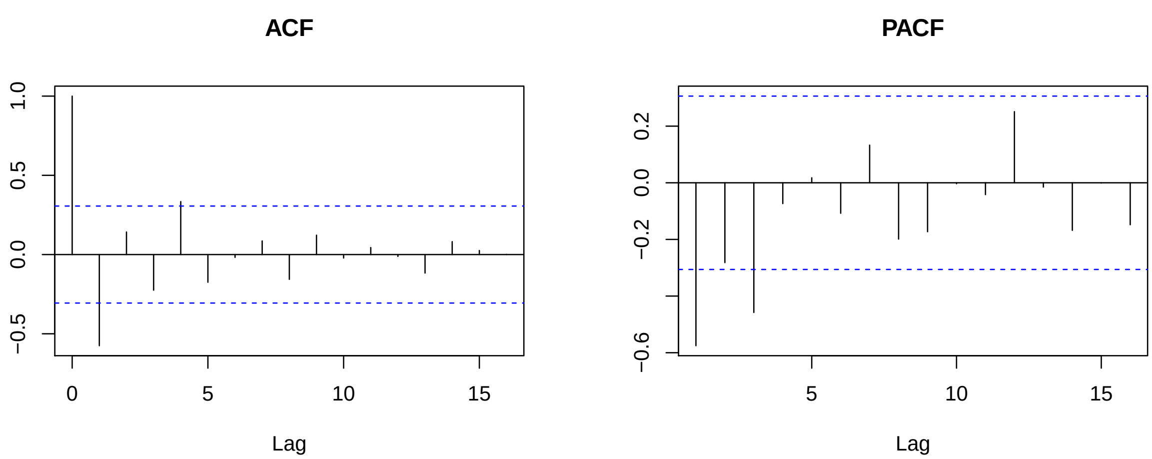
De fato, a tendência é eliminada após a segunda diferença, conforme gráficos a seguir.

```
seq1 <- 1:42
dif1 <- diff(sudoeste)
trend2 <- lm(formula = dif1 ~ seq1, data=as.data.frame(dif1))
seq2 <- 1:41
dif2 <- diff(dif1)
trend3 <- lm(formula = dif2 ~ seq2, data = as.data.frame(dif2))
par(mfrow=c(1,2))
```

```r
ts.plot(dif1, ylab='mil Litros', xlab='Ano', main='Primeira Diferença') +
  abline(trend2, col = 'red')
ts.plot(dif2, ylab='mil Litros', xlab='Ano', main='Segunda Diferença') +
  abline(trend3, col = 'red')
```



Avaliando os resíduos para a segunda diferença, temos os seguintes gráficos:



A função de autocorrelação apresentou valor significativo para ordem um, o que indica um modelo de médias móveis de ordem 1. Por outro lado, a função de autocorrelação parcial apresentou valor significativo até ordem 3, o que indica um modelo autorregressivo de ordem 3, informação que difere daquela dada pelo modelo sugerido por auto.arima (AR(0) e MA(1)). Por conta disso, vamos comparar com o modelo ARIMA(3,2,1).

Comecemos analisando um modelo ARIMA(0,2,1) para os dados originais.

```r
arima(x = sudoeste, order = c(0,2,1), xreg = time(sudoeste))
```

```
##
```

```
## Call:
## arima(x = sudoeste, order = c(0, 2, 1), xreg = time(sudoeste))
##
## Coefficients:
##           ma1  time(sudoeste)
##       -0.8396        17834.35
## s.e.   0.0782             NaN
##
## sigma^2 estimated as 1.83e+09:  log likelihood = -496.01,  aic = 998.01
```

Podemos comparar com um modelo ARIMA(3,0,1) para a segunda diferença.

```r
arima(x = dif2, order = c(3,0,1), xreg = time(dif2))
```

```
##
## Call:
## arima(x = dif2, order = c(3, 0, 1), xreg = time(dif2))
##
## Coefficients:
##           ar1      ar2      ar3      ma1  intercept  time(dif2)
##       -0.8788  -0.6956  -0.6199  -0.0090   2438.529    -59.0575
## s.e.   0.2449   0.2218   0.1590   0.3021   4157.652    175.7512
##
## sigma^2 estimated as 1.498e+09:  log likelihood = -492.3,  aic = 998.6
```

Podemos ver que os valores de AIC e de `log likelihood` são muito próximos, indicando que qualquer um dos dois modelos estaria adequado. Portanto, podemos seguir com o modelo mais simples que é ARIMA(0,2,1).

Mas, analisando o gráfico da segunda diferença (anteriormente apresentado), é possível ver que a variância não é constante e isto indica que devemos fazer alguma mudança antes de considerar modelos AR, MA ou ARMA pois estes supõe série estacionária, ou considerar modelos diferentes.

Modelos de Médias Móveis

Vamos separar apenas os dados da região sudoeste e converter os dados para o formato de séries temporais reconhecido pelo R.

```r
leite_SO <- leite_meso[,c('Ano','SO')]
leite_SO <- ts(leite_SO$SO, start=leite_SO$Ano[1], frequency = 1)
```

Agora, vamos criar uma função que retorna a série com n lags.

```r
lags <- function(serie, n){
  ts_merged <- NULL

  # Criando n lags
  for(i in 1:n){
    ts_merged <- ts.union(ts_merged, stats::lag(serie, k = -i))
  }

  # Unindo os lags com a série original
  ts_merged <- ts.union(serie, ts_merged)
```

```
  # Nomeando as colunas
  colnames(ts_merged) <- c("y", paste0("y_", 1:i))

  # Removendo valores ausentes criados nos lags
  ts_merged <- window(ts_merged,
                      start = start(serie) + n,
                      end = end(serie))
  return(ts_merged)
}
```

Vamos também criar uma função que calcula a média aritmética simples de uma série com n lags.

```
ts_mean <- function(serie){
  ts_avg <- ts_sum(serie) / dim(serie)[2]
  return(ts_avg)
}
```

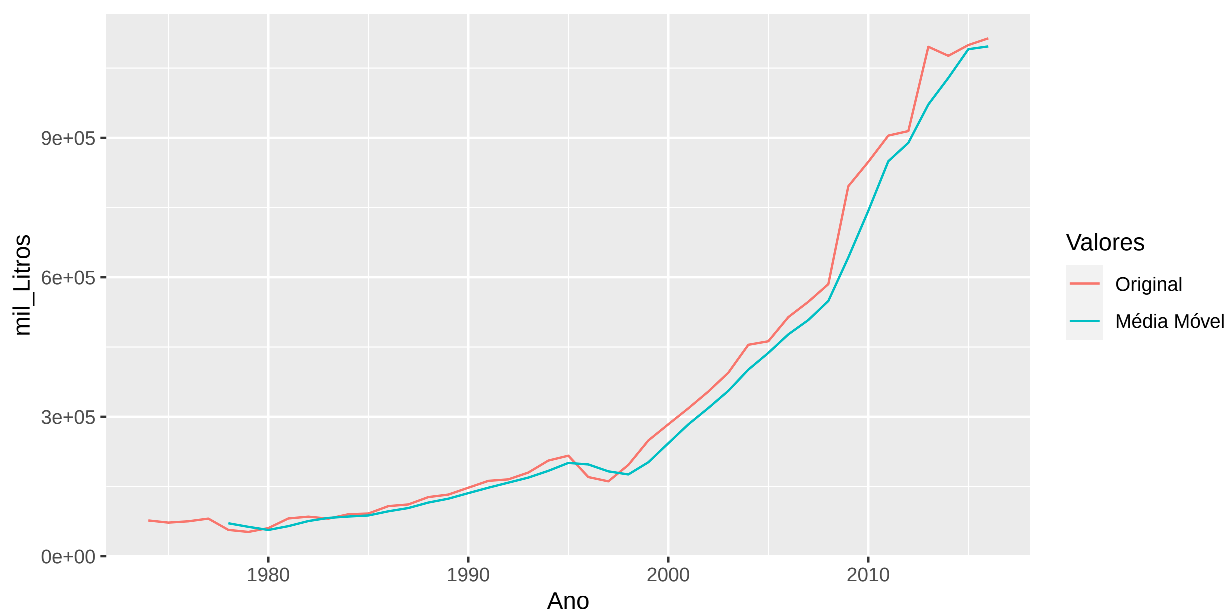
Por fim, vamos combinar as duas funções anteriores para criar uma função que retorna uma série suavizada.

```
sma <- function(serie, ordem){
  l <- ordem - 1
  l <- lags(serie = serie, n = l)
  m <- ts_mean(l)
  u <- ts.union(serie, m)
  colnames(u) <- c("original", "transformada")
  return(u)
}
```

Vamos criar uma série do tipo MA com média simples e ordem 3.

```
sma_3 <- sma(leite_SO, ordem = 3)
```

Podemos plotar a série original e a série suavizada para observamos as diferenças.



Poderíamos usar esta estratégia para reduzir efeitos sazonais, mas no nosso caso não há tais efeitos visto que os valores analisados são anuais.

Forecasting

Para fazer previsões precisamos separar nossos dados em dois conjuntos, um deles (treino) serve para criar o modelo e o outro (teste) serve para ver se nosso modelo está realmente acertando as previsões.

Essa separação pode ser feita de maneira simples no R. Vamos deixar os últimos 12 valores para teste.

```
treino <- window(leite_SO,
                 start = time(leite_SO)[1],
                 end = time(leite_SO)[length(leite_SO) - 12])

teste <- window(leite_SO,
                 start = time(leite_SO)[length(leite_SO) - 12 + 1],
                 end = time(leite_SO)[length(leite_SO)])
```

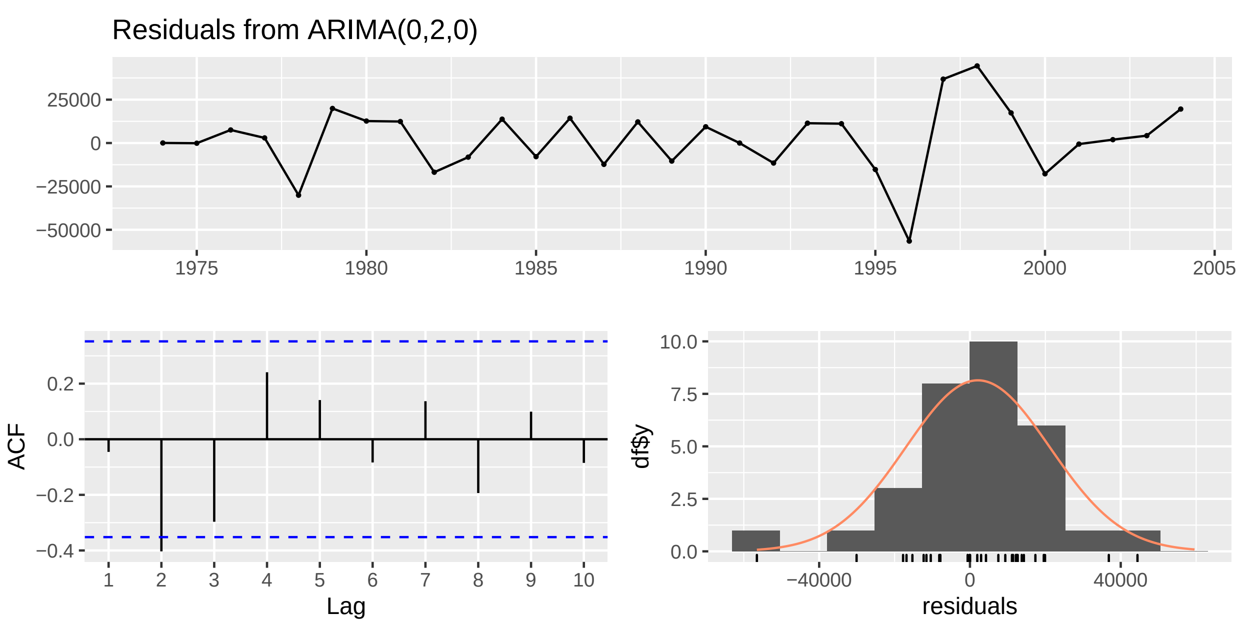
Podemos criar o modelo usando novamente a função `auto.arima`, mas agora considerando apenas o conjunto de treino.

```
md <- auto.arima(treino)
md
```

```
## Series: treino
## ARIMA(0,2,0)
##
## sigma^2 estimated as 384528173:  log likelihood=-327.78
## AIC=657.56   AICc=657.7   BIC=658.92
```

Podemos ver que o modelo sugerido foi ARIMA(0,2,0) diferente do modelo sugerido quando se considerava toda a série. Esta indicação sugere que há uma forte componente linear na série de treino.

Podemos verificar os resíduos para ver a qualidade do ajuste do modelo.



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,2,0)
## Q* = 12.292, df = 6, p-value = 0.05577
##
## Model df: 0.   Total lags used: 6
```

Observa-se que os resíduos apresentam média zero e variância constante, a não ser por uma pequena região entre os anos de 1995 e 2000. Em conjunto com a ACF e a distribuição dos resíduos, parece que o modelo tem um bom ajuste.

Também foi apresentado o teste de Ljung-Box. Este é um tipo de teste estatístico para verificar se há alguma autocorrelação da série diferente de zero. Podemos ver que temos um p-valor de 0.05577 e isso indica que não podemos rejeitar a hipótese nula a um nível de 5% de significância. Em outras palavras, pelo teste de Ljung-Box podemos afirmar com 95% de confiança que a auto correlação entre todos os lags é nula, o que está de acordo com o plot da ACF apresentado. O teste de Ljung-Box é uma maneira mais formal de confirmar o que o ACF estava sugerindo.

Vamos usar o modelo treinado para fazer previsões para as 12 observações deixadas como teste e, então, avaliar a performance do modelo.

Podemos fazer as previsões para o conjunto de teste.

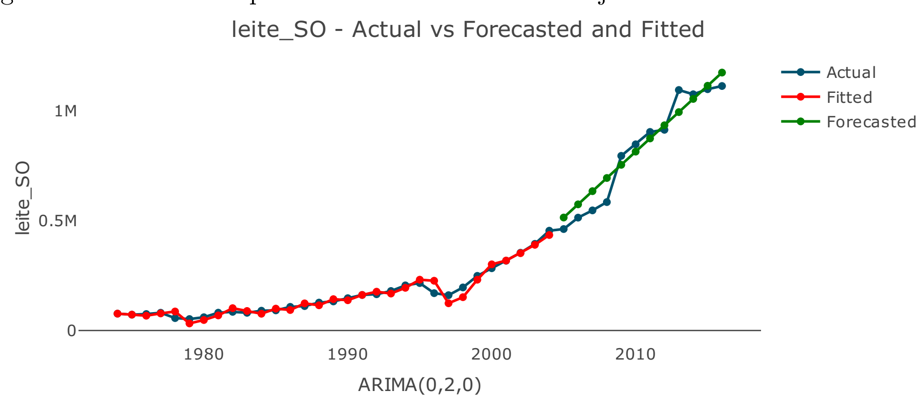
```
forecast_arima020 <- as.data.frame(forecast(md, h=12))[,1]
```

Vamos calcular algumas métricas para ver a performance do modelo no conjunto de treino e de teste.

```
fc <- forecast(md, h = 12)
accuracy(fc, teste)
```

```
##                      ME     RMSE      MAE       MPE      MAPE      MASE
## Training set   2085.313 18966.28 14177.23  1.237302 11.837163 0.7547825
## Test set     -15178.167 61115.46 52809.00 -3.630095  7.547441 2.8115018
##                     ACF1 Theil's U
## Training set -0.04546999        NA
## Test set      0.35109830 0.7429267
```

O erro no conjunto de teste ser maior que no conjunto de treino é algo natural. Podemos também observar graficamente o desempenho do modelo nos dois conjuntos.

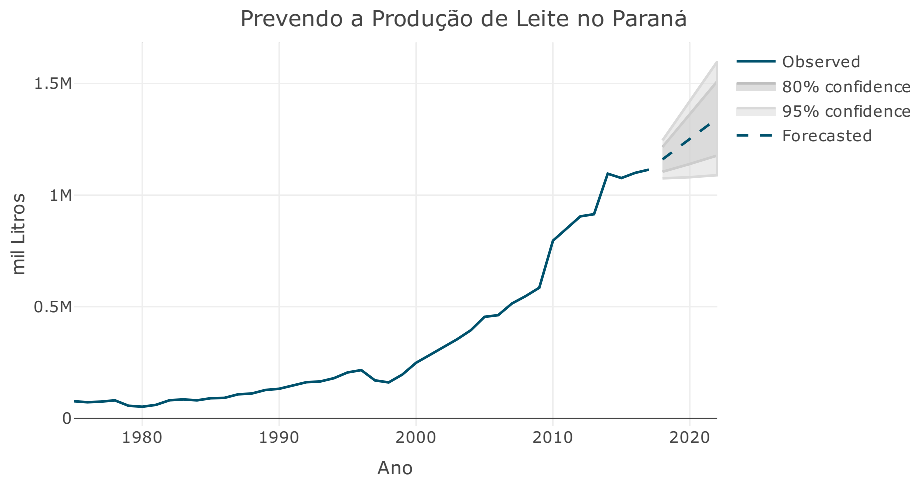


Para entender se os erros obtidos são altos, precisamos comparar com outros modelos comumente usados ou considerados satisfatórios. Ao longo deste trabalho apresentaremos mais modelos e, ao final, iremos comparar todos para ver qual tem melhor capacidade preditiva.

Após selecionar o melhor modelo, em geral, ele é treinado novamente, mas usando toda a série e, assim, pode-se tentar prever valores futuros. Vamos ilustrar esse procedimento com o modelo em questão e tentar prever cinco valores futuros.

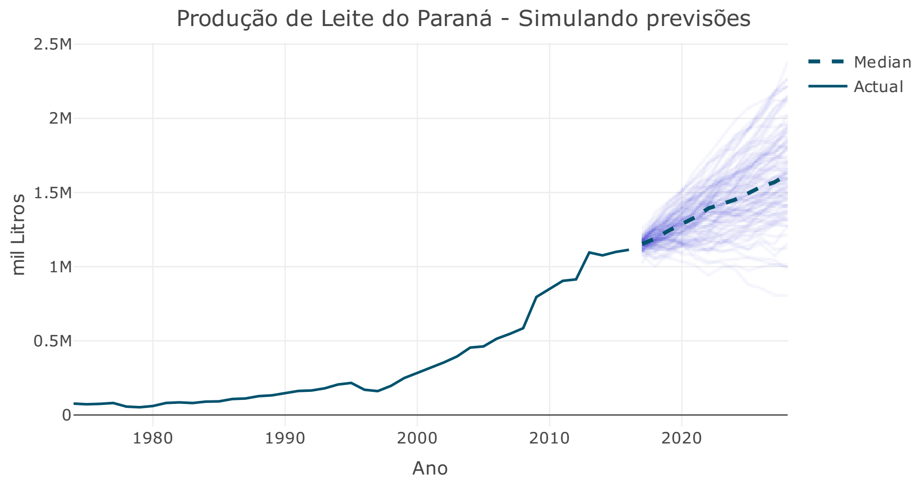
```
md_final <- auto.arima(leite_SO)
fc_final <- forecast(md_final, h = 5)
```

Podemos observar melhor as previsões no gráfico a seguir.



Outra maneira de observar a variação das predições é fazendo simulações de possíveis caminhos. Vamos criar 100 simulações que vão tentar prever os próximos 12 anos.

```
fc_final3 <- forecast_sim(model=md_final, h=12, n=100)

fc_final3_plotly <- fc_final3$plot %>%
            layout(title = "Produção de Leite do Paraná - Simulando previsões",
                   yaxis = list(title = "mil Litros"),
                   xaxis = list(title = "Ano"))
fc_final3_plotly
```



Uma maneira mais robusta de tentar obter o melhor modelo é treinar vários modelos com o conjunto de treino e avaliá-los com relação a suas performances no conjunto de teste, para então determinar qual foi o melhor. O pacote `TSstudio` conduz todo o processo de treino, teste, avaliação e forecasting.

Vamos testar os modelos `arima` para ARIMA e `ets` para Exponential Smoothing State Space com diferentes parâmetros.

```
methods <- list(ets1 = list(method = "ets",
                            method_arg = list(opt.crit = "lik"),
                            notes = "ETS model with opt.crit = lik"),
                ets2 = list(method = "ets",
                            method_arg = list(opt.crit = "amse"),
                            notes = "ETS model with opt.crit = amse"),
                arima1 = list(method = "arima",
                              method_arg = list(order = c(0,2,1)),
                              notes = "ARIMA(0,2,1)"))
```

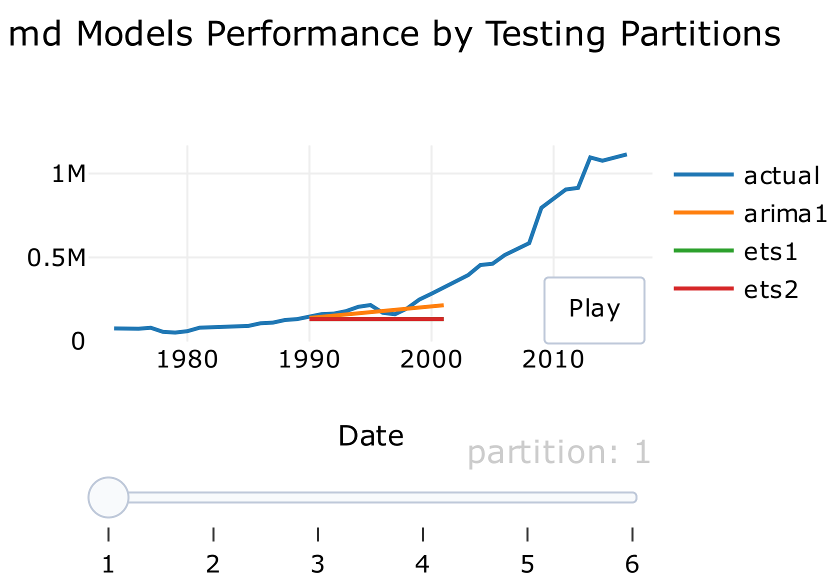
Vamos treinar e comparar os modelos.

```
md <- train_model(input = leite_SO,
                  methods = methods,
                  train_method = list(partitions = 6,
                                      sample.out = 12,
                                      space = 3),
                  horizon = 5,
                  error = "MAPE")
```

```
## # A tibble: 3 x 7
##   model_id model notes        avg_mape avg_rmse `avg_coverage_8~ `avg_coverage_9~
##   <chr>    <chr> <chr>           <dbl>    <dbl>            <dbl>            <dbl>
## 1 arima1   arima ARIMA(0,2,~     0.229  142231.            0.569            0.694
## 2 ets1     ets   ETS model ~     0.375  253102.            0.222            0.375
## 3 ets2     ets   ETS model ~     0.397  266438.            0.194            0.319
```

Podemos ver que o modelo ARIMA foi o melhor, pois apresentou menores erros.

Podemos ver como cada modelo se comporta conforme a janela vai avançando (apenas para `html`).



Forecasting com Modelos de Médias Móveis

Vamos criar uma função que possa fazer previsões usando modelos de médias móveis.

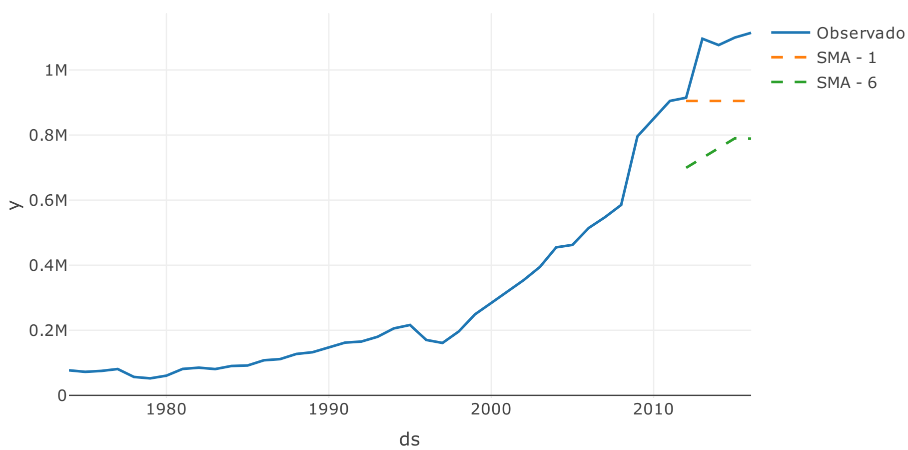
```
sma_forecast <- function(df, h, m, w = NULL){

  # Configurando os pesos da média
  if(is.null(w)){
    w <- rep(1/m, m)
  }

  # Mudando o nome da coluna de data no dataframe
  names(df)[1] <- "date"

  # Separando os dados em treino e teste de acordo com o horizonte de previsão
  df$type <- c(rep("train", nrow(df) - h), rep("test", h))
  df1 <- df %>% spread(key = type, value = y)

  # Criar a variável alvo
  df1$yhat <- df1$train

  # função de média móvel simples
```

```r
  for(i in (nrow(df1) - h + 1):nrow(df1)){
    r <- (i-m):(i-1)
    df1$yhat[i] <- sum(df1$yhat[r] * w)
  }

  # descartando os valores reais do yhat que foram usadas na janela móvel
  df1$yhat <- ifelse(is.na(df1$test), NA, df1$yhat)
  df1$y <- ifelse(is.na(df1$test), df1$train, df1$test)
  return(df1)
}
```

Vamos transformar os dados para o formato de dataframe e calcular a média móvel para fazer previsões para um intervalo de cinco anos. Usaremos média móvel com apenas uma observação e com seis observações, apenas a título de ilustração.

```r
leite_SO_df <- ts_to_prophet(leite_SO)
leite_SO_m1 <- sma_forecast(leite_SO_df, h = 5, m = 1)
leite_SO_m6 <- sma_forecast(leite_SO_df, h = 5, m = 6)
```

Vamos plotar as previsões.



Podemos ver que modelos de médias móveis não são muito bons para forecasting no nosso caso, pois nossa série tem forte tendência de crescimento e não há sazonalidade.

Forecasting com Função de Holt

O modelo de Holt extende a suavização exponencial simples para permitir a previsão de dados que possuem tendência, como no nosso caso. Este método envolve uma equação de previsão e duas equações de suavização (uma para o nível e outra para a tendência).

Equação de previsão: $\hat{y}_{t+h|t} = l_t + hb_t$

Equação de Nível: $l_t = \alpha y_t + (1 - \alpha)(l_{t-1} + b_{t-1})$

Equação de Tendência: $b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$

em que l_t denota uma estimativa para o nível da série no tempo t, b_t denota uma estimativa da tendência (inclinação) da série no tempo t, $0 \leq \alpha \leq 1$ é o parâmetro de suavização para nível e $0 \leq \beta \leq 1$ é o parâmetros de suavização para a tendência.

A equação de nível mostra que l_t é uma média ponderada da observação y_t e de $l_{t-1} + b_{t-1}$ que é $\hat{y}_{t+1|t}$ (previsão de um passo a frente); enquanto que a equação de tendência mostra que b_t é uma média ponderada da tendência estimada, no tempo t, de $l_t - l_{t-1}$ e b_{t-1} (que é a tendência previamente estimada).

Podemos usar o modelo de Holt com a função `holt` do R:

```
fc_holt <- holt(treino, h = 12, initial = "optimal")
fc_holt$model
```

```
## Holt's method
##
## Call:
##  holt(y = treino, h = 12, initial = "optimal")
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.4182
##
##   Initial states:
##     l = 69181.356
##     b = 5231.7159
##
##   sigma:  20049.77
##
##      AIC     AICc      BIC
## 726.3413 728.7413 733.5112
```

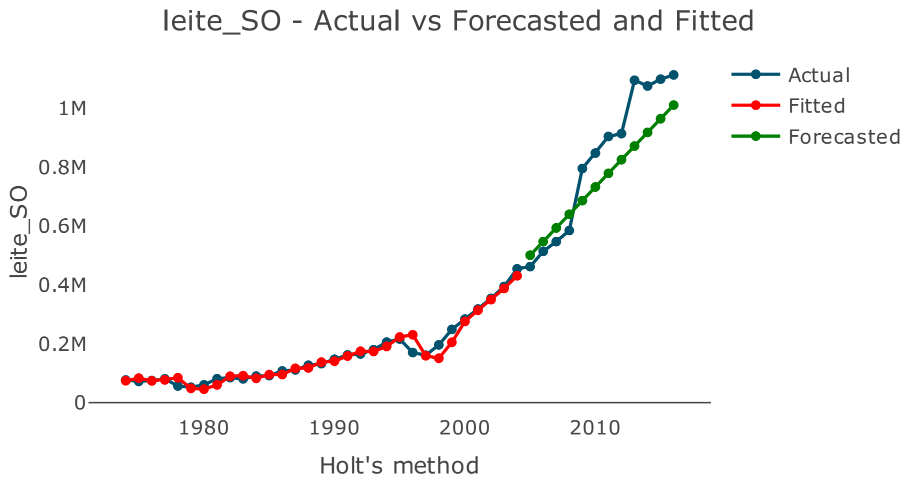
Calculando a acurácia para este modelo:

```
accuracy(fc_holt, teste)
```

```
##                     ME      RMSE       MAE       MPE      MAPE      MASE
## Training set  3173.771  18711.58  12512.53 0.7779418  9.813718 0.6661554
## Test set     73581.522 115503.32 102556.00 6.2119862 11.679454 5.4599853
##                   ACF1 Theil's U
## Training set 0.2240982        NA
## Test set     0.6511842  1.037244
```

Observando um gráfico com as previsões:

```
test_forecast(leite_SO, forecast.obj = fc_holt, test = teste)
```



Podemos adaptar o parâmetro exponencial do modelo de Holt para diminuir o crescimento exponencial das previsões e assim melhorar o modelo.

```
fc_holt_exp <- holt(treino,
                    h = 12,
                    beta = 0.1,
                    initial = "optimal",
                    exponential = TRUE)
```

Podemos fazer as previsões para o conjunto de teste.

```
forecast_holt <- as.data.frame(predict(fc_holt_exp, teste))[,1]
```

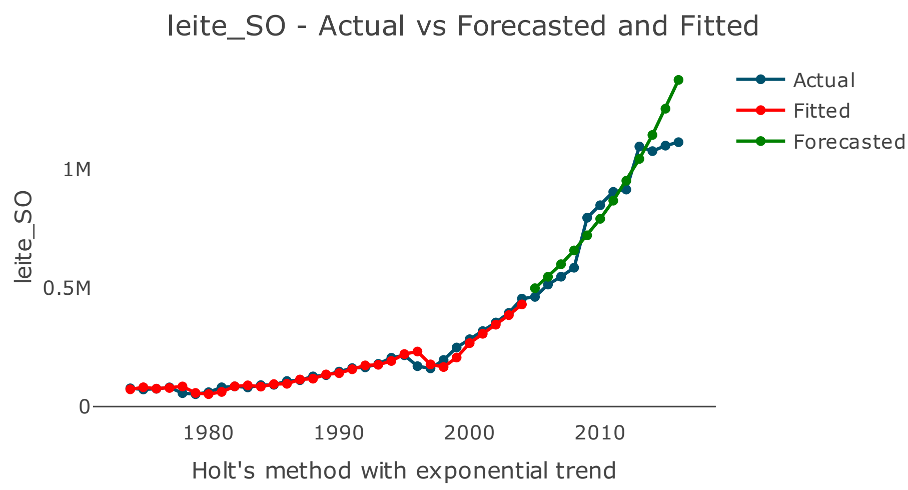
Vamos calcular a acurácia para o modelo de Holt com ajuste exponencial.

```
accuracy(fc_holt_exp, teste)
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set   2603.964  17971.5 12405.79 -0.2173794 9.170272 0.6604729
## Test set     -41399.969 101019.9 78205.57 -4.9504144 9.108096 4.1635912
##                   ACF1 Theil's U
## Training set 0.3201595        NA
## Test set     0.4155805 0.8447938
```

Vamos plotar os gráfico com as previsões.

```
test_forecast(leite_SO, forecast.obj = fc_holt_exp, test = teste)
```



Forecasting com Machine Learning

Vamos utilizar o framework h2o para executar um AutoML e tentar encontrar algum modelo de machine learning para comparar com os modelos previamente apresentados. Usaremos AutoML pois o foco da disciplina não é nos modelos de Machine Learning, e faremos essa comparação apenas a título de curiosidade para comparar as performances e explorar diferentes técnicas que poderiam ser utilizadas.

Após carregar o pacote h2o, precisamos inicializar o cluster que irá executar o processamento com o h2o.

```
h2o.init(max_mem_size = "4G")
```

Lembrando que vamos usar os dados de treino e de teste para treinar os modelos, conforme havíamos definido anteriormente.

Mas precisamos converter os dados do formato dataframe para o formato utilizado pelo h2o que é um h2o cluster.

```
train_h <- as.h2o(treino)
test_h <- as.h2o(teste)
```

Vamos usar o AutoML do h2o para testar vários modelos (random forest, GBM, redes neurais, entre outros) e obter o melhor dos modelos testados.

```
autoML1 <- h2o.automl(training_frame = train_h,
                      x = 'ds',
                      y = 'y',
                      nfolds = 5,
                      max_runtime_secs = 60*20,
                      seed = 1234)
```

Podemos obter uma lista com o desempenho dos modelos.

```
autoML1@leaderboard
```

```
##                                                model_id mean_residual_deviance
## 1  DeepLearning_grid_2_AutoML_1_20220609_174041_model_4              143763103
## 2  DeepLearning_grid_1_AutoML_1_20220609_174041_model_4              155860623
## 3 DeepLearning_grid_1_AutoML_1_20220609_174041_model_34              182413395
## 4 DeepLearning_grid_1_AutoML_1_20220609_174041_model_67              263982609
## 5 DeepLearning_grid_1_AutoML_1_20220609_174041_model_95              276046537
## 6  DeepLearning_grid_3_AutoML_1_20220609_174041_model_4              286272001
##       rmse       mse       mae      rmsle
## 1 11990.13 143763103  8305.417 0.09748887
## 2 12484.42 155860623  9746.574 0.12006893
## 3 13506.05 182413395 10605.559 0.10579677
## 4 16247.54 263982609 10898.213 0.11744860
## 5 16614.65 276046537 12515.146 0.13513551
## 6 16919.57 286272001 11750.614 0.11255723
##
## [560 rows x 6 columns]
```

Finalmente, podemos usar o melhor dos modelos para fazer previsões e assim comparar com os demais modelos testados ao longo do trabalho.

```
test_h$pred_autoML <- h2o.predict(autoML1@leader, test_h)
```

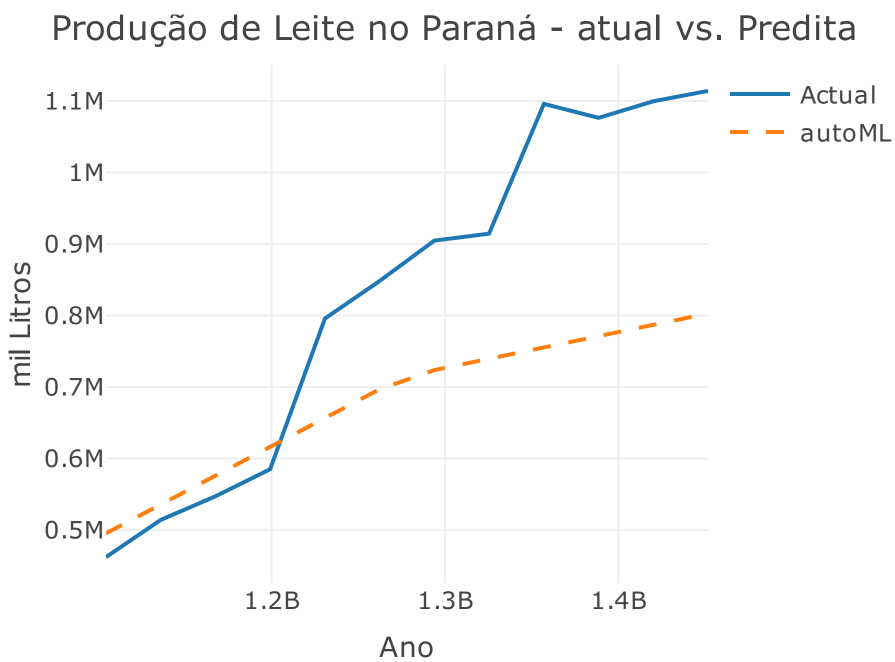
```
##   |                                                                              |
```

```
forecast_automl <- as.data.frame(test_h)
mape_autoML <- mean(abs(forecast_automl$y - forecast_automl$pred_autoML) / forecast_automl$y)
mape_autoML
```

```
## [1] 0.1768254
```

Podemos também visualizar as performances.

```
plot_ly(data = forecast_automl) %>%
  add_lines(x = ~ ds, y = ~y, name = "Actual") %>%
  add_lines(x = ~ ds, y = ~ pred_autoML, name = "autoML", line =
list(dash = "dash")) %>%
  layout(title = "Produção de Leite no Paraná - atual vs. Predita",
         yaxis = list(title = "mil Litros"),
         xaxis = list(title = "Ano"))
```




```r
forecast_df <- data.frame('Ano'=c(2005:2016),
                          'Real'=as.data.frame(teste),
                          'Arima(0,2,0)'= forecast_arima020,
                          'Holt' = forecast_holt,
                          'AutoML' = forecast_automl)
forecast_df <- forecast_df[,c(1,3,4,5,8)]
forecast_df
```

```
##     Ano  Real.y Arima.0.2.0.      Holt AutoML.pred_autoML
## 1  2005  462354       514824  498753.2           495565.3
## 2  2006  514303       574861  546972.8           535825.3
## 3  2007  547328       634898  599854.2           576085.3
## 4  2008  585127       694935  657848.2           616345.3
## 5  2009  795827       754972  721449.2           656715.6
## 6  2010  848341       815009  791199.0           696975.6
## 7  2011  904743       875046  867692.3           723743.6
## 8  2012  914474       935083  951580.9           739546.1
## 9  2013 1095843       995120 1043580.0           755392.0
## 10 2014 1076335      1055157 1144473.5           771194.6
## 11 2015 1099507      1115194 1255121.4           786997.1
## 12 2016 1114010      1175231 1376466.8           802799.7
```

Pode-se ver que nos primeiros anos (de 2005 a 2008) o modelo de rede neural foi melhor, mas depois há uma leve mudança no comportamento dos nossos dados e então outros modelos começar a fazer melhores previsões. Para entender qual dos modelos foi melhor, precisamos calcular algumas métricas em todo o conjunto de previsões.

Vamos calcular o erro quadrático médio para comparar o modelo ARIMA(0,2,0), o modelo de Holt e o melhor modelo do AutoML.

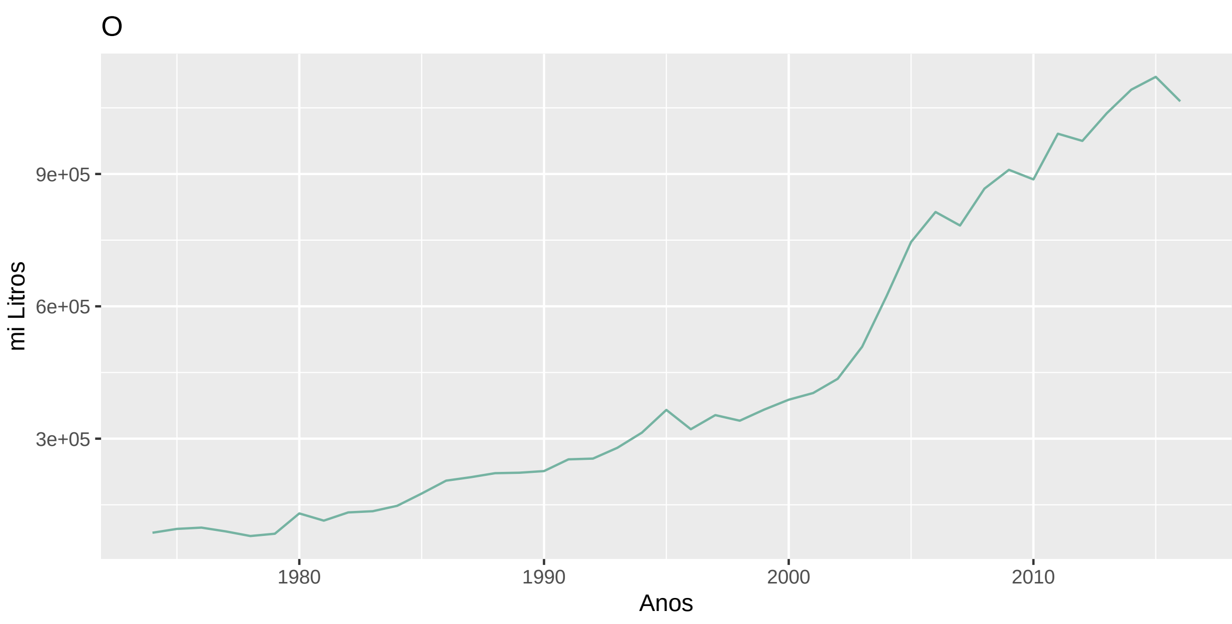
```
mae_vec <- c()
rmse_vec <- c()
for(i in 3:5){
  mae_vec[i-2] <- mae(forecast_df$Real.y, forecast_df[,i])
  rmse_vec[i-2] <- rmse(forecast_df$Real.y, forecast_df[,i])
}

metricas <- data.frame(mae_vec, rmse_vec)
rownames(metricas) <- c('ARIMA(0,2,0)','Holt','AutoML')
colnames(metricas) <- c('MAE','RMSE')
metricas
```

```
##                    MAE      RMSE
## ARIMA(0,2,0)  52809.00  61115.46
## Holt          78205.57 101019.92
## AutoML       169202.08 206664.70
```

Podemos ver que o modelo ARIMA(0,2,0) foi melhor no conjunto de teste. Vale ressaltar que temos poucos dados para essa série temporal e que se houvesse mais dados, outros modelos poderiam ter se saído melhor.

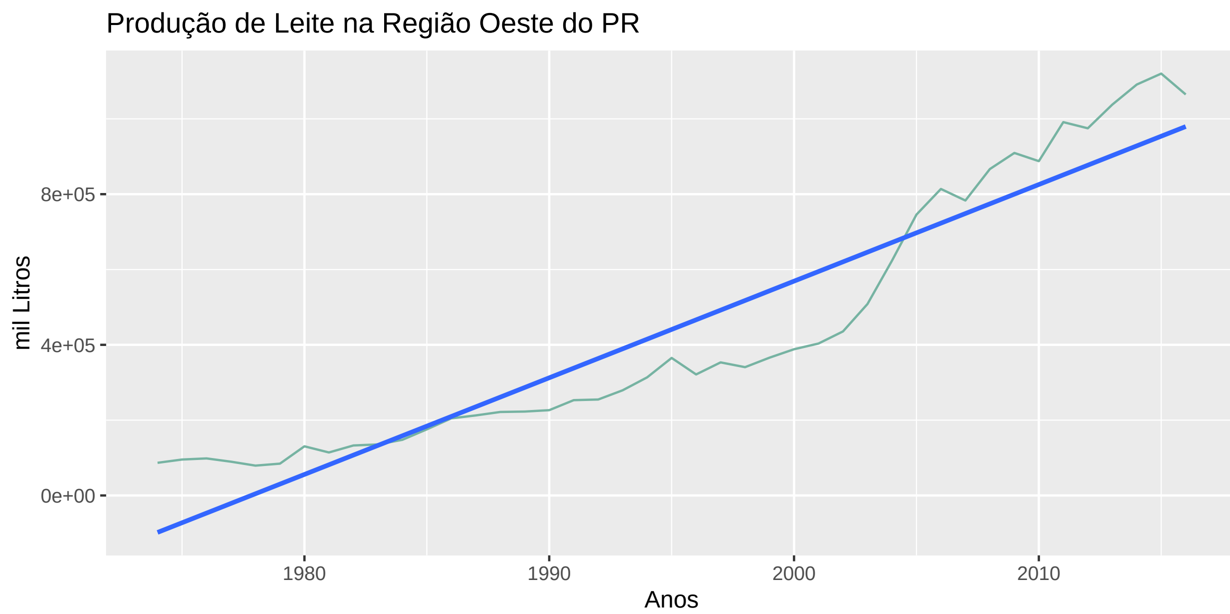
Região Oeste

Agora vamos analisar a região oeste, à qual corresponde o gráfico a seguir:



Podemos observar que até o início dos anos 2000 a produção estava aumentando, mas após 2002 houve um salto muito maior. Os motivos que levaram a essa mudança de comportamento são provavelmente os mesmos que causaram a mudança de regime na região sudoeste.

Vamos ajustar um modelo linear para ter uma noção da tendência apresentada pela série.



Executando o comando **auto.arima**, podemos investigar um primeiro modelo, ARIMA não sazonal, candidato a modelar o comportamento da série histórica.

```
auto.arima(oeste, seasonal = FALSE)
```

```
## Series: oeste
## ARIMA(0,2,1)
##
## Coefficients:
##           ma1
##       -0.8573
## s.e.   0.1143
##
## sigma^2 estimated as 1.583e+09:  log likelihood=-492.58
## AIC=989.16   AICc=989.48   BIC=992.59
```

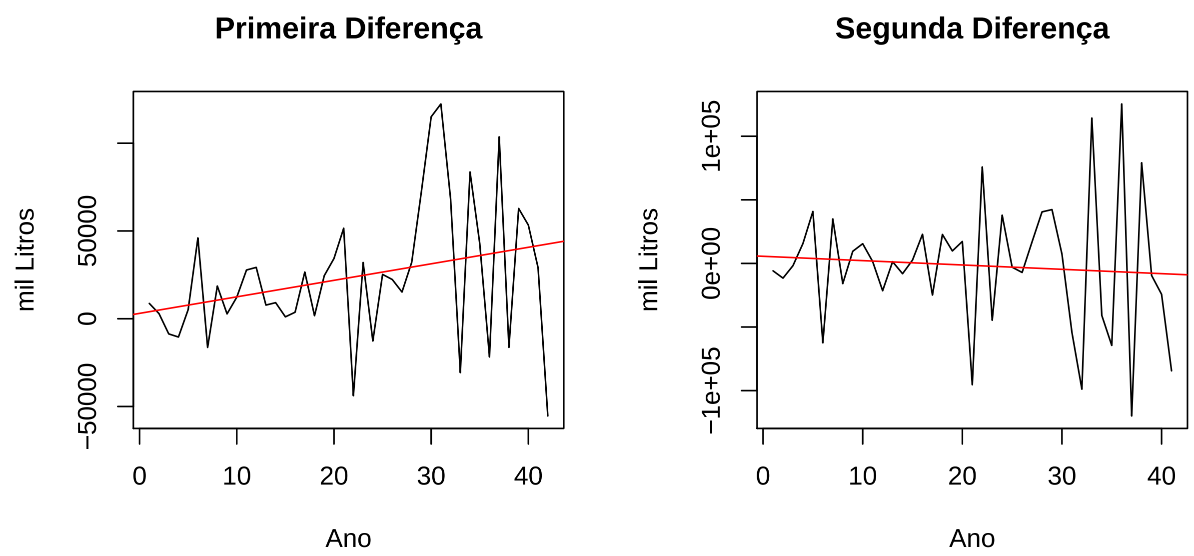
Obtemos um modelo ARIMA(0,2,1), ou seja, o modelo sugere que tomemos duas diferenças para eliminar a tendência e o que nos restará será um modelo de médias móveis de primeira ordem estacionário. Podemos escrevê-lo como

$$(1-B)^2 x_t = (1-0,86B)\varepsilon_t.$$

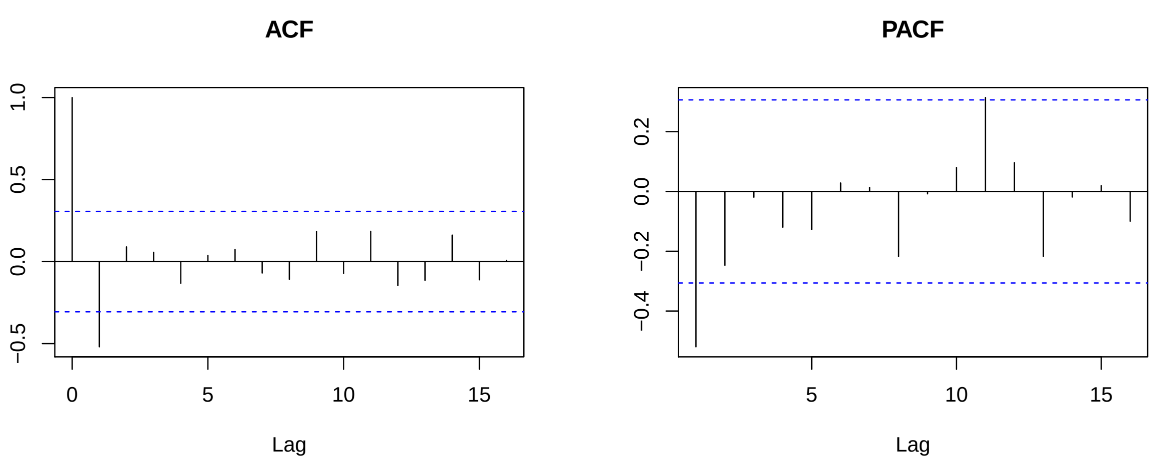
De fato, a tendência é bastante reduzida após a segunda diferença, conforme gráficos a seguir.

```
seq1 <- 1:42
dif1_O <- diff(oeste)
trend2_O <- lm(formula = dif1_O ~ seq1, data=as.data.frame(dif1_O))
seq2 <- 1:41
dif2_O <- diff(dif1_O)
trend3_O <- lm(formula = dif2_O ~ seq2, data = as.data.frame(dif2_O))
par(mfrow=c(1,2))
ts.plot(dif1_O, ylab='mil Litros', xlab='Ano', main='Primeira Diferença') +
```

```r
  abline(trend2_0, col = 'red')
ts.plot(dif2_0, ylab='mil Litros', xlab='Ano', main='Segunda Diferença') +
  abline(trend3_0, col = 'red')
```



Podemos observar que mesmo com a segunda diferença ainda há um pouco de tendência. Avaliando os resíduos para a segunda diferença, temos os seguintes gráficos:



A função de autocorrelação apresentou valor significativo para ordem um, o que indica um modelo de médias móveis de ordem 1, assim como a função de autocorrelação parcial. Vamos avaliar o modelo ARIMA(1,0,1) para a segunda diferença.

```r
arima(x = dif2_0, order = c(1,0,1), xreg = time(dif2_0))
```

```
##
## Call:
## arima(x = dif2_0, order = c(1, 0, 1), xreg = time(dif2_0))
##
```

```
## Coefficients:
##           ar1      ma1  intercept  time(dif2_O)
##       -0.0105  -1.0000   2754.406      -86.1760
## s.e.   0.1708   0.0817   1874.220       86.7338
##
## sigma^2 estimated as 1.367e+09:  log likelihood = -491.3,  aic = 992.59
```

Analisando o gráfico da segunda diferença (anteriormente apresentado), é possível ver que a variância não é constante e isto indica que devemos fazer alguma mudança antes de considerar modelos AR, MA ou ARMA pois estes supõe série estacionária, ou considerar modelos diferentes.

Modelos de Médias Móveis

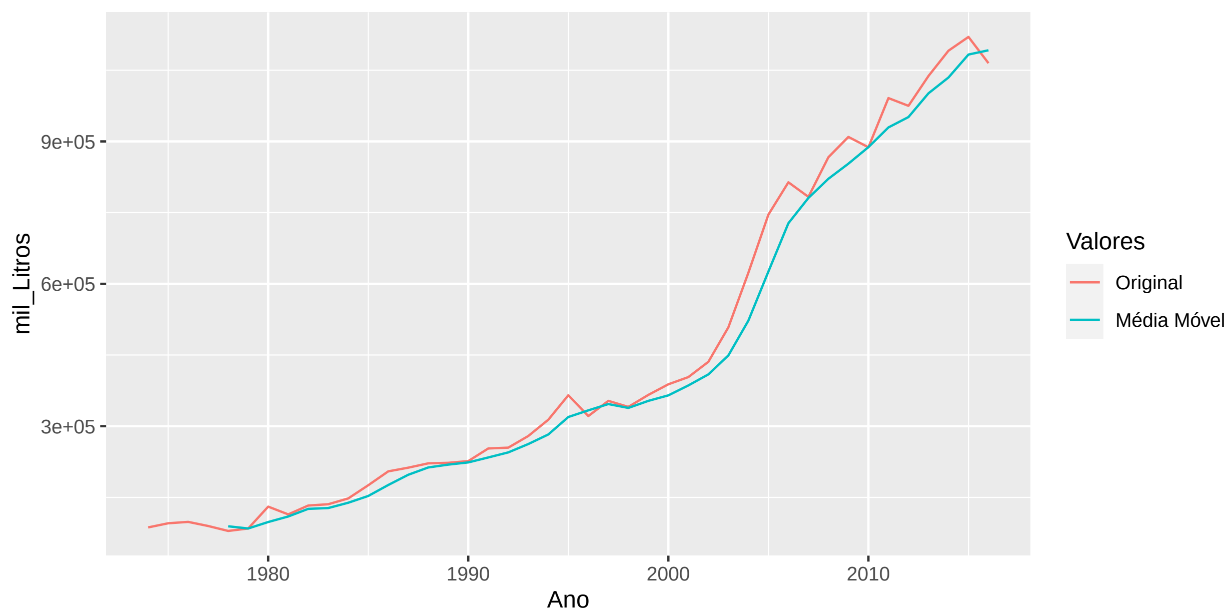
Vamos separar apenas os dados da região oeste e converter os dados para o formato de séries temporais reconhecido pelo R.

```r
leite_O <- leite_meso[,c('Ano','O')]
leite_O <- ts(leite_O$O, start=leite_O$Ano[1], frequency = 1)
```

Vamos usar a função criada anteriormente para criar uma série do tipo MA com média simples e ordem 3.

```r
sma_3_O <- sma(leite_O, ordem = 3)
```

Podemos plotar a série original e a série suavizada para observamos as diferenças.



Poderíamos usar esta estratégia para reduzir efeitos sazonais, mas no nosso caso não há tais efeitos visto que os valores analisados são anuais.

Forecasting

Vamos considerar os dados de treino e teste, semelhante ao que já fizemos para a região sudoeste.

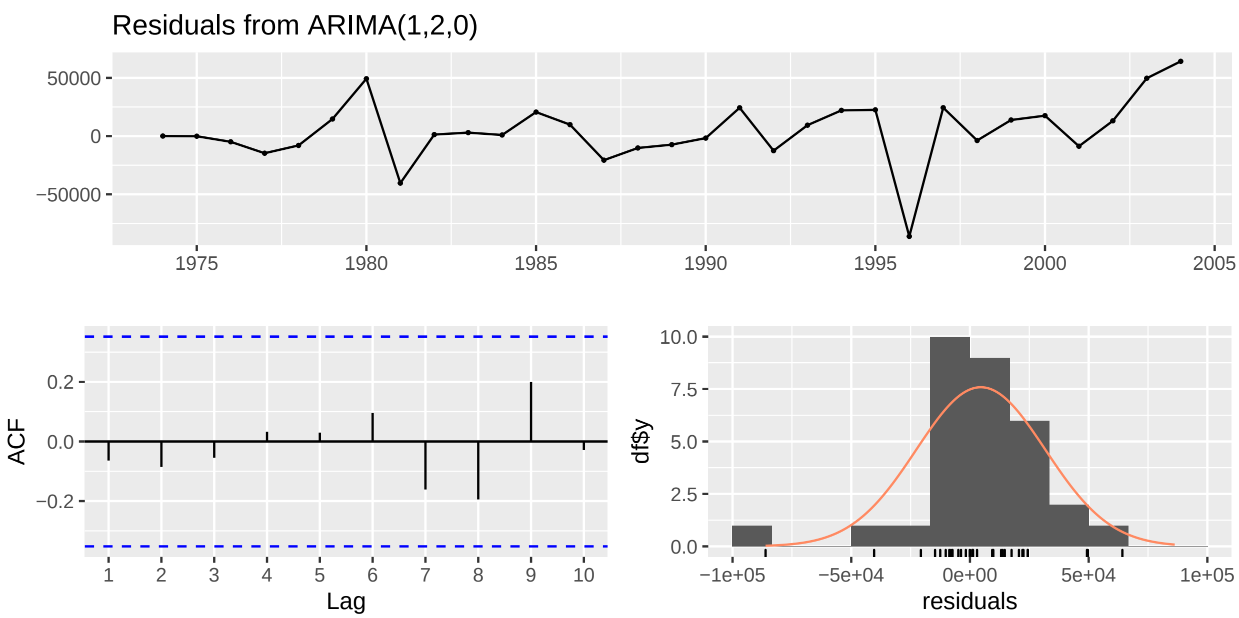
Vamos criar o modelo usando novamente a função `auto.arima`, mas agora considerando apenas o conjunto de treino.

```r
md_O <- auto.arima(treino_O)
md_O
```

```
## Series: treino_O
## ARIMA(1,2,0)
##
## Coefficients:
##           ar1
##       -0.5396
## s.e.   0.1556
##
## sigma^2 estimated as 817542750:  log likelihood=-338.38
## AIC=680.76   AICc=681.22   BIC=683.49
```

Podemos ver que o modelo sugerido foi ARIMA(1,2,0) diferente do modelo sugerido quando se considerava toda a série. Esta indicação sugere que há uma forte componente linear na série de treino.

Podemos verificar os resíduos para ver a qualidade do ajuste do modelo.




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,2,0)
## Q* = 0.95733, df = 5, p-value = 0.9659
##
## Model df: 1.   Total lags used: 6
```

Observa-se que os resíduos apresentam média zero e variância constante. Em conjunto com a ACF e a distribuição dos resíduos, parece que o modelo tem um bom ajuste.

O teste de Ljung-Box apresenta p-valor de 0.9659 e isto indica que não podemos rejeitar a hipótese nula a nível de 5% de significância. Em outras palavras, pelo teste de Ljung-Box, podemos afirmar com 95% de confiança que a auto correlação entre todos os lags é nula, o que está de acordo com o plot da ACF apresentado. O teste de Ljung-Box é uma maneira mais formal de confirmar o que o ACF estava sugerindo.

Vamos usar o modelo treinado para fazer previsões para as 12 observações deixadas como teste e, então, avaliar a performance do modelo.

Podemos fazer as previsões para o conjunto de teste.

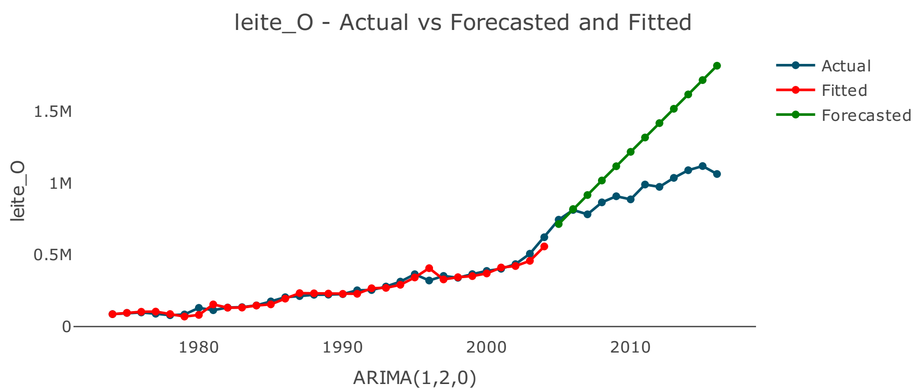
```
forecast_arima120 <- as.data.frame(forecast(md_O, h=12))[,1]
```

Vamos calcular algumas métricas para ver a performance do modelo no conjunto de treino e de teste.

```
fc_O <- forecast(md_O, h = 12)
accuracy(fc_O, teste_O)
```

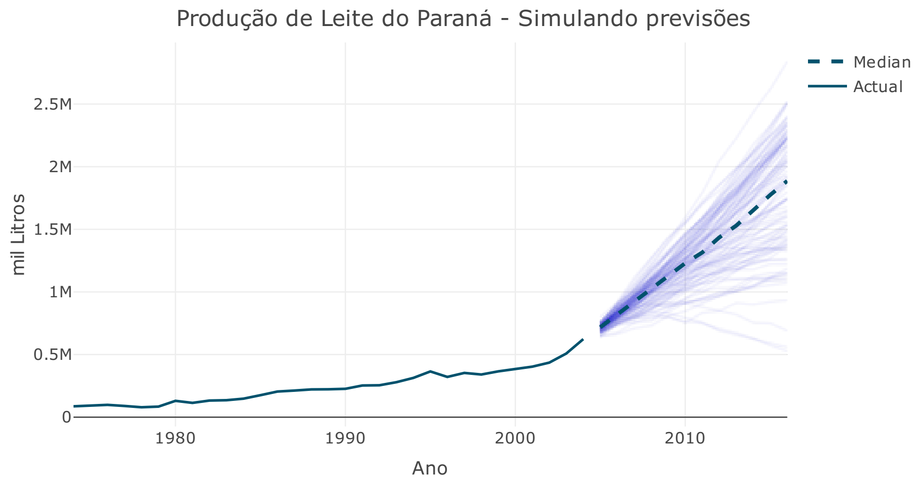
```
##                      ME      RMSE       MAE         MPE     MAPE       MASE
## Training set    4570.656  27173.99  18706.15   0.6369293  8.39659  0.7788291
## Test set     -328684.947 402612.88 333713.28 -32.5089600 33.18326 13.8941258
##                     ACF1 Theil's U
## Training set -0.06395047        NA
## Test set      0.69638332  6.218367
```

O erro no conjunto de teste é muito maior do que no conjunto de treino, o que indica que o modelo não capturou adequadamente o padrão da série temporal. Podemos também observar graficamente o desempenho do modelo nos dois conjuntos.



Vamos criar 100 simulações que vão tentar prever os próximos 12 anos.

```
fc_final3_O <- forecast_sim(model=md_O, h=12, n=100)

fc_final3_O_plotly <- fc_final3_O$plot %>%
            layout(title = "Produção de Leite do Paraná - Simulando previsões",
                   yaxis = list(title = "mil Litros"),
                   xaxis = list(title = "Ano"))
fc_final3_O_plotly
```



Vamos testar os modelos `arima` para ARIMA e `ets` para Exponential Smoothing State Space com diferentes parâmetros.

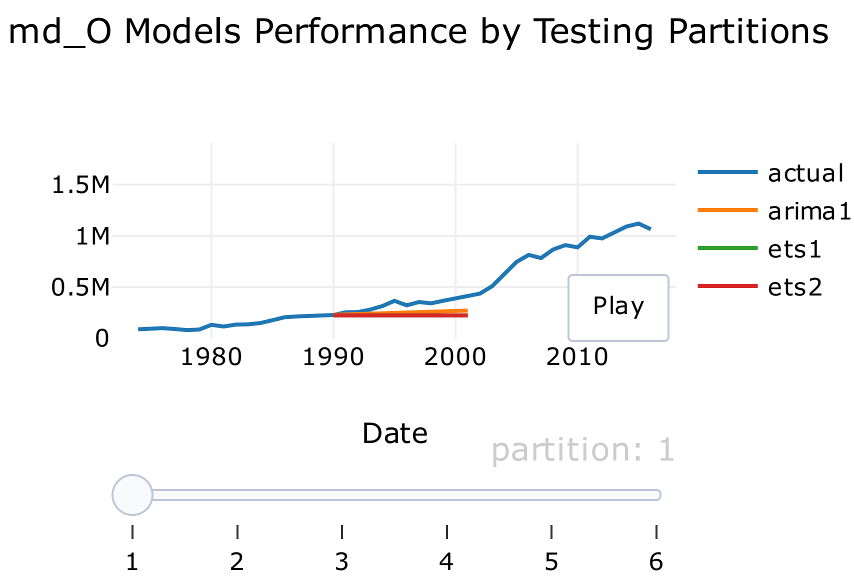
```
methods_0 <- list(ets1 = list(method = "ets",
                              method_arg = list(opt.crit = "lik"),
                              notes = "ETS model with opt.crit = lik"),
                  ets2 = list(method = "ets",
                              method_arg = list(opt.crit = "amse"),
                              notes = "ETS model with opt.crit = amse"),
                  arima1 = list(method = "arima",
                                method_arg = list(order = c(1,2,0)),
                                notes = "ARIMA(1,2,0)"))
```

Vamos treinar e comparar os modelos.

```
md_0 <- train_model(input = leite_0,
                    methods = methods_0,
                    train_method = list(partitions = 6,
                                        sample.out = 12,
                                        space = 3),
                    horizon = 5,
                    error = "MAPE")
```

```
## # A tibble: 3 x 7
##   model_id model notes       avg_mape avg_rmse `avg_coverage_8~ `avg_coverage_9~
##   <chr>    <chr> <chr>          <dbl>    <dbl>            <dbl>            <dbl>
## 1 ets2     ets   ETS model ~    0.267  202466.            0.347            0.431
## 2 arima1   arima ARIMA(1,2,~    0.284  227995.            0.556            0.903
## 3 ets1     ets   ETS model ~    0.292  220583.            0.208            0.458
```

Podemos ver que o segundo modelo ets foi o melhor, pois apresenta menores erros.

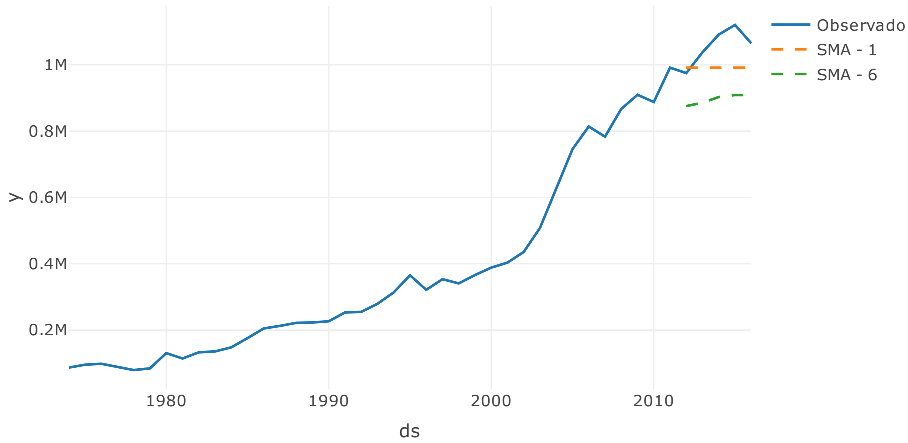


Forecasting com Modelos de Médias Móveis

Vamos transformar os dados para o formato de dataframe e calcular a média móvel para fazer previsões para um intervalo de cinco anos. Usaremos média móvel com apenas uma observação e com seis observações, apenas a título de ilustração.

```
leite_O_df <- ts_to_prophet(leite_O)
leite_O_m1 <- sma_forecast(leite_O_df, h = 5, m = 1)
leite_O_m6 <- sma_forecast(leite_O_df, h = 5, m = 6)
```

Vamos plotar as previsões.



Novamente podemos ver que modelos de médias móveis não são muito bons para forecasting no nosso caso, pois a série tem forte tendência de crescimento e não há sazonalidade.

Forecasting com Função de Holt

Podemos usar o modelo de Holt com a função `holt` do R:

```
fc_holt_0 <- holt(treino_0, h = 12, initial = "optimal")
fc_holt_0$model
```

```
## Holt's method
##
## Call:
##  holt(y = treino_0, h = 12, initial = "optimal")
##
##   Smoothing parameters:
##     alpha = 0.8546
##     beta  = 0.4919
##
##   Initial states:
##     l = 74454.2064
##     b = 10632.407
##
##   sigma:  29203.19
##
##      AIC     AICc      BIC
## 749.6570 752.0570 756.8269
```

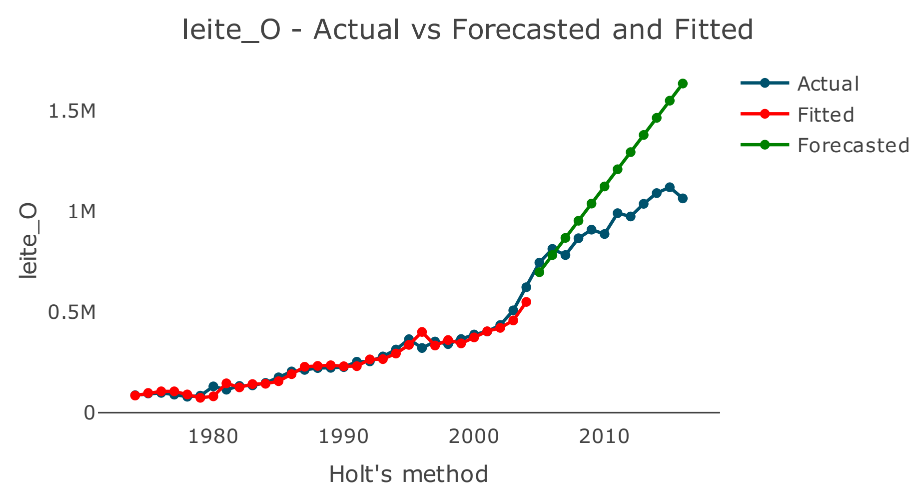
Calculando a acurácia para este modelo, temos

```
accuracy(fc_holt_0, teste_0)
```

```
##                        ME      RMSE      MAE         MPE      MAPE      MASE
## Training set    4893.078  27254.07  19733.9   0.5014601  8.882579 0.8216195
## Test set     -226226.064 289874.40 239288.5 -22.1393255 23.833617 9.9627563
##                     ACF1 Theil's U
## Training set -0.03095446        NA
## Test set      0.67540178  4.441982
```

Observando um gráfico com as previsões:

```
test_forecast(leite_O, forecast.obj = fc_holt_O, test = teste_O)
```



Podemos adaptar o parâmetro exponencial do modelo de Holt para diminuir o crescimento exponencial das previsões e assim melhorar o modelo.

```
fc_holt_exp_O <- holt(treino_O,
                     h = 12,
                     beta = 0.03,
                     initial = "optimal",
                     exponential = TRUE)
```

Podemos fazer as previsões para o conjunto de teste.

```
forecast_holt_O <- as.data.frame(predict(fc_holt_exp_O, teste_O))[,1]
```

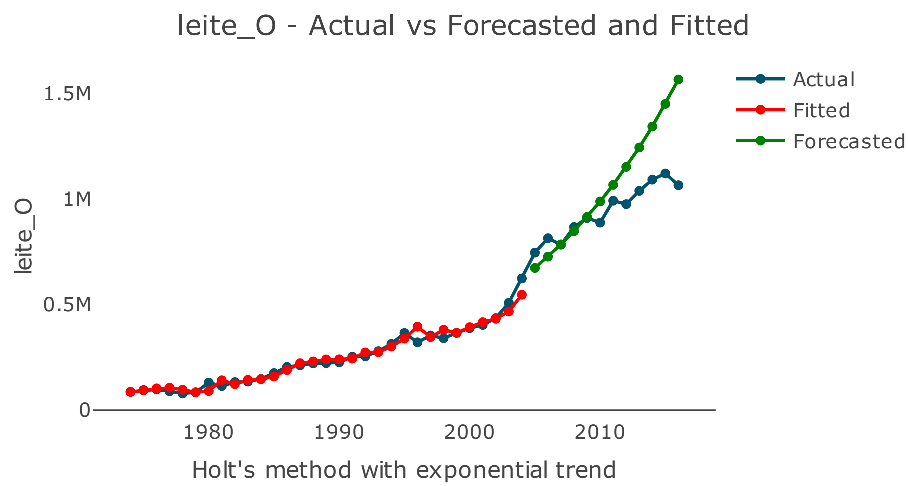
Vamos calcular a acurácia para o modelo de Holt com ajuste exponencial.

```
accuracy(fc_holt_exp_O, teste_O)
```

```
##                         ME      RMSE       MAE        MPE      MAPE      MASE
## Training set     -360.6724  25703.59  17419.77  -1.579853  7.804665 0.7252709
## Test set     -122095.7412 208974.70 151996.12 -11.182984 14.968360 6.3283462
##                      ACF1 Theil's U
## Training set -0.001616461        NA
## Test set      0.640553342  3.091037
```

Vamos plotar os gráfico com as previsões.

```
test_forecast(leite_O, forecast.obj = fc_holt_exp_O, test = teste_O)
```



Forecasting com Machine Learning

Lembrando que vamos usar os dados de treino e de teste para treinar os modelos, conforme havíamos definido anteriormente.

Mas precisamos converter os dados do formato dataframe para o formato utilizado pelo h2o que é um h2o cluster.

```
train_h_O <- as.h2o(treino_O)
test_h_O <- as.h2o(teste_O)
```

Vamos usar o AutoML do h2o.

```
autoML1_O <- h2o.automl(training_frame = train_h_O,
                        x = 'ds',
                        y = 'y',
                        nfolds = 5,
                        max_runtime_secs = 60*20,
                        seed = 1234)
```

Podemos obter uma lista com o desempenho dos modelos.

```
autoML1_O@leaderboard
```

```
##                                                 model_id mean_residual_deviance
## 1 DeepLearning_grid_1_AutoML_2_20220609_180314_model_84              419591252
## 2  DeepLearning_grid_1_AutoML_2_20220609_180314_model_4              463840009
```

```
## 3 DeepLearning_grid_1_AutoML_2_20220609_180314_model_34              471154821
## 4 DeepLearning_grid_1_AutoML_2_20220609_180314_model_54              565494543
## 5  DeepLearning_grid_2_AutoML_2_20220609_180314_model_4              585987876
## 6  DeepLearning_grid_3_AutoML_2_20220609_180314_model_4              627224765
##       rmse       mse      mae      rmsle
## 1 20483.93 419591252 16191.71 0.09181381
## 2 21536.95 463840009 15729.80 0.09662900
## 3 21706.10 471154821 15511.29 0.10444312
## 4 23780.13 565494543 17305.87 0.09985620
## 5 24207.19 585987876 14564.28 0.09422809
## 6 25044.46 627224765 15840.59 0.08448589
##
## [301 rows x 6 columns]
```

Finalmente, podemos usar o melhor dos modelos para fazer previsões e assim comparar com os demais modelos testados ao longo do trabalho.

```
test_h_O$pred_autoML <- h2o.predict(autoML1_O@leader, test_h_O)
```

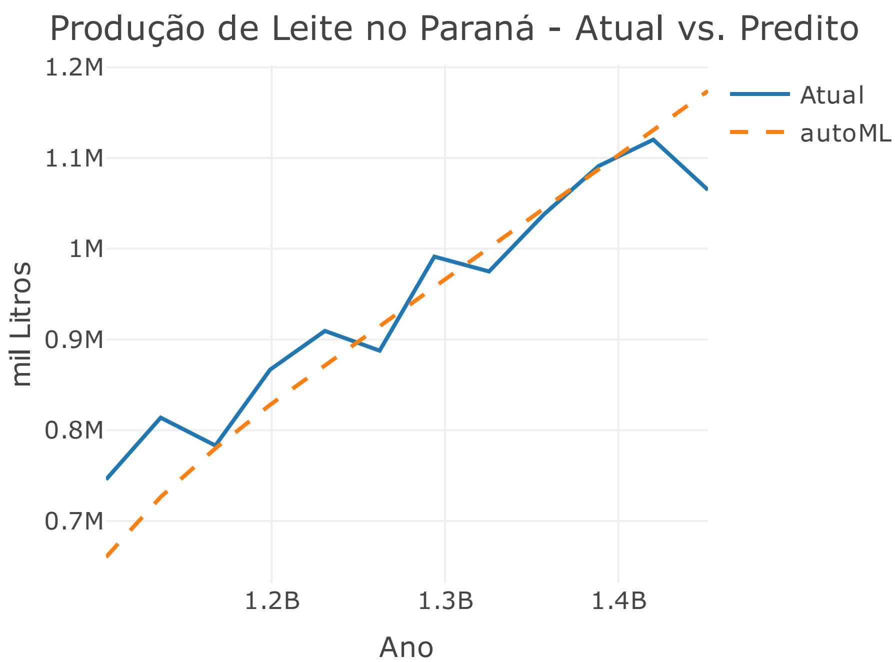
```
##   |                                                                              |
```

```
forecast_automl_O <- as.data.frame(test_h_O)
mape_autoML_O <- mean(abs(forecast_automl_O$y - forecast_automl_O$pred_autoML) / forecast_automl_O$y)
mape_autoML_O
```

```
## [1] 0.0437223
```

Podemos também visualizar as performances.

```
plot_ly(data = forecast_automl_O) %>%
  add_lines(x = ~ ds, y = ~ y, name = "Atual") %>%
  add_lines(x = ~ ds, y = ~ pred_autoML, name = "autoML", line =
list(dash = "dash")) %>%
  layout(title = "Produção de Leite no Paraná - Atual vs. Predito",
         yaxis = list(title = "mil Litros"),
         xaxis = list(title = "Ano"))
```




```
forecast_df_0 <- data.frame('Ano'=c(2005:2016),
                          'Real'=as.data.frame(teste_0),
                          'Arima(0,2,0)'= forecast_arima120,
                          'Holt' = fc_holt_0,
                         'AutoML' = forecast_automl_0$pred_autoML)
forecast_df_0 <- forecast_df_0[,c(1,3,4,5,10)]
forecast_df_0
```

```
##       Ano  Real.y Arima.0.2.0. Holt.Point.Forecast    AutoML
## 2005 2005  745714     715544.0            697989.5  660266.2
## 2006 2006  813879     820058.0            783229.1  726547.3
## 2007 2007  783175     917921.3            868468.6  780074.3
## 2008 2008  866780    1019373.1            953708.1  828055.0
## 2009 2009  909485    1118888.6           1038947.7  871376.9
## 2010 2010  887706    1219448.9           1124187.2  914580.5
## 2011 2011  991315    1319445.5           1209426.8  957784.8
## 2012 2012  974993    1419746.2           1294666.3 1000989.0
## 2013 2013 1037798    1519882.8           1379905.8 1044311.6
## 2014 2014 1091138    1620108.0           1465145.4 1087515.9
## 2015 2015 1120190    1720285.4           1550384.9 1130720.1
## 2016 2016 1064798    1820488.6           1635624.4 1173924.4
```

Pode-se ver que nos primeiros anos (2005 e 2006) o modelo de rede neural foi melhor, mas depois há uma mudança no comportamento dos nossos dados e então outros modelos começam a fazer melhores previsões.

Vamos calcular o erro quadrático médio para comparar o modelo ARIMA(1,2,0), o modelo de Holt e o melhor modelo do AutoML.

```
mae_vec_O <- c()
rmse_vec_O <- c()
for(i in 3:5){
  mae_vec_O[i-2] <- mae(forecast_df_O$Real.y, forecast_df_O[,i])
  rmse_vec_O[i-2] <- rmse(forecast_df_O$Real.y, forecast_df_O[,i])
}

metricas_O <- data.frame(mae_vec_O, rmse_vec_O)
rownames(metricas_O) <- c('ARIMA(1,2,0)','Holt','AutoML')
colnames(metricas_O) <- c('MAE','RMSE')
metricas_O
```

```
##                    MAE      RMSE
## ARIMA(1,2,0) 333713.28 402612.88
## Holt         239288.47 289874.40
## AutoML        39075.52  52031.18
```

Podemos ver que o melhor modelo selecionado foi o modelo de Holt. Vale ressaltar que temos poucos dados para essa série temporal e que se houvesse mais dados, outros modelos poderiam ter se saído melhor.

Conclusão

Neste trabalho focamos em duas séries temporais principais que são referentes às duas principais mesorregiões produtoras de leite no Paraná. Nossos dados são anuais e não apresentam sazonalidade.

Podemos identificar que há uma mudança de comportamento em meados dos anos 90 em que a produção de leite destas regiões aumenta muito. Observando os gráficos pode-se notar que em 2015 e 2016 parece haver uma nova mudança de comportamento dos dados. O período de grande elevação da produção de leite coincide aproximadamente com a situação econômica do país e pode, possivelmente, estar relacionada a isto.

Temos poucas observações sobre o comportamento mais recente e este apresenta forte tendência. Outro problema enfrentando é falta de dados mais atuais disponíveis para fazermos comparações.

Devido à falta de sazonalidade e forte tendência, os modelos de médias móveis e de suavização exponencial não produziram bons resultados de previsão. Os melhores modelos foram ARIMA com duas diferenças, isto é ARIMA(p,2,q), o modelo de Holt e modelos de redes neurais selecionados por AutoML.

Para diferentes séries teremos diferentes modelos ou modelos com diferentes parâmetros estimados. Observando os gráficos para os anos mais recentes vemos que pode haver uma nova mudança de comportamento das séries e, portanto, todos os modelos estimados podem eventualmente apresentar um desempenho não muito satisfatório se esse novo comportamento se confirmar.

Esperamos que com os resultados apresentados com futuras melhorias possa embasar algumas decisões de políticas públicas regionais, beneficiando a população e os produtores de leite do Paraná.

Referências

Alves, L. R., Ostapechen, L. A. P., Porcé, M., & Parré, J. L. (2020). Atividade leiteira no Paraná: uma análise espacial e econométrica. Redes, 25, 2432-2453. https://doi.org/10.17058/redes.v25i0.14974 https://sindileiteparana.com.br/dados-do-setor/

https://www.idrparana.pr.gov.br/Pagina/Bovinocultura-de-Leite

http://www.ipeadata.gov.br/Default.aspx