Estatística Aplicada II

Aluno: Victor Lima

Data: 02/07/2023

Segunda Lista de Exercícios

Com a base de dados “prodbebidas” (dados mensais do índice de produção de bebidas no Brasil) obter os seguintes resultados com o auxílio do “R”

Fazer a todos os testes estatísticos e gráficos necessários e a predição para os próximos 6 meses do índice de produção de bebidas para os seguintes modelos:

I. ETS;
II. ARIMA OU SARIMA (verificar se existe sazonalidade ou não e decidir qual modelo é mais adequado)

Obs: separe os últimos 12 meses da série para testar o modelo.

1 - Criando e analisando os dados

1.1 - Carregar conjunto de dados e organizar para utilizar nos modelos

```
library(readxl)
prodbebidas <- read_excel('prodbebidas.xls')
head(prodbebidas)
```

Data <chr>	**Prodbebidas** <dbl>
2002.01	62.55626
2002.02	57.84550
2002.03	60.69372
2002.04	62.65435
2002.05	62.31734
2002.06	60.49615

6 rows

1.2 - Convertendo os valores da coluna “Datas” para o tipo data

```
dates <- unlist(prodbebidas[,1])
datesFormat <- c()
for (date in dates) {
  year <- substr(date, start = 1, stop = 4)
  month <- substr(date, start = 6, stop = 8)
  # cat(month, '/' , year, '\n')
  datesFormat <- c(datesFormat, paste0(year,'-',month,'-01'))
}
prodbebidas <- subset(prodbebidas, select = -Data)
prodbebidas <- cbind(prodbebidas, Datas = as.Date(datesFormat))
```

1.3 - Criando uma série temporal com a tabela de dados

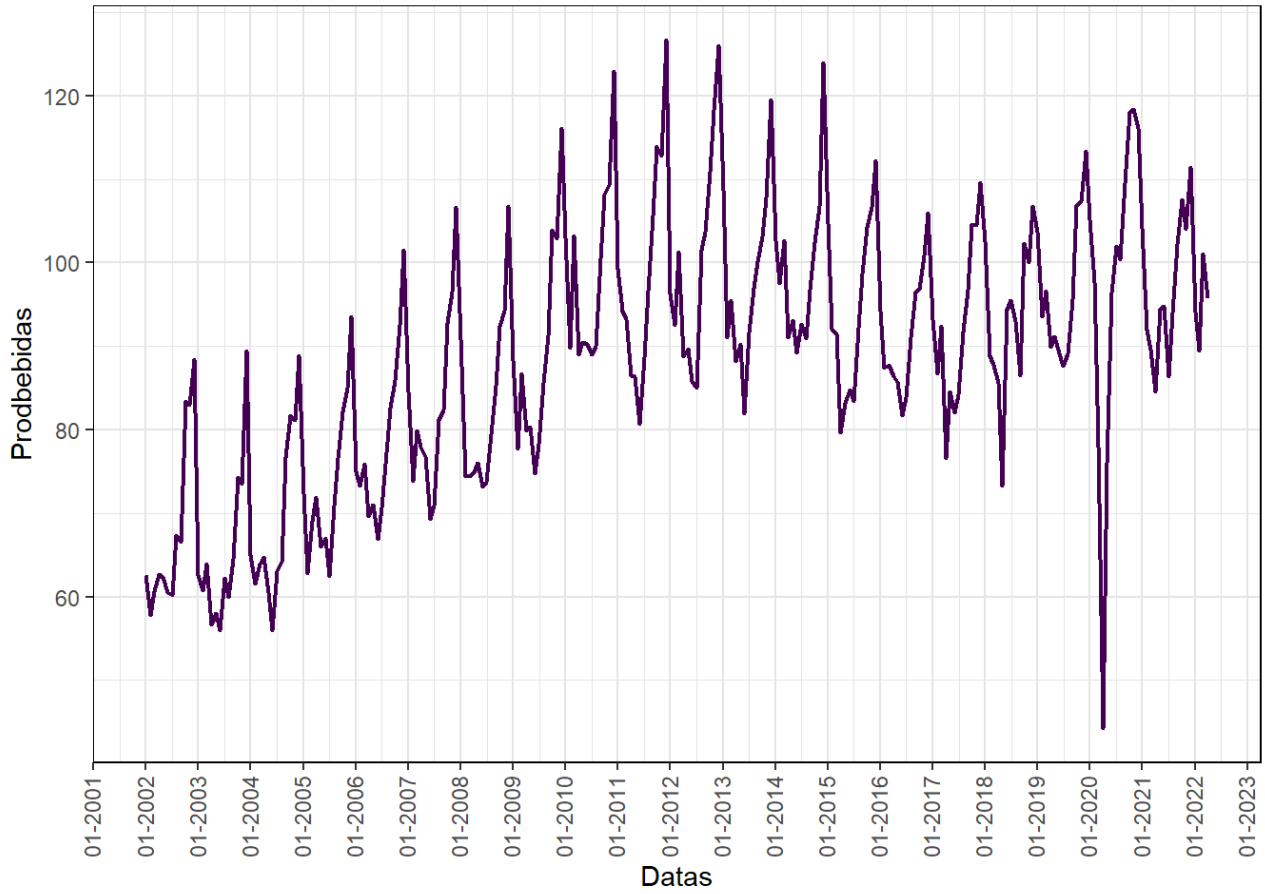
```
prodbebidas_ts <- ts(data = prodbebidas[, 1], start = c(2002, 1), end = c(2022, 4), frequency = 1
2)
```

1.4 - Protando um gráfico para ver a distribuição dos valores no tempo

```
library('ggplot2')
library('forecast')
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
prodbebidas %>%
  ggplot() +
  geom_line(aes(x = Datas, y = Prodbebidas, group = TRUE, color = "Prodbebidas"), linewidth = 0.8)
+
  scale_color_viridis_d() +
  scale_y_continuous(labels = scales::comma) +
  scale_x_date(date_labels = "%m-%Y", date_breaks = "1 year") +
  theme(axis.text.x = element_text(angle = 90,  vjust = 0.4),
        panel.background = element_rect(fill = "white", color = "black"),
        panel.grid = element_line(color = "grey90"),
        panel.border = element_rect(color = "black", fill = NA),
        legend.position = "none")
```



1.5 - Decompondo os valores pelo modelo aditivo

```
decprodbebidas <- decompose(x = prodbebidas_ts, type = "additive")
```

1.6 - Inserindo os objetos decompostos em um dataframe

```
library('dplyr')
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
decprodbebidas_df <- data.frame(tempo = prodbebidas$Datas, serie = unlist(prodbebidas$Prodbebida
s), tendencia = unlist(decprodbebidas$trend), sazonalidade = unlist(decprodbebidas$seasonal), dess
azonalizada = prodbebidas_ts - decprodbebidas$seasonal, erro = unlist(decprodbebidas$random)) %>%
rename(tempo = 1, serie = 2, tendencia = 3, sazonalidade = 4, dessazonalizada = 5, erro = 6)
```

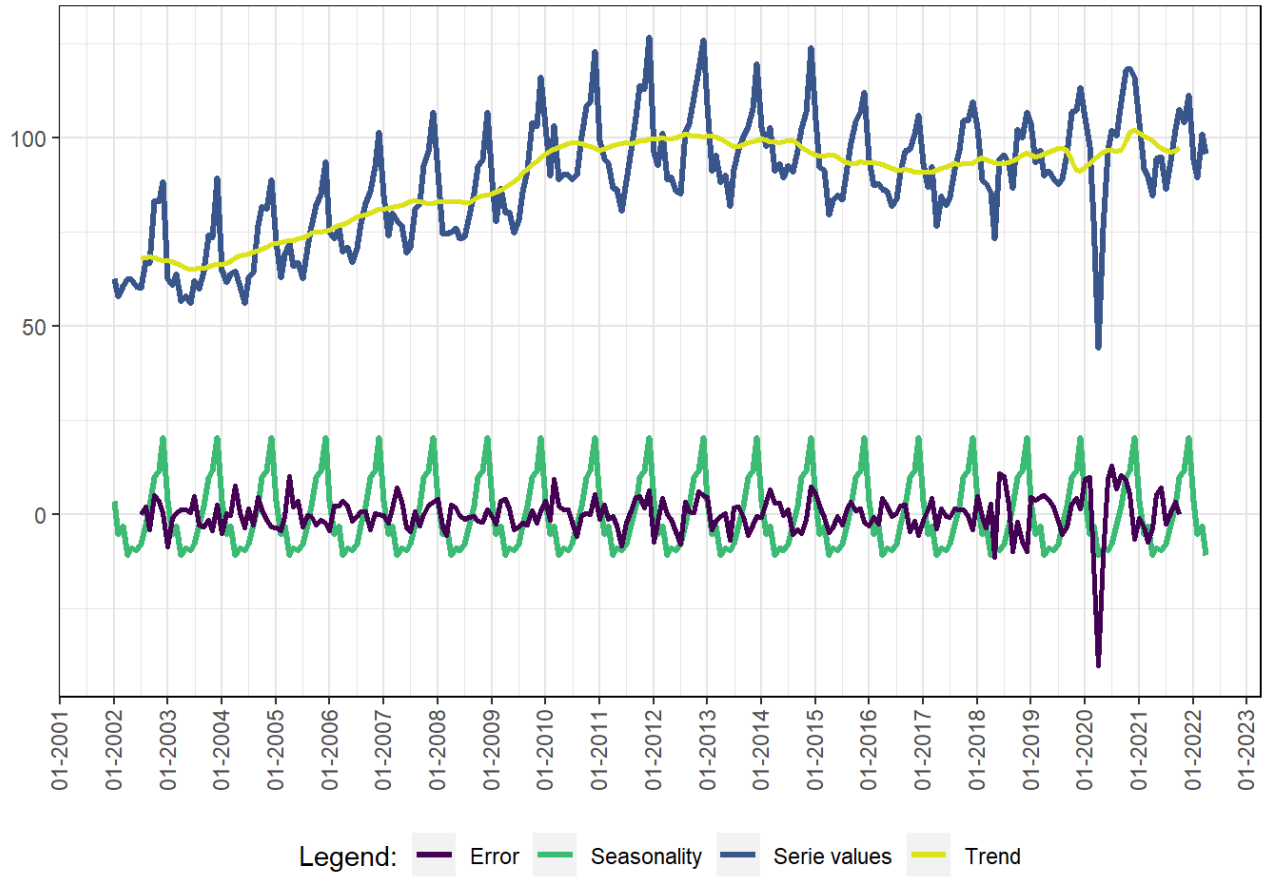
1.7 - Inserindo os objetos decompostos em um dataframe

```
library('dplyr')
decprodbebidas_df <- data.frame(tempo = prodbebidas$Datas, serie = unlist(prodbebidas$Prodbebida
s), tendencia = unlist(decprodbebidas$trend), sazonalidade = unlist(decprodbebidas$seasonal), dess
azonalizada = prodbebidas_ts - decprodbebidas$seasonal, erro = unlist(decprodbebidas$random)) %>%
rename(tempo = 1,serie = 2, tendencia = 3, sazonalidade = 4, dessazonalizada = 5,erro = 6)
```

1.8 - Plotando todos os dados da decomposição juntos

```
decprodbebidas_df %>%
  ggplot() + geom_line(aes(x = tempo, y = serie, color = "Serie values"), linewidth = 1.2) + geom_
line(aes(x = tempo, y = tendencia, color = "Trend"), linewidth = 1) + geom_line(aes(x = tempo, y =
sazonalidade, color = "Seasonality"), linewidth = 1.2) + geom_line(aes(x = tempo, y = erro, color
= "Error"), linewidth = 1) +  scale_x_date(date_labels = "%m-%Y", date_breaks = "1 year") + scale_
y_continuous(labels = scales::comma) + labs(color = "Legend:", x = NULL, y = NULL) + scale_color_m
anual(values = c("#440154FF", "#3CBB75FF", "#39568CFF", "#DCE319FF")) + theme(axis.text.x = elemen
t_text(angle = 90, vjust = 0.4), panel.background = element_rect(fill = "white", color = "black"),
panel.grid = element_line(color = "grey90"), panel.border = element_rect(color = "black", fill = N
A), legend.position = "bottom")
```

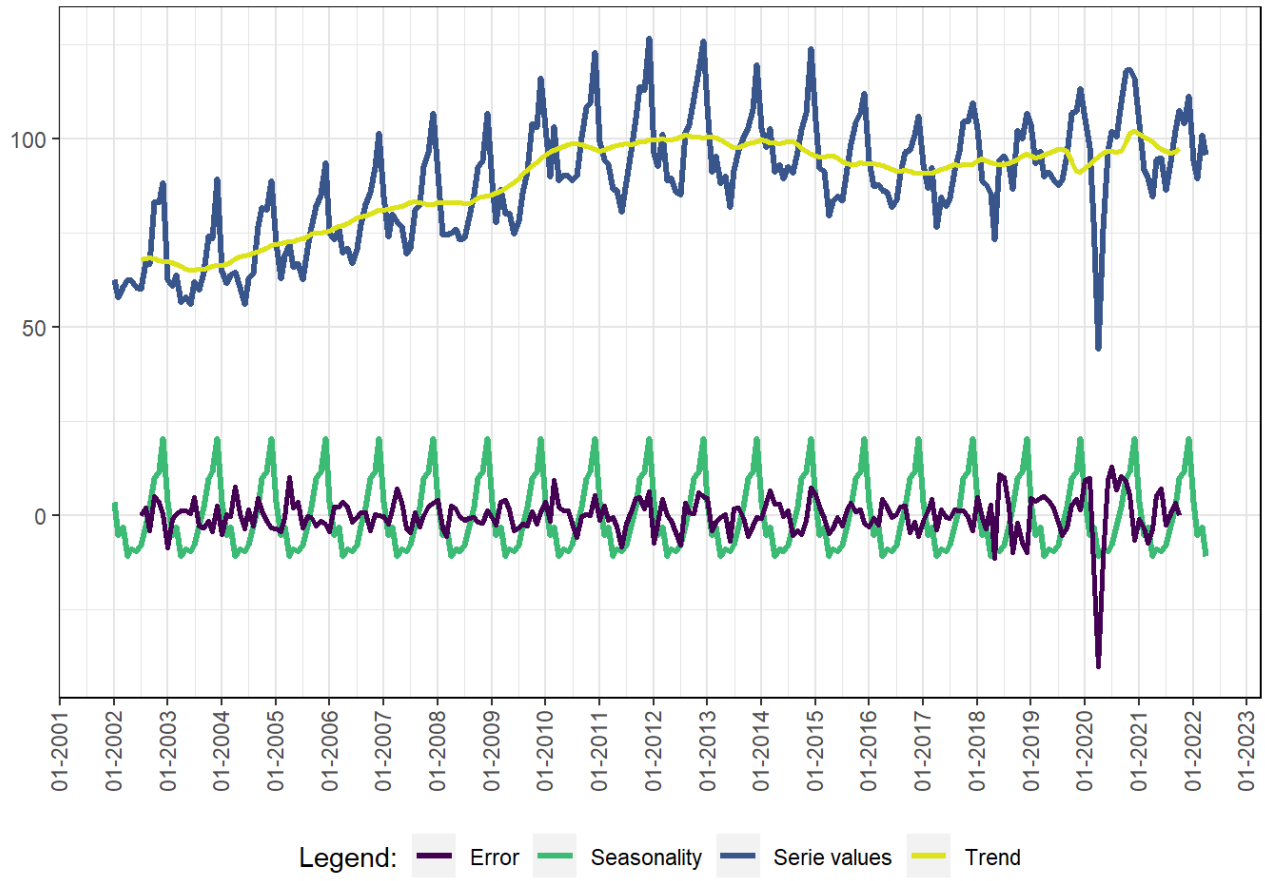
```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
## Removed 12 rows containing missing values (`geom_line()`).
```



1.9 - Plotando todos os dados da decomposição juntos

```
decprodbebidas_df %>%
  ggplot() + geom_line(aes(x = tempo, y = serie, color = "Serie values"), linewidth = 1.2) + geom_
line(aes(x = tempo, y = tendencia, color = "Trend"), linewidth = 1) + geom_line(aes(x = tempo, y =
sazonalidade, color = "Seasonality"), linewidth = 1.2) + geom_line(aes(x = tempo, y = erro, color
= "Error"), linewidth = 1) +  scale_x_date(date_labels = "%m-%Y", date_breaks = "1 year") + scale_
y_continuous(labels = scales::comma) + labs(color = "Legend:", x = NULL, y = NULL) + scale_color_m
anual(values = c("#440154FF", "#3CBB75FF", "#39568CFF", "#DCE319FF")) + theme(axis.text.x = elemen
t_text(angle = 90, vjust = 0.4), panel.background = element_rect(fill = "white", color = "black"),
panel.grid = element_line(color = "grey90"), panel.border = element_rect(color = "black", fill = N
A), legend.position = "bottom")
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
## Removed 12 rows containing missing values (`geom_line()`).
```



2 - Modelo ETS

2.1 - Isolando apenas os valores

```
prodbebidasvalues=prodbebidas[1]
```

2.2 - Criando a série temporal

```
prodbebidasvaluests=ts(prodbebidasvalues, start=c(2002, 1), end=c(2022, 4), frequency=12)
```

2.3 - Separando os dados de treino e teste do modelo

Foi solicitado que os último 12 meses, devem ser usados para teste

2.3.1 - Separando os dados de treino

```
prodbebidastreino=window(prodbebidasvaluests,start=c(2002,1), end=c(2021,4))
```

2.3.2 - Separando os dados de teste

```
prodbebidasteste=window(prodbebidasvaluests,start=c(2021,5), end=c(2022,4))
```

2.4 - Estimando o modelo ETS

```
prodbebidastreino.ets <- ets(prodbebidastreino)
summary(prodbebidastreino.ets)
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = prodbebidastreino)
##
##   Smoothing parameters:
##     alpha = 0.5324
##     gamma = 0.1855
##
##   Initial states:
##     l = 68.3188
##     s = 22.2949 11.6767 10.4517 0.0984 -2.9925 -7.4788
##            -10.2453 -6.8968 -6.2432 -3.005 -6.982 -0.678
##
##   sigma:  5.6518
##
##      AIC     AICc      BIC
## 2082.839 2085.061 2134.540
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 0.2423636 5.478619 3.752648 -0.02880651 4.494228 0.6607421
##                   ACF1
## Training set 0.1047277
```

2.5 - Fazendo a previsao no modelo ETS para 12 meses

```
prodbebidasvaluests.ets.forecasts <- forecast.ets(prodbebidastreino.ets, h = 12)
summary(prodbebidasvaluests.ets.forecasts)
```

```
##
## Forecast method: ETS(A,N,A)
##
## Model Information:
## ETS(A,N,A)
##
## Call:
##  ets(y = prodbebidastreino)
##
##   Smoothing parameters:
##     alpha = 0.5324
##     gamma = 0.1855
##
##   Initial states:
##     l = 68.3188
##     s = 22.2949 11.6767 10.4517 0.0984 -2.9925 -7.4788
##            -10.2453 -6.8968 -6.2432 -3.005 -6.982 -0.678
##
##   sigma:  5.6518
##
##      AIC     AICc      BIC
## 2082.839 2085.061 2134.540
##
## Error measures:
##                     ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 0.2423636 5.478619 3.752648 -0.02880651 4.494228 0.6607421
##                   ACF1
## Training set 0.1047277
##
## Forecasts:
##          Point Forecast     Lo 80     Hi 80    Lo 95    Hi 95
## May 2021       91.68192  84.43884  98.92499 80.60459 102.7592
## Jun 2021       95.22489  87.01919 103.43060 82.67535 107.7744
## Jul 2021       94.00474  84.93804 103.07144 80.13842 107.8711
## Aug 2021       95.38278  85.53005 105.23552 80.31432 110.4512
## Sep 2021      100.58180  90.00126 111.16235 84.40026 116.7633
## Oct 2021      109.23093  97.96952 120.49234 92.00809 126.4538
## Nov 2021      110.04264  98.13924 121.94603 91.83797 128.2473
## Dec 2021      116.56363 104.05114 129.07611 97.42744 135.6998
## Jan 2022      106.04728  92.95401 119.14055 86.02286 126.0717
## Feb 2022       94.99386  81.34450 108.64322 74.11896 115.8688
## Mar 2022       92.56915  78.38548 106.75283 70.87710 114.2612
## Apr 2022       83.17693  68.07010  98.28376 60.07303 106.2808
```

2.6 - Plotando gráfico com a predição

```
library('plotly')
```

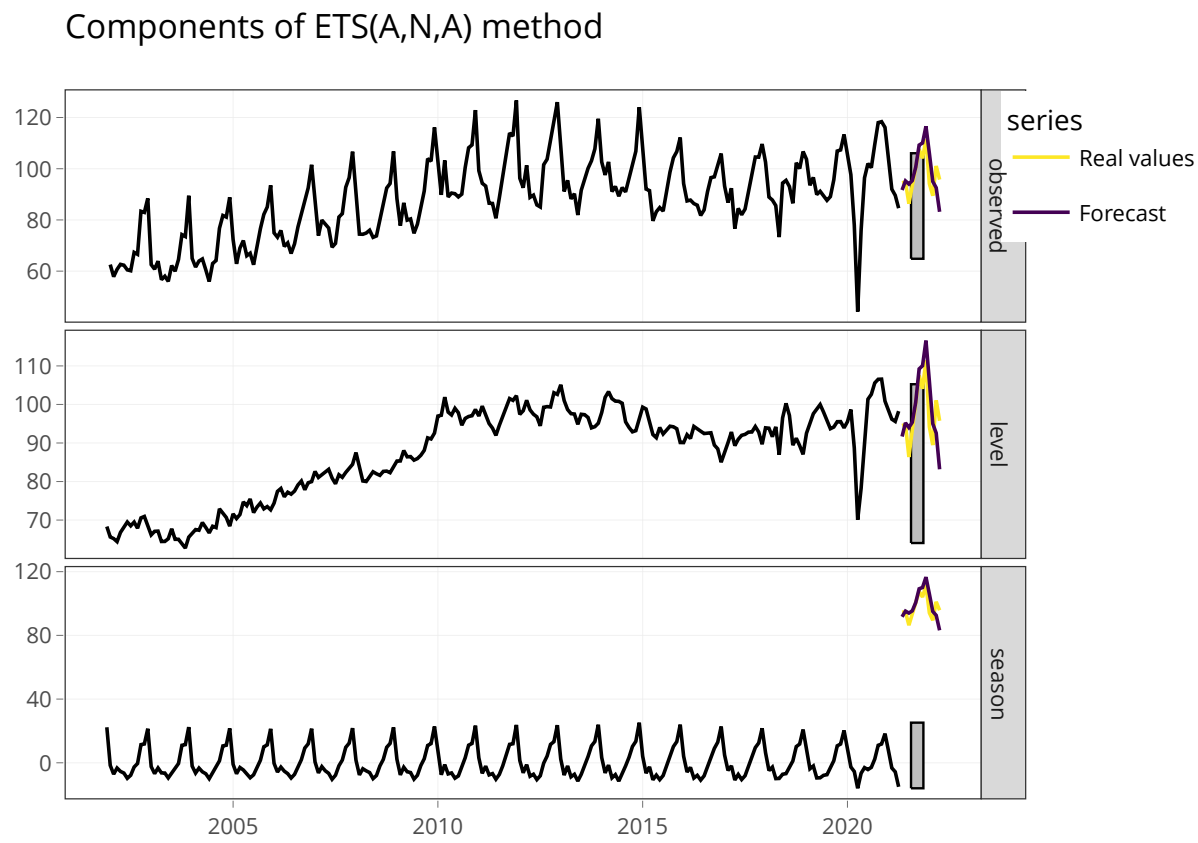
```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
ggplotly(
  autoplot(prodbebidastreino.ets)+
    autolayer(prodbebidasteste, serie="Real values")+
    autolayer(prodbebidasvaluests.ets.forecasts$mean, serie="Forecast")+
    scale_colour_viridis_d()+
    scale_y_continuous(labels=scales::comma)+
    theme_bw()
)
```



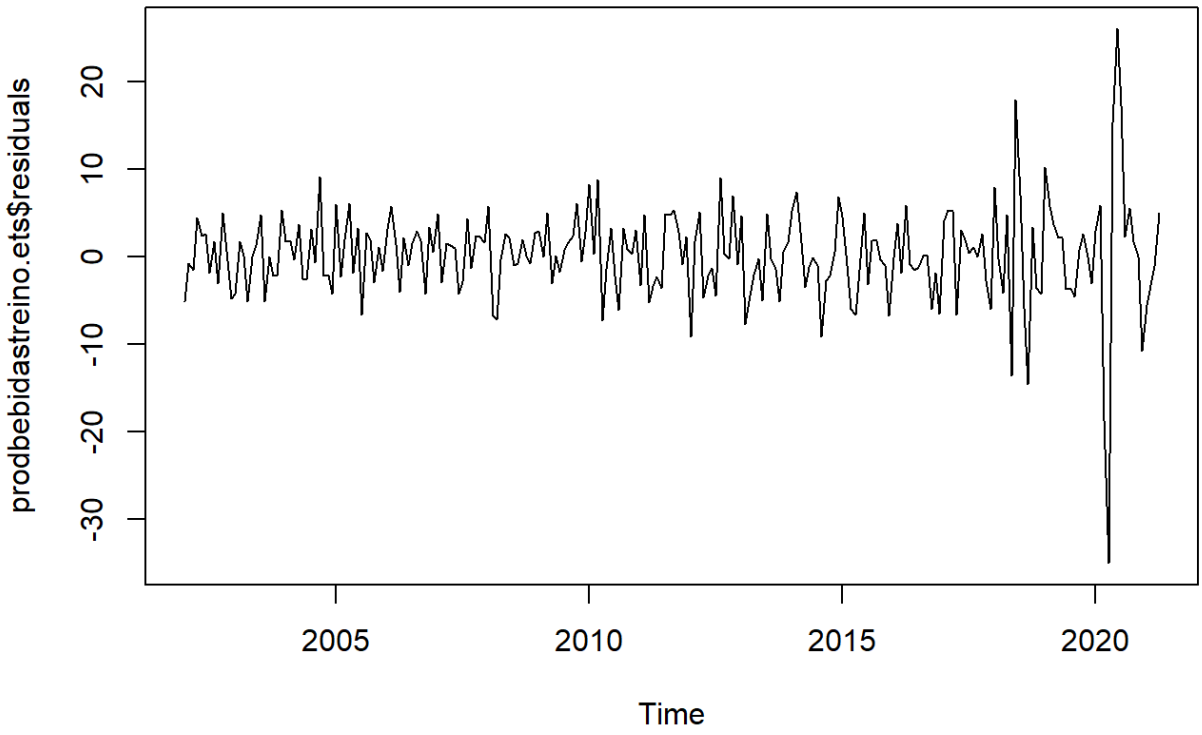
2.7 - Verificando as estatísticas de precisão no conjunto de dados de teste

```
library('forecast')
forecast::accuracy(prodbebidasvaluests.ets.forecasts$mean,prodbebidasteste)
```

```
##                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -1.182813 6.69891 5.432951 -1.285254 5.634968 0.3820066 0.8175471
```

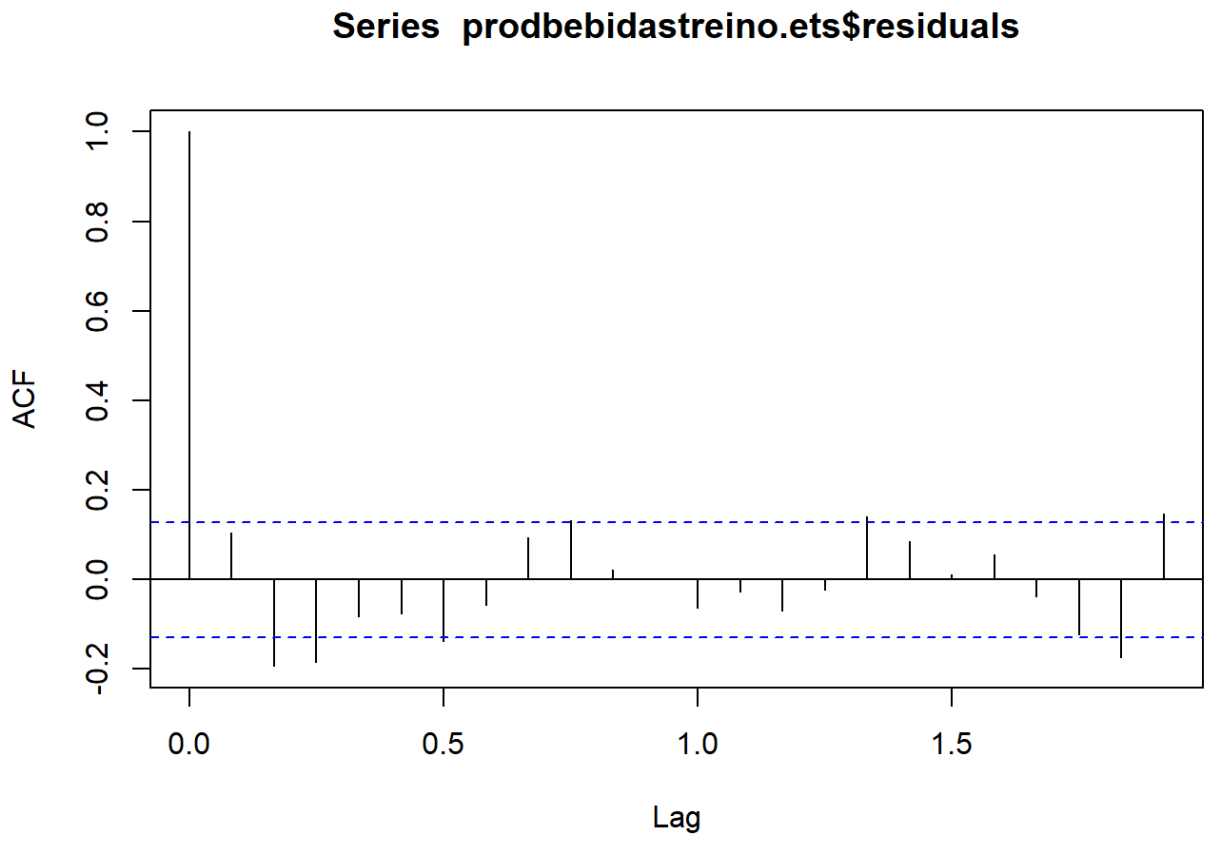
2.8 - Plotando os resíduos estimados

```
plot(prodbebidastreino.ets$residuals)
```



2.9 - Plotando o gráfico de autocorrelação dos resíduos

```
acf(prodbebidastreino.ets$residuals)
```



Pela visualização do gráfico podemos afirmar que não existe correlação significante entre os resíduos, já que nenhum resultado (além do primeiro) ultrapassa a tolerância da média dos resíduos.

2.10 - Verificando se o modelo possui desajustes

```
Box.test(prodbebidastreino.ets$residuals, lag=1, type=c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  prodbebidastreino.ets$residuals
## X-squared = 2.5776, df = 1, p-value = 0.1084
```

Como o valor do p é maior que 0.05, podemos afirmar que o mesmo não possui desajustes.

3 - Modelo ARIMA/SARIMA

PS - Vamos usar os mesmos valores de treino e teste do modelo ETS

```
# prodbebidasvaluests # série temporal
# prodbebidastreino # dados de treino
# prodbebidasteste # dados de teste
```

3.1 - Teste de sazonalidade

Vamos verificar a sazonalidade do da série, pois o modelo ARIMA não é utilizado em séries com sazonalidade, para isso usamos o modelo SARIMA

```
library(seastests)
seastests::combined_test(prodbebidasvaluests)
```

```
## Test used:  WO
##
## Test statistic:  1
## P-value:  0 0 0
```

Como os valores de P são menores do que 0,01 (QS-test) e 0,002 (kw-test) respectivamente, então podemos afirmar que se trata de uma série com sazonalidade. Desta forma deve-se usar o modelo SARIMA

3.2 - Teste Estacionário (Dickey-Fuller)

```
library('urca')
prodbebidastreinodf=ur.df(prodbebidastreino, selectlags = 'BIC', type = 'none')
prodbebidastreinodf
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -0.6081
```

```
prodbebidastreinodf=ur.df(prodbebidastreino, selectlags = 'BIC', type = 'drift')
prodbebidastreinodf
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -4.9646 12.3445
```

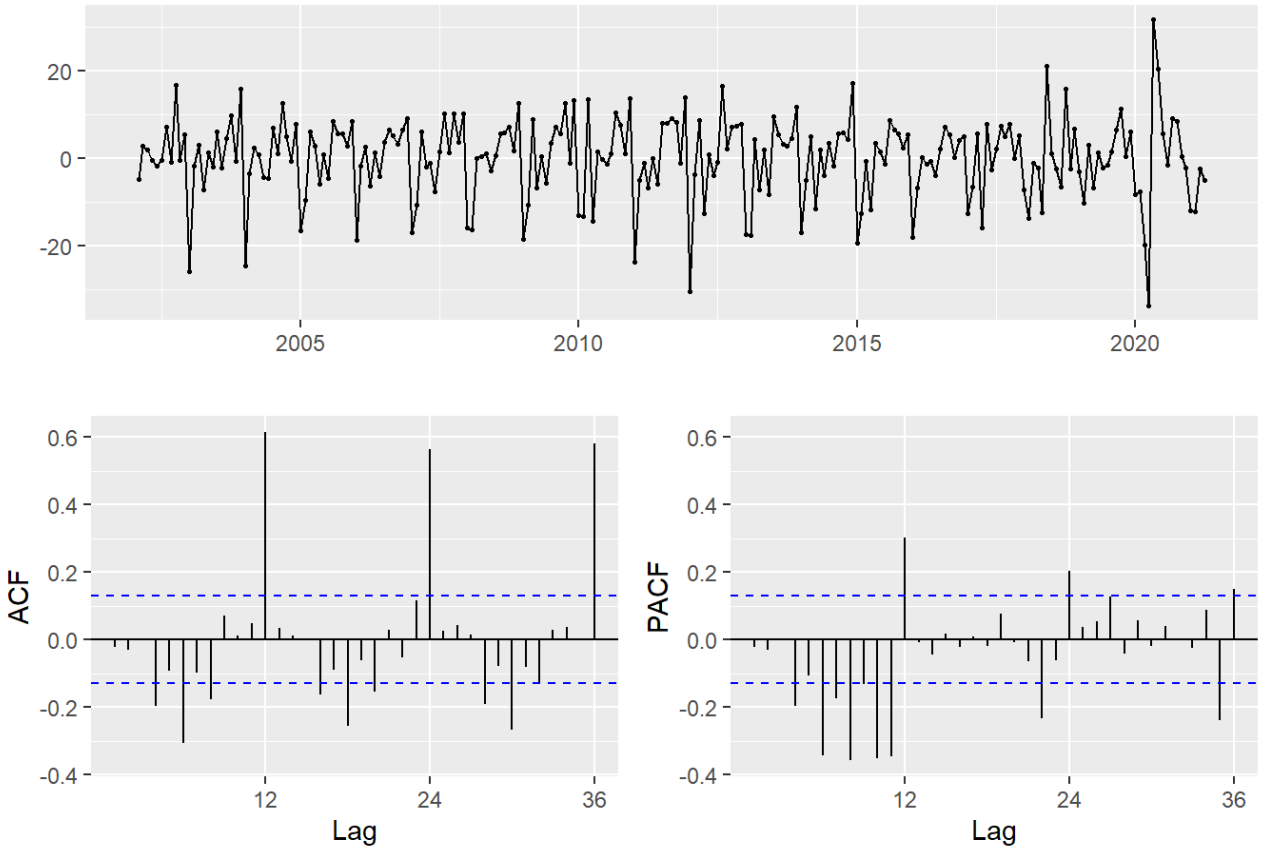
```
prodbebidastreinodf=ur.df(prodbebidastreino, selectlags = 'BIC', type = 'trend')
prodbebidastreinodf
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -6.2351 13.0424 19.5416
```

Como os valores de `z.lag.1` são significativamente diferentes nos 3 testes, não podemos afirmar que este modelo está estacionário.

3.3 - Aplicando a função de diferenciação para torna-la estacionária

```
difprodbebidastreino <- diff(prodbebidastreino)
ggtsdisplay(difprodbebidastreino)
```



3.4 - Repetindo o teste Dickey-Fuller para verificar se está estacionária

```
difprodbebidastreinodf=ur.df(difprodbebidastreino, selectlags = 'BIC', type = 'none')
difprodbebidastreinodf
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -11.0857
```

```
difprodbebidastreinodf=ur.df(difprodbebidastreino, selectlags = 'BIC', type = 'drift')
difprodbebidastreinodf
```

```
##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -11.0634 61.201
```

```
difprodbebidastreinodf=ur.df(difprodbebidastreino, selectlags = 'BIC', type = 'trend')
difprodbebidastreinodf
```

```
##
## ################################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ################################################################
##
## The value of the test statistic is: -11.0497 40.7016 61.0512
```

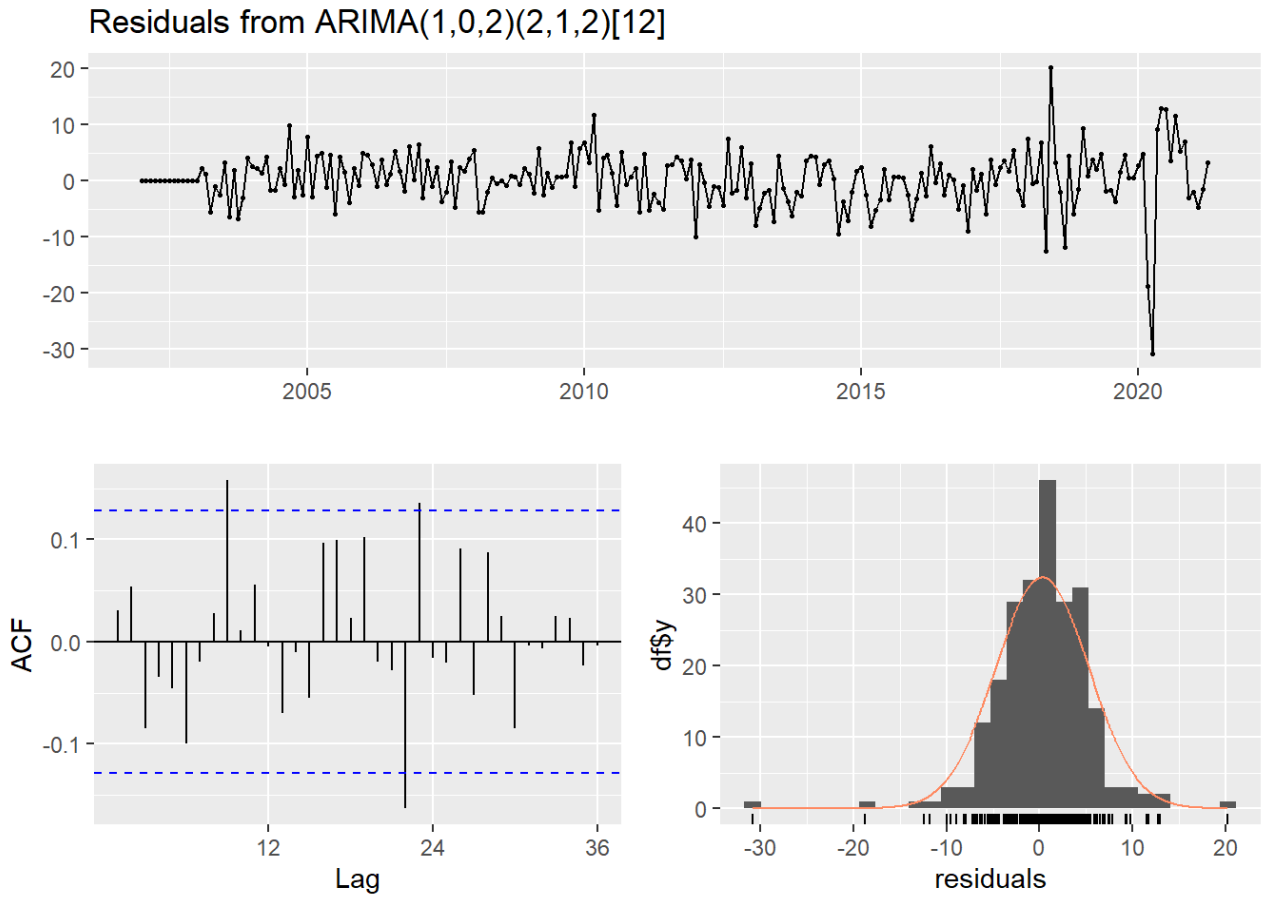
Agora podemos verificar que os valores de z.lag.1 são siginificativamente iguais. Logo podemos afirmar que a série após aplicar as diferenciações está estacionária.

3.5 - Aplicando a função auto.arima, que sozinha identifica que existe a sazonalidade e aplica o modelo SARIMA

```
arimaprodbebidas=auto.arima(prodbebidastreino, trace=F)
```

3.6 - Teste de resíduo de Ljung-Box

```
checkresiduals(arimaprodbebidas)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,2)(2,1,2)[12]
## Q* = 34.597, df = 17, p-value = 0.007024
##
## Model df: 7.   Total lags used: 24
```

No teste de resíduos estamos verificando se os mesmos nao são correlacionados. Logo, como o valor de p (0.007024), é maior do que 0.05 podemos aceitar a hipótese de que os resíduos não são correlacionados.

3.7 - Teste de normalidade

```
ks.test(arimaprodbebidas$residuals, "pnorm", mean(arimaprodbebidas$residuals), sd(arimaprodbebidas
$residuals))
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  arimaprodbebidas$residuals
## D = 0.067978, p-value = 0.234
## alternative hypothesis: two-sided
```

Como o valor de p é maior do que 0.05, podemos afirmar a hipótese de que existe normalidade nos resíduos

3.8 - Teste de estacionariedade da variância

Verificando de existe o efeito ARCH/GARCH, ou seja, evidência de heterocedasticidade

```
library('FinTS')
```

```
## Carregando pacotes exigidos: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'FinTS'
```

```
## The following object is masked from 'package:forecast':
##
##     Acf
```

```
ArchTest(arimaprodbebidas$residuals)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  arimaprodbebidas$residuals
## Chi-squared = 31.514, df = 12, p-value = 0.001644
```

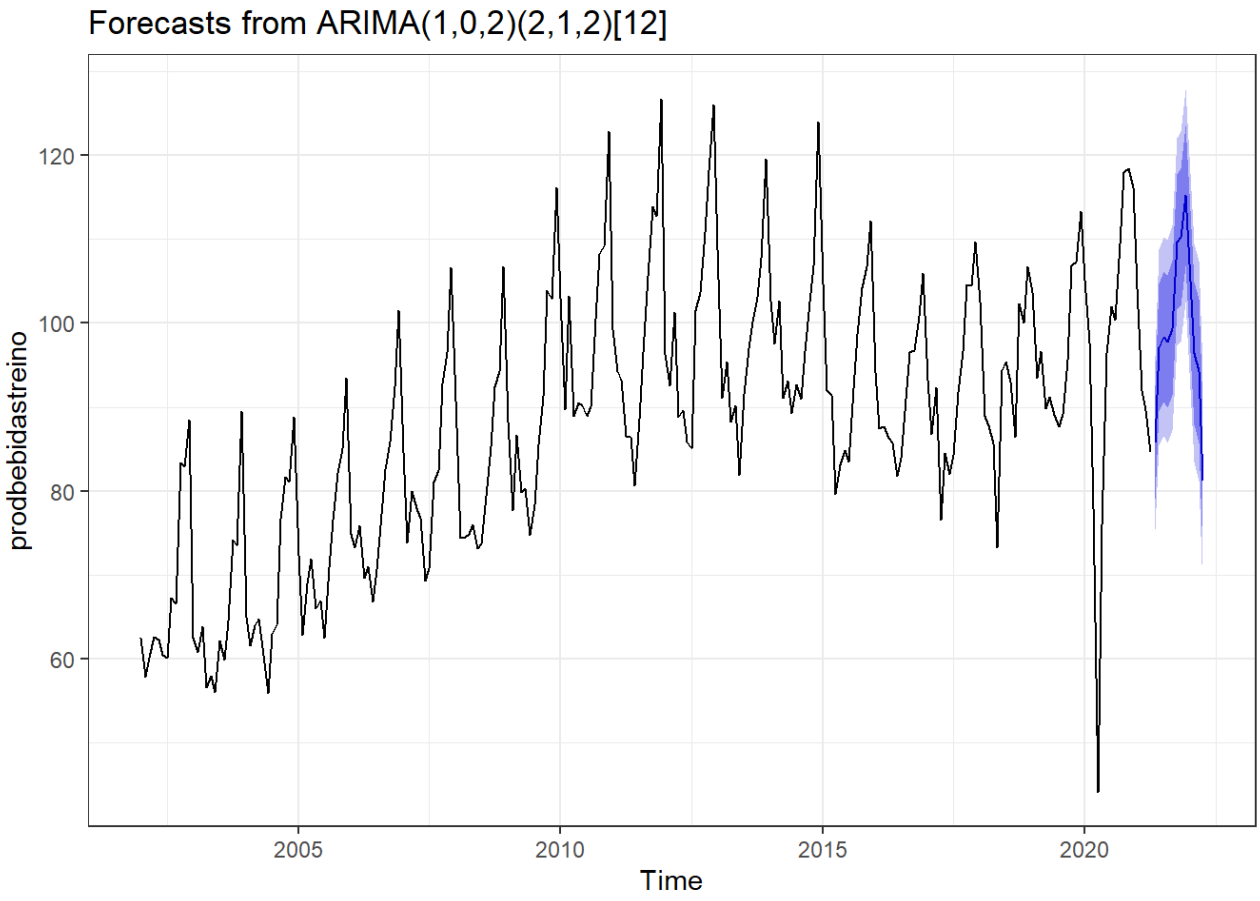
Como o Valor de p (0.001644) é menor do que 0.05, podemos afirmar que a hipótese válida é que existe heterocedasticidade da variância.

3.9 - Realizando a predição dos valores

```
prevprodbebidas=forecast::forecast(arimaprodbebidas, h=12)
prevprodbebidas
```

```
##          Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## May 2021       85.79982  79.06923  92.53040  75.50627  96.09336
## Jun 2021       97.11409  89.48287 104.74532  85.44314 108.78505
## Jul 2021       98.38603  90.63347 106.13859  86.52952 110.24255
## Aug 2021       97.83107  89.96183 105.70031  85.79611 109.86603
## Sep 2021       99.41268  91.43116 107.39421  87.20600 111.61937
## Oct 2021      109.70548 101.61581 117.79515  97.33340 122.07756
## Nov 2021      110.27721 102.08332 118.47110  97.74573 122.80869
## Dec 2021      115.19485 106.90045 123.48924 102.50967 127.88003
## Jan 2022      107.03256  98.64122 115.42391  94.19911 119.86602
## Feb 2022       96.43855  87.95356 104.92353  83.46188 109.41521
## Mar 2022       94.29273  85.71730 102.86816  81.17774 107.40772
## Apr 2022       81.20666  72.54382  89.86951  67.95799  94.45534
```

```
autoplot(prevprodbebidas) + theme_bw()
```



3.10 - Verificando a acurácia do modelo

```
forecast::accuracy(prevprodbebidas, prodbebidasteste)
```

```
##                       ME     RMSE      MAE         MPE     MAPE      MASE
## Training set  0.2754421 5.032150 3.576535  0.04480483 4.228811 0.6297333
## Test set     -1.4487435 8.049874 6.915202 -1.62271540 7.259174 1.2175843
##                    ACF1 Theil's U
## Training set 0.0302036        NA
## Test set     0.2701566 0.9459907
```

3.10 - Plotando gráfico de valores Preditos e valores reais

```
ggplotly(
  autoplot(prodbebidastreino)+
    autolayer(prodbebidasteste,serie="Valores Reais")+
    autolayer(prevprodbebidas$mean, serie="Forecast")+
    scale_colour_viridis_d()+
    scale_y_continuous(labels=scales::comma)+
    theme_bw()
)
```