Universidade de Brasília

DEPARTAMENTO DE ESTATÍSTICA

25 junho 2024

Lista 6

Prof. Dr. Raul Yukihiro Matsushita

Aluno: Bruno Gondim Toledo

Matrícula: 15/0167636

Análise de séries temporais

1º/2024

Tente analisar separadamente as seguintes séries temporais mensais encontradas no arquivo "ALGONQUIN_PARK_Ontario_Canada.csv":

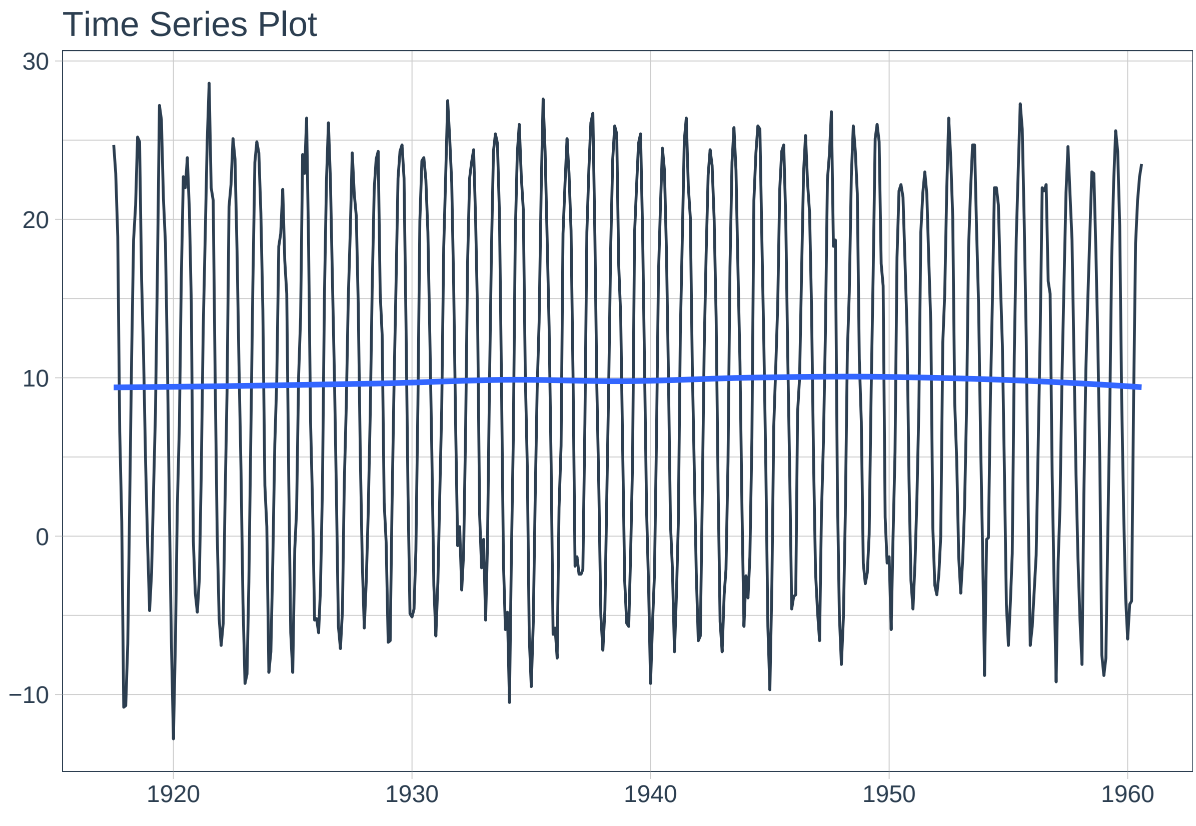
1. "Mean Max Temp (°C)";

```
df1 = df[,c(5,8)]
df1 = df1 %>% drop_na()
colnames(df1)[2] = "mean_max_temp"
```

```
df1$data = as.Date(df1$data)
df1 = tsibble::as_tsibble(df1,index=data,regular=F)
```

Visualizando a série com uma linha suavizada

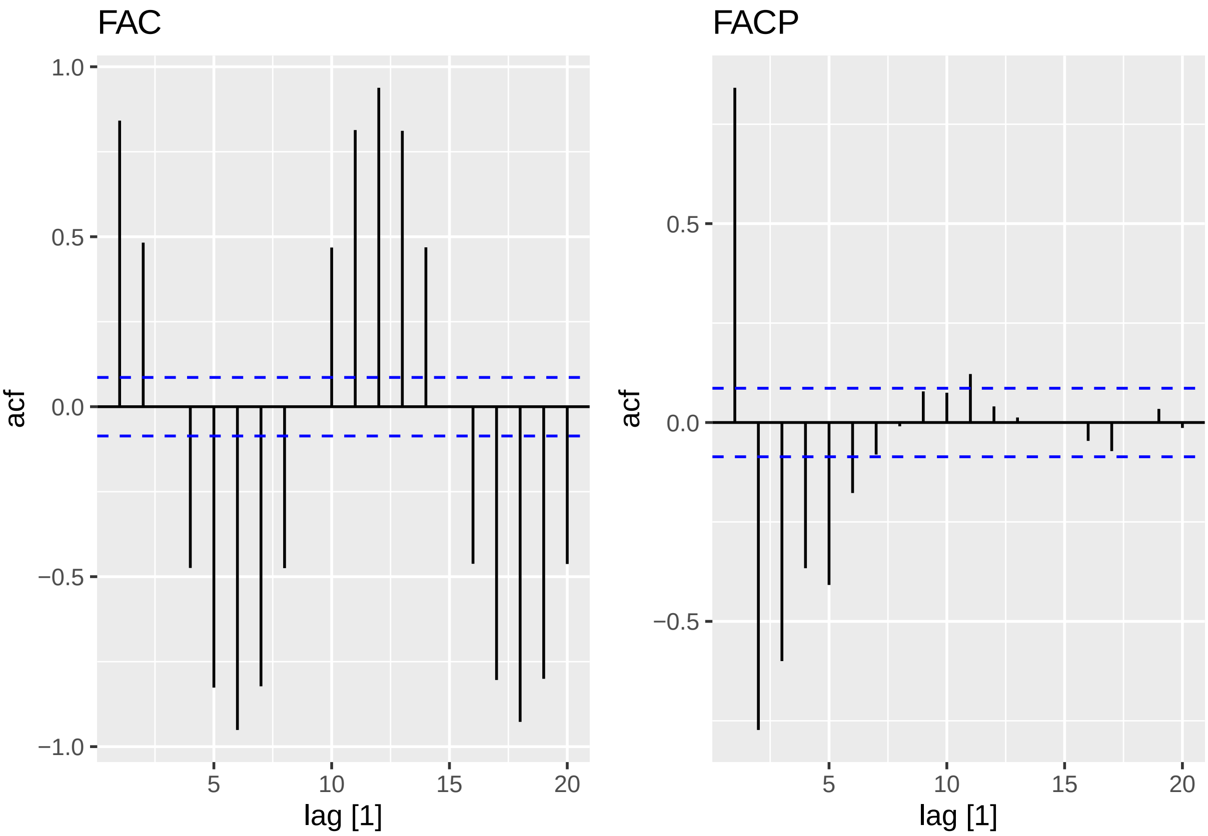
```
df1 %>%
  plot_time_series(data,mean_max_temp, .interactive = F, .smooth = T)
```




```
FAC1 = df1 %>%
  ACF(var = mean_max_temp,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC")

FACP1 = df1 %>%
  ACF(var = mean_max_temp,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP")
```

```r
grid.arrange(FAC1, FACP1, nrow = 1)
```




```r
aTSA::adf.test(ts(df1$mean_max_temp), nlag = 5)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0  -4.94    0.01
## [2,]   1 -12.03    0.01
## [3,]   2 -13.70    0.01
## [4,]   3  -8.66    0.01
## [5,]   4  -5.86    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0  -6.61    0.01
## [2,]   1 -18.27    0.01
## [3,]   2 -28.92    0.01
## [4,]   3 -26.80    0.01
## [5,]   4 -27.87    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0  -6.61    0.01
## [2,]   1 -18.26    0.01
## [3,]   2 -28.91    0.01
## [4,]   3 -26.78    0.01
## [5,]   4 -27.87    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

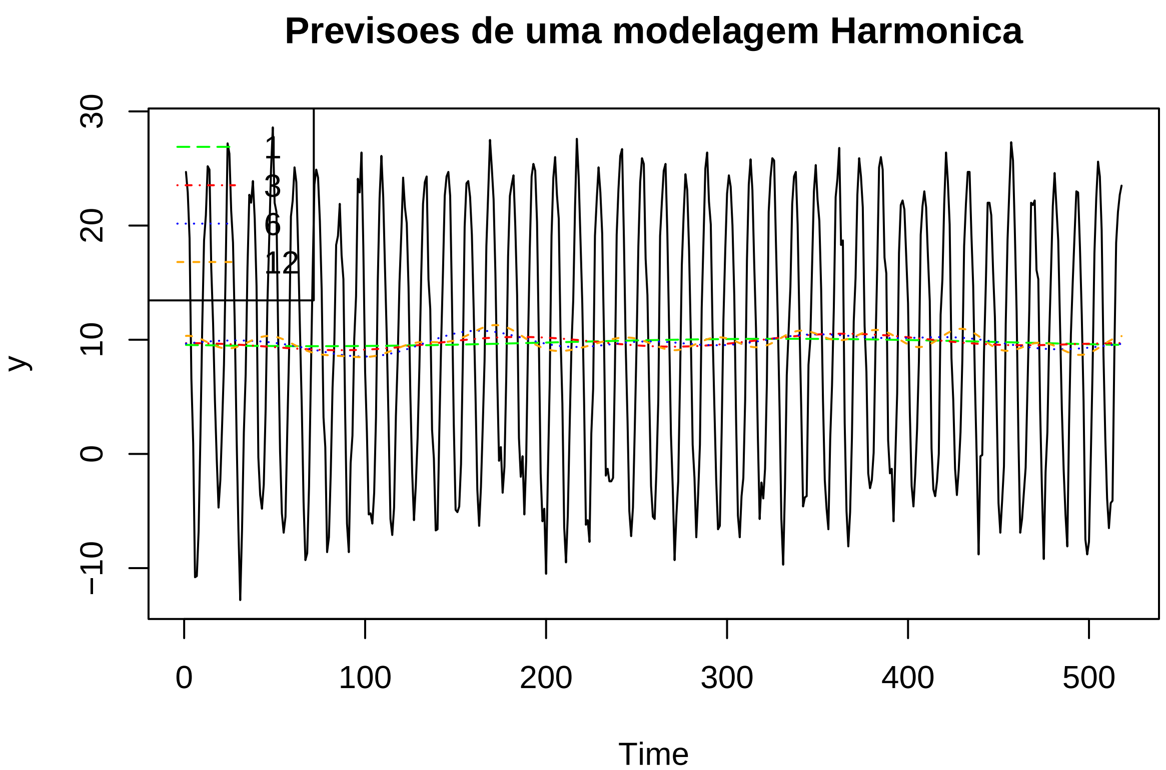
O teste de Dickey-Fuller sugere a estacionariedade da série

```
y <- ts(df1$mean_max_temp)

fit1 <- haRmonics(y = y,
                  numFreq = 1,
                  delta = 0.1)
fit3 <- haRmonics(y = y,
                   numFreq = 3,
                   delta = 0.1)
fit6 <- haRmonics(y = y,
                    numFreq = 6,
                    delta = 0.1)
fit12 <- haRmonics(y = y,
                    numFreq = 12,
                    delta = 0.1)

x = ts(fit1$fitted)
w = ts(fit3$fitted)
z = ts(fit6$fitted)
v = ts(fit12$fitted)

plot(y, pch = 16, main = "Previsoes de uma modelagem Harmonica")
lines(x ,lty = 5, col = "green")
lines(w, lty = 4, col = "red")
lines(z, lty = 3, col = "blue")
lines(v, lty = 2, col = "orange")
legend("topleft", legend = c("1", "3", "6","12"),
       col = c("green", "red", "blue","orange"), lty = c(5, 4, 3,2))
```



A modelagem harmônica foi satisfatória em capturar todas as estruturas da série, deixando apenas ruido branco.

Visto que a série não necessita de diferenciação, farei a modelagem sarima com ordem de sazonalidade = 12, argumentos d = D = 0. Iniciei com grid de parâmetros p,P,q,Q no intervalo [0,2]; e após uma iteração e avaliação, expandi o grid para intervalo [0,3] para alguns parâmetros cujo melhor modelo se encontrava na fronteira do intervalo.

```
modelos <- data.frame(p = integer(),
                      P = integer(),
                      q = integer(),
                      Q = integer(),
                      AIC = numeric(),
                      BIC = numeric())

tic()
for(p in 0:3){
  for(P in 0:2){
      for(q in 0:3){
          for(Q in 0:3){
            tryCatch({
              fit = astsa::sarima(y, details = FALSE, Model = FALSE,
                          p = p, d = 0, q = q, P = P, D = 0, Q = Q, S = 12)
              AIC = fit$ICs[1]
              BIC = fit$ICs[3]
              mod = c(p, P, q, Q, AIC, BIC)
              modelos = rbind(modelos, mod)
            }, error = function(e) {
            })
        }
      }
    }
 }
toc()

colnames(modelos) = c("p","P","q","Q","AIC","BIC")

modelo = modelos %>%
  arrange(BIC, AIC) %>% head(1)
```

Logo, o melhor modelo (minimiza BIC) é o com os parâmetros:

```
kable(modelo)
```

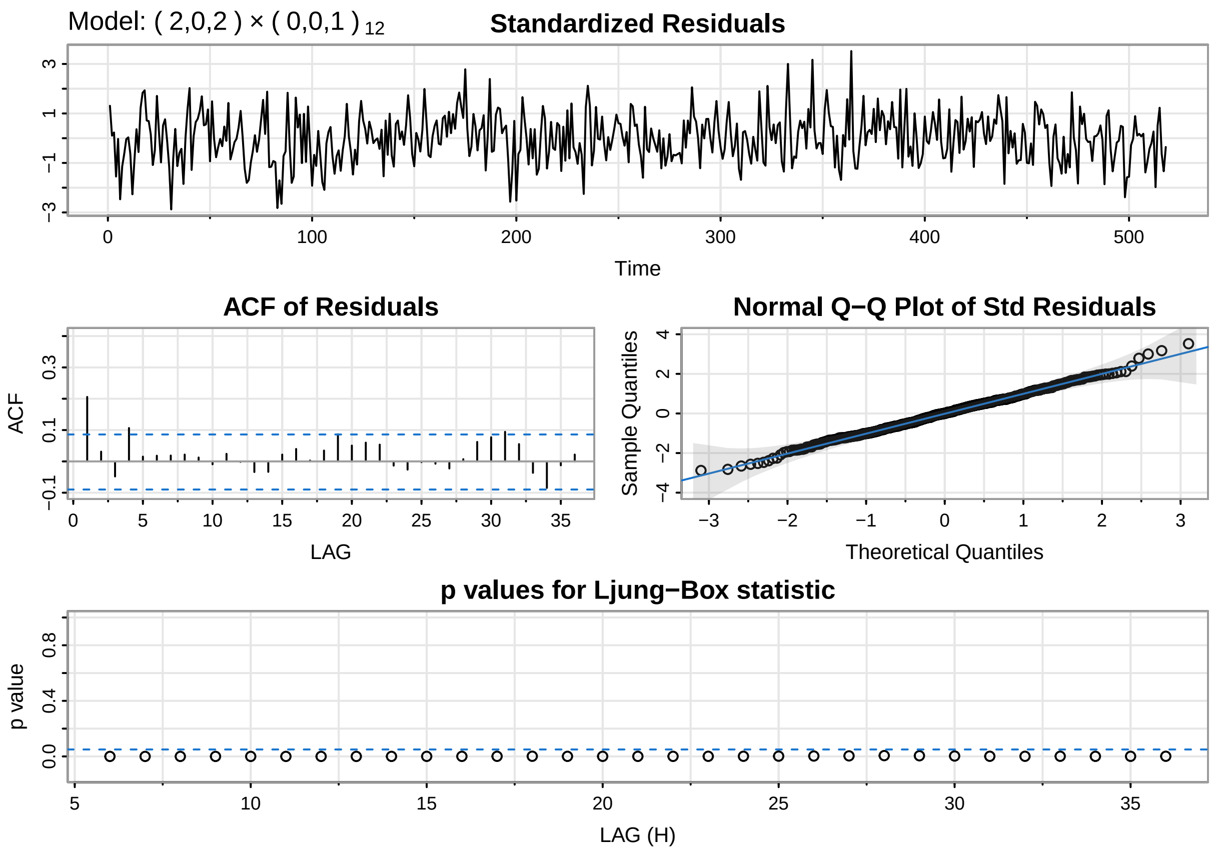
p	P	q	Q	AIC	BIC
2	0	3	0	4.409052	4.466484

```
fit = astsa::sarima(y,p = 2, d = 0, q = 2, P = 0, D = 0, Q = 1, S = 12)
```

```
## initial  value 2.431138
## iter   2 value 2.375536
## iter   3 value 2.223394
## iter   4 value 1.654624
## iter   5 value 1.604353
## iter   6 value 1.471418
## iter   7 value 1.426926
## iter   8 value 1.399208
## iter   9 value 1.395788
## iter  10 value 1.385441
## iter  11 value 1.377935
```

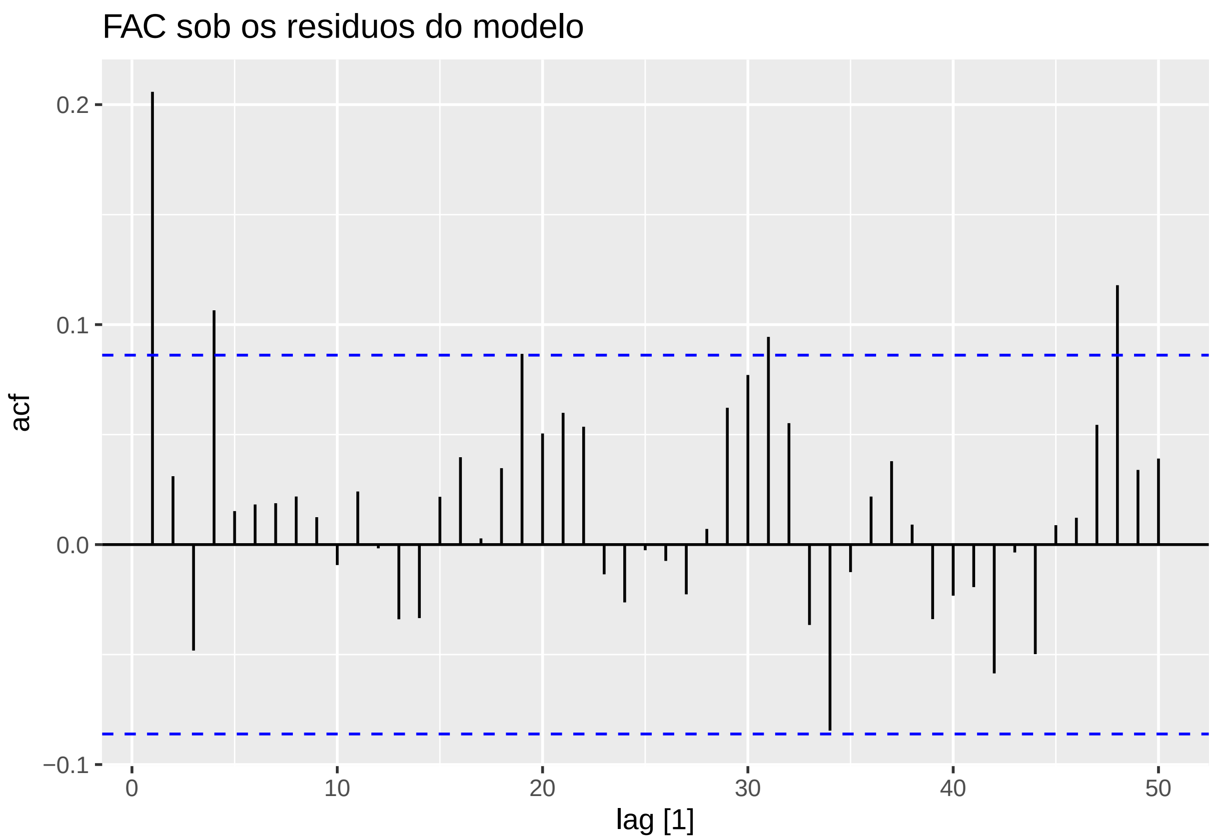
```
## iter  12 value 1.328844
## iter  13 value 1.285380
## iter  14 value 1.246665
## iter  15 value 1.213701
## iter  16 value 1.170776
## iter  17 value 1.096903
## iter  18 value 0.986754
## iter  19 value 0.978539
## iter  20 value 0.945537
## iter  21 value 0.914273
## iter  22 value 0.896354
## iter  23 value 0.872350
## iter  24 value 0.839701
## iter  25 value 0.819017
## iter  26 value 0.807159
## iter  27 value 0.806953
## iter  28 value 0.806706
## iter  29 value 0.804080
## iter  30 value 0.797026
## iter  31 value 0.787555
## iter  32 value 0.787142
## iter  33 value 0.786900
## iter  34 value 0.786678
## iter  35 value 0.786668
## iter  36 value 0.786627
## iter  37 value 0.786582
## iter  38 value 0.786571
## iter  39 value 0.786570
## iter  40 value 0.786570
## iter  41 value 0.786570
## iter  42 value 0.786569
## iter  43 value 0.786568
## iter  44 value 0.786568
## iter  45 value 0.786568
## iter  45 value 0.786568
## iter  45 value 0.786568
## final  value 0.786568
## converged
## initial  value 0.794626
## iter   2 value 0.794581
## iter   3 value 0.794467
## iter   4 value 0.794466
## iter   5 value 0.794449
## iter   6 value 0.794447
## iter   7 value 0.794444
## iter   8 value 0.794443
## iter   9 value 0.794441
## iter  10 value 0.794435
## iter  11 value 0.794423
## iter  12 value 0.794403
## iter  13 value 0.794388
## iter  14 value 0.794381
## iter  15 value 0.794381
## iter  16 value 0.794381
## iter  17 value 0.794381
## iter  18 value 0.794380
## iter  18 value 0.794380
## iter  18 value 0.794380
```

```
## final  value 0.794380
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##         Estimate     SE    t.value p.value
## ar1       1.7322 0.0003  5104.7528  0.0000
## ar2      -0.9999 0.0001 -9291.5585  0.0000
## ma1      -1.6595 0.0173   -95.7827  0.0000
## ma2       0.9426 0.0169    55.8322  0.0000
## sma1     -0.0184 0.0470    -0.3911  0.6959
## xmean     9.7097 0.1000    97.1005  0.0000
##
## sigma^2 estimated as 4.795605 on 512 degrees of freedom
##
## AIC = 4.453665  AICc = 4.453982  BIC = 4.511097
##
```

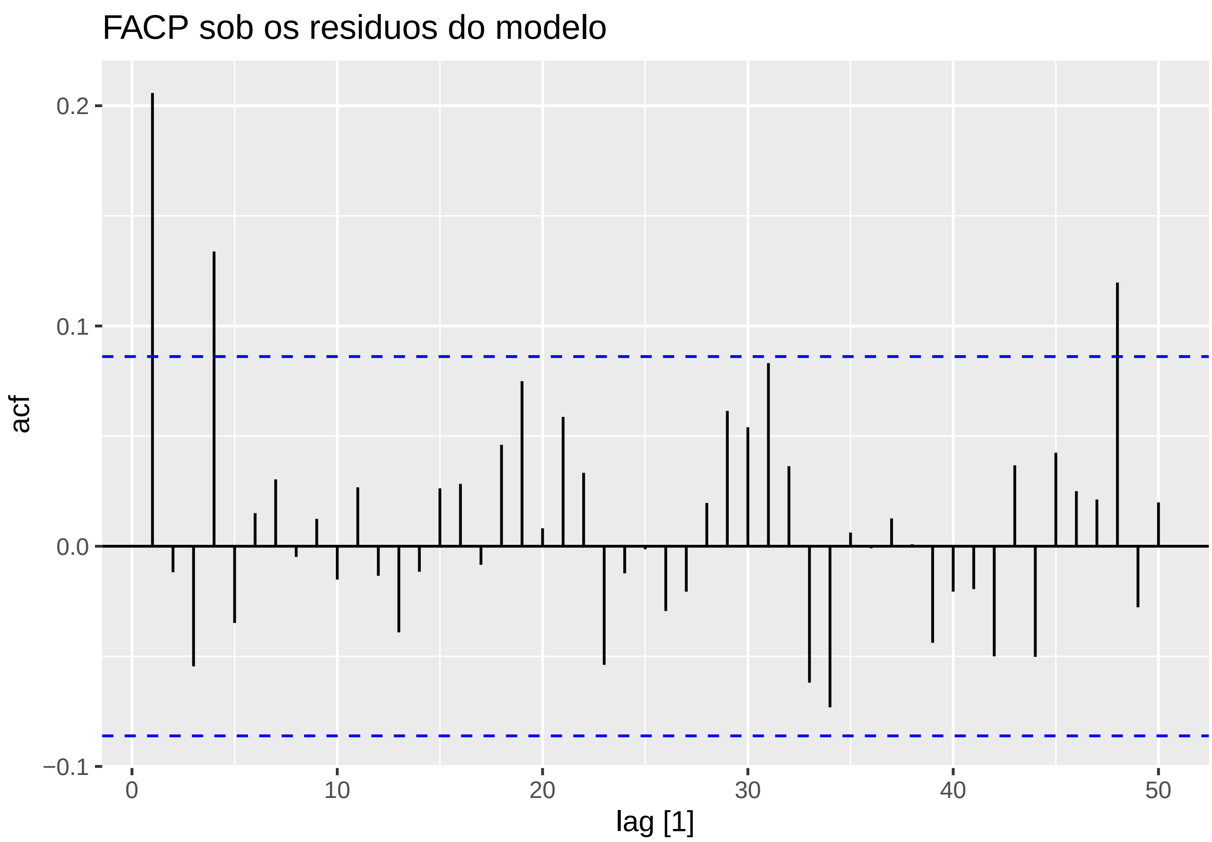



```
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 50) %>%
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```
res %>%
  ACF(value,lag_max = 50,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```



```r
Box.test(fit[["fit"]][["residuals"]], lag = 50, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 77.434, df = 50, p-value = 0.007698
```

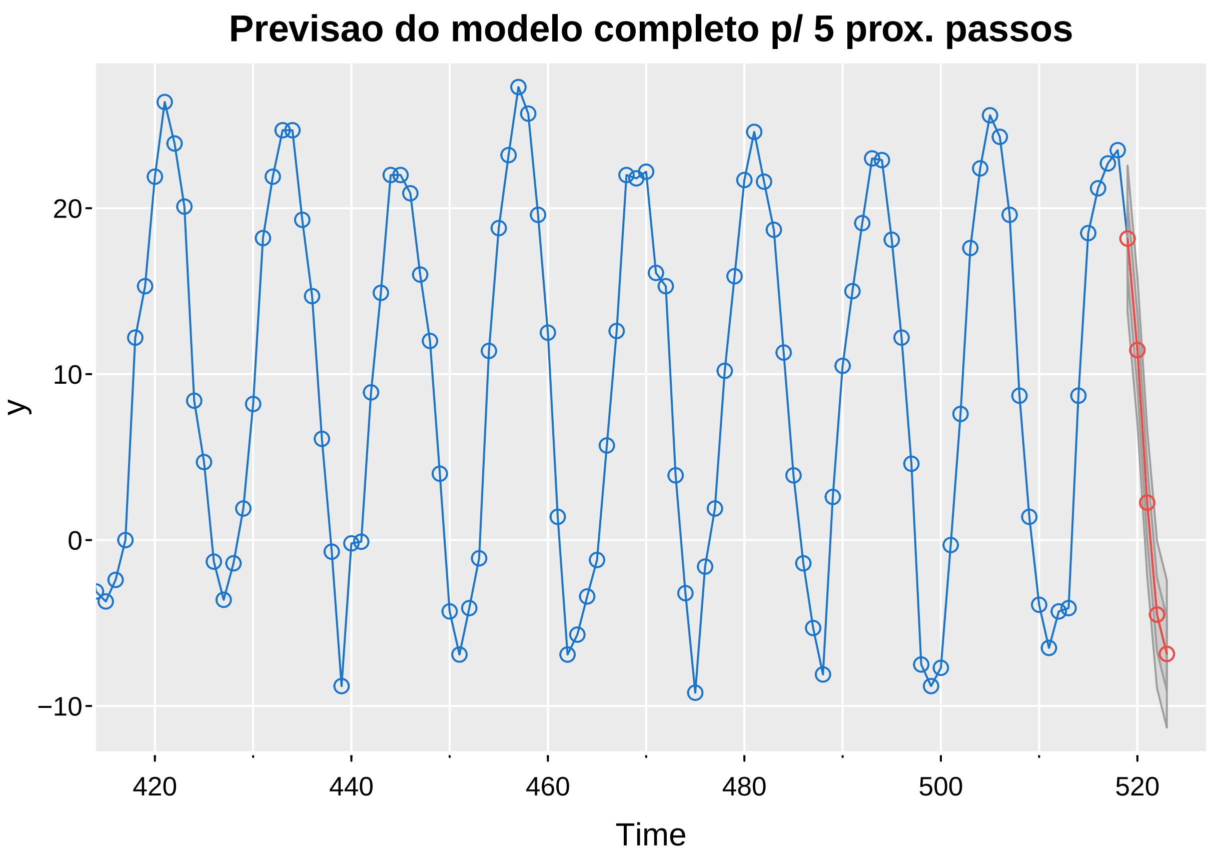
```r
shapiro.test(fit[["fit"]][["residuals"]])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.99735, p-value = 0.5794
```

Sob a hipótese nula de independência, o teste de Ljung-Box não rejeita a hipótese nula. Portanto, existem indícios de independência na série dos resíduos.

Avaliando as formas da FAC, FACP e resultado do teste de Ljung-Box, não aparenta haver autocorrelação significativa com lag $= 50$ para os resíduos, sugerindo que os resíduos se tratam de ruído branco. Logo, este modelo aparenta ter ajustado quanto a estrutura de dados, sendo um modelo adequado para previsões. Além disso, o teste de Shapiro-Wilk não rejeita a normalidade dos resíduos.

```r
prev = sarima.for(y,p = 0, d = 1, q = 1, P = 0, D = 1, Q = 1, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Previsao do modelo completo p/ 5 prox. passos")
```

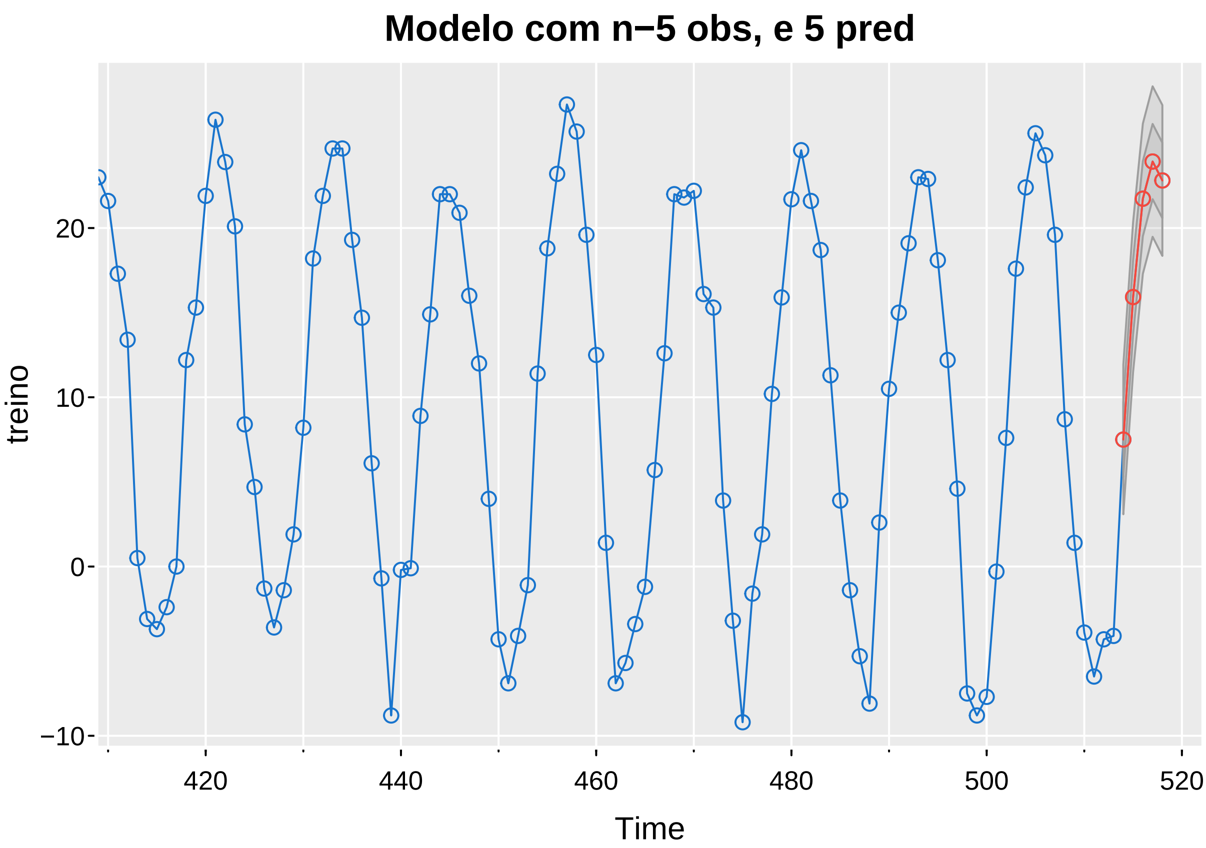


```
prev
```

```
## $pred
## Time Series:
## Start = 519
## End = 523
## Frequency = 1
## [1] 18.172860 11.449605  2.254256 -4.489930 -6.862023
##
## $se
## Time Series:
## Start = 519
## End = 523
## Frequency = 1
## [1] 2.199776 2.204762 2.209737 2.214700 2.219653
```

```
treino = ts(y[1:513])
teste = ts(y[514:518])
```

```
prev = sarima.for(treino,p = 0, d = 1, q = 1, P = 0, D = 1, Q = 1, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Modelo com n-5 obs, e 5 pred")
```




```
prev$pred
```

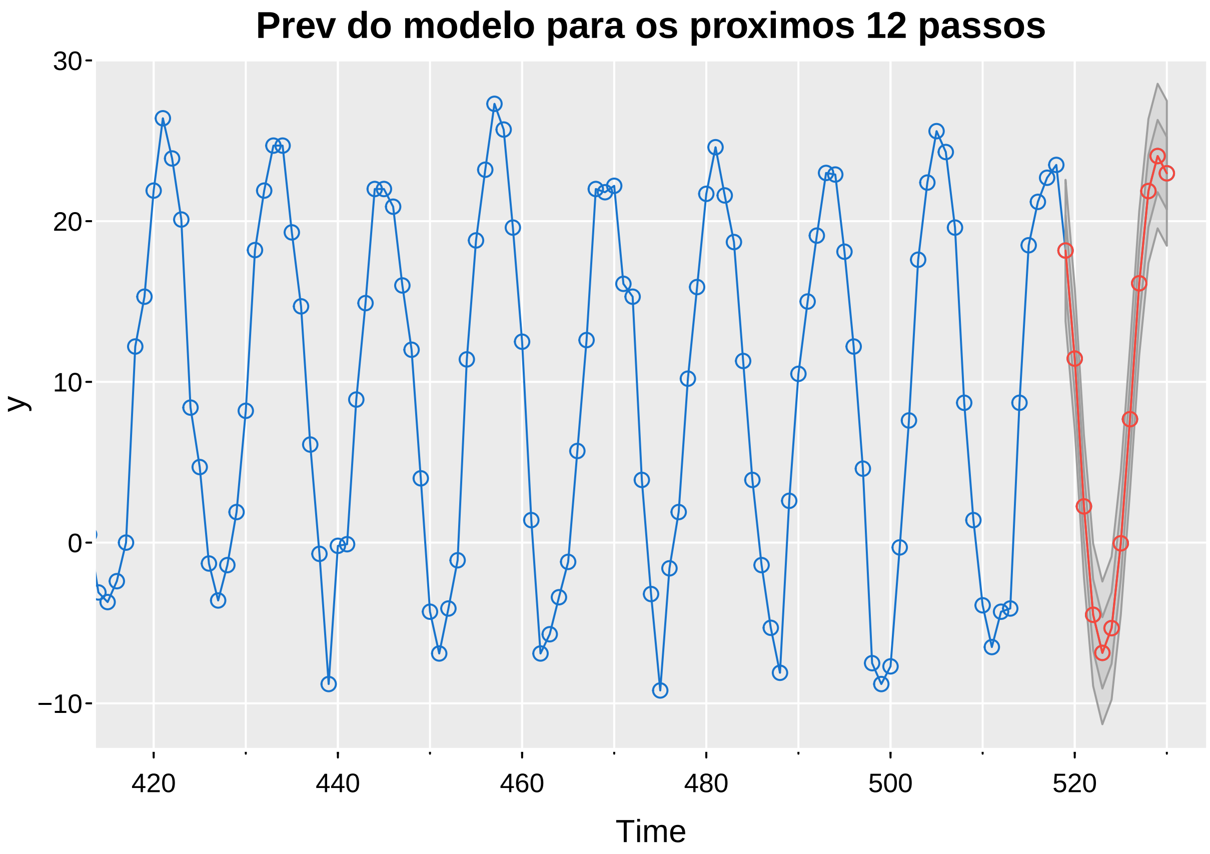
```
## Time Series:
## Start = 514
## End = 518
## Frequency = 1
## [1]  7.501256 15.922684 21.734588 23.929051 22.810446
```

```r
MAPE = MAPE(prev$pred, as.vector(teste))
MAPE
```

```
## [1] 0.07716069
```

Com um MAPE $= 0.0771607$, ou seja, estando abaixo de 10%, podemos dizer que se trata de um excelente modelo preditivo.

```r
prev = sarima.for(y,p = 0, d = 1, q = 1, P = 0, D = 1, Q = 1, S = 12,
           n.ahead = 12, gg=TRUE, col=4,main="Prev do modelo para os proximos 12 passos")
```



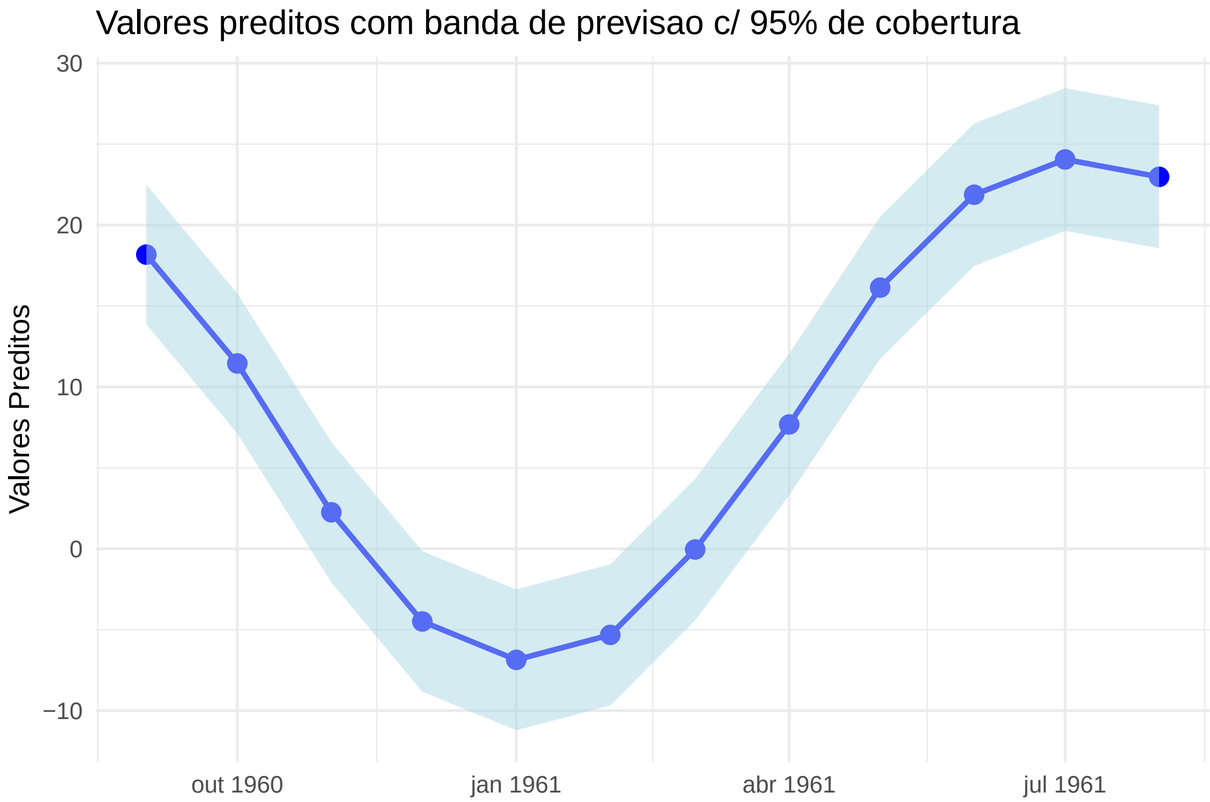

```r
datas <- seq.Date(from = as.Date("1960-09-01"), to = as.Date("1961-08-31"), by = "months")

previsoes = data.frame(datas,prev$pred,prev$se)
colnames(previsoes) = c("data","Valores_preditos","Erro_padrao")

previsoes = previsoes %>%
  mutate(Limite_superior = Valores_preditos + (1.96 * Erro_padrao),
         Limite_inferior = Valores_preditos - (1.96 * Erro_padrao))

ggplot(previsoes, aes(x = data)) +
  geom_line(aes(y = Valores_preditos), color = "blue", size = 1) +
  geom_point(aes(y = Valores_preditos), color = "blue", size = 3) +
  geom_ribbon(aes(ymin = Limite_inferior, ymax = Limite_superior), fill = "lightblue", alpha = 0.5)
  labs(title = "Valores preditos com banda de previsao c/ 95% de cobertura",
       x = "",
       y = "Valores Preditos") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



Os diagnósticos do modelo foram todos atendidos com folga. Portanto, as previsões deste modelo apresentam alta credibilidade. Considerando a banda de confiança e a especificidade dos dados, é bastante improvável que este modelo se torne inútil, mesmo a longo prazo. Ainda assim, novos valores devem ser constantemente adicionados para modelagem, visto que em algum ponto este comportamento pode variar.

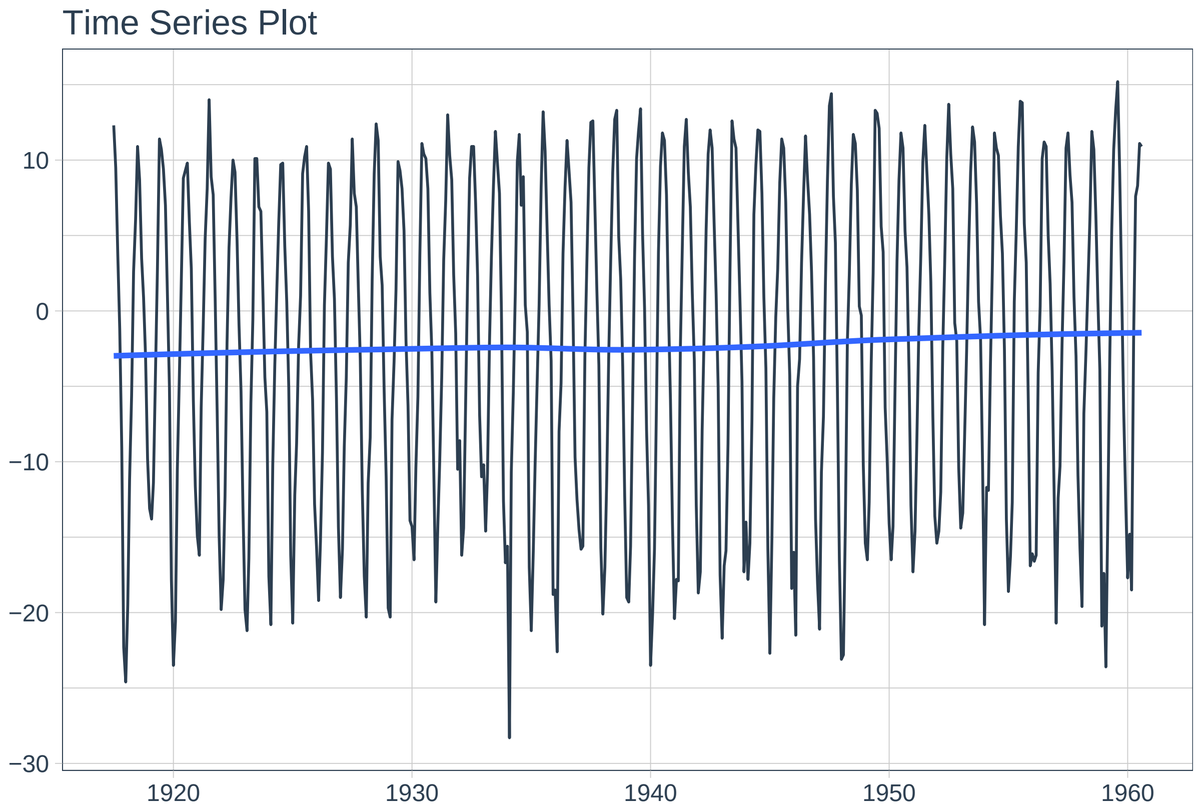
2. "Mean Min Temp (°C)";

```
df2 = df[,c(5,10)]
df2 = df2 %>% drop_na()
colnames(df2)[2] = "mean_min_temp"
```

```
df2$data = as.Date(df2$data)
df2 = tsibble::as_tsibble(df2,index=data,regular=F)
```

Visualizando a série com uma linha suavizada

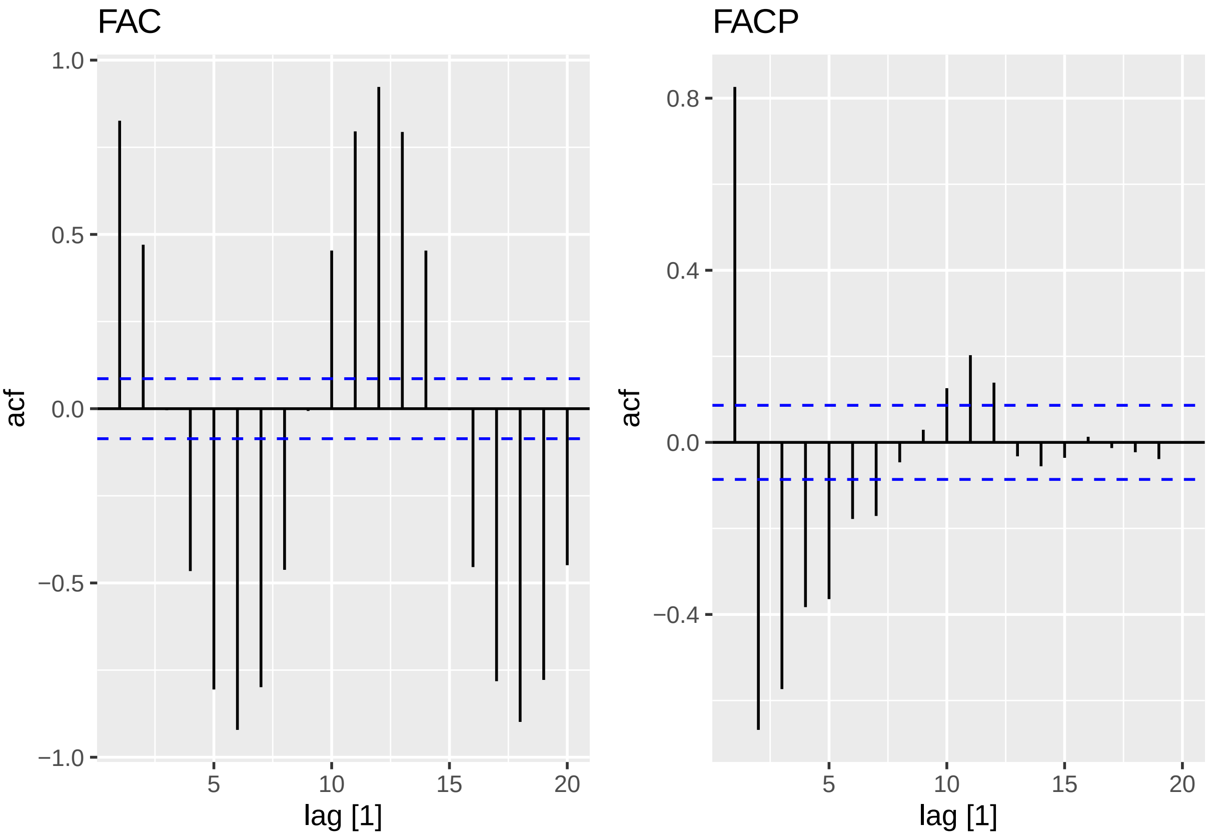
```
df2 %>%
  plot_time_series(data,mean_min_temp, .interactive = F, .smooth = T)
```




```
FAC2 = df2 %>%
  ACF(var = mean_min_temp,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC")

FACP2 = df2 %>%
  ACF(var = mean_min_temp,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP")

grid.arrange(FAC2, FACP2, nrow = 1)
```




```
aTSA::adf.test(ts(df2$mean_min_temp), nlag = 5)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0  -6.79    0.01
## [2,]   1 -14.63    0.01
## [3,]   2 -22.81    0.01
## [4,]   3 -21.36    0.01
## [5,]   4 -18.66    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0  -6.96    0.01
## [2,]   1 -15.14    0.01
## [3,]   2 -24.56    0.01
## [4,]   3 -24.92    0.01
## [5,]   4 -25.11    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0  -6.97    0.01
## [2,]   1 -15.15    0.01
## [3,]   2 -24.60    0.01
## [4,]   3 -25.04    0.01
## [5,]   4 -25.40    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

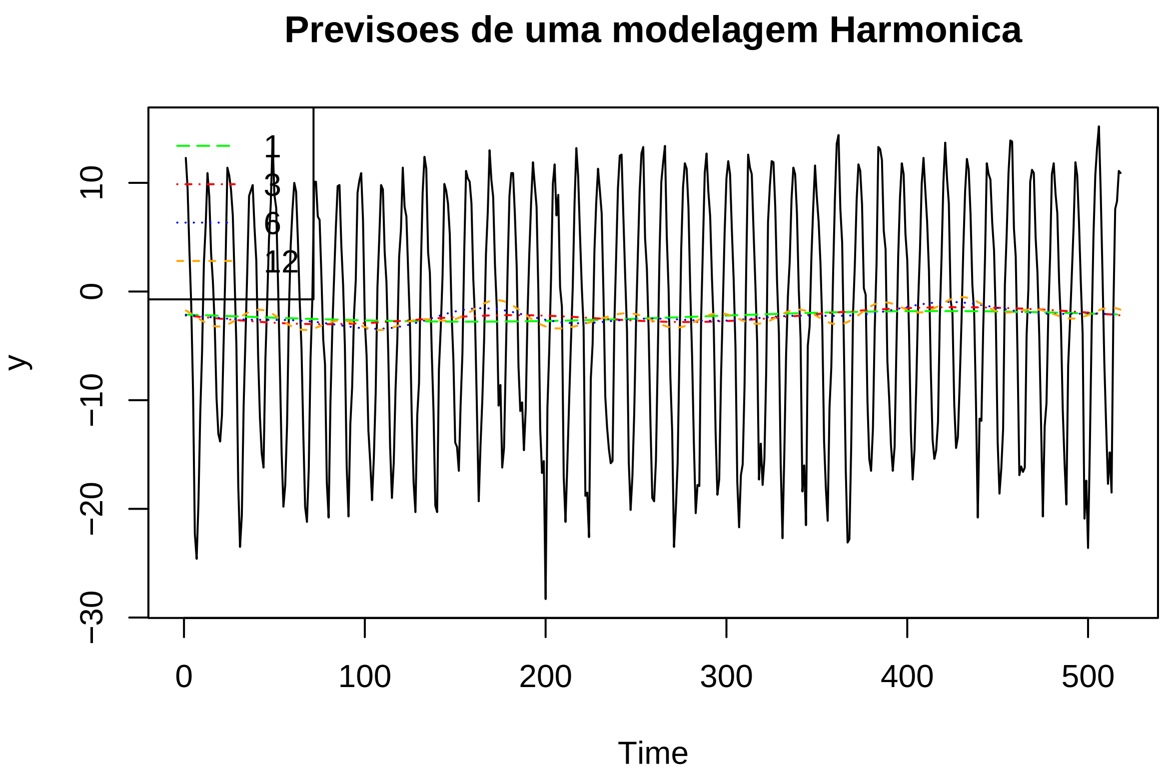
O teste de Dickey-Fuller sugere a estacionariedade da série

```r
y <- ts(df2$mean_min_temp)

fit1 <- haRmonics(y = y,
                  numFreq = 1,
                  delta = 0.1)
fit3 <- haRmonics(y = y,
                   numFreq = 3,
                   delta = 0.1)
fit6 <- haRmonics(y = y,
                    numFreq = 6,
                    delta = 0.1)
fit12 <- haRmonics(y = y,
                    numFreq = 12,
                    delta = 0.1)

x = ts(fit1$fitted)
w = ts(fit3$fitted)
z = ts(fit6$fitted)
v = ts(fit12$fitted)

plot(y, pch = 16, main = "Previsoes de uma modelagem Harmonica")
lines(x ,lty = 5, col = "green")
lines(w, lty = 4, col = "red")
lines(z, lty = 3, col = "blue")
lines(v, lty = 2, col = "orange")
legend("topleft", legend = c("1", "3", "6","12"),
       col = c("green", "red", "blue","orange"), lty = c(5, 4, 3,2))
```



A modelagem harmônica foi satisfatória em capturar todas as estruturas da série, deixando apenas ruido branco.

Visto que a série não necessita de diferenciação, farei a modelagem sarima com ordem de sazonalidade = 12, argumentos d = D = 0. Iniciei com grid de parâmetros p,P,q,Q no intervalo [0,2]; e após uma iteração e avaliação, expandi o grid para intervalo [0,3] para o parâmetro q = 3.

```r
modelos <- data.frame(p = integer(),
                      P = integer(),
                      q = integer(),
                      Q = integer(),
                      AIC = numeric(),
                      BIC = numeric())

tic()
for(p in 0:2){
  for(P in 0:2){
      for(q in 0:3){
          for(Q in 0:2){
            tryCatch({
              fit = astsa::sarima(y, details = FALSE, Model = FALSE,
                                  p = p, d = 0, q = q, P = P, D = 0, Q = Q, S = 12)
              AIC = fit$ICs[1]
              BIC = fit$ICs[3]
              mod = c(p, P, q, Q, AIC, BIC)
              modelos = rbind(modelos, mod)
            }, error = function(e) {
            })
        }
      }
    }
 }
toc()

colnames(modelos) = c("p","P","q","Q","AIC","BIC")

modelo = modelos %>%
  arrange(BIC, AIC) %>% head(1)
```

Logo, o melhor modelo (minimiza BIC) é o com os parâmetros:

```r
kable(modelo)
```

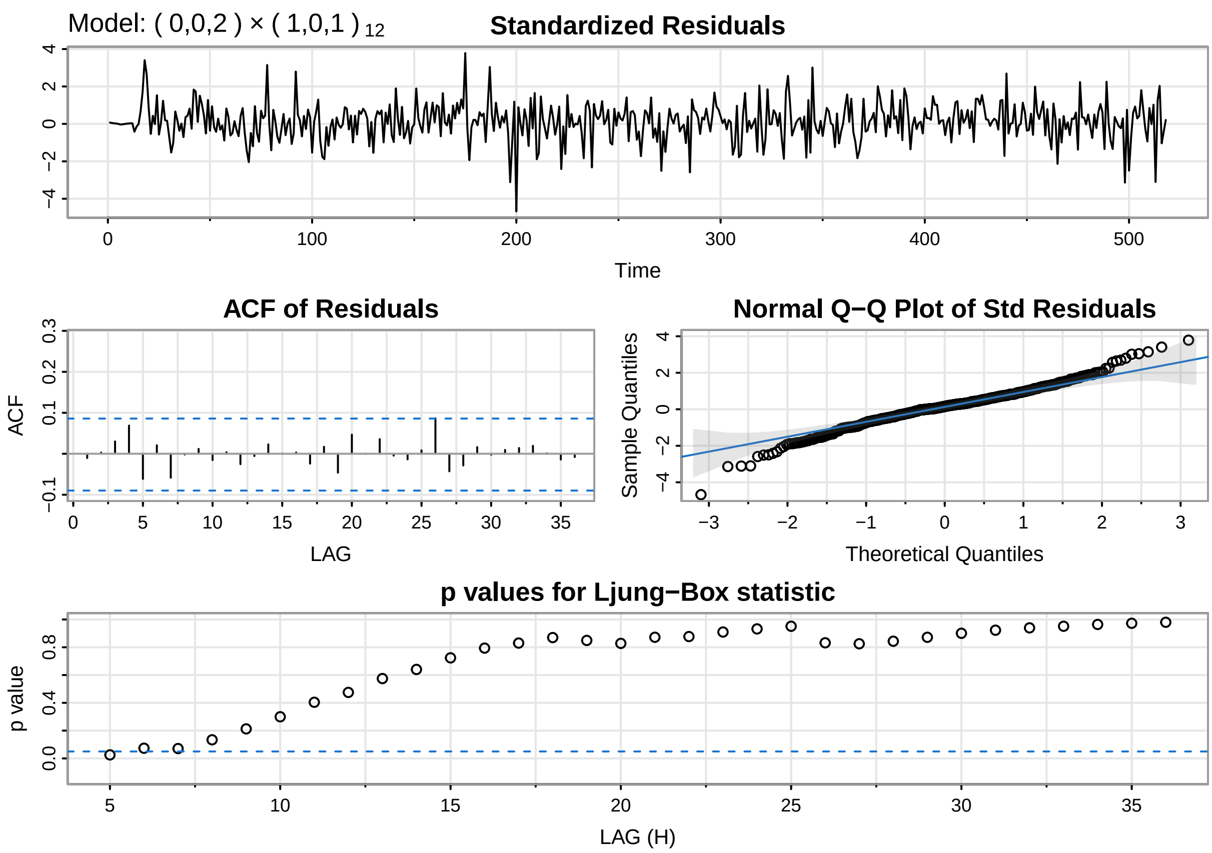
p	P	q	Q	AIC	BIC
0	1	2	1	4.670384	4.719612

```r
fit = astsa::sarima(y,p = 0, d = 0, q = 2, P = 1, D = 0, Q = 1, S = 12)
```

```
## initial  value 2.371297
## iter   2 value 1.865771
## iter   3 value 1.480287
## iter   4 value 1.286134
## iter   5 value 1.190228
## iter   6 value 1.087254
## iter   7 value 1.051798
## iter   8 value 1.008333
## iter   9 value 0.978323
## iter  10 value 0.976892
## iter  11 value 0.974638
## iter  12 value 0.952440
## iter  13 value 0.951631
## iter  14 value 0.949033
```

```
## iter  15 value 0.947835
## iter  16 value 0.947461
## iter  17 value 0.947436
## iter  18 value 0.947433
## iter  19 value 0.947426
## iter  20 value 0.947402
## iter  21 value 0.947339
## iter  22 value 0.947160
## iter  23 value 0.946786
## iter  24 value 0.946700
## iter  25 value 0.946116
## iter  26 value 0.946028
## iter  27 value 0.946024
## iter  28 value 0.946020
## iter  29 value 0.946001
## iter  30 value 0.945960
## iter  31 value 0.945851
## iter  32 value 0.945605
## iter  33 value 0.945156
## iter  34 value 0.944650
## iter  35 value 0.944643
## iter  36 value 0.944450
## iter  37 value 0.944438
## iter  38 value 0.944437
## iter  39 value 0.944437
## iter  40 value 0.944437
## iter  41 value 0.944435
## iter  42 value 0.944430
## iter  43 value 0.944421
## iter  44 value 0.944400
## iter  45 value 0.944369
## iter  46 value 0.944368
## iter  47 value 0.944355
## iter  48 value 0.944353
## iter  49 value 0.944350
## iter  50 value 0.944350
## iter  50 value 0.944350
## iter  50 value 0.944350
## final  value 0.944350
## converged
## initial  value 1.014428
## iter   2 value 0.996512
## iter   3 value 0.991479
## iter   4 value 0.987061
## iter   5 value 0.985804
## iter   6 value 0.977017
## iter   7 value 0.971866
## iter   8 value 0.961924
## iter   9 value 0.953235
## iter  10 value 0.949671
## iter  11 value 0.946990
## iter  12 value 0.945854
## iter  13 value 0.945658
## iter  14 value 0.945512
## iter  15 value 0.945456
## iter  16 value 0.944360
## iter  17 value 0.943117
## iter  18 value 0.940807
```

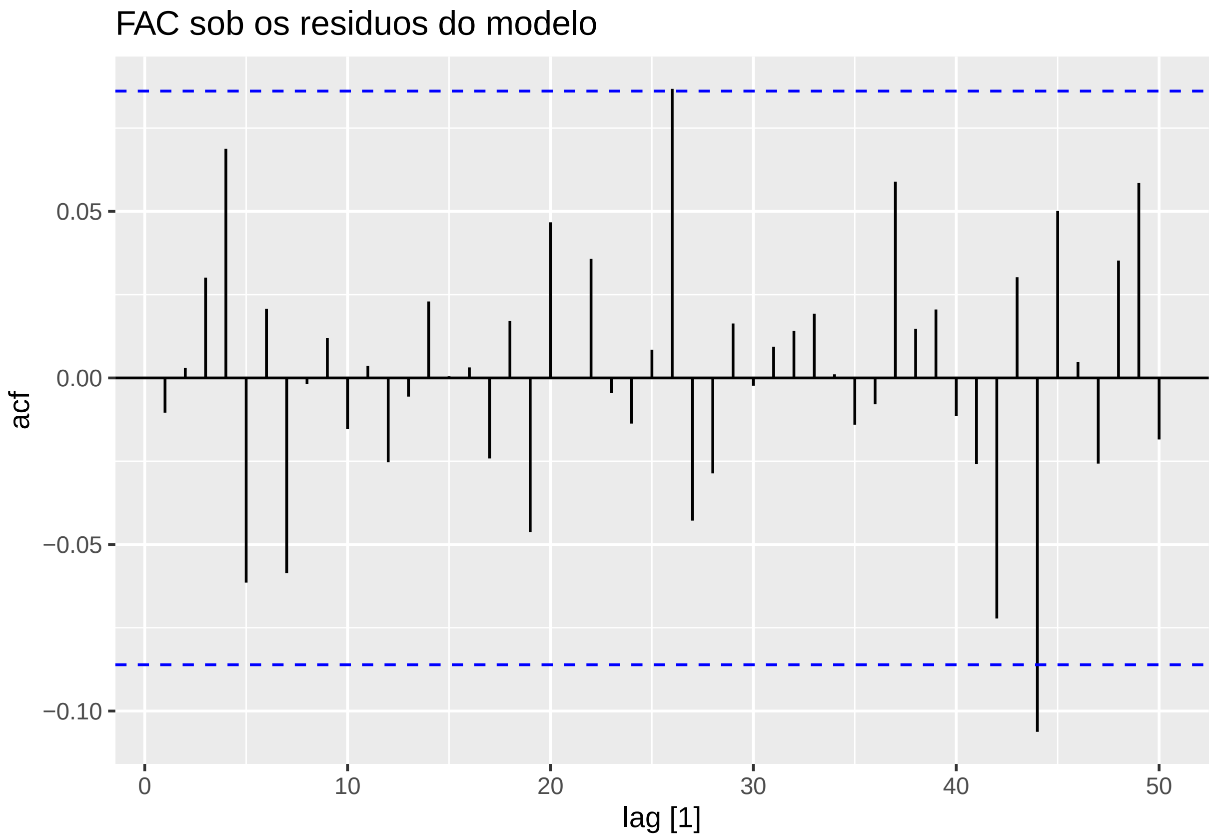
```
## iter  19 value 0.939635
## iter  20 value 0.938835
## iter  21 value 0.907689
## iter  22 value 0.907056
## iter  23 value 0.904938
## iter  24 value 0.904759
## iter  25 value 0.904672
## iter  26 value 0.904671
## iter  26 value 0.904671
## iter  26 value 0.904671
## final  value 0.904671
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate      SE       t.value p.value
## ma1     0.1951  0.0440        4.4385  0.0000
## ma2     0.1185  0.0400        2.9580  0.0032
## sar1    1.0000  0.0000 14240374.7594  0.0000
## sma1   -0.9471  0.0256      -37.0337  0.0000
## xmean -13.2005 51.1424       -0.2581  0.7964
##
## sigma^2 estimated as 5.652465 on 513 degrees of freedom
##
## AIC = 4.670384  AICc = 4.670611  BIC = 4.719612
##
```



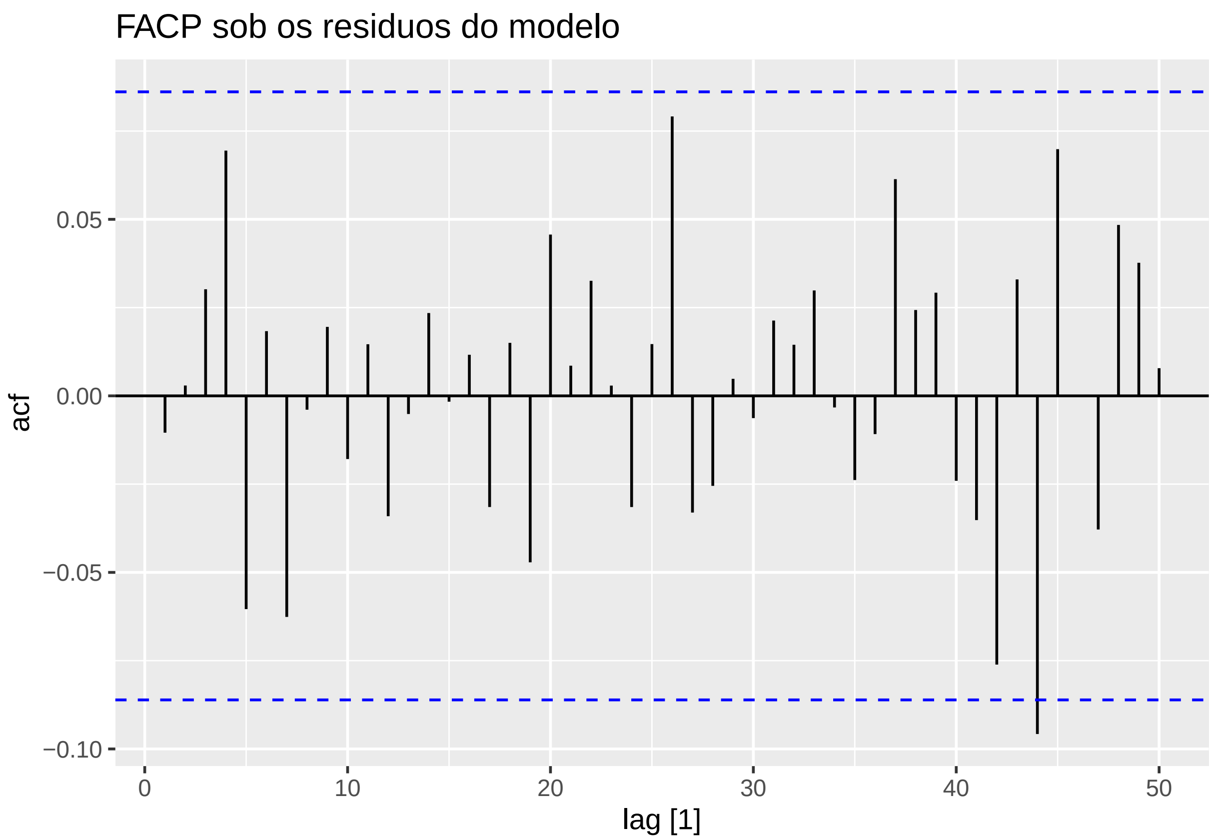

```
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 50) %>%
```

```r
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```r
res %>%
  ACF(value,lag_max = 50,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```




```
Box.test(fit[["fit"]][["residuals"]], lag = 50, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 35.117, df = 50, p-value = 0.9452
```

```
shapiro.test(fit[["fit"]][["residuals"]])
```

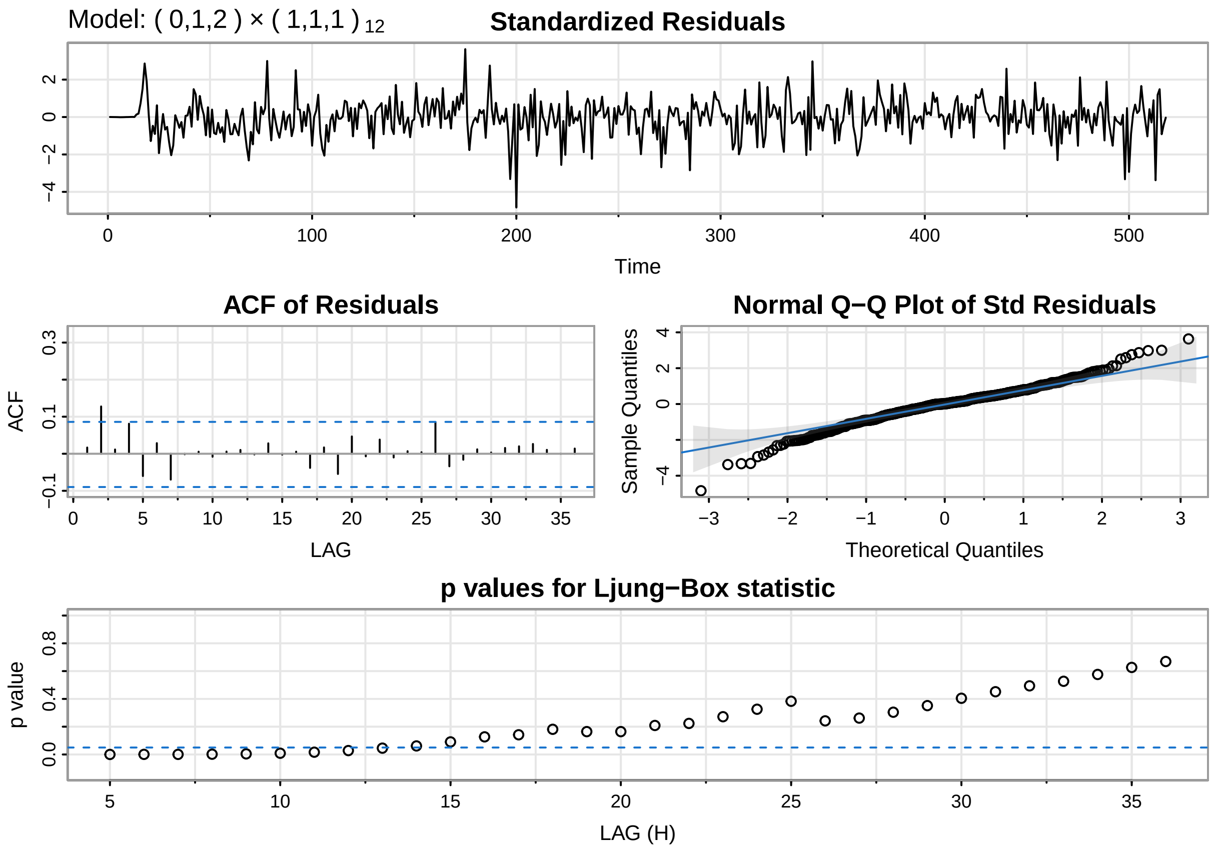
```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.97665, p-value = 2.339e-07
```

Diferenciando 1 vez a série

```
fit = astsa::sarima(y,p = 0, d = 1, q = 2, P = 1, D = 1, Q = 1, S = 12)
```

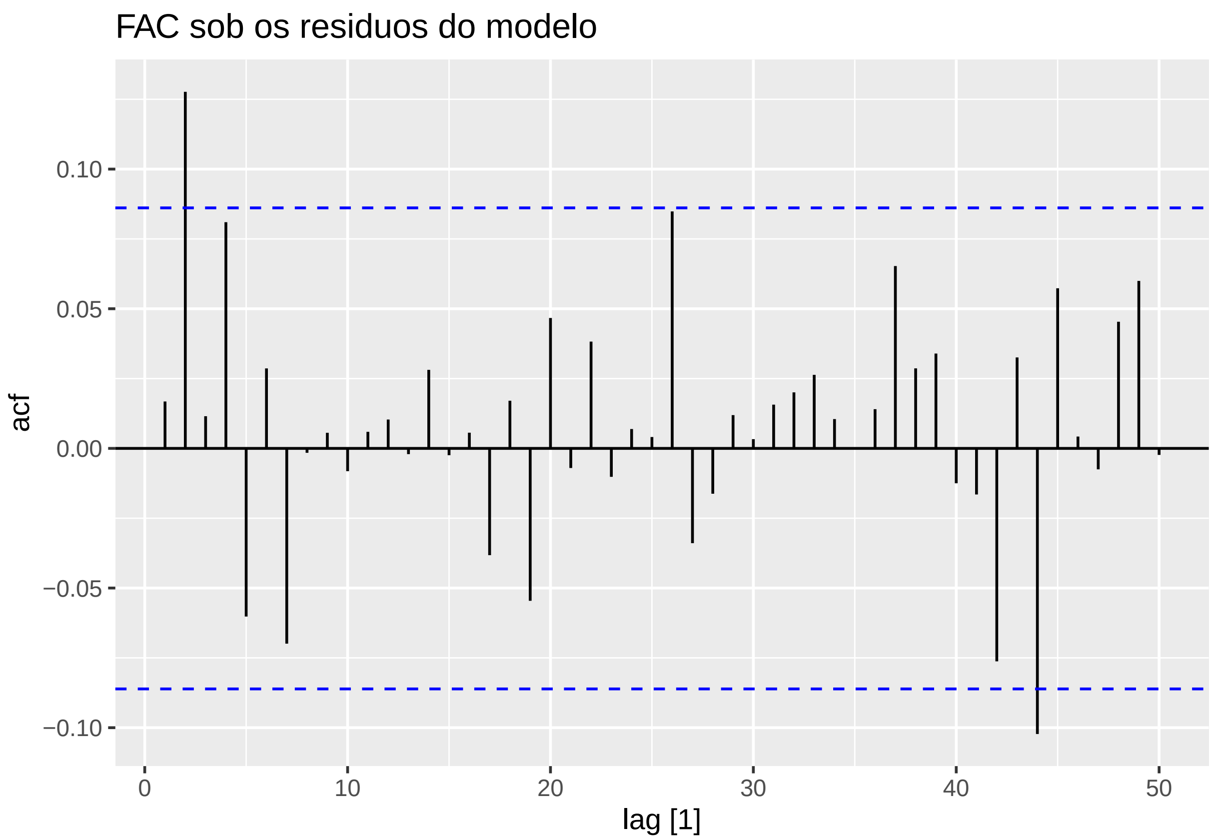
```
## initial  value 1.455741
## iter   2 value 1.098315
## iter   3 value 1.063737
## iter   4 value 0.962210
## iter   5 value 0.959977
## iter   6 value 0.934565
## iter   7 value 0.926750
```

```
## iter   8 value 0.924607
## iter   9 value 0.923563
## iter  10 value 0.923211
## iter  11 value 0.923201
## iter  12 value 0.923196
## iter  13 value 0.923185
## iter  14 value 0.923185
## iter  15 value 0.923185
## iter  15 value 0.923185
## iter  15 value 0.923185
## final  value 0.923185
## converged
## initial  value 0.929202
## iter   2 value 0.923151
## iter   3 value 0.918878
## iter   4 value 0.916654
## iter   5 value 0.916089
## iter   6 value 0.915980
## iter   7 value 0.915963
## iter   8 value 0.915961
## iter   9 value 0.915961
## iter   9 value 0.915961
## iter   9 value 0.915961
## final  value 0.915961
## converged
## <><><><><><><><><><><><><><><>
##
## Coefficients:
##      Estimate     SE  t.value p.value
## ma1   -0.8332 0.0442 -18.8410  0.0000
## ma2   -0.1668 0.0393  -4.2488  0.0000
## sar1  -0.0209 0.0486  -0.4288  0.6683
## sma1  -0.9816 0.0629 -15.6113  0.0000
##
## sigma^2 estimated as 5.673546 on 501 degrees of freedom
##
## AIC = 4.6896  AICc = 4.689759  BIC = 4.731428
##
```

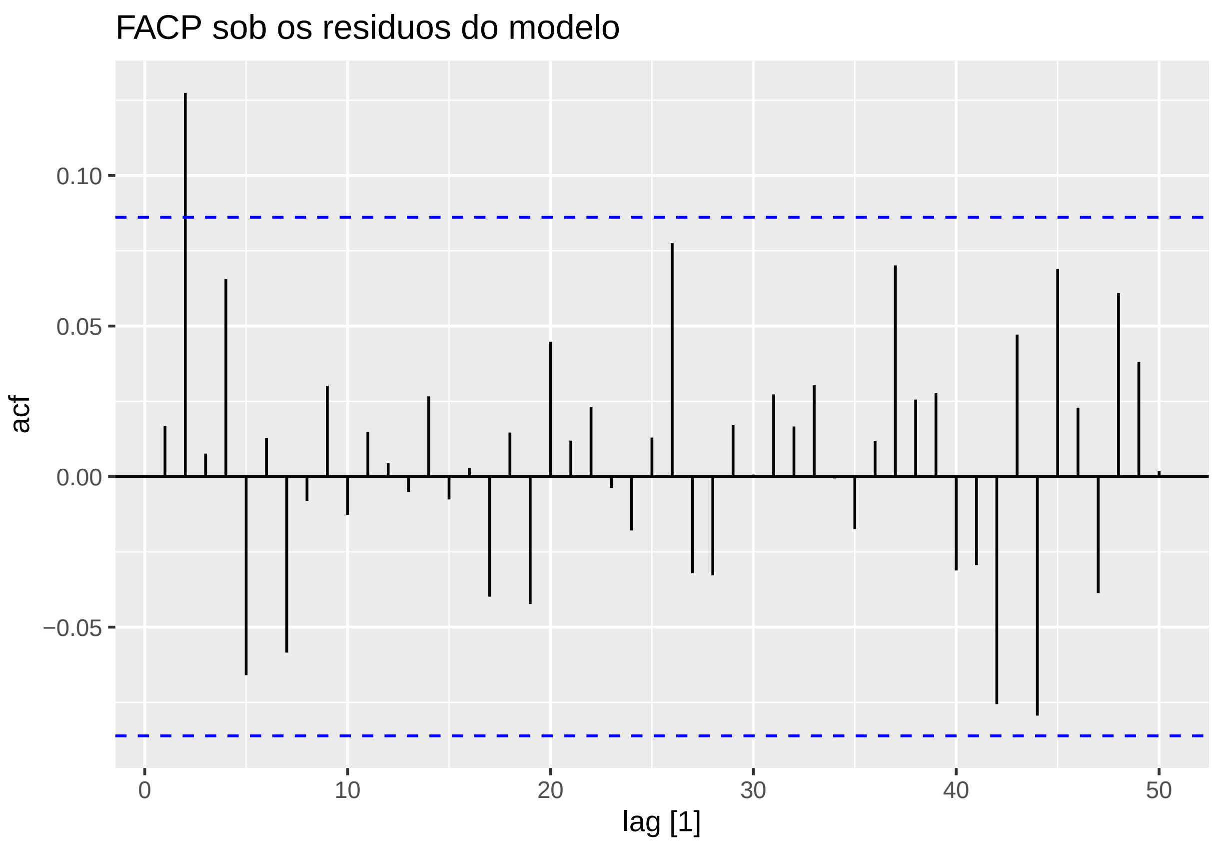



```
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 50) %>%
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```r
res %>%
  ACF(value,lag_max = 50,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```



```r
Box.test(fit[["fit"]][["residuals"]], lag = 50, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 46.739, df = 50, p-value = 0.605
```

```r
shapiro.test(fit[["fit"]][["residuals"]])
```

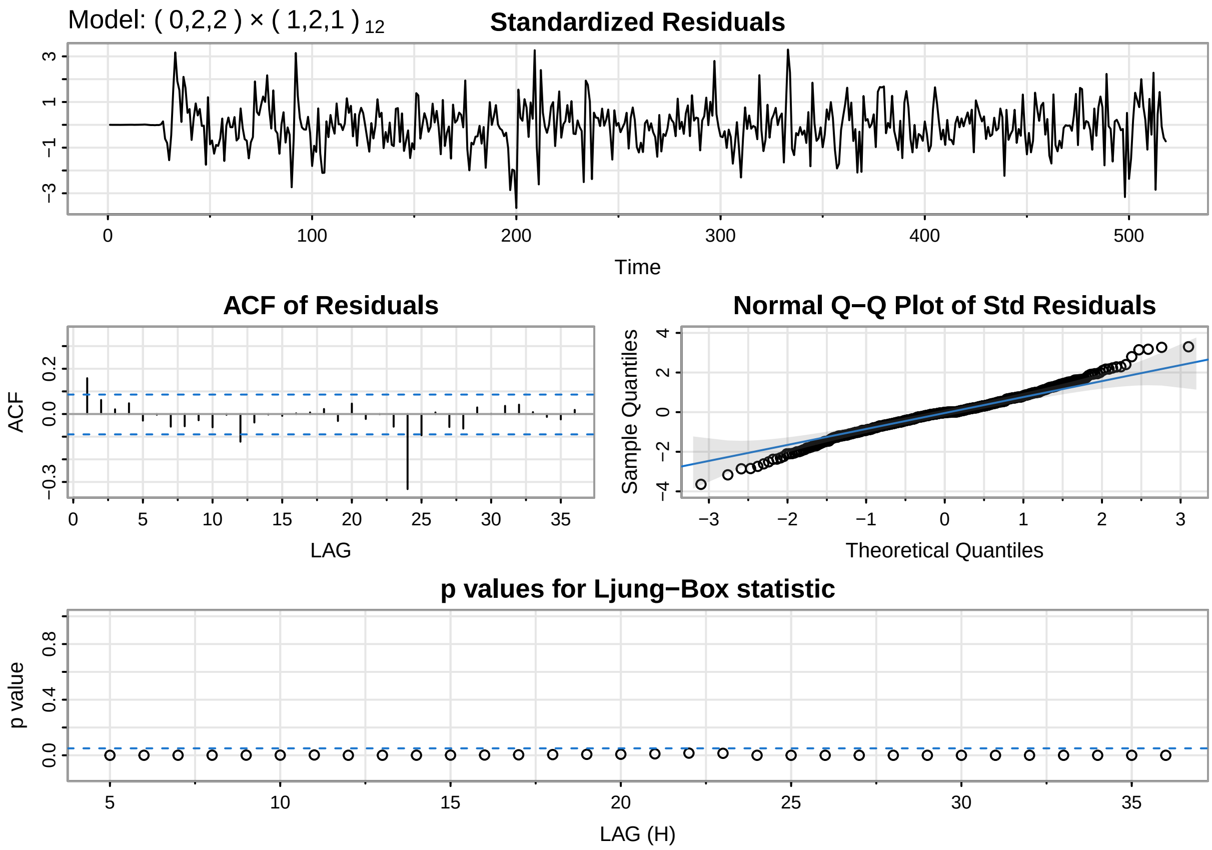
```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.97514, p-value = 1.049e-07
```

Diferenciando 2 vezes a série

```r
fit = astsa::sarima(y,p = 0, d = 2, q = 2, P = 1, D = 2, Q = 1, S = 12)
```

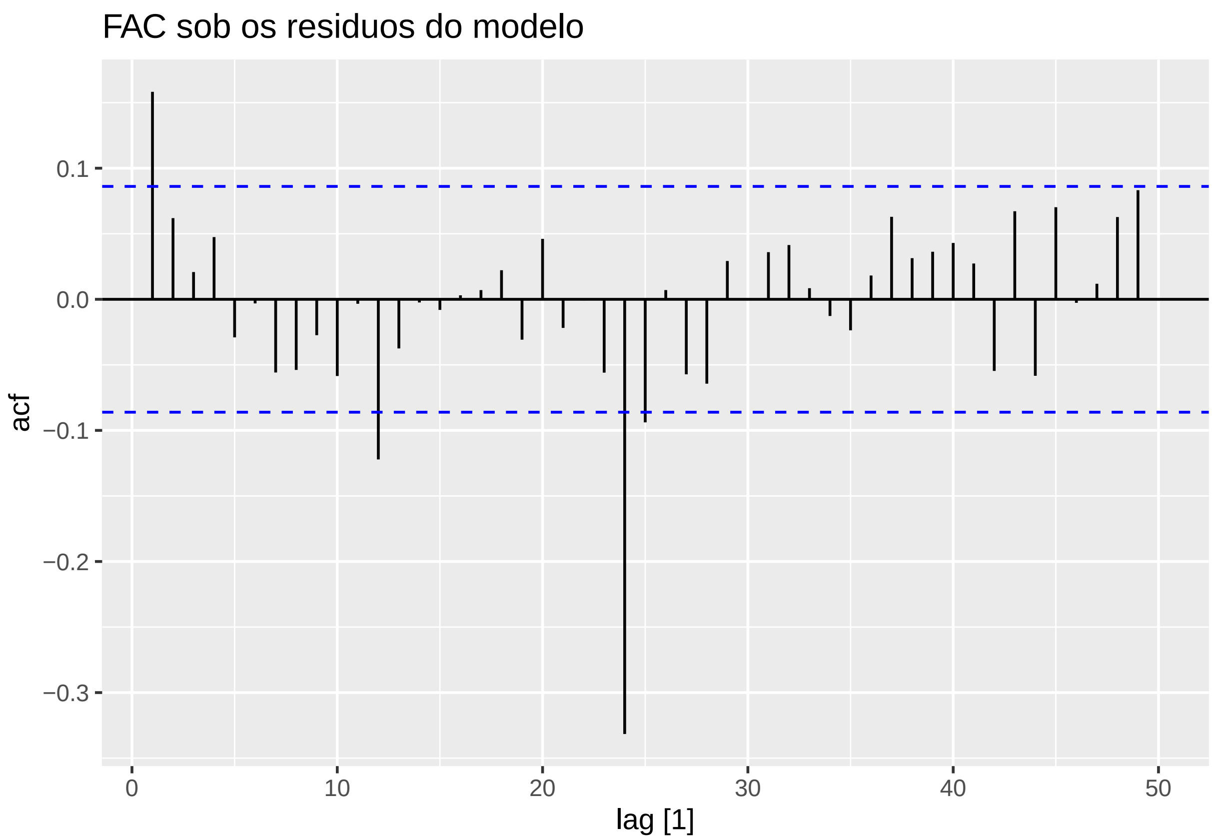
```
## initial  value 2.537551
## iter   2 value 1.666748
## iter   3 value 1.511014
## iter   4 value 1.433703
## iter   5 value 1.395875
## iter   6 value 1.381877
## iter   7 value 1.353994
## iter   8 value 1.302681
## iter   9 value 1.275193
## iter  10 value 1.264601
## iter  11 value 1.263881
## iter  12 value 1.257535
## iter  13 value 1.253714
## iter  14 value 1.252665
## iter  15 value 1.252653
## iter  16 value 1.252651
## iter  16 value 1.252651
## iter  16 value 1.252651
## final  value 1.252651
## converged
## initial  value 1.260935
## iter   2 value 1.240580
## iter   3 value 1.222744
## iter   4 value 1.200828
## iter   5 value 1.197361
## iter   6 value 1.194795
## iter   7 value 1.194367
## iter   8 value 1.191716
## iter   9 value 1.191539
## iter  10 value 1.186473
## iter  11 value 1.186271
## iter  12 value 1.183333
## iter  13 value 1.182268
## iter  14 value 1.180194
```

```
## iter  15 value 1.179887
## iter  16 value 1.179864
## iter  16 value 1.179864
## iter  17 value 1.179791
## iter  18 value 1.179776
## iter  18 value 1.179776
## iter  18 value 1.179776
## final  value 1.179776
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##      Estimate     SE  t.value p.value
## ma1   -1.9300 0.0589 -32.7426       0
## ma2    0.9300 0.0585  15.9061       0
## sar1  -0.5271 0.0398 -13.2452       0
## sma1  -1.0000 0.0197 -50.8754       0
##
## sigma^2 estimated as 8.962925 on 488 degrees of freedom
##
## AIC = 5.217755  AICc = 5.217922  BIC = 5.260422
##
```

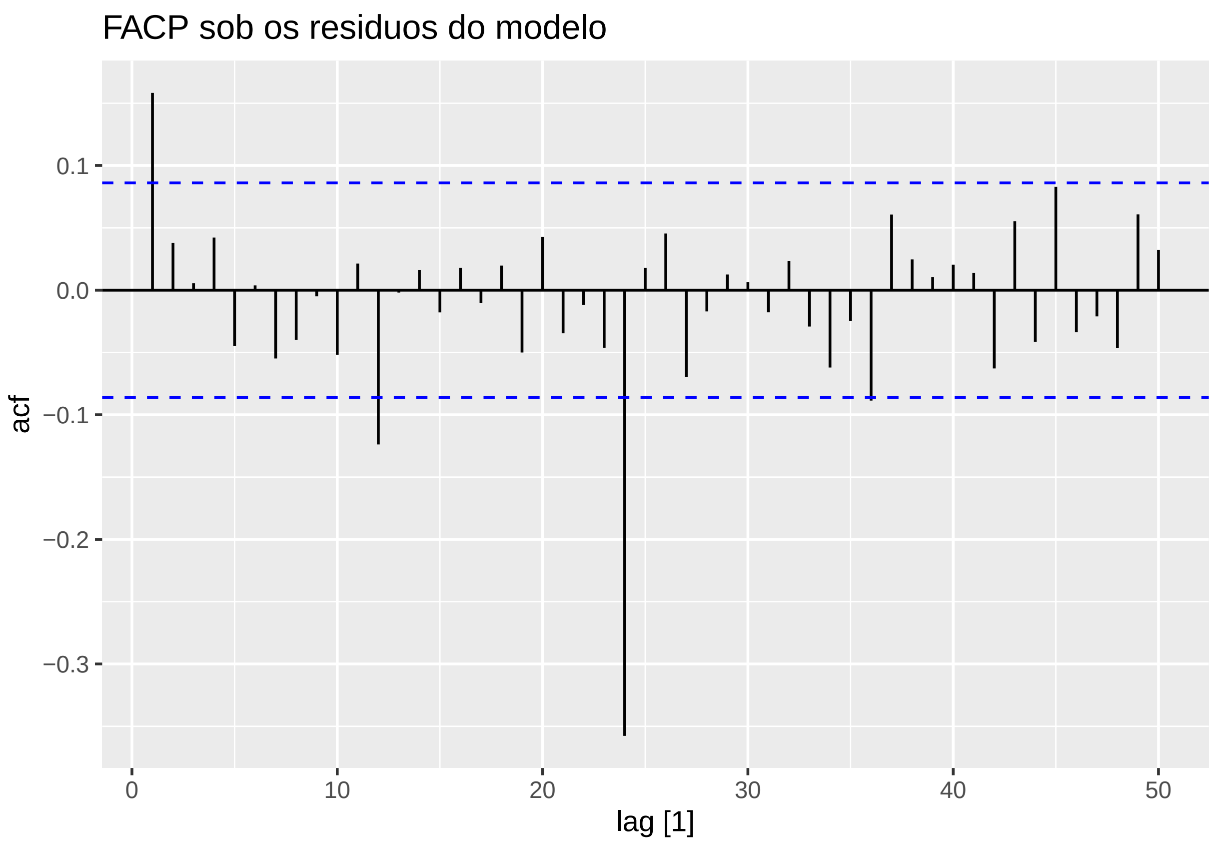



```r
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 50) %>%
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```r
res %>%
  ACF(value,lag_max = 50,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```



```r
Box.test(fit[["fit"]][["residuals"]], lag = 50, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 126.86, df = 50, p-value = 1.305e-08
```

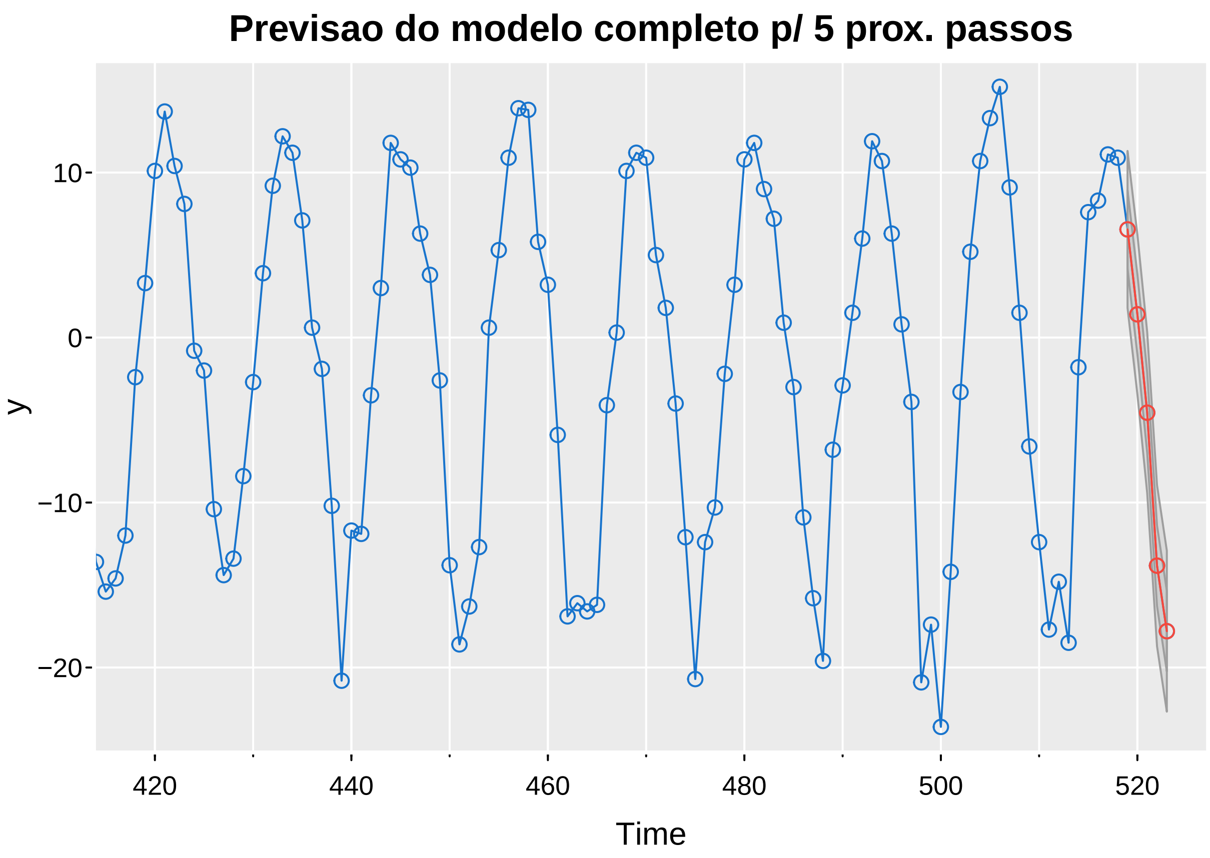
```r
shapiro.test(fit[["fit"]][["residuals"]])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.98363, p-value = 1.434e-05
```

Sob a hipótese nula de independência, o teste de Ljung-Box não rejeita a hipótese nula. Portanto, existem indícios de independência na série dos resíduos.

Avaliando as formas da FAC, FACP e resultado do teste de Ljung-Box, aparentam haver ainda algumas correlações positivas, porém ao diferenciar a série uma ou duas vezes, essas correlações aumentaram. Logo, este modelo aparenta ser o com melhor ajuste quanto a estrutura de dados, sendo um modelo possivelmente problemático, porém ainda útil, para previsões. Além disso, o teste de Shapiro-Wilk rejeita a normalidade dos resíduos, o que é outro sinal de alerta quando formos observar as bandas de confiança

```r
prev = sarima.for(y,p = 0, d = 0, q = 2, P = 1, D = 0, Q = 1, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Previsao do modelo completo p/ 5 prox. passos")
```

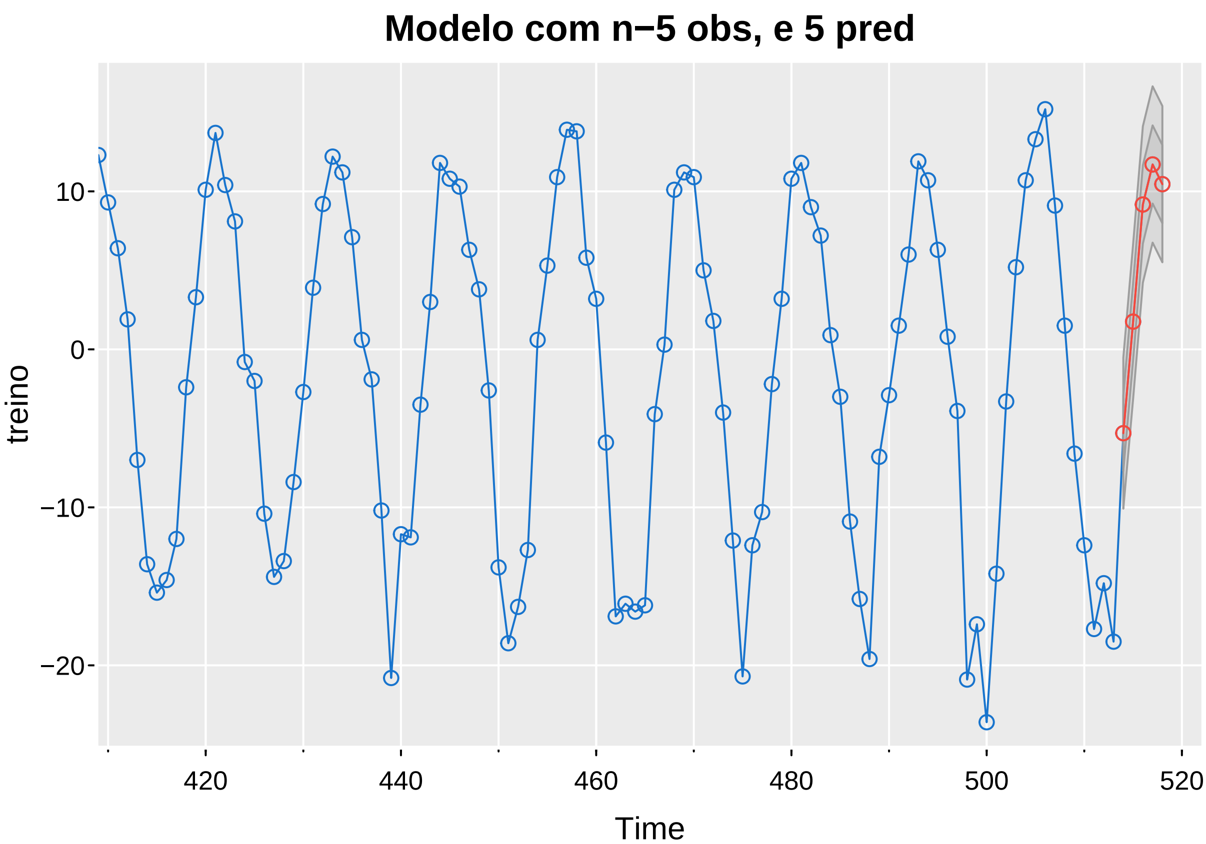


```r
prev
```

```
## $pred
## Time Series:
## Start = 519
## End = 523
## Frequency = 1
## [1]   6.555272   1.407973  -4.554210 -13.826321 -17.797139
##
## $se
## Time Series:
## Start = 519
## End = 523
## Frequency = 1
## [1] 2.378646 2.423492 2.439817 2.439817 2.439817
```

```r
treino = ts(y[1:513])
teste = ts(y[514:518])
```

```r
prev = sarima.for(treino,p = 0, d = 0, q = 2, P = 1, D = 0, Q = 1, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Modelo com n-5 obs, e 5 pred")
```




```r
prev$pred
```

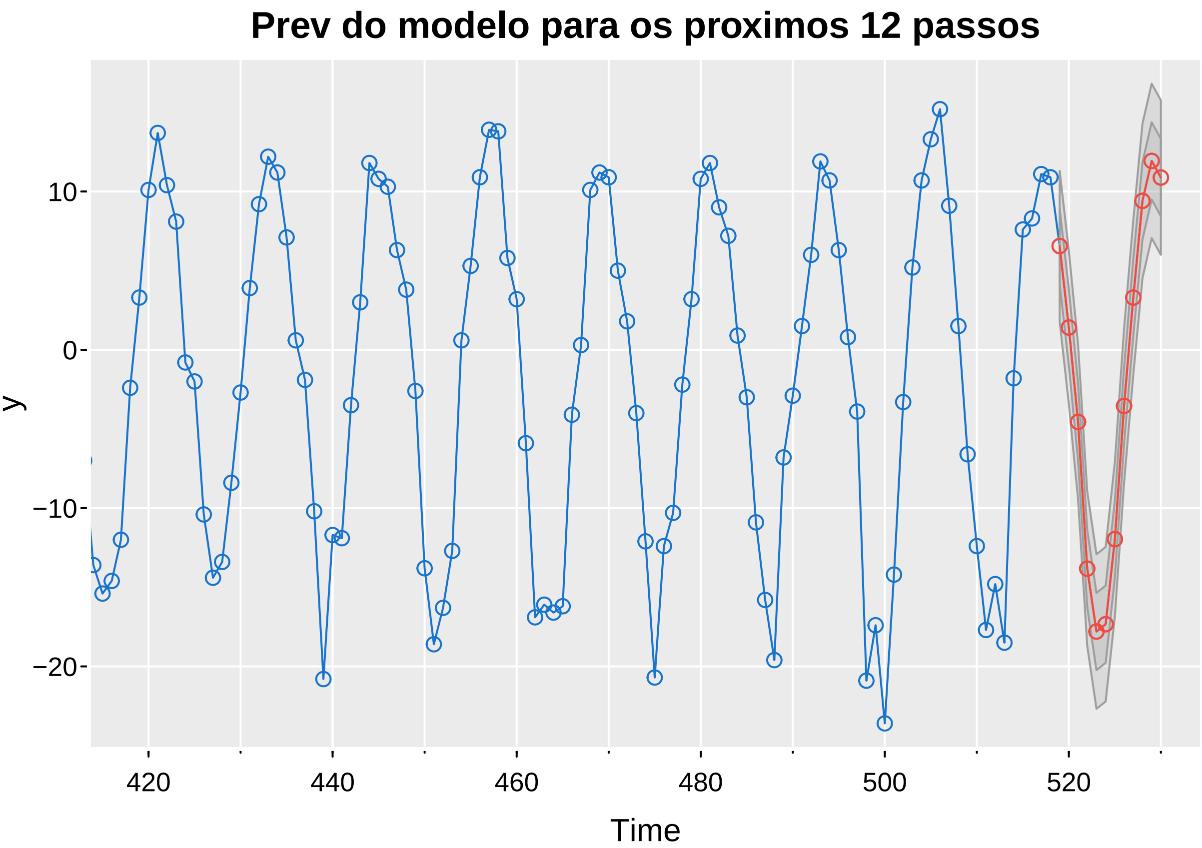
```
## Time Series:
## Start = 514
## End = 518
## Frequency = 1
## [1] -5.305152  1.754691  9.168654 11.702967 10.455767
```

```
MAPE = MAPE(prev$pred, as.vector(teste))
MAPE
```

```
## [1] 0.5832321
```

Com um MAPE = 0.5832321, ou seja, estando acima de 50%, podemos dizer que se trata de um modelo muito ruim para modelagem preditiva, além da não normalidade dos resíduos

```
prev = sarima.for(y,p = 0, d = 0, q = 2, P = 1, D = 0, Q = 1, S = 12,
                  n.ahead = 12, gg=TRUE, col=4,main="Prev do modelo para os proximos 12 passos")
```



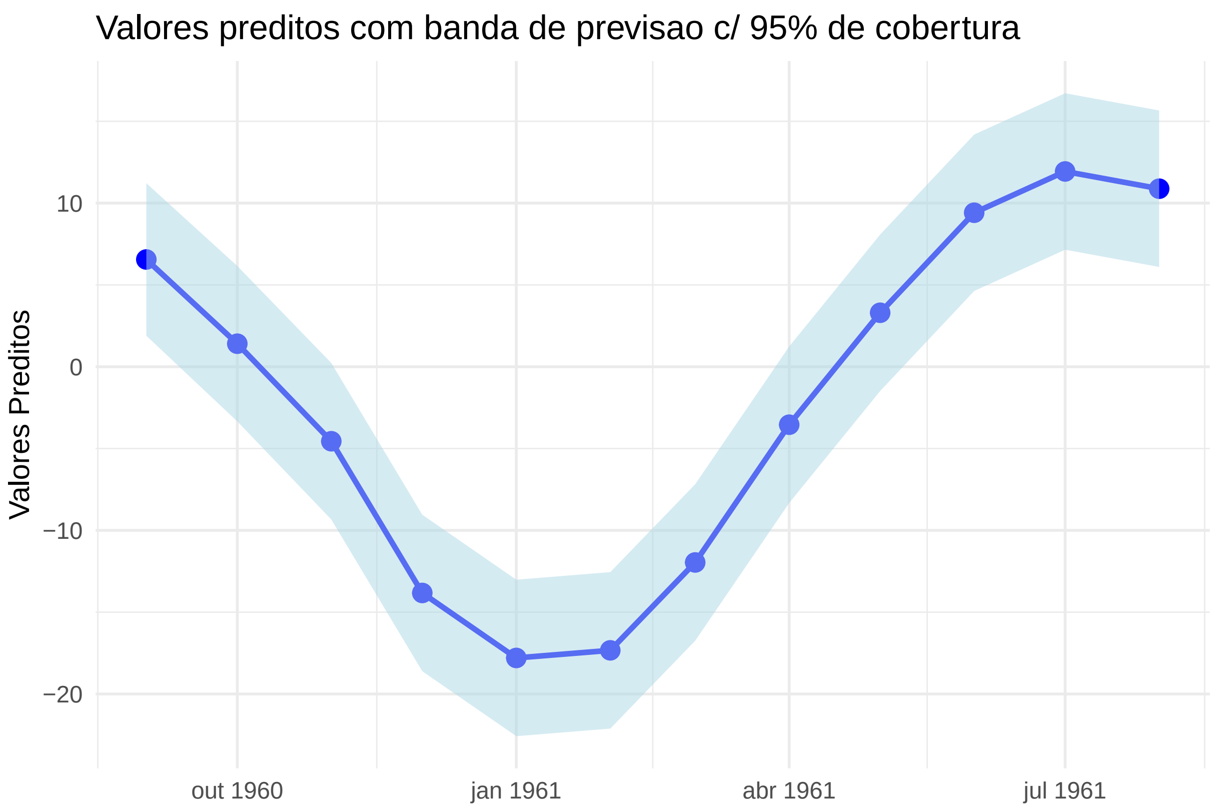

```
datas <- seq.Date(from = as.Date("1960-09-01"), to = as.Date("1961-08-31"), by = "months")

previsoes = data.frame(datas,prev$pred,prev$se)
colnames(previsoes) = c("data","Valores_preditos","Erro_padrao")

previsoes = previsoes %>%
  mutate(Limite_superior = Valores_preditos + (1.96 * Erro_padrao),
         Limite_inferior = Valores_preditos - (1.96 * Erro_padrao))

ggplot(previsoes, aes(x = data)) +
  geom_line(aes(y = Valores_preditos), color = "blue", size = 1) +
  geom_point(aes(y = Valores_preditos), color = "blue", size = 3) +
  geom_ribbon(aes(ymin = Limite_inferior, ymax = Limite_superior), fill = "lightblue", alpha = 0.5)
  labs(title = "Valores preditos com banda de previsao c/ 95% de cobertura",
       x = "",
       y = "Valores Preditos") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



Os diagnósticos do modelo não foram cumpridos, e este é um modelo bem ruim para modelagem preditiva! Podemos dizer que é "melhor que nada", ainda assim, não deve ser levado muito a sério os resultados preditivos deste modelo parauma intervenção real.

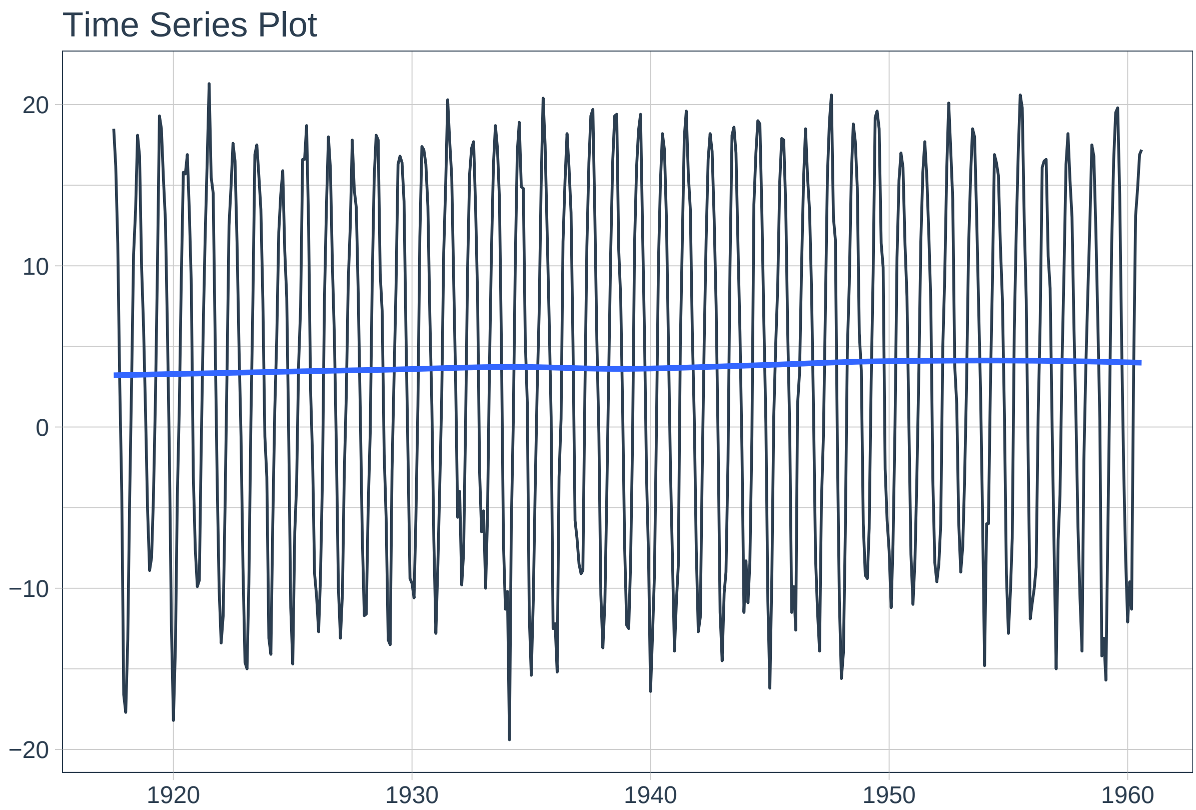
3. "Mean Temp (°C)";

```
df3 = df[,c(5,12)]
df3 = df3 %>% drop_na()
colnames(df3)[2] = "mean_temp"
```

```
df3$data = as.Date(df3$data)
df3 = tsibble::as_tsibble(df3,index=data,regular=F)
```

Visualizando a série com uma linha suavizada

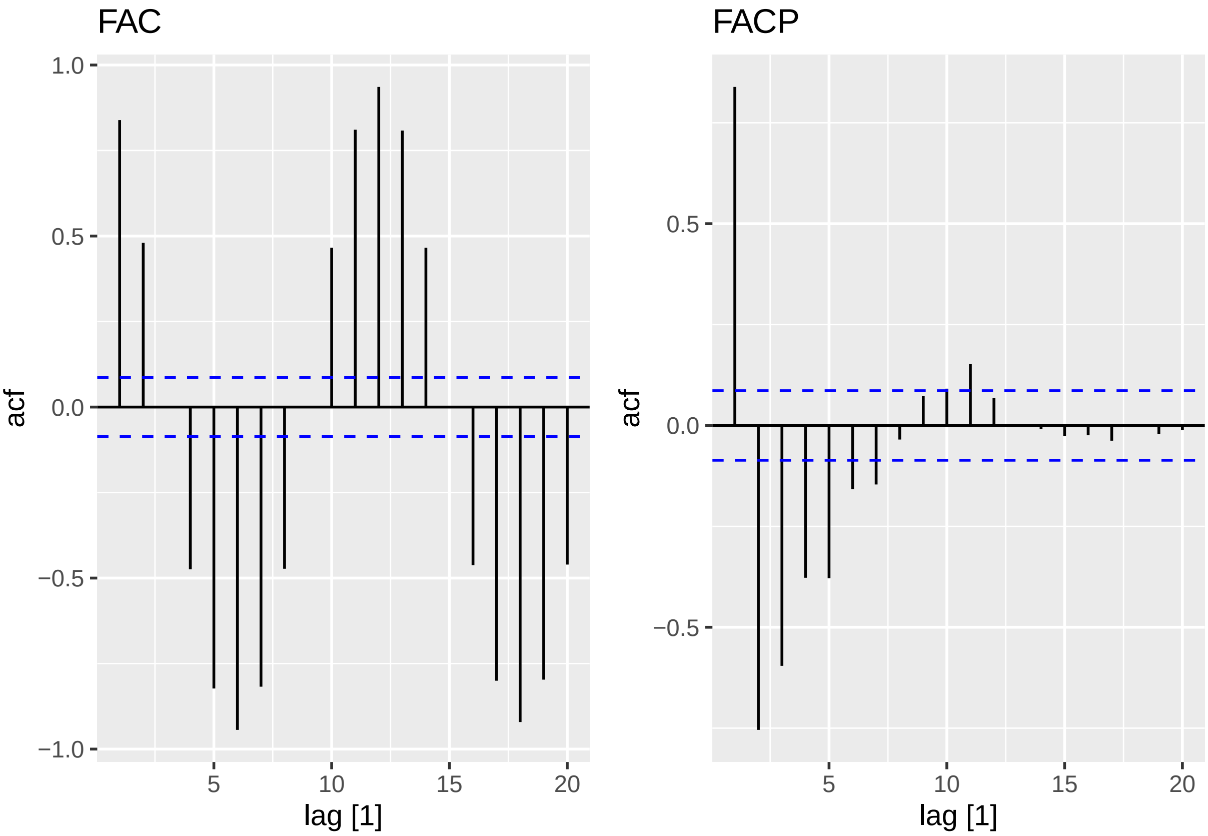
```
df3 %>%
  plot_time_series(data,mean_temp, .interactive = F, .smooth = T)
```




```
FAC3 = df3 %>%
  ACF(var = mean_temp,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC")

FACP3 = df3 %>%
  ACF(var = mean_temp,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP")

grid.arrange(FAC3, FACP3, nrow = 1)
```




```r
aTSA::adf.test(ts(df3$mean_temp), nlag = 5)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0  -6.3    0.01
## [2,]   1 -16.1    0.01
## [3,]   2 -23.0    0.01
## [4,]   3 -17.9    0.01
## [5,]   4 -13.5    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0  -6.67    0.01
## [2,]   1 -17.56    0.01
## [3,]   2 -28.10    0.01
## [4,]   3 -26.49    0.01
## [5,]   4 -26.58    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0  -6.68    0.01
## [2,]   1 -17.56    0.01
## [3,]   2 -28.11    0.01
## [4,]   3 -26.54    0.01
## [5,]   4 -26.72    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

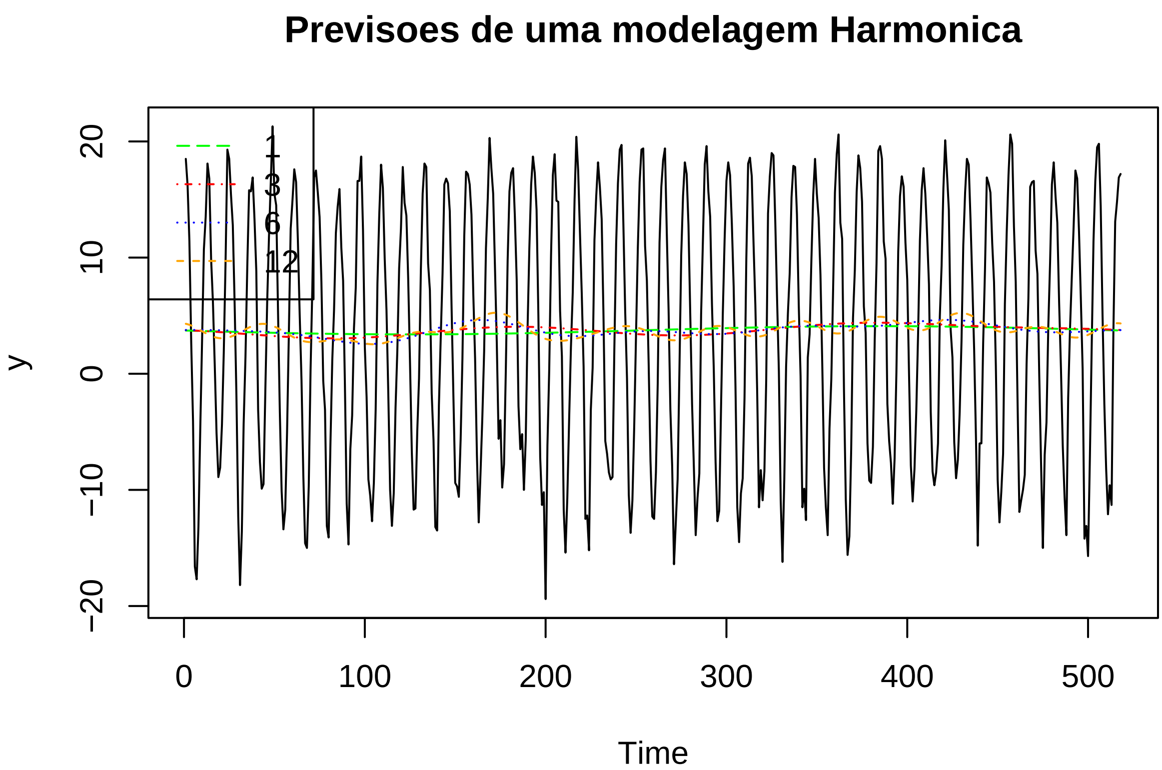
O teste de Dickey-Fuller sugere a estacionariedade da série

```r
y <- ts(df3$mean_temp)

fit1 <- haRmonics(y = y,
                  numFreq = 1,
                  delta = 0.1)
fit3 <- haRmonics(y = y,
                   numFreq = 3,
                   delta = 0.1)
fit6 <- haRmonics(y = y,
                    numFreq = 6,
                    delta = 0.1)
fit12 <- haRmonics(y = y,
                    numFreq = 12,
                    delta = 0.1)

x = ts(fit1$fitted)
w = ts(fit3$fitted)
z = ts(fit6$fitted)
v = ts(fit12$fitted)

plot(y, pch = 16, main = "Previsoes de uma modelagem Harmonica")
lines(x ,lty = 5, col = "green")
lines(w, lty = 4, col = "red")
lines(z, lty = 3, col = "blue")
lines(v, lty = 2, col = "orange")
legend("topleft", legend = c("1", "3", "6","12"),
       col = c("green", "red", "blue","orange"), lty = c(5, 4, 3,2))
```



A modelagem harmônica foi satisfatória em capturar todas as estruturas da série, deixando apenas ruido branco.

Visto que não foi testado se a série necessita de diferenciação, farei a modelagem sarima com ordem de sazonalidade = 12. Iniciei com grid de parâmetros p,P,q,Q,d,D no intervalo [0,2]; e o melhor modelo selecionado não tinha parâmetros na fronteira.

```r
modelos <- data.frame(p = integer(),
                      P = integer(),
                      q = integer(),
                      Q = integer(),
                      d = integer(),
                      D = integer(),
                      AIC = numeric(),
                      BIC = numeric())

tic()
for(p in 0:2){
  for(P in 0:2){
      for(q in 0:2){
          for(Q in 0:2){
              for(d in 0:2){
                for(D in 0:2){
            tryCatch({
              fit = astsa::sarima(y, details = FALSE, Model = FALSE,
                          p = p, d = d, q = q, P = P, D = D, Q = Q, S = 12)
              AIC = fit$ICs[1]
              BIC = fit$ICs[3]
              mod = c(p, P, q, Q,d,D, AIC, BIC)
              modelos = rbind(modelos, mod)
            }, error = function(e) {
            })
        }}}}}}
toc()

colnames(modelos) = c("p","P","q","Q","d","D","AIC","BIC")

modelo = modelos %>%
  arrange(BIC, AIC) %>% head(1)
```

Logo, o melhor modelo (minimiza BIC) é o com os parâmetros:

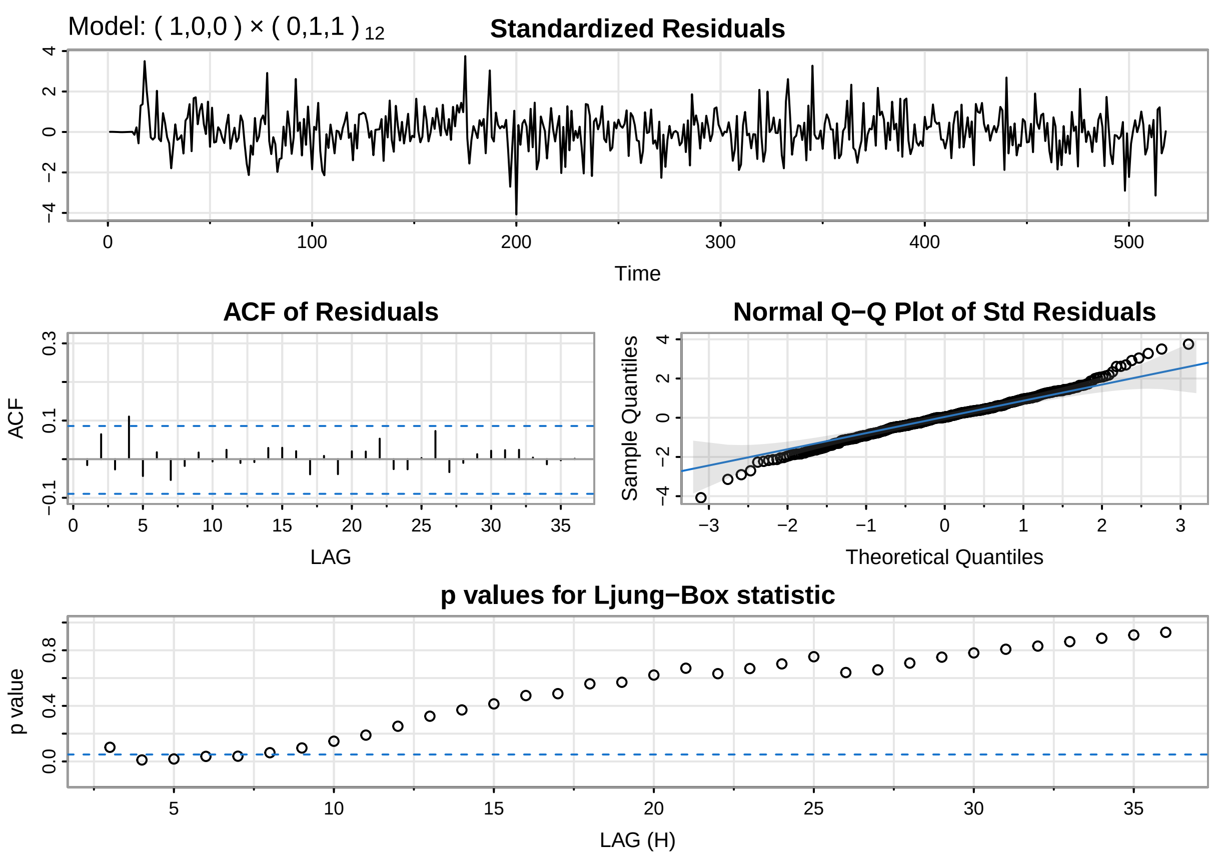
```r
kable(modelo)
```

p	P	q	Q	d	D	AIC	BIC
1	0	0	1	0	1	4.414454	4.447865

```r
fit = astsa::sarima(y,p = 1, d = 0, q = 0, P = 0, D = 1, Q = 1, S = 12)
```

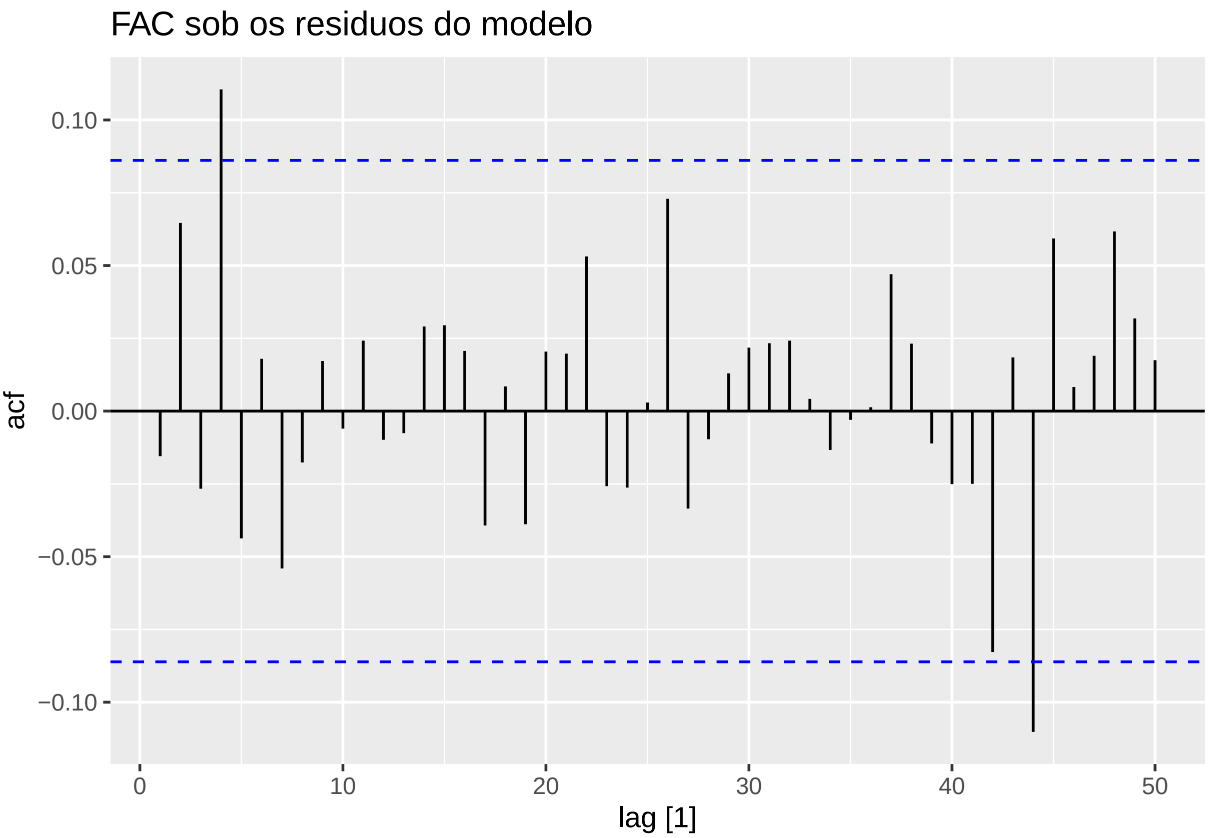
```
## initial  value 1.114982
## iter   2 value 0.903300
## iter   3 value 0.861161
## iter   4 value 0.838780
## iter   5 value 0.833551
## iter   6 value 0.832454
## iter   7 value 0.831450
## iter   8 value 0.831194
## iter   9 value 0.830992
## iter  10 value 0.830973
## iter  11 value 0.830963
## iter  12 value 0.830963
## iter  12 value 0.830963
```

```
## iter  12 value 0.830963
## final  value 0.830963
## converged
## initial  value 0.806390
## iter   2 value 0.796673
## iter   3 value 0.785879
## iter   4 value 0.783631
## iter   5 value 0.781031
## iter   6 value 0.780403
## iter   7 value 0.780386
## iter   8 value 0.780383
## iter   9 value 0.780383
## iter  10 value 0.780383
## iter  11 value 0.780383
## iter  11 value 0.780383
## iter  11 value 0.780383
## final  value 0.780383
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE  t.value p.value
## ar1        0.2341 0.0432   5.4239  0.0000
## sma1      -1.0000 0.0326 -30.7144  0.0000
## constant   0.0017 0.0008   2.1846  0.0294
##
## sigma^2 estimated as 4.355183 on 503 degrees of freedom
##
## AIC = 4.414454  AICc = 4.414549  BIC = 4.447865
##
```

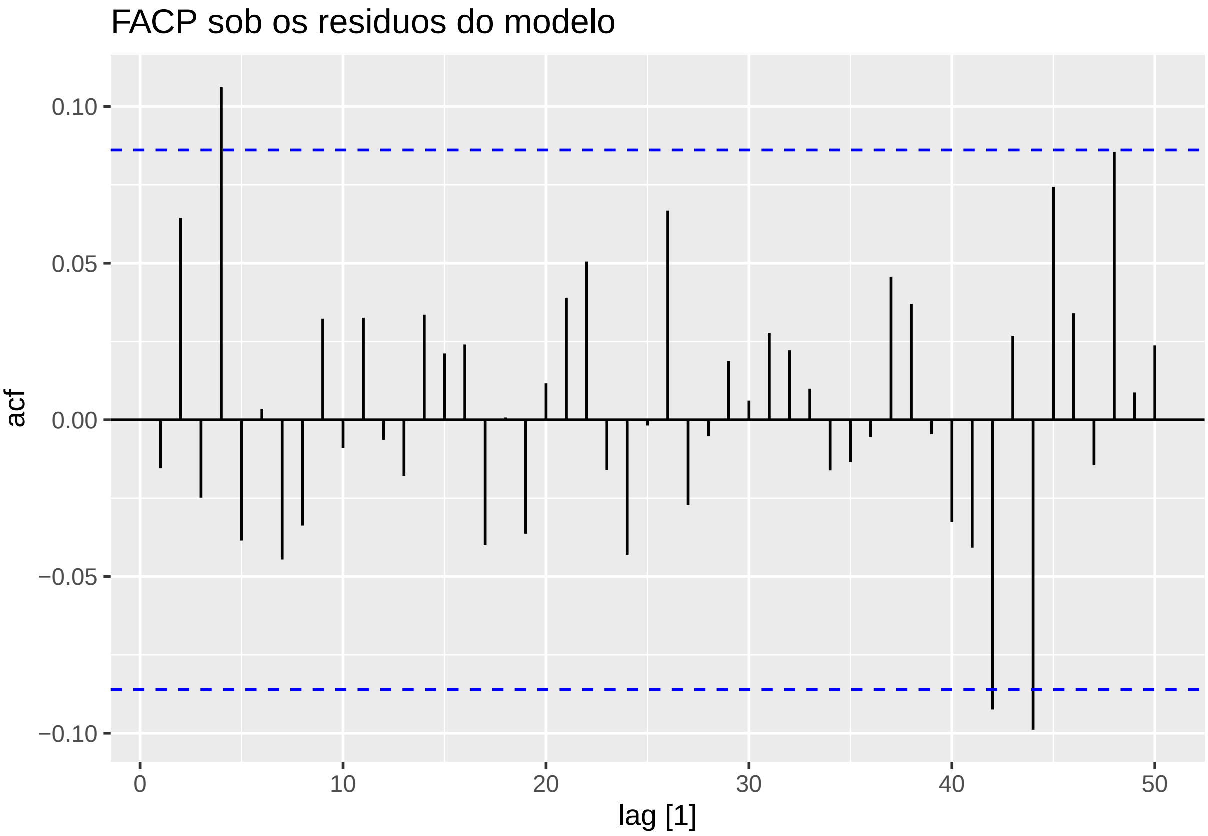


```r
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 50) %>%
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```r
res %>%
  ACF(value,lag_max = 50,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```




```
Box.test(fit[["fit"]][["residuals"]], lag = 50, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 41.215, df = 50, p-value = 0.8073
```

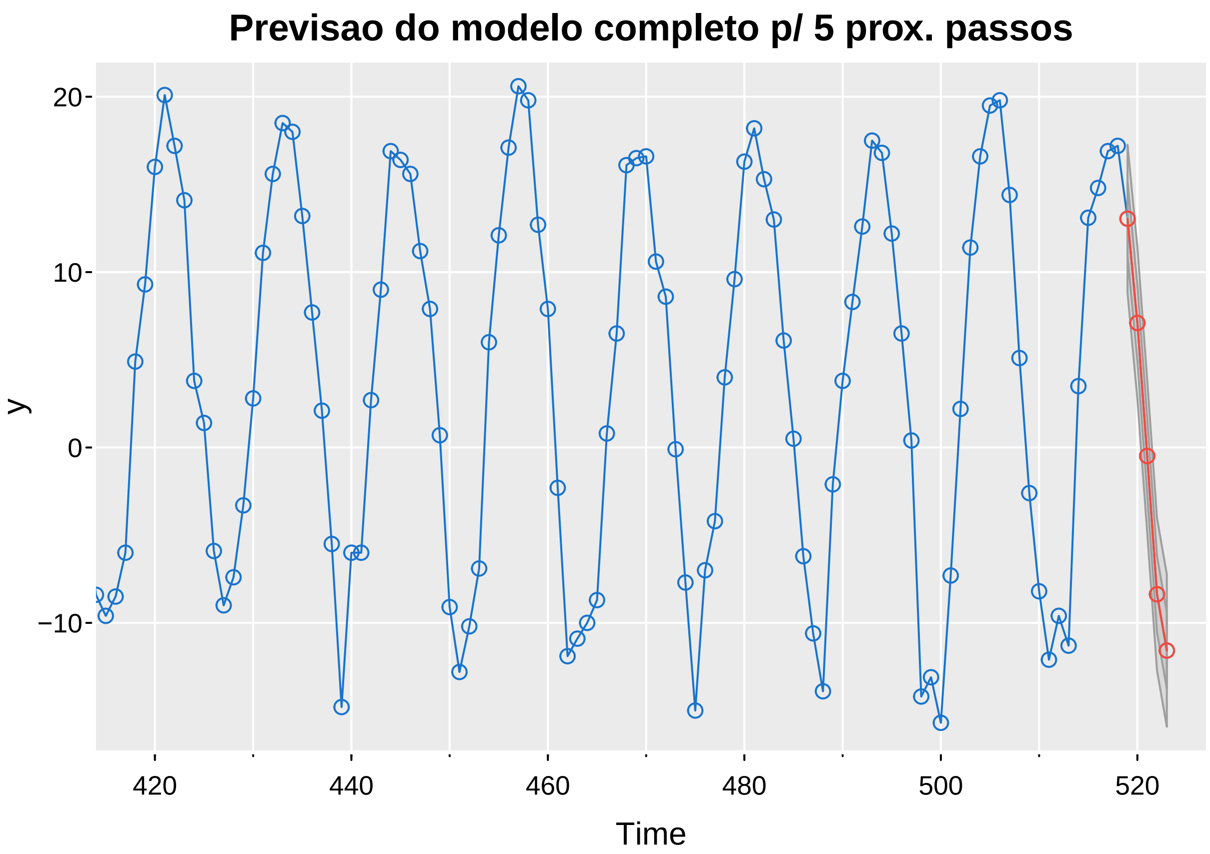
```
shapiro.test(fit[["fit"]][["residuals"]])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.98784, p-value = 0.0002618
```

Sob a hipótese nula de independência, o teste de Ljung-Box não rejeita a hipótese nula. Portanto, existem indícios de independência na série dos resíduos.

Avaliando as formas da FAC, FACP e resultado do teste de Ljung-Box, aparenta haver alguma correlação nos resíduos, além da rejeição da normalidade pelo teste de Shapiro-Wilk. Como já foram feitos testes de diferenciação anteriormente, este é o melhor modelo que pode se obter com este método.

```
prev = sarima.for(y,p = 1, d = 0, q = 0, P = 0, D = 1, Q = 1, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Previsao do modelo completo p/ 5 prox. passos")
```

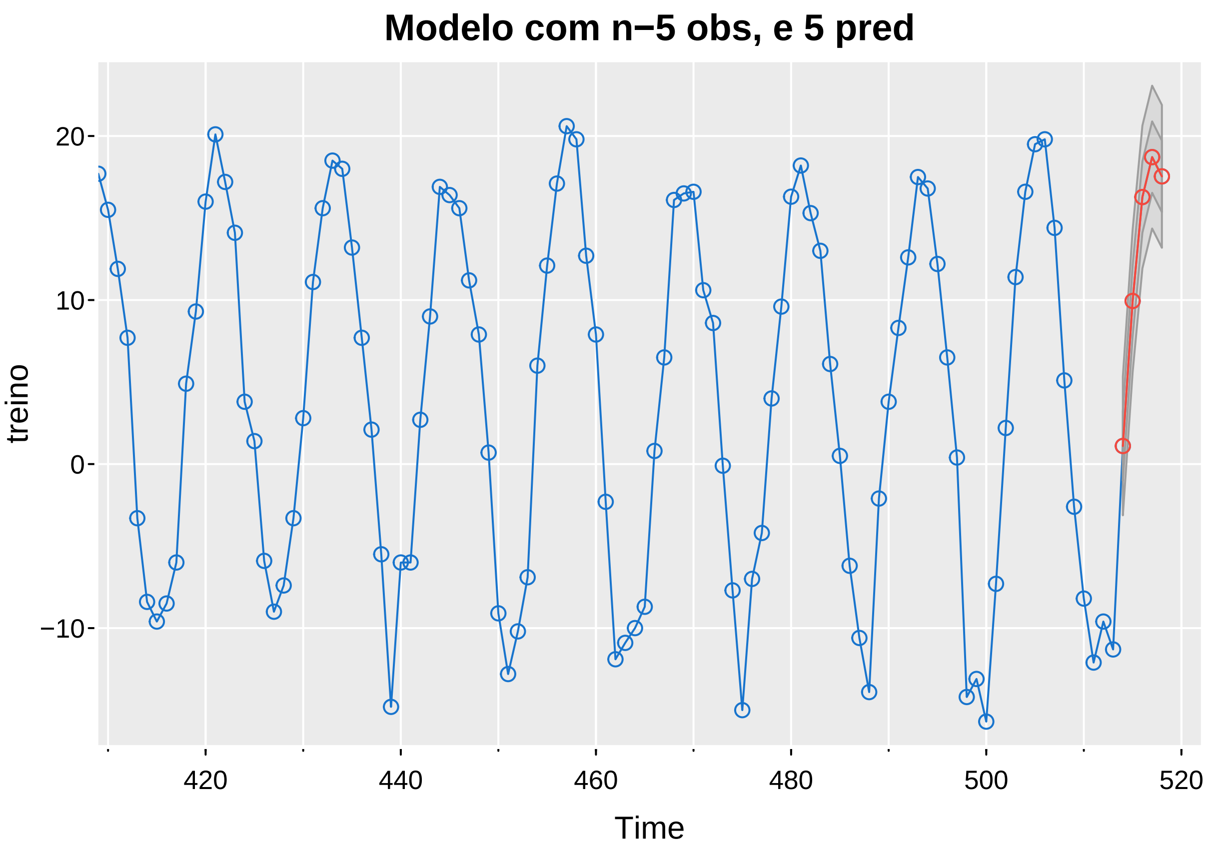



```r
prev
```

```
## $pred
## Time Series:
## Start = 519
## End = 523
## Frequency = 1
## [1]  13.0461183   7.1004247  -0.4823577  -8.3695683 -11.5803779
##
## $se
## Time Series:
## Start = 519
## End = 523
## Frequency = 1
## [1] 2.111034 2.168114 2.171199 2.171368 2.171377
```

```r
treino = ts(y[1:513])
teste = ts(y[514:518])
```

```r
prev = sarima.for(treino,p = 1, d = 0, q = 0, P = 0, D = 1, Q = 1, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Modelo com n-5 obs, e 5 pred")
```




```
prev$pred
```

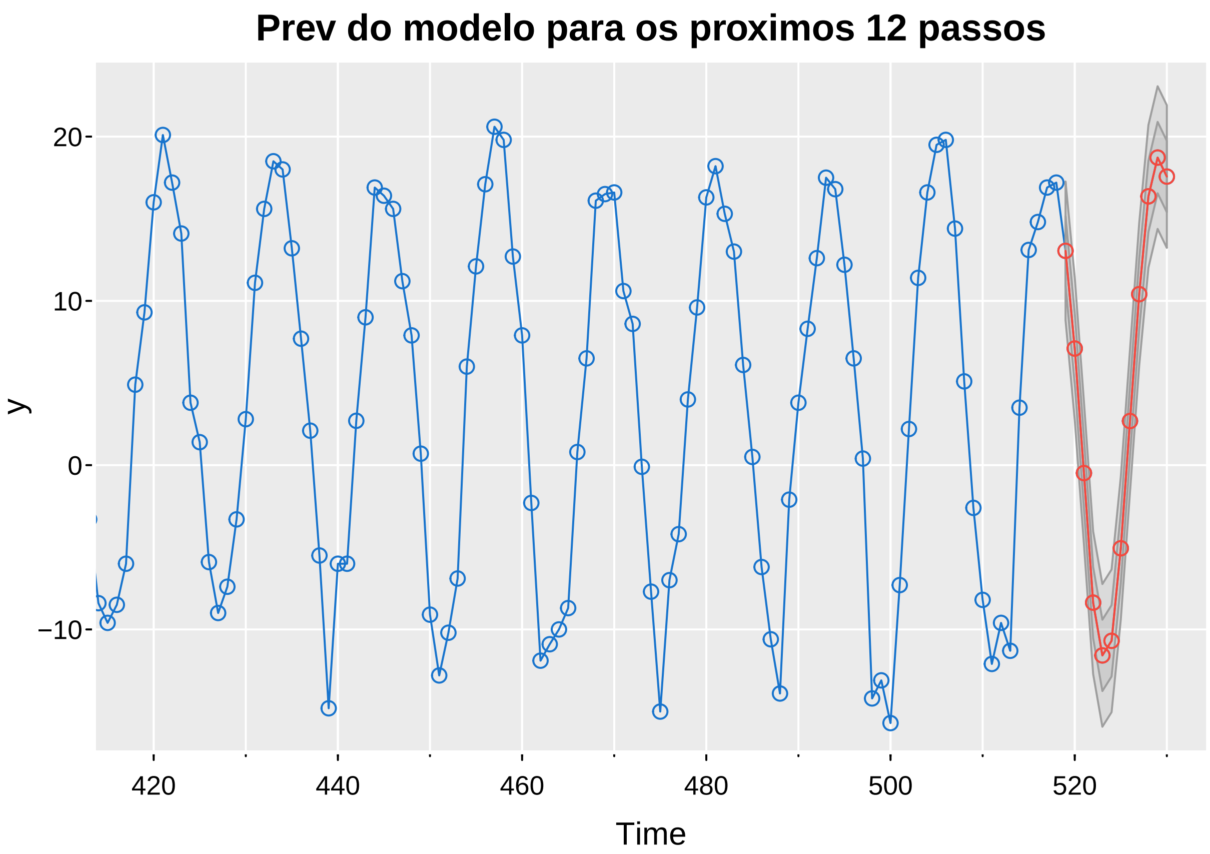
```
## Time Series:
## Start = 514
## End = 518
## Frequency = 1
## [1]  1.102669  9.943533 16.279603 18.713841 17.539487
```

```
MAPE = MAPE(prev$pred, as.vector(teste))
MAPE
```

```
## [1] 0.2305884
```

Com um MAPE = 0.2305884, ou seja,entre 20 e 30%, podemos dizer que se trata de um bom modelo preditivo.

```
prev = sarima.for(y,p = 1, d = 0, q = 0, P = 0, D = 1, Q = 1, S = 12,
           n.ahead = 12, gg=TRUE, col=4,main="Prev do modelo para os proximos 12 passos")
```



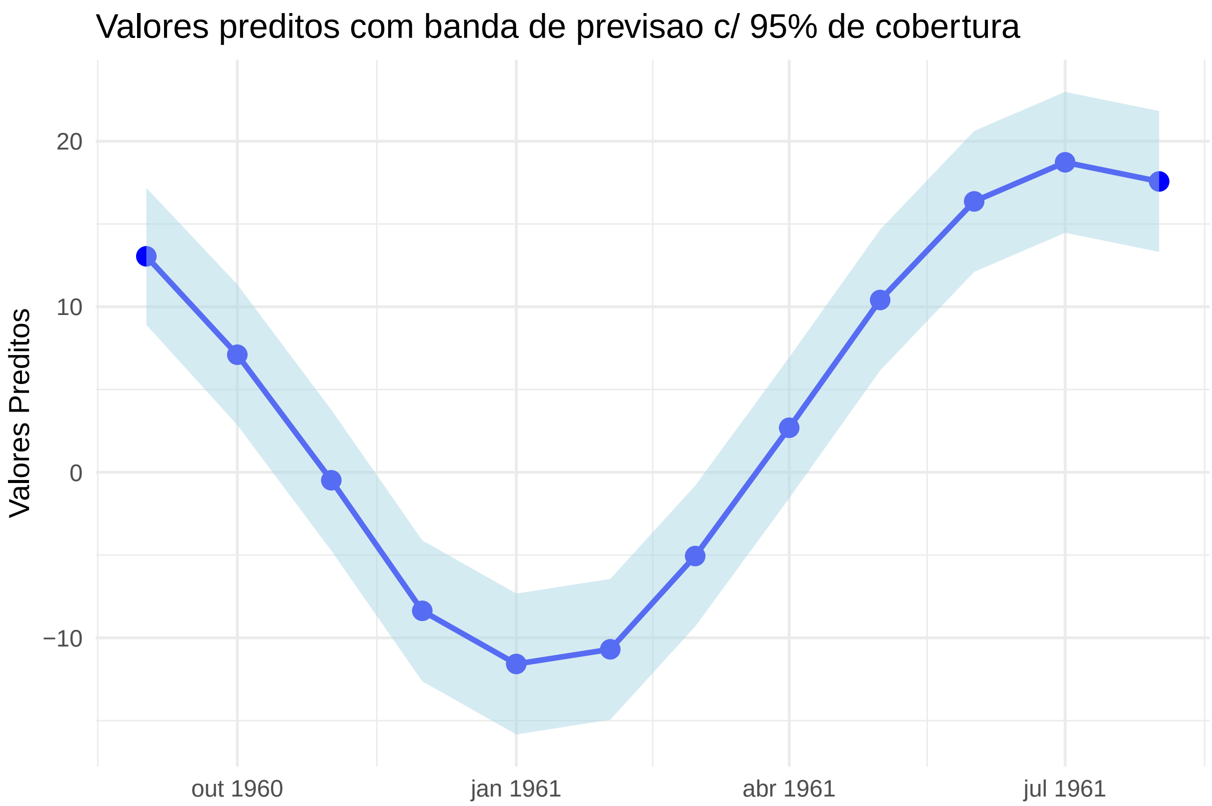

```r
datas <- seq.Date(from = as.Date("1960-09-01"), to = as.Date("1961-08-31"), by = "months")

previsoes = data.frame(datas,prev$pred,prev$se)
colnames(previsoes) = c("data","Valores_preditos","Erro_padrao")

previsoes = previsoes %>%
  mutate(Limite_superior = Valores_preditos + (1.96 * Erro_padrao),
         Limite_inferior = Valores_preditos - (1.96 * Erro_padrao))

ggplot(previsoes, aes(x = data)) +
  geom_line(aes(y = Valores_preditos), color = "blue", size = 1) +
  geom_point(aes(y = Valores_preditos), color = "blue", size = 3) +
  geom_ribbon(aes(ymin = Limite_inferior, ymax = Limite_superior), fill = "lightblue", alpha = 0.5)
  labs(title = "Valores preditos com banda de previsao c/ 95% de cobertura",
       x = "",
       y = "Valores Preditos") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



Os diagnósticos do modelo foram todos atendidos. Portanto, as previsões deste modelo apresentam alguma credibilidade. Considerando a banda de confiança e a especificidade dos dados, temos de tomar cuidado pois os resíduos não são normais. Esta banda pode estar incorreta. Ainda assim, se trata de um modelo relevante para análises preditivas.

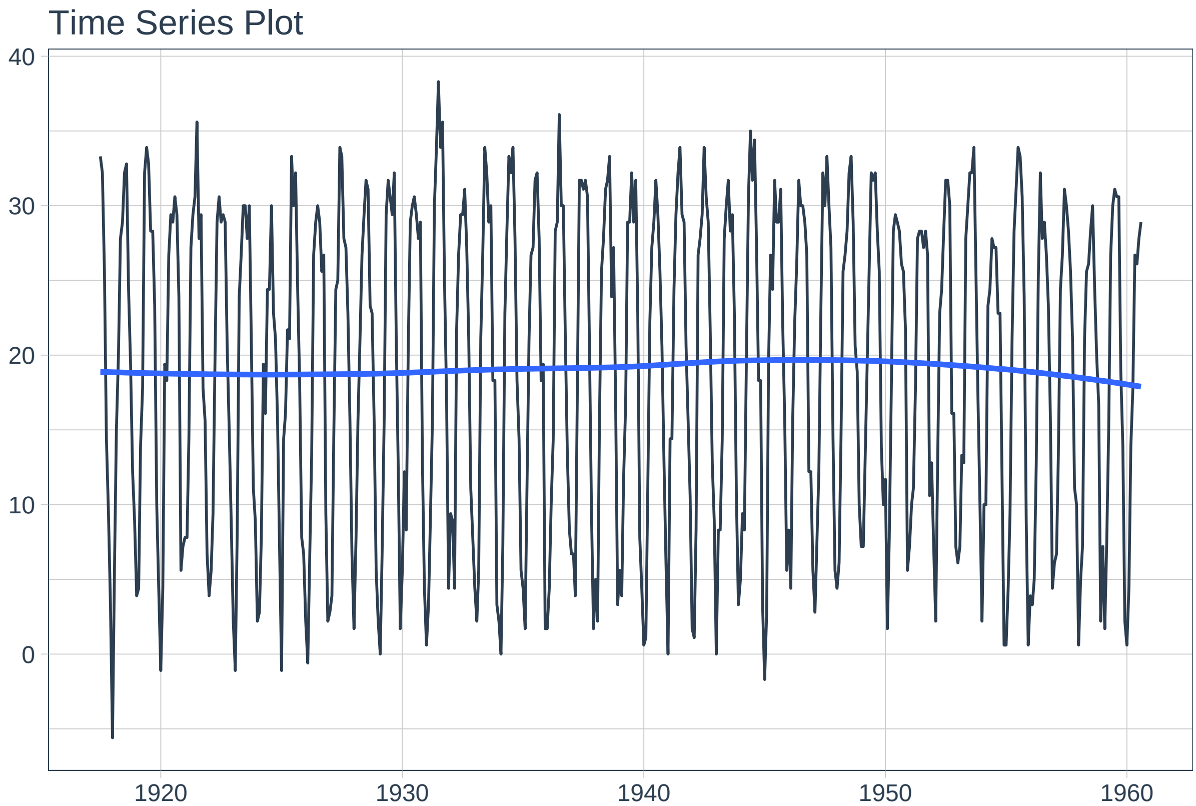
4. "Extr Max Temp (°C)";

```
df4 = df[,c(5,14)]
df4 = df4 %>% drop_na()
colnames(df4)[2]="extr_max_temp"
```

```
df4$data = as.Date(df4$data)
df4 = tsibble::as_tsibble(df4,index=data,regular=F)
```

Visualizando a série com uma linha suavizada

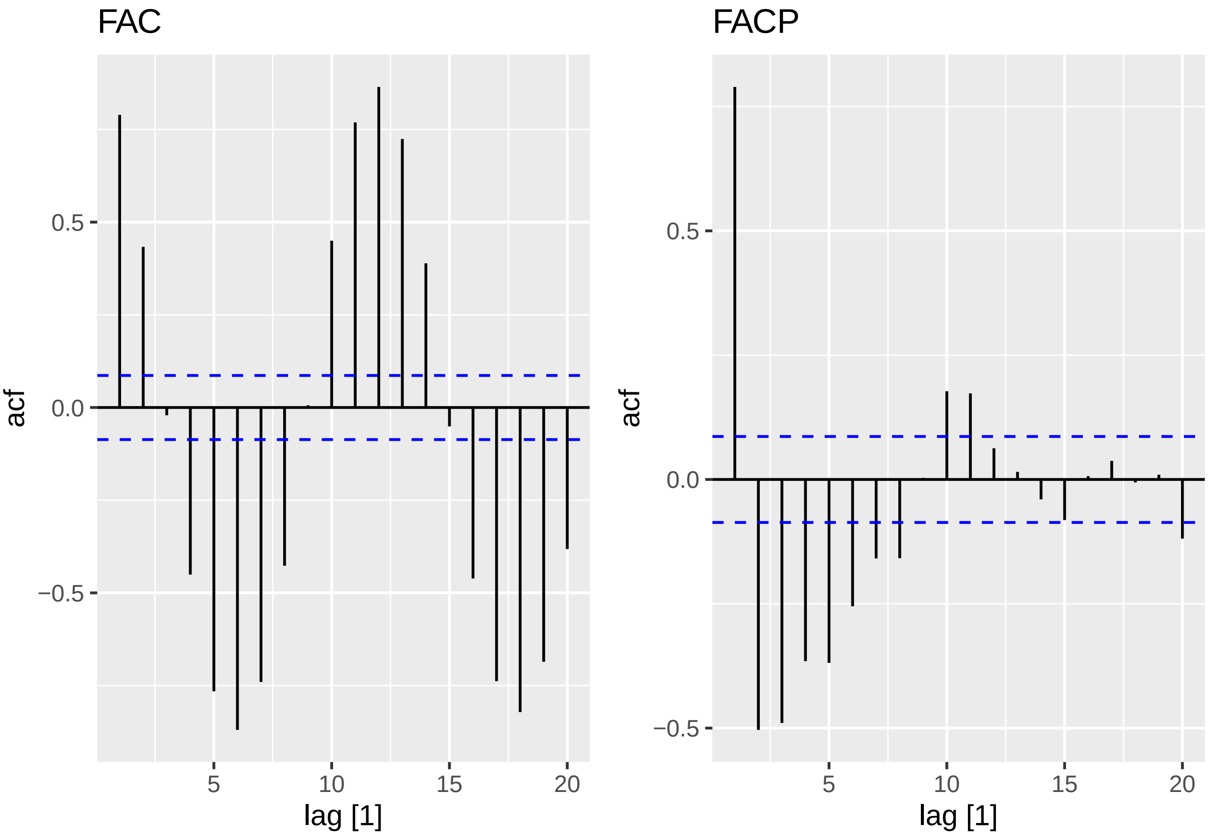
```
df4 %>%
  plot_time_series(data,extr_max_temp, .interactive = F, .smooth = T)
```




```r
FAC4 = df4 %>%
  ACF(var = extr_max_temp,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC")

FACP4 = df4 %>%
  ACF(var = extr_max_temp,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP")

grid.arrange(FAC4, FACP4, nrow = 1)
```




```
aTSA::adf.test(ts(df4$extr_max_temp), nlag = 5)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -3.69    0.01
## [2,]   1 -5.43    0.01
## [3,]   2 -6.38    0.01
## [4,]   3 -5.27    0.01
## [5,]   4 -4.01    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0  -7.75    0.01
## [2,]   1 -12.78    0.01
## [3,]   2 -19.08    0.01
## [4,]   3 -21.46    0.01
## [5,]   4 -23.08    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0  -7.74    0.01
## [2,]   1 -12.77    0.01
## [3,]   2 -19.06    0.01
## [4,]   3 -21.44    0.01
## [5,]   4 -23.06    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

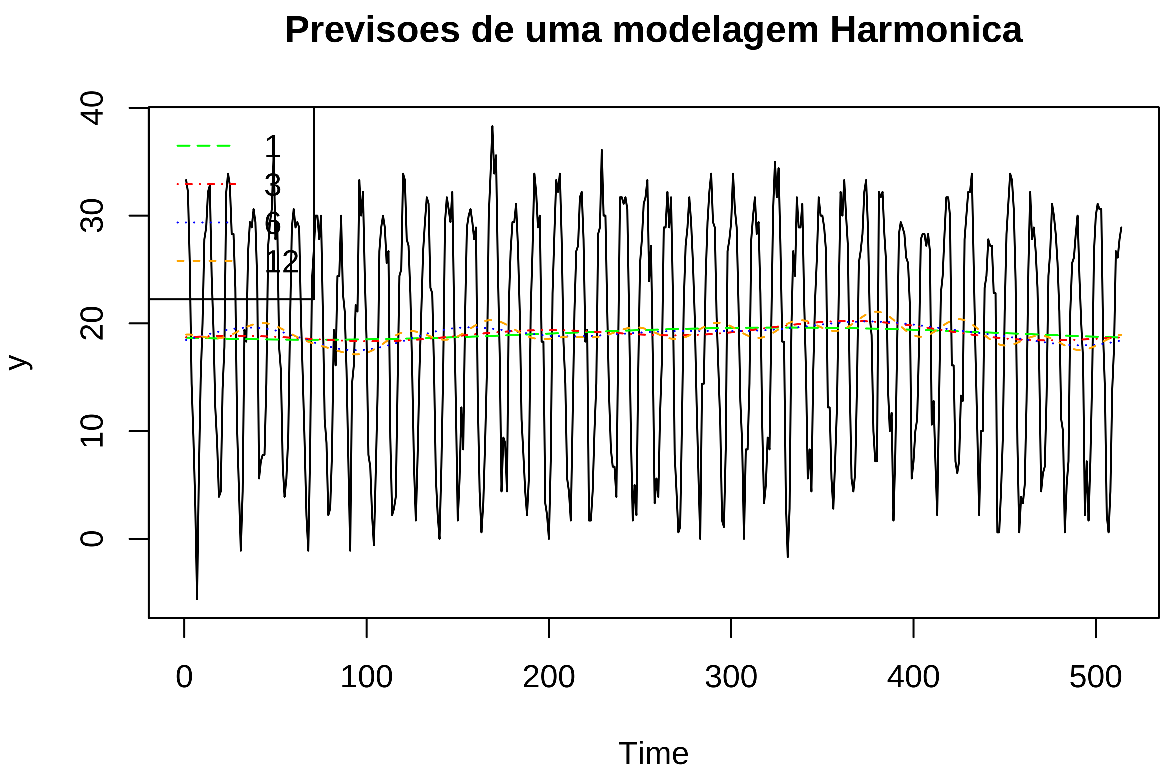
O teste de Dickey-Fuller sugere a estacionariedade da série

```r
y <- ts(df4$extr_max_temp)

fit1 <- haRmonics(y = y,
                  numFreq = 1,
                  delta = 0.1)
fit3 <- haRmonics(y = y,
                   numFreq = 3,
                   delta = 0.1)
fit6 <- haRmonics(y = y,
                    numFreq = 6,
                    delta = 0.1)
fit12 <- haRmonics(y = y,
                    numFreq = 12,
                    delta = 0.1)

x = ts(fit1$fitted)
w = ts(fit3$fitted)
z = ts(fit6$fitted)
v = ts(fit12$fitted)

plot(y, pch = 16, main = "Previsoes de uma modelagem Harmonica")
lines(x ,lty = 5, col = "green")
lines(w, lty = 4, col = "red")
lines(z, lty = 3, col = "blue")
lines(v, lty = 2, col = "orange")
legend("topleft", legend = c("1", "3", "6","12"),
       col = c("green", "red", "blue","orange"), lty = c(5, 4, 3,2))
```



A modelagem harmônica foi satisfatória em capturar todas as estruturas da série, deixando apenas ruido branco.

Visto que não foi testada a necessidade de diferenciação da série, farei a modelagem sarima com ordem de sazonalidade = 12, argumentos d = D = 0. Iniciei com grid de parâmetros p,P,q,Q,d,D no intervalo [0,2]; e ajustou-se o limite dos parâmetros até a convergência de um modelo ótimo sem parâmetros na fronteira.

```
modelos <- data.frame(p = integer(),
                      P = integer(),
                      q = integer(),
                      Q = integer(),
                      d = integer(),
                      D = integer(),
                      AIC = numeric(),
                      BIC = numeric())

tic()
for(p in 0:3){
  for(P in 0:1){
      for(q in 0:4){
          for(Q in 0:1){
              for(d in 0:1){
                for(D in 0:1){
            tryCatch({
              fit = astsa::sarima(y, details = FALSE, Model = FALSE,
                            p = p, d = d, q = q, P = P, D = D, Q = Q, S = 12)
              AIC = fit$ICs[1]
              BIC = fit$ICs[3]
              mod = c(p, P, q, Q,d,D, AIC, BIC)
              modelos = rbind(modelos, mod)
            }, error = function(e) {
            })
        }}}}}}
toc()

colnames(modelos) = c("p","P","q","Q","d","D","AIC","BIC")

modelo = modelos %>%
  arrange(BIC, AIC) %>% head(1)
```

Logo, o melhor modelo (minimiza BIC) é o com os parâmetros:

```
kable(modelo)
```

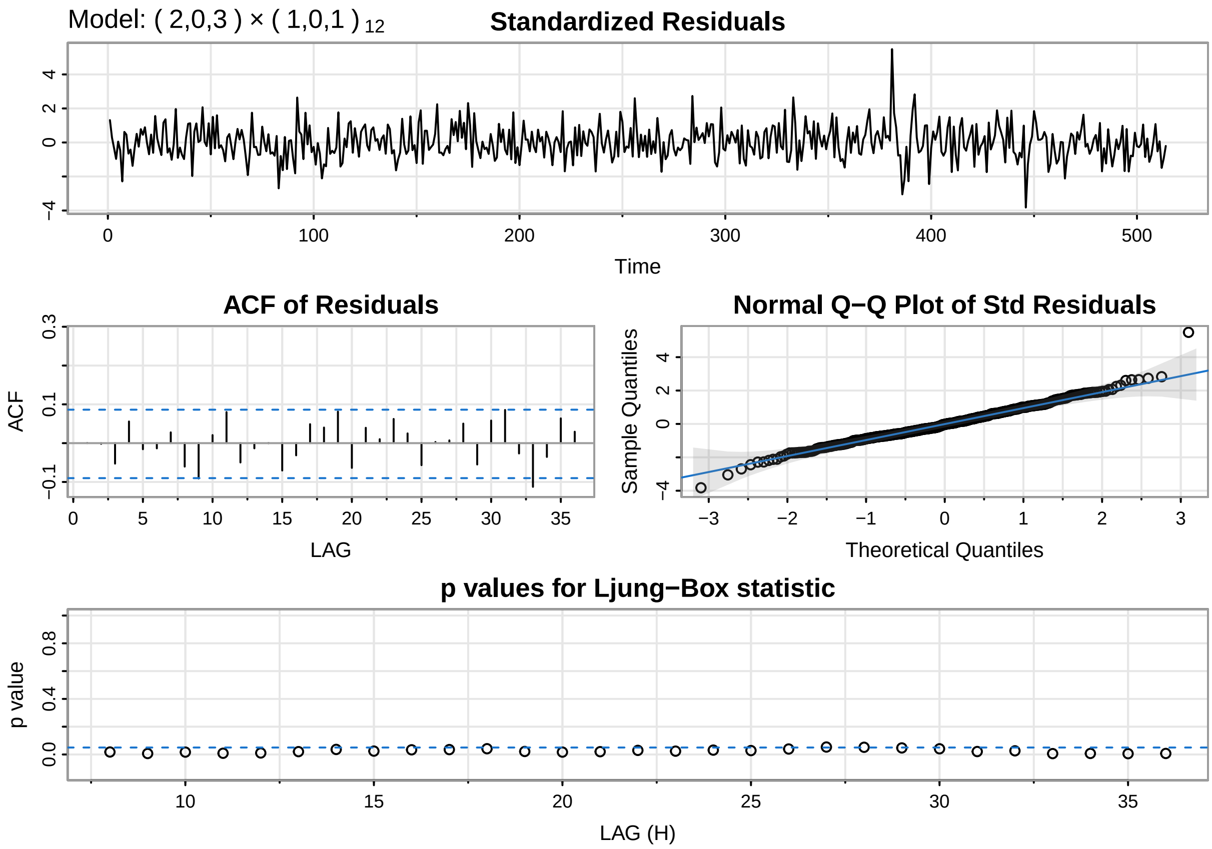
p	P	q	Q	d	D	AIC	BIC
2	1	3	1	0	0	5.52764	5.60192

```
fit = astsa::sarima(y,p = 2,P = 1, q = 3,Q = 1,d = 0, D = 0, S = 12)
```

```
## initial  value 2.374028
## iter   2 value 1.837765
## iter   3 value 1.701837
## iter   4 value 1.620495
## iter   5 value 1.586689
## iter   6 value 1.509525
## iter   7 value 1.472640
## iter   8 value 1.455921
## iter   9 value 1.448312
## iter  10 value 1.443313
## iter  11 value 1.440106
```

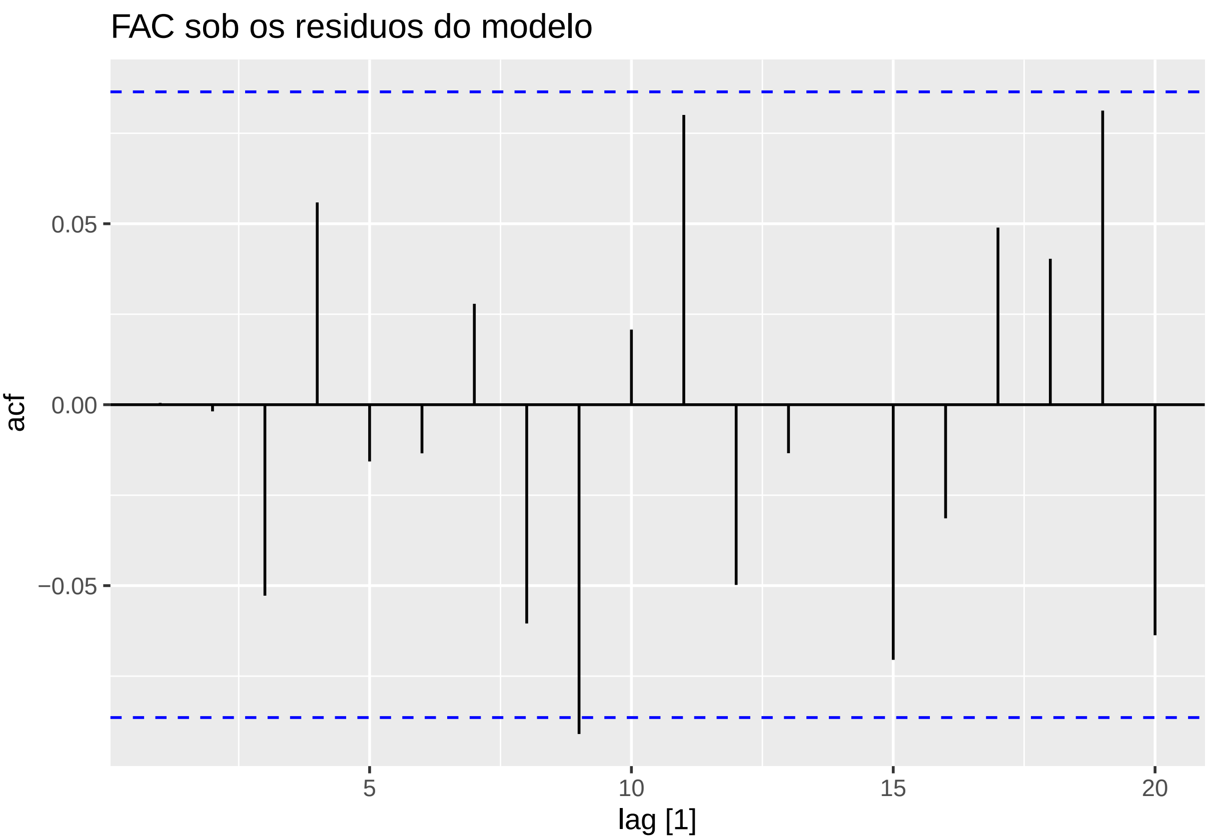
```
## iter  12 value 1.439908
## iter  13 value 1.439897
## iter  14 value 1.439895
## iter  15 value 1.439883
## iter  16 value 1.439876
## iter  17 value 1.439872
## iter  18 value 1.439871
## iter  19 value 1.439870
## iter  20 value 1.439869
## iter  21 value 1.439861
## iter  22 value 1.439786
## iter  23 value 1.439758
## iter  24 value 1.439700
## iter  25 value 1.439649
## iter  26 value 1.439528
## iter  27 value 1.439361
## iter  28 value 1.438932
## iter  29 value 1.436223
## iter  30 value 1.435474
## iter  31 value 1.434541
## iter  32 value 1.434274
## iter  33 value 1.433717
## iter  34 value 1.433038
## iter  35 value 1.432548
## iter  36 value 1.430645
## iter  37 value 1.427862
## iter  38 value 1.411846
## iter  39 value 1.411188
## iter  40 value 1.409867
## iter  41 value 1.404636
## iter  42 value 1.397075
## iter  43 value 1.393794
## iter  44 value 1.387473
## iter  45 value 1.382286
## iter  46 value 1.376430
## iter  47 value 1.370568
## iter  48 value 1.365502
## iter  49 value 1.360909
## iter  50 value 1.354005
## iter  51 value 1.349275
## iter  52 value 1.345474
## iter  53 value 1.340689
## iter  54 value 1.340159
## iter  55 value 1.337409
## iter  56 value 1.336043
## iter  57 value 1.334902
## iter  58 value 1.334876
## iter  59 value 1.334327
## iter  60 value 1.333187
## iter  61 value 1.332878
## iter  62 value 1.332547
## iter  63 value 1.332306
## iter  64 value 1.332294
## iter  65 value 1.332291
## iter  66 value 1.332288
## iter  67 value 1.332287
## iter  68 value 1.332287
## iter  68 value 1.332287
```

```
## iter  68 value 1.332287
## final  value 1.332287
## converged
## initial  value 1.331001
## iter   2 value 1.330433
## iter   3 value 1.330302
## iter   4 value 1.330149
## iter   5 value 1.329883
## iter   6 value 1.329696
## iter   7 value 1.329261
## iter   8 value 1.328359
## iter   9 value 1.327944
## iter  10 value 1.327665
## iter  11 value 1.327572
## iter  12 value 1.327540
## iter  13 value 1.327521
## iter  14 value 1.327508
## iter  15 value 1.327502
## iter  16 value 1.327468
## iter  17 value 1.327429
## iter  18 value 1.327391
## iter  19 value 1.327390
## iter  20 value 1.327390
## iter  21 value 1.327387
## iter  22 value 1.327381
## iter  23 value 1.327380
## iter  24 value 1.327378
## iter  25 value 1.327377
## iter  26 value 1.327376
## iter  27 value 1.327375
## iter  28 value 1.327374
## iter  29 value 1.327373
## iter  30 value 1.327372
## iter  31 value 1.327372
## iter  32 value 1.327372
## iter  33 value 1.327372
## iter  34 value 1.327372
## iter  35 value 1.327372
## iter  35 value 1.327372
## final  value 1.327372
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE   t.value p.value
## ar1     1.7219 0.0059  292.9485  0.0000
## ar2    -0.9918 0.0057 -174.4726  0.0000
## ma1    -1.4016 0.0441  -31.8020  0.0000
## ma2     0.5611 0.0712    7.8855  0.0000
## ma3     0.1323 0.0446    2.9654  0.0032
## sar1    0.9051 0.0520   17.3979  0.0000
## sma1   -0.8123 0.0695  -11.6819  0.0000
## xmean  18.9257 0.3185   59.4199  0.0000
##
## sigma^2 estimated as 14.02405 on 506 degrees of freedom
##
## AIC = 5.52764  AICc = 5.528194  BIC = 5.60192
##
```

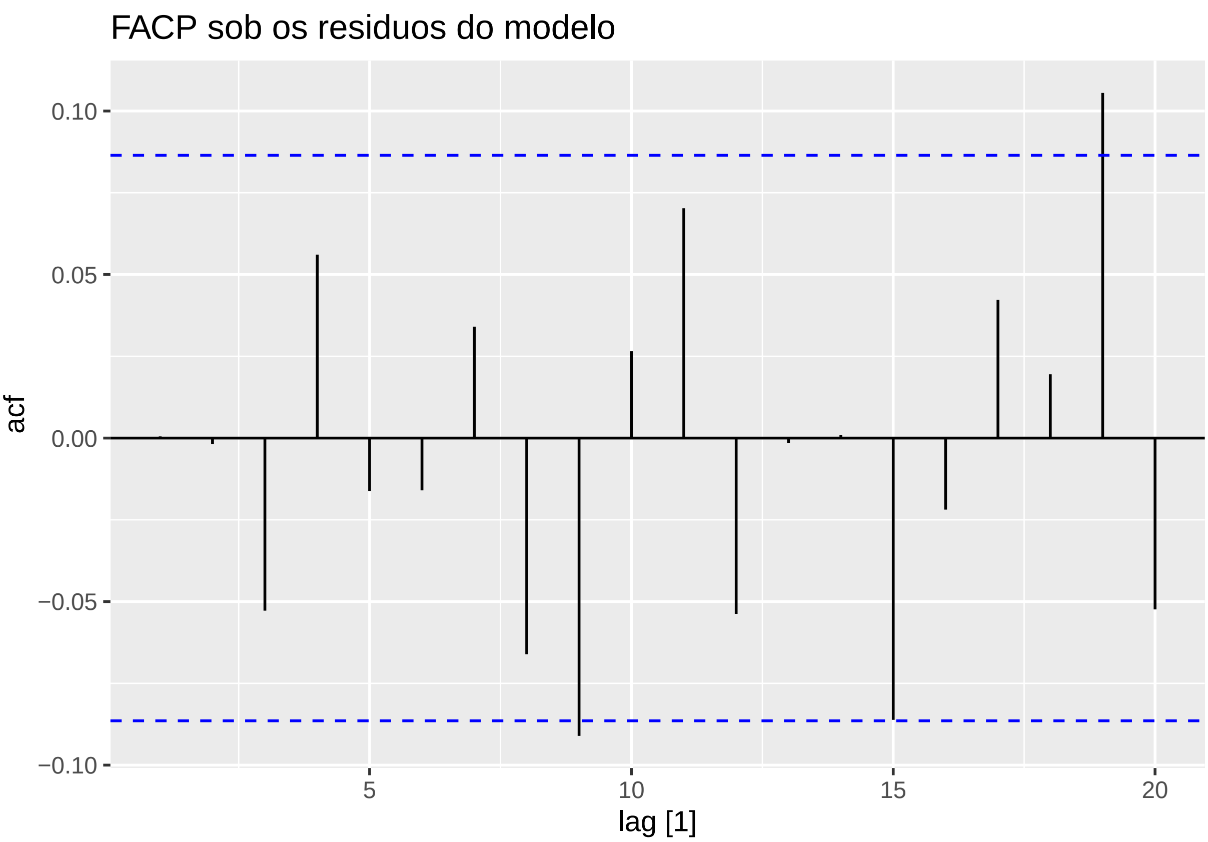



```
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```r
res %>%
  ACF(value,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```



```r
Box.test(fit[["fit"]][["residuals"]], lag = 20, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 26.006, df = 20, p-value = 0.1656
```

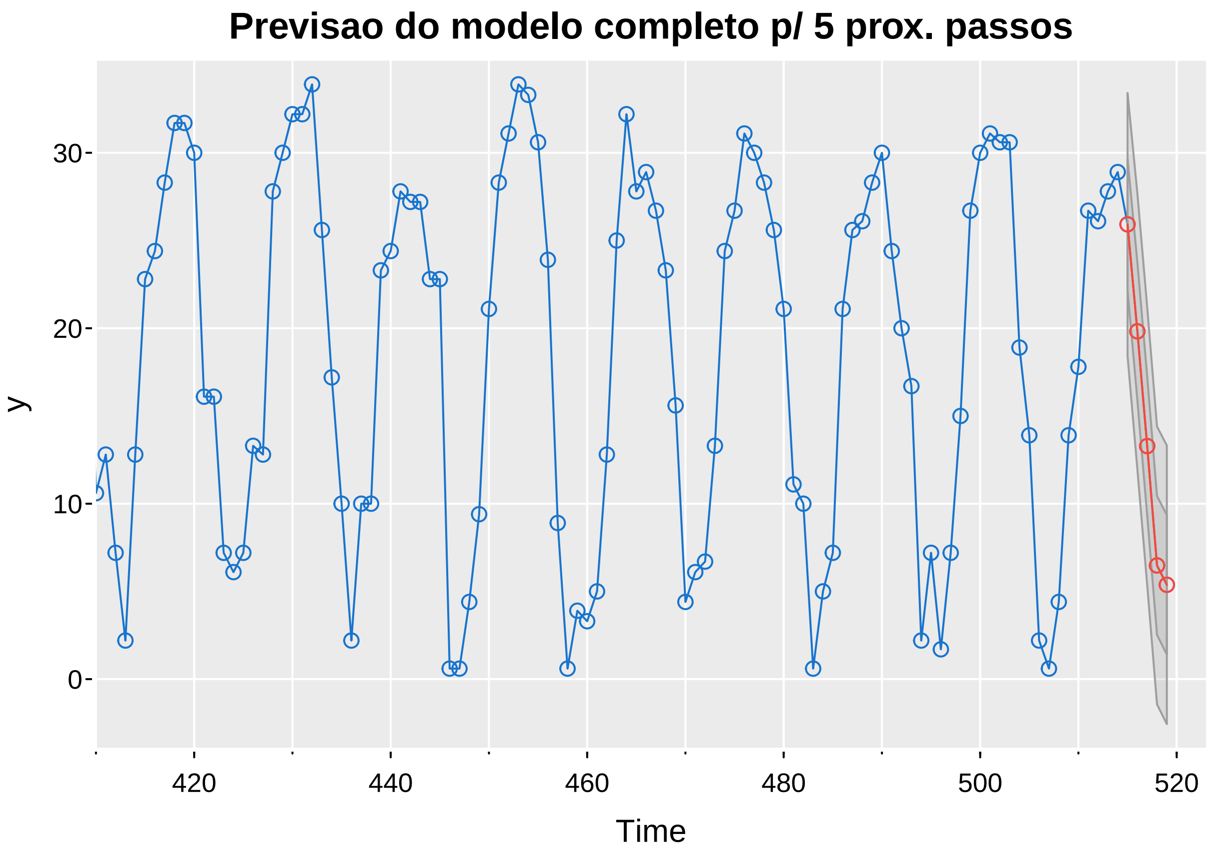
```r
shapiro.test(fit[["fit"]][["residuals"]])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.98501, p-value = 3.849e-05
```

Sob a hipótese nula de independência, o teste de Ljung-Box não rejeita a hipótese nula. Portanto, existem indícios de independência na série dos resíduos.

Avaliando as formas da FAC, FACP e resultado do teste de Ljung-Box, aparenta haver alguma correlação nos resíduos, além da rejeição da normalidade pelo teste de Shapiro-Wilk. Apesar disso, o gráfico Q-Q mostra certa aderência aos quantis esperados de uma distribuição normal, salvo um outlier isolado. Como já foram feitos testes de diferenciação anteriormente, este é o melhor modelo que pode se obter com este método.

```r
prev = sarima.for(y,p = 2,P = 1,q = 3,Q = 1, d = 0, D = 0, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Previsao do modelo completo p/ 5 prox. passos")
```

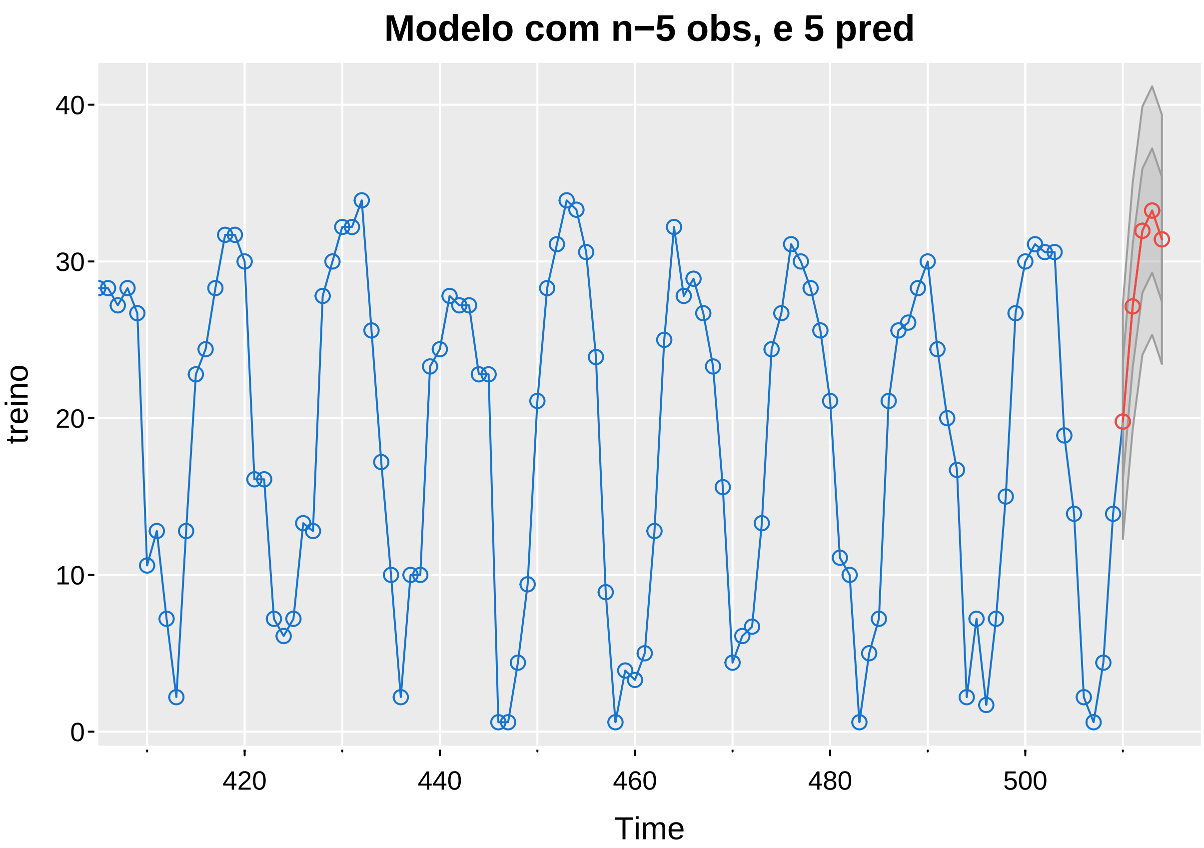


```r
prev
```

```
## $pred
## Time Series:
## Start = 515
## End = 519
## Frequency = 1
## [1] 25.921961 19.823627 13.284658  6.479668  5.378258
##
## $se
## Time Series:
## Start = 515
## End = 519
## Frequency = 1
## [1] 3.744869 3.932272 3.958214 3.959128 3.970647
```

```r
treino = ts(y[1:509])
teste = ts(y[510:514])
```

```r
prev = sarima.for(treino, p = 2, P = 1, q = 3,Q = 1, d = 0, D = 0, S = 12,
        n.ahead = 5, gg=TRUE, col=4,main="Modelo com n-5 obs, e 5 pred")
```




```r
prev$pred
```

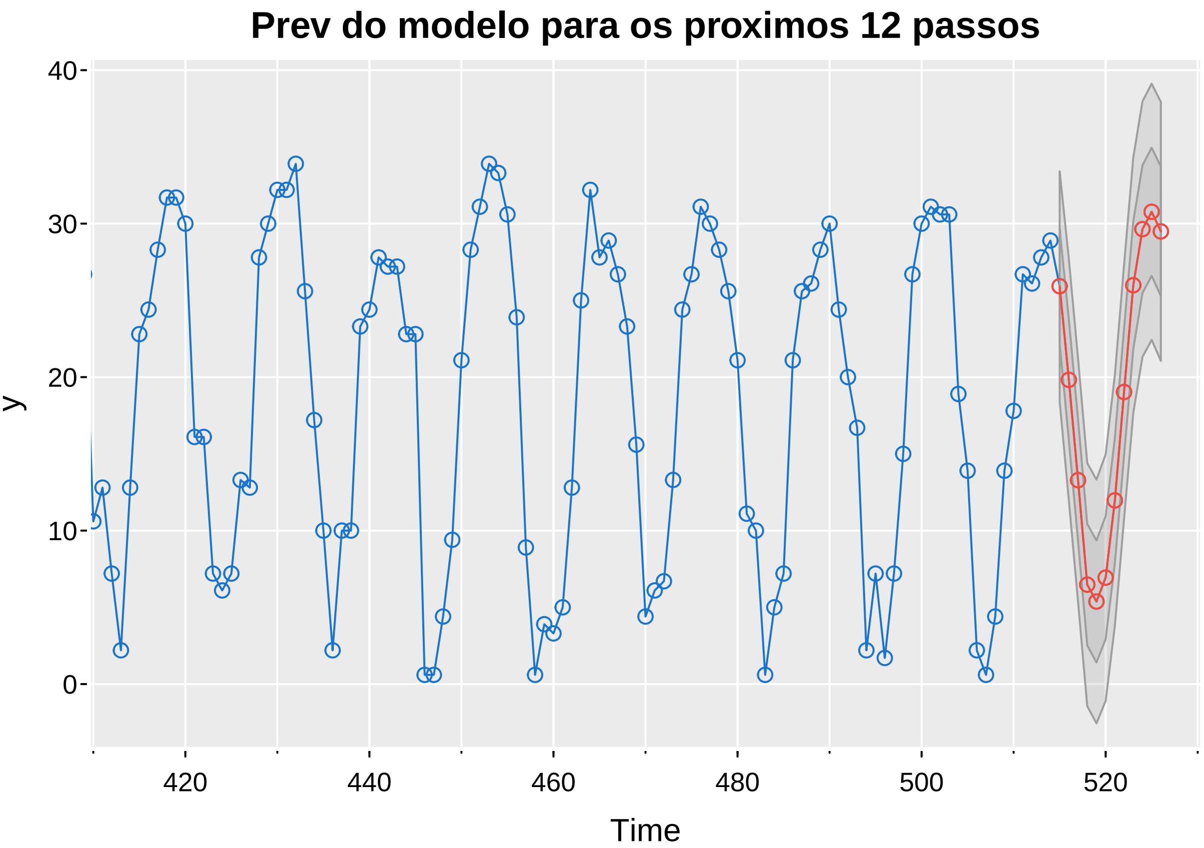
```
## Time Series:
## Start = 510
## End = 514
## Frequency = 1
## [1] 19.78464 27.13199 31.95792 33.25017 31.41203
```

```r
MAPE = MAPE(prev$pred, as.vector(teste))
MAPE
```

```
## [1] 0.1270177
```

Com um MAPE = 0.1270177, ou seja,entre 10 e 20%, podemos dizer que se trata de um excelente modelo preditivo.

```r
prev = sarima.for(y, p = 2, P = 1, q = 3,Q = 1, d = 0, D = 0, S = 12,
                  n.ahead = 12, gg=TRUE, col=4,main="Prev do modelo para os proximos 12 passos")
```



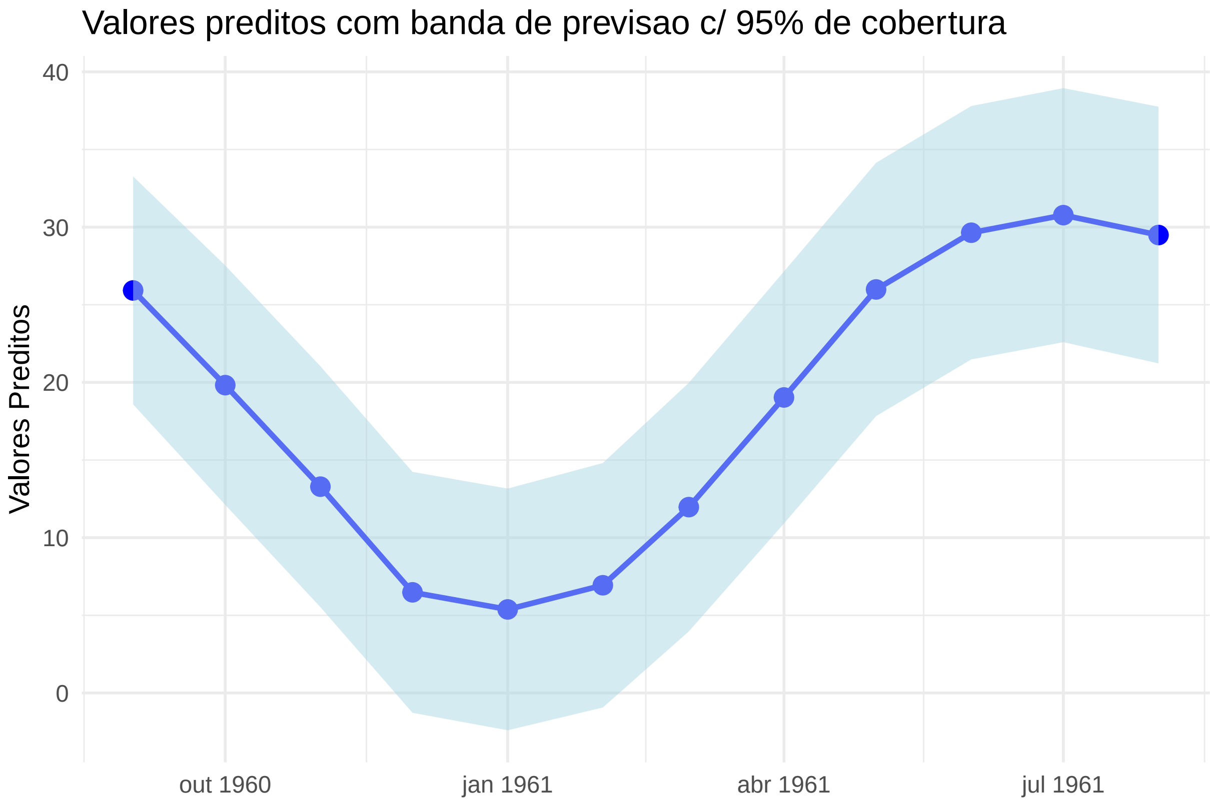

```r
datas <- seq.Date(from = as.Date("1960-09-01"), to = as.Date("1961-08-31"), by = "months")

previsoes = data.frame(datas,prev$pred,prev$se)
colnames(previsoes) = c("data","Valores_preditos","Erro_padrao")

previsoes = previsoes %>%
  mutate(Limite_superior = Valores_preditos + (1.96 * Erro_padrao),
         Limite_inferior = Valores_preditos - (1.96 * Erro_padrao))

ggplot(previsoes, aes(x = data)) +
  geom_line(aes(y = Valores_preditos), color = "blue", size = 1) +
  geom_point(aes(y = Valores_preditos), color = "blue", size = 3) +
  geom_ribbon(aes(ymin = Limite_inferior, ymax = Limite_superior), fill = "lightblue", alpha = 0.5)
  labs(title = "Valores preditos com banda de previsao c/ 95% de cobertura",
       x = "",
       y = "Valores Preditos") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



Alguns diagnósticos do modelo foram parcialmente todos atendidos. Portanto, as previsões deste modelo apresentam alguma credibilidade. Considerando a banda de confiança e a especificidade dos dados, temos de tomar cuidado pois os resíduos não são normais. Esta banda pode estar incorreta, apesar da fuga da normalidade aparentar ser ligeira pelo gráfico Q-Q. Ainda assim, se trata de um modelo relevante para análises preditivas.

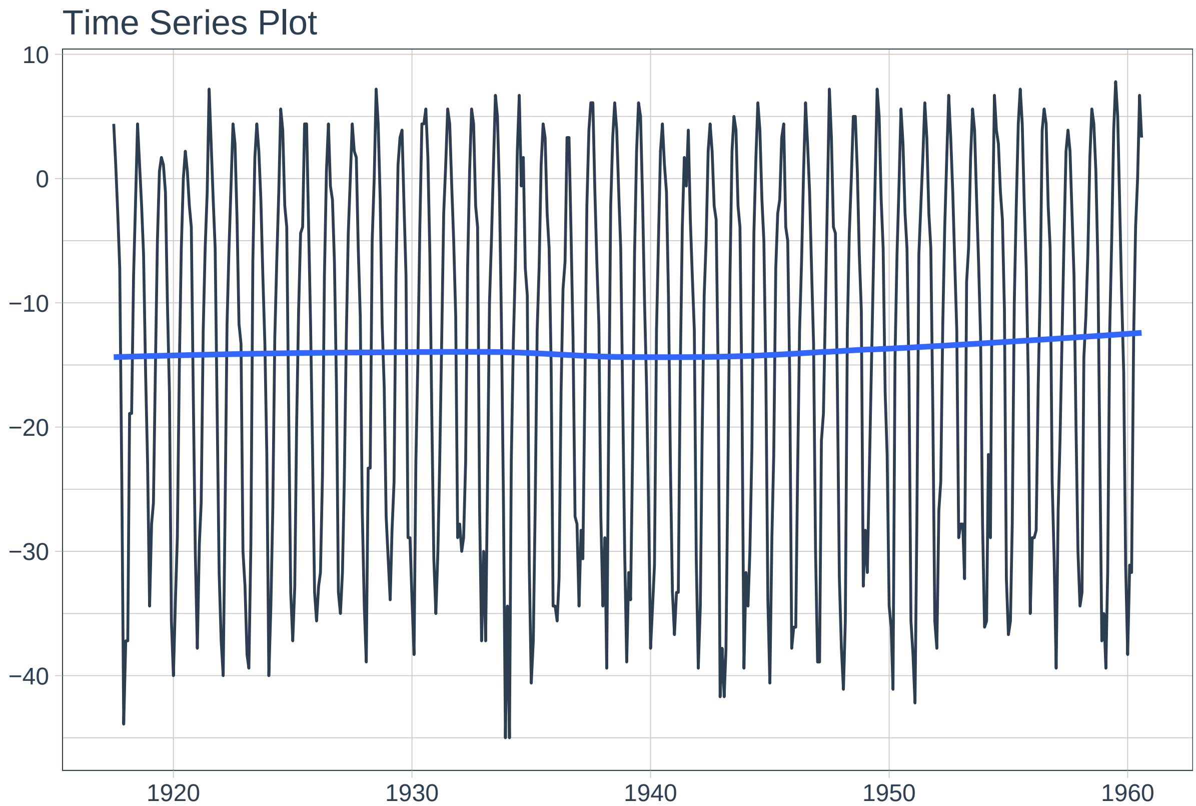
5. "Extr Min Temp (°C)";

```
df5 = df[,c(5,16)]
df5 = df5 %>% drop_na()
colnames(df5)[2]="extr_min_temp"
```

```
df5$data = as.Date(df5$data)
df5 = tsibble::as_tsibble(df5,index=data,regular=F)
```

Visualizando a série com uma linha suavizada

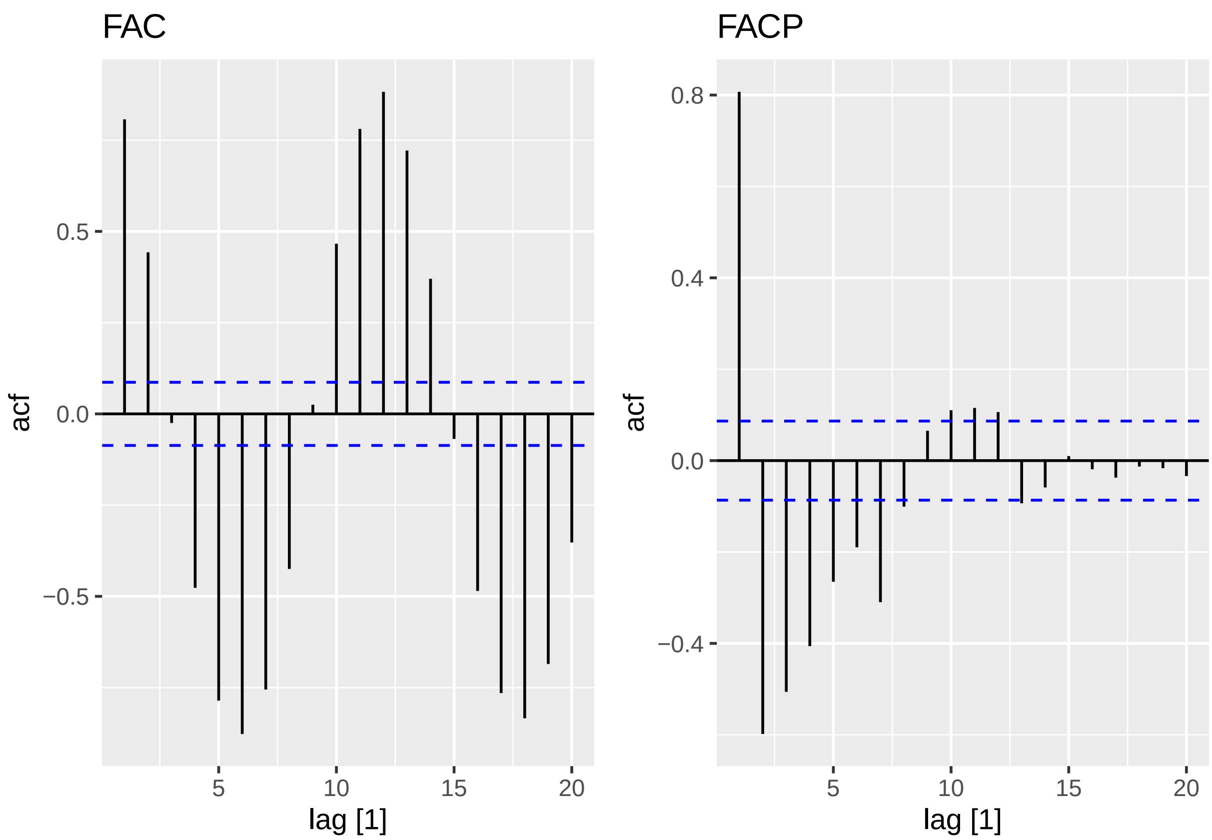
```
df5 %>%
  plot_time_series(data,extr_min_temp, .interactive = F, .smooth = T)
```




```
FAC5 = df5 %>%
  ACF(var = extr_min_temp,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC")

FACP5 = df5 %>%
  ACF(var = extr_min_temp,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP")

grid.arrange(FAC5, FACP5, nrow = 1)
```




```r
aTSA::adf.test(ts(df5$extr_min_temp), nlag = 5)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag    ADF p.value
## [1,]   0  -5.28    0.01
## [2,]   1  -9.26    0.01
## [3,]   2 -11.34    0.01
## [4,]   3  -9.58    0.01
## [5,]   4  -6.63    0.01
## Type 2: with drift no trend
##       lag    ADF p.value
## [1,]   0  -7.34    0.01
## [2,]   1 -14.02    0.01
## [3,]   2 -20.95    0.01
## [4,]   3 -23.68    0.01
## [5,]   4 -21.64    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
## [1,]   0  -7.34    0.01
## [2,]   1 -14.02    0.01
## [3,]   2 -20.95    0.01
## [4,]   3 -23.70    0.01
## [5,]   4 -21.68    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

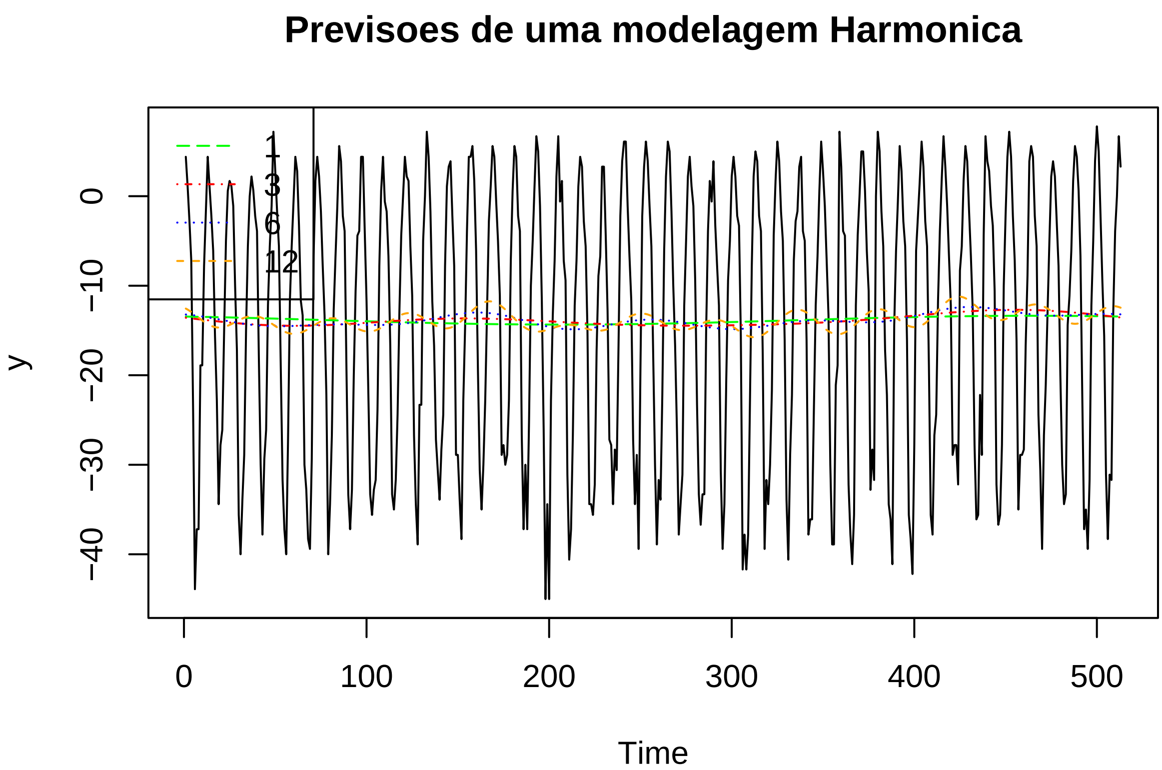
O teste de Dickey-Fuller sugere a estacionariedade da série

```r
y <- ts(df5$extr_min_temp)

fit1 <- haRmonics(y = y,
                  numFreq = 1,
                  delta = 0.1)
fit3 <- haRmonics(y = y,
                   numFreq = 3,
                   delta = 0.1)
fit6 <- haRmonics(y = y,
                    numFreq = 6,
                    delta = 0.1)
fit12 <- haRmonics(y = y,
                    numFreq = 12,
                    delta = 0.1)

x = ts(fit1$fitted)
w = ts(fit3$fitted)
z = ts(fit6$fitted)
v = ts(fit12$fitted)

plot(y, pch = 16, main = "Previsoes de uma modelagem Harmonica")
lines(x ,lty = 5, col = "green")
lines(w, lty = 4, col = "red")
lines(z, lty = 3, col = "blue")
lines(v, lty = 2, col = "orange")
legend("topleft", legend = c("1", "3", "6","12"),
       col = c("green", "red", "blue","orange"), lty = c(5, 4, 3,2))
```



A modelagem harmônica foi satisfatória em capturar todas as estruturas da série, deixando apenas ruido branco.

Visto que não foi testada a necessidade de diferenciação da série, farei a modelagem sarima com ordem de sazonalidade = 12, argumentos d = D = 0. Iniciei com grid de parâmetros p,P,q,Q,d,D no intervalo [0,2]; e ajustou-se o limite dos parâmetros até a convergência de um modelo ótimo sem parâmetros na fronteira.

```r
modelos <- data.frame(p = integer(),
                      P = integer(),
                      q = integer(),
                      Q = integer(),
                      d = integer(),
                      D = integer(),
                      AIC = numeric(),
                      BIC = numeric())

tic()
for(p in 0:4){
  for(P in 0:2){
      for(q in 0:3){
          for(Q in 0:2){
              for(d in 0:1){
                for(D in 0:1){
            tryCatch({
              fit = astsa::sarima(y, details = FALSE, Model = FALSE,
                             p = p, d = d, q = q, P = P, D = D, Q = Q, S = 12)
              AIC = fit$ICs[1]
              BIC = fit$ICs[3]
              mod = c(p, P, q, Q,d,D, AIC, BIC)
              modelos = rbind(modelos, mod)
            }, error = function(e) {
            })
        }}}}}}
toc()

colnames(modelos) = c("p","P","q","Q","d","D","AIC","BIC")

modelo = modelos %>%
  arrange(BIC, AIC) %>% head(1)
```

Logo, o melhor modelo (minimiza BIC) é o com os parâmetros:

```r
kable(modelo)
```

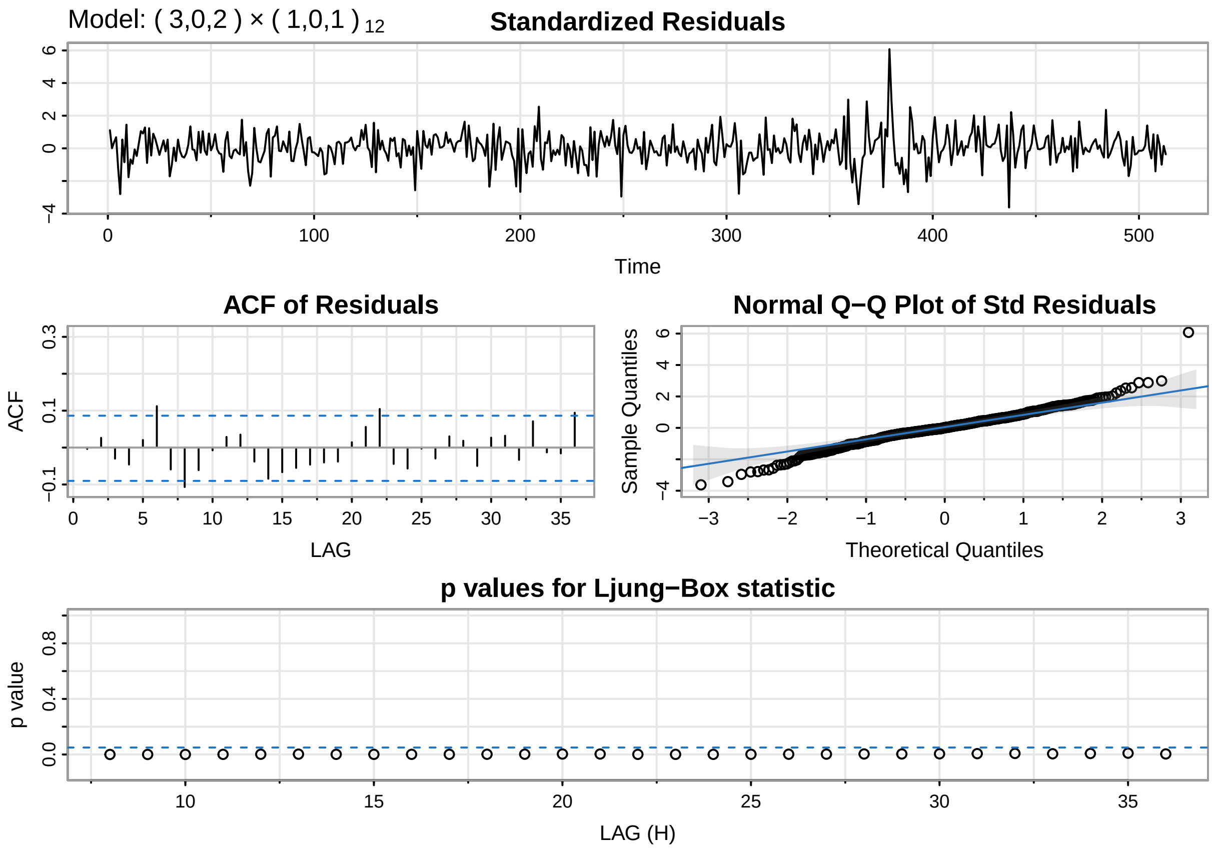
p	P	q	Q	d	D	AIC	BIC
3	1	2	1	0	0	5.982323	6.056714

```r
fit = astsa::sarima(y,p = 3,P = 1, q = 2,Q = 1,d = 0, D = 0, S = 12)
```

```
## initial  value 2.715863
## iter   2 value 2.121298
## iter   3 value 1.827911
## iter   4 value 1.796501
## iter   5 value 1.768303
## iter   6 value 1.740574
## iter   7 value 1.703015
## iter   8 value 1.699634
## iter   9 value 1.691798
## iter  10 value 1.690421
## iter  11 value 1.682692
```

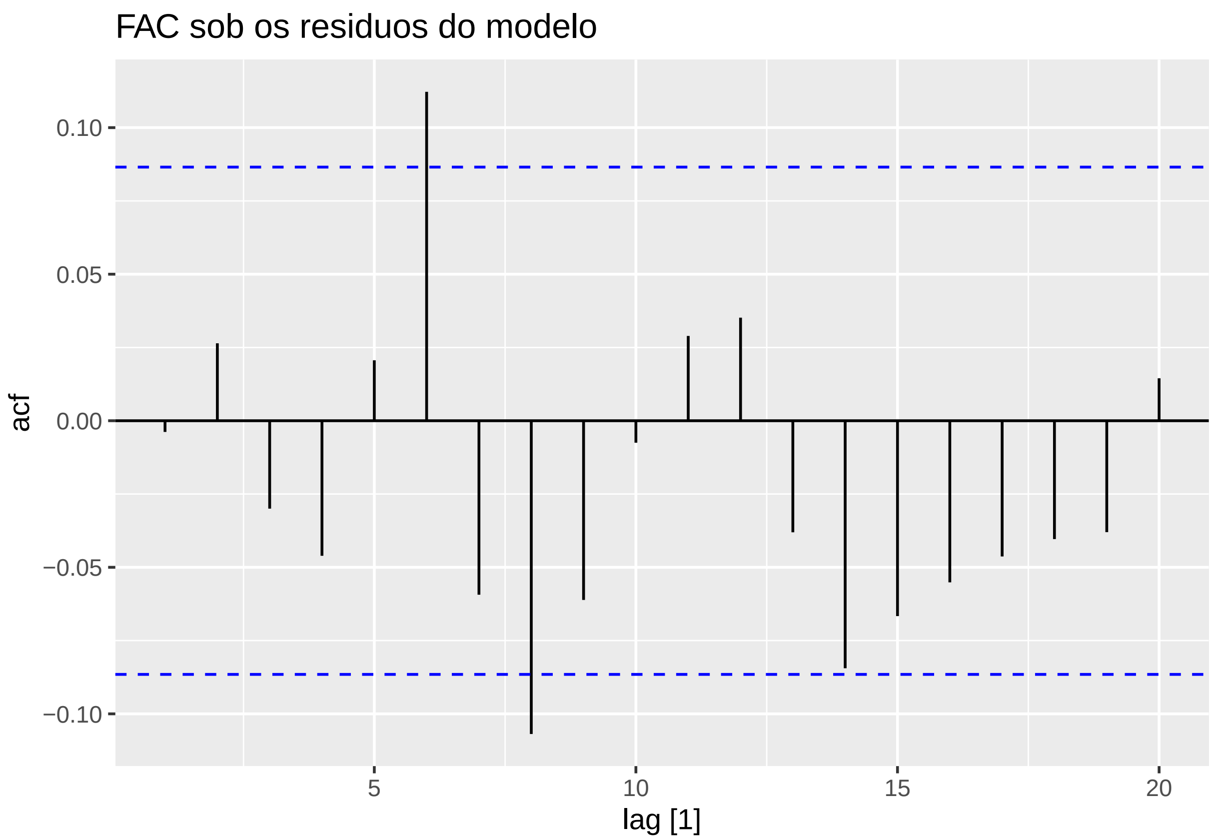
```
## iter  12 value 1.678060
## iter  13 value 1.644793
## iter  14 value 1.640460
## iter  15 value 1.630282
## iter  16 value 1.629403
## iter  17 value 1.627258
## iter  18 value 1.626527
## iter  19 value 1.625635
## iter  20 value 1.624606
## iter  21 value 1.621009
## iter  22 value 1.616368
## iter  23 value 1.605102
## iter  24 value 1.596112
## iter  25 value 1.594906
## iter  26 value 1.592906
## iter  27 value 1.586376
## iter  28 value 1.583892
## iter  29 value 1.580461
## iter  30 value 1.579187
## iter  31 value 1.577279
## iter  32 value 1.576483
## iter  33 value 1.574712
## iter  34 value 1.572016
## iter  35 value 1.570186
## iter  36 value 1.567527
## iter  37 value 1.566363
## iter  38 value 1.565636
## iter  39 value 1.565163
## iter  40 value 1.564724
## iter  41 value 1.564016
## iter  42 value 1.563828
## iter  43 value 1.563781
## iter  44 value 1.563768
## iter  45 value 1.563767
## iter  46 value 1.563766
## iter  47 value 1.563763
## iter  48 value 1.563762
## iter  49 value 1.563762
## iter  50 value 1.563762
## iter  51 value 1.563762
## iter  52 value 1.563762
## iter  53 value 1.563762
## iter  54 value 1.563762
## iter  55 value 1.563762
## iter  56 value 1.563762
## iter  56 value 1.563762
## iter  56 value 1.563762
## final  value 1.563762
## converged
## initial  value 1.566511
## iter   2 value 1.566454
## iter   3 value 1.566355
## iter   4 value 1.565936
## iter   5 value 1.565253
## iter   6 value 1.564227
## iter   7 value 1.563162
## iter   8 value 1.562018
## iter   9 value 1.559890
```

```
## iter  10 value 1.558200
## iter  11 value 1.557809
## iter  12 value 1.557329
## iter  13 value 1.557016
## iter  14 value 1.556005
## iter  15 value 1.555807
## iter  16 value 1.555666
## iter  17 value 1.555481
## iter  18 value 1.555283
## iter  19 value 1.555263
## iter  20 value 1.555250
## iter  21 value 1.555238
## iter  22 value 1.555196
## iter  23 value 1.555146
## iter  24 value 1.555065
## iter  25 value 1.554989
## iter  26 value 1.554921
## iter  27 value 1.554825
## iter  28 value 1.554735
## iter  29 value 1.554691
## iter  30 value 1.554683
## iter  31 value 1.554680
## iter  32 value 1.554680
## iter  33 value 1.554679
## iter  33 value 1.554679
## iter  33 value 1.554679
## final  value 1.554679
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE  t.value p.value
## ar1     1.9942 0.0803  24.8333   0e+00
## ar2    -1.4662 0.1355 -10.8228   0e+00
## ar3     0.2764 0.0771   3.5862   4e-04
## ma1    -1.5350 0.0611 -25.1426   0e+00
## ma2     0.7512 0.0510  14.7293   0e+00
## sar1    0.9582 0.0300  31.9135   0e+00
## sma1   -0.8717 0.0563 -15.4783   0e+00
## xmean -13.9060 0.5310 -26.1874   0e+00
##
## sigma^2 estimated as 22.00784 on 505 degrees of freedom
##
## AIC = 5.982324  AICc = 5.98288  BIC = 6.056714
##
```

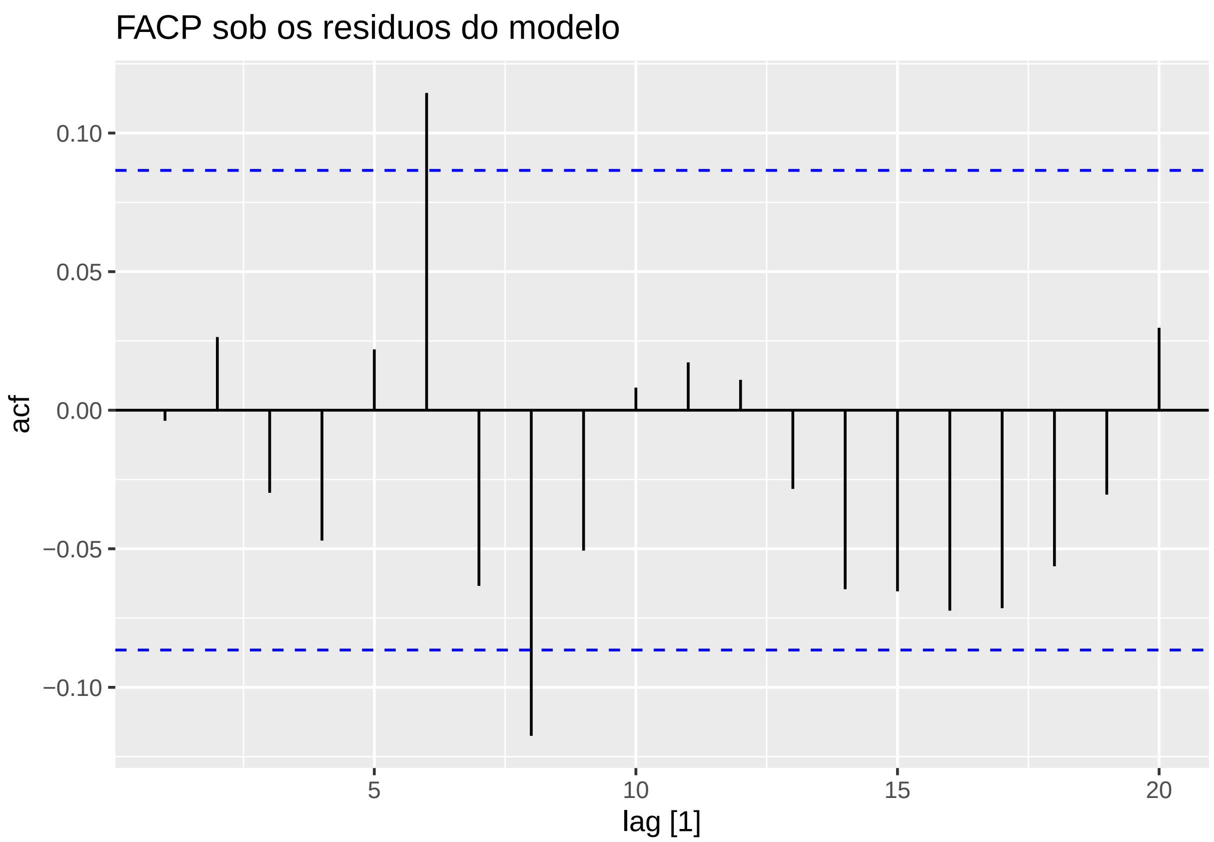



```
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```r
res %>%
  ACF(value,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```



```r
Box.test(fit[["fit"]][["residuals"]], lag = 20, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 31.024, df = 20, p-value = 0.05487
```

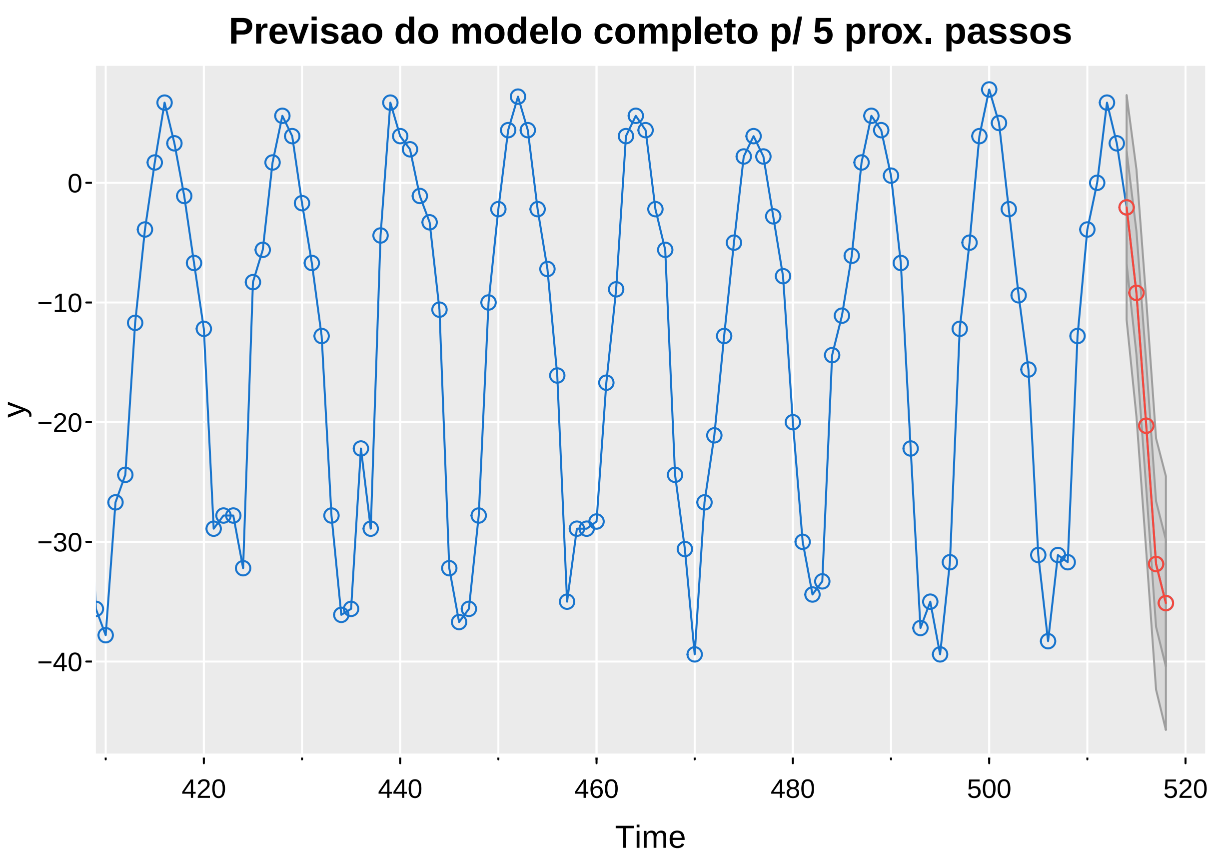
```r
shapiro.test(fit[["fit"]][["residuals"]])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.97064, p-value = 1.292e-08
```

Sob a hipótese nula de independência, o teste de Ljung-Box não rejeita a hipótese nula a 5%, porém numa fronteira próxima à rejeição. Portanto, existem indícios de independência na série dos resíduos, ainda que não tão fortes.

Avaliando as formas da FAC, FACP e resultado do teste de Ljung-Box, aparenta haver alguma correlação nos resíduos, além de uma acentuada fuga da normalidade dos resíduos, tanto pelo teste de Shapiro-WWilk, quanto pelo gráfico Q-Q. Como já foram feitos testes de diferenciação anteriormente, este é o melhor modelo que pode se obter com este método.

```r
prev = sarima.for(y,p = 3,P = 1,q = 2,Q = 1, d = 0, D = 0, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Previsao do modelo completo p/ 5 prox. passos")
```

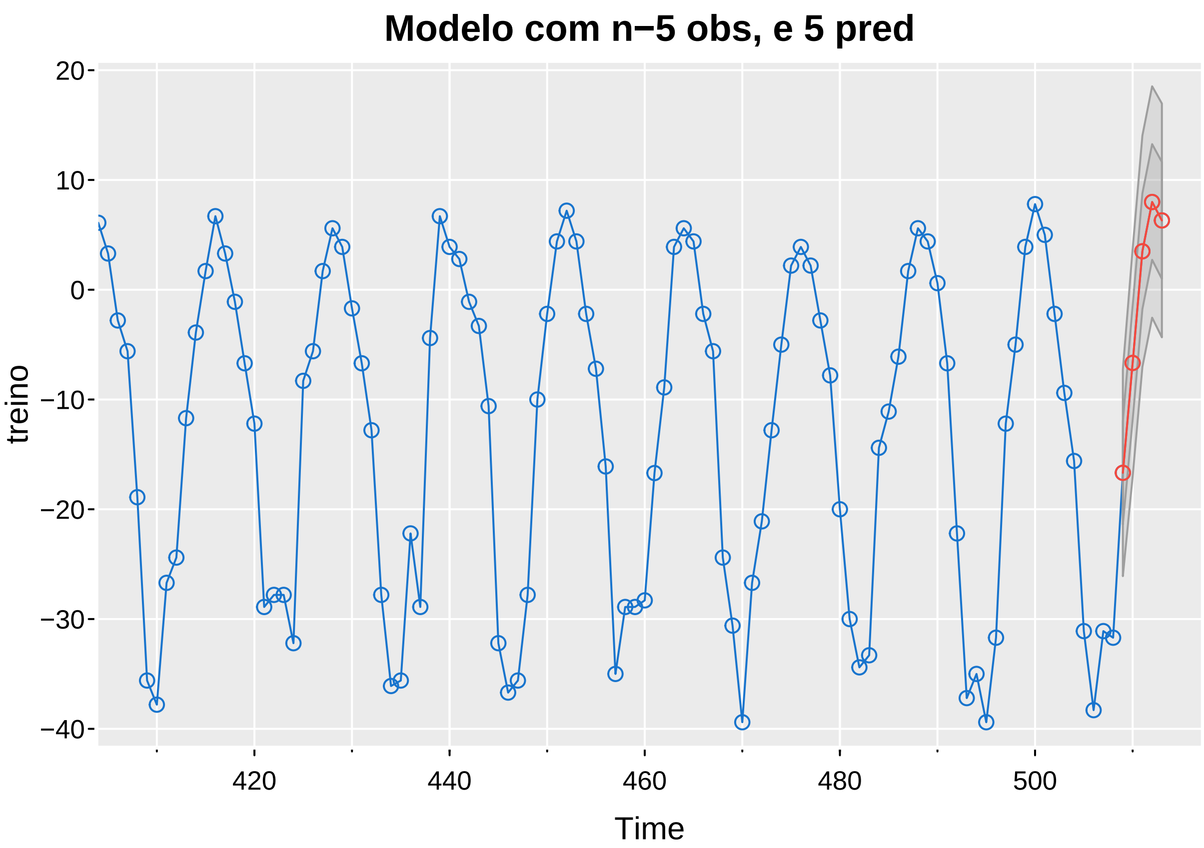


```r
prev
```

```
## $pred
## Time Series:
## Start = 514
## End = 518
## Frequency = 1
## [1]  -2.051815  -9.194738 -20.297438 -31.852327 -35.115978
##
## $se
## Time Series:
## Start = 514
## End = 518
## Frequency = 1
## [1] 4.691252 5.162202 5.247420 5.247445 5.301205
```

```r
treino = ts(y[1:508])
teste = ts(y[509:513])
```

```r
prev = sarima.for(treino, p = 3, P = 1, q = 2,Q = 1, d = 0, D = 0, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Modelo com n-5 obs, e 5 pred")
```




```r
prev$pred
```

```
## Time Series:
## Start = 509
## End = 513
## Frequency = 1
## [1] -16.678175  -6.668206   3.503964   7.992485   6.313424
```

```r
MAPE = MLmetrics::MAPE(prev$pred, as.vector(teste))
MAPE
```

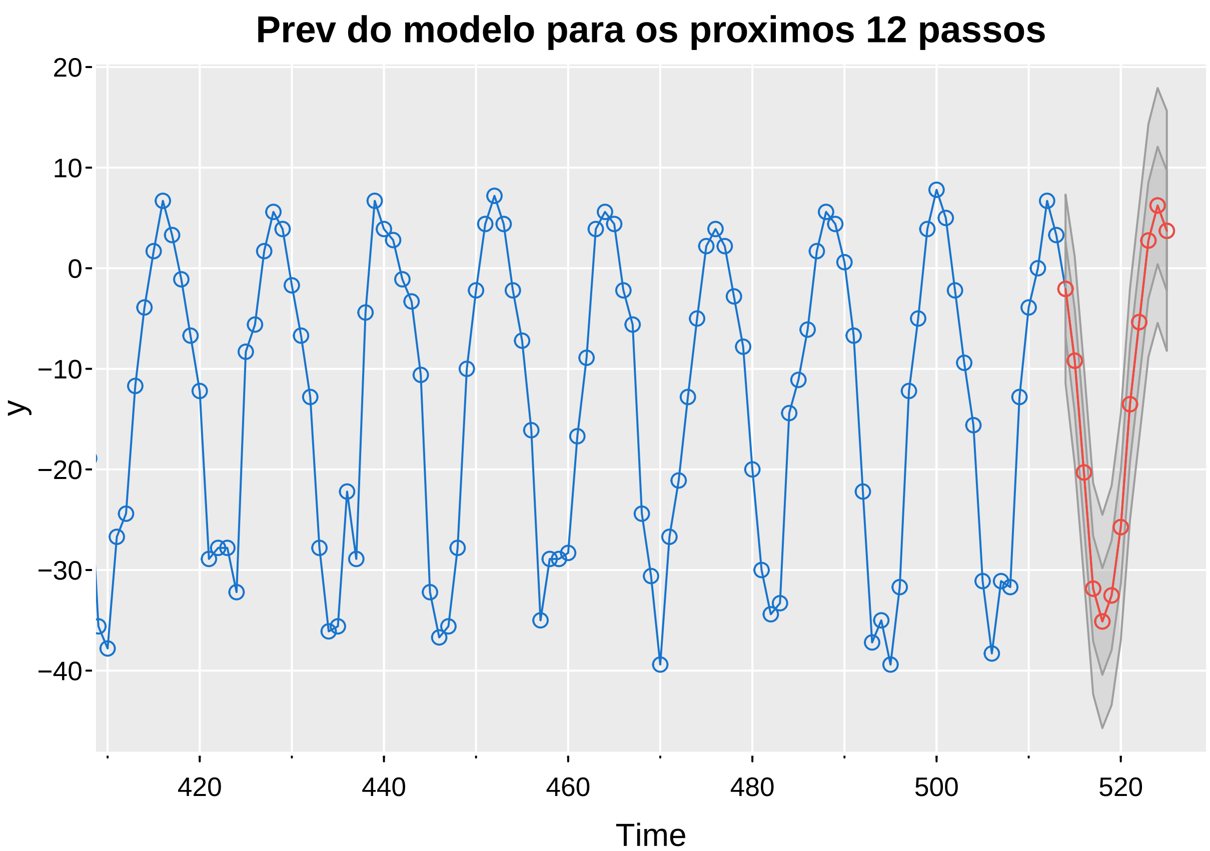
```
## [1] Inf
```

```r
# Pseudo-MAPE
set.seed(150167636)
teste[3] <- rnorm(1)
MAPE = MAPE(prev$pred, as.vector(teste))
MAPE
```

```
## [1] 4.740971
```

Não foi possível calcular um MAPE para este conjunto, visto que uma das observações era igual à zero, e isso retorna um MAPE infinito por conta da divisão por zero. Calculando um "pseudo-MAPE" (ou seja, incluí um ruído aleatório na observação para que deixasse de ser zero) para assim ser possível obter algum valor; ainda assim o valor retornado foi 4.7409709, o que não faz muito sentido.

```r
prev = sarima.for(y, p = 3, P = 1, q = 2,Q = 1, d = 0, D = 0, S = 12,
        n.ahead = 12, gg=TRUE, col=4,main="Prev do modelo para os proximos 12 passos")
```



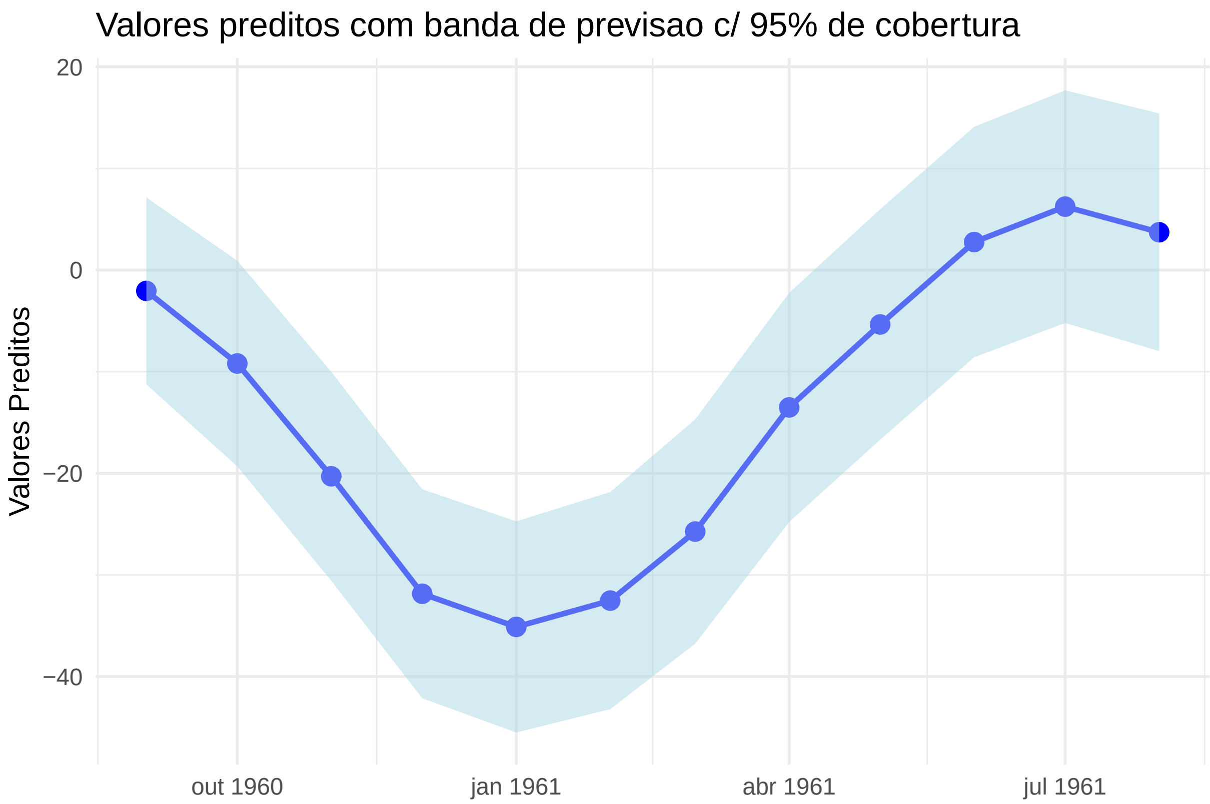

```r
datas <- seq.Date(from = as.Date("1960-09-01"), to = as.Date("1961-08-31"), by = "months")

previsoes = data.frame(datas,prev$pred,prev$se)
colnames(previsoes) = c("data","Valores_preditos","Erro_padrao")

previsoes = previsoes %>%
  mutate(Limite_superior = Valores_preditos + (1.96 * Erro_padrao),
         Limite_inferior = Valores_preditos - (1.96 * Erro_padrao))
```

```r
ggplot(previsoes, aes(x = data)) +
  geom_line(aes(y = Valores_preditos), color = "blue", size = 1) +
  geom_point(aes(y = Valores_preditos), color = "blue", size = 3) +
  geom_ribbon(aes(ymin = Limite_inferior, ymax = Limite_superior), fill = "lightblue", alpha = 0.5)
  labs(title = "Valores preditos com banda de previsao c/ 95% de cobertura",
       x = "",
       y = "Valores Preditos") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



Quase todos os diagnósticos do modelo foram rejeitados. A fuga da normalidade é extremamente acentuada. Sequer foi possível calcular um MAPE. Este modelo não é nem um pouco confiável para previsões. A indicação é não utilizá-lo, e buscar alguma modelagem mais sofisticada para entender o fenômeno desta variável, ainda que visualmente a previsão tenha o comportamento senoidal aproximado da série registrada.

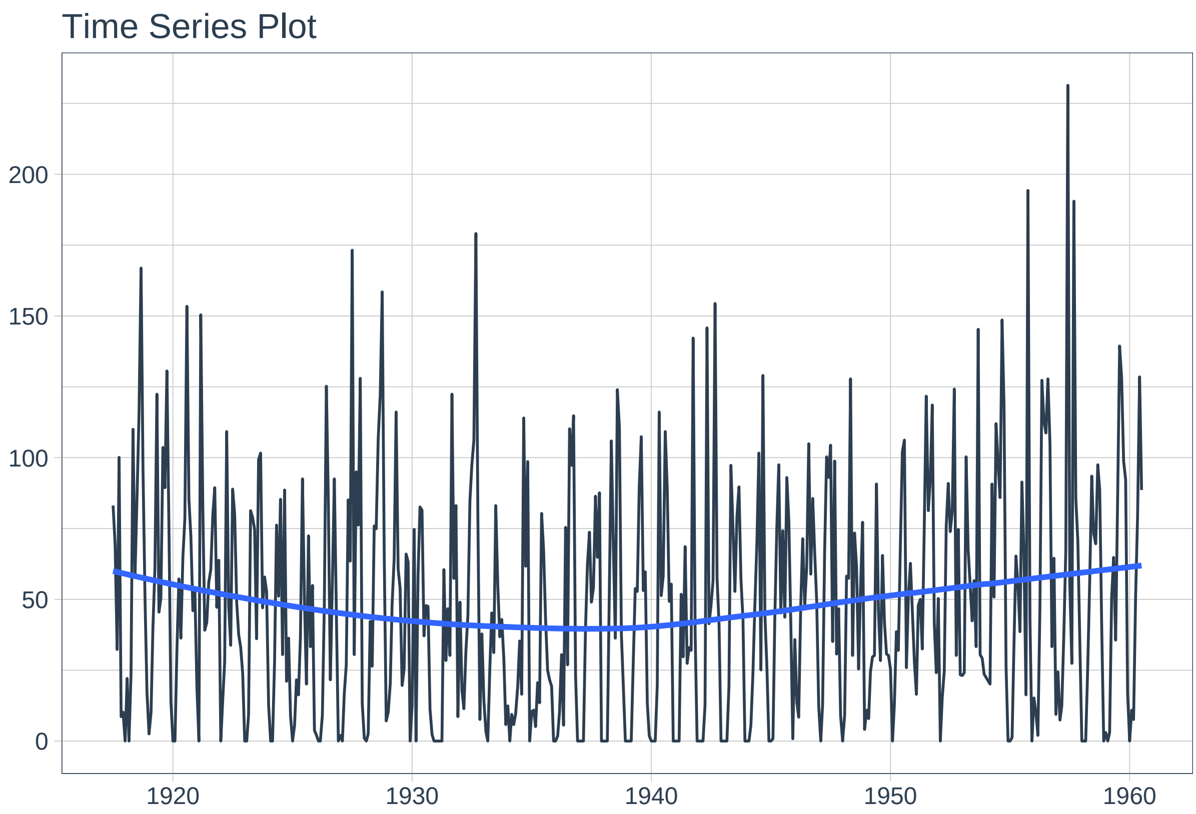
6. "Total Rain (mm)";

```r
df6 = df[,c(5,18)]
df6 = df6 %>% drop_na()
colnames(df6)[2]="total_rain"
```

```r
df6$data = as.Date(df6$data)
df6 = tsibble::as_tsibble(df6,index=data,regular=F)
```

Visualizando a série com uma linha suavizada

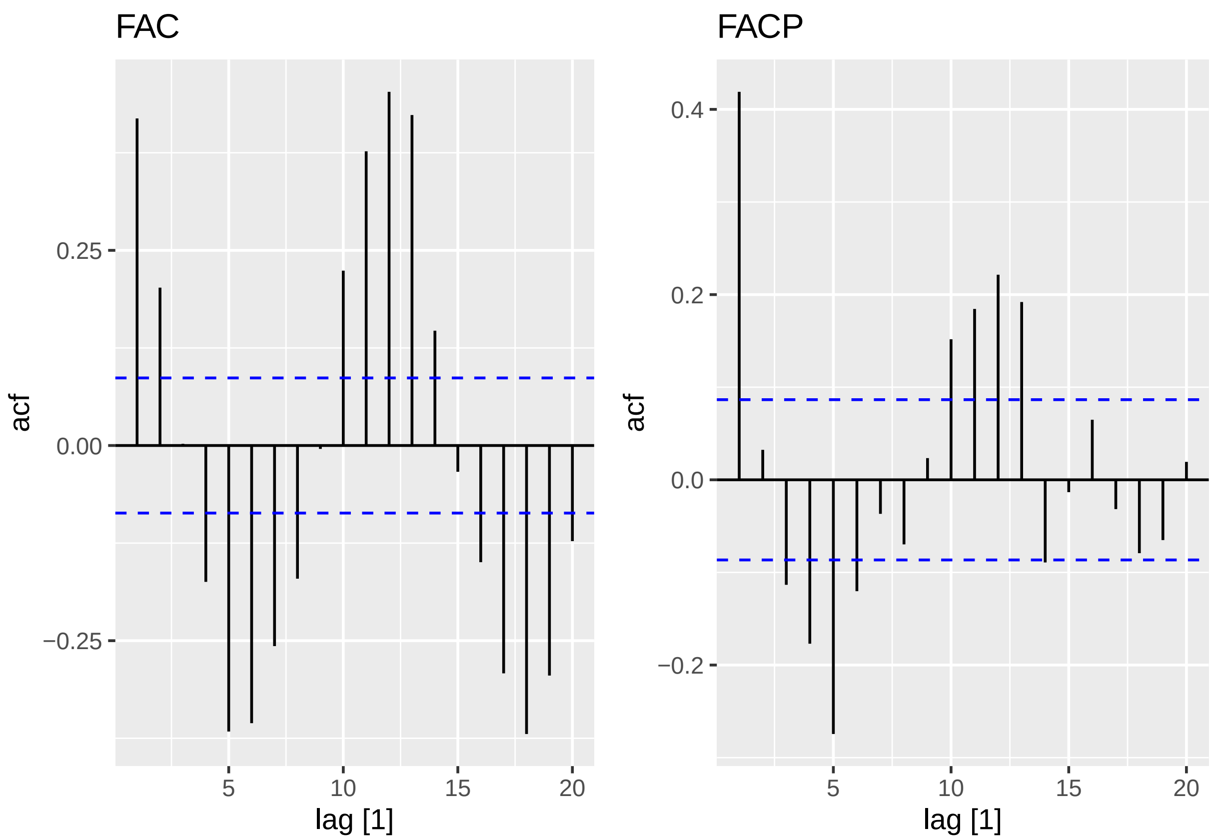
```
df6 %>%
  plot_time_series(data,total_rain, .interactive = F, .smooth = T)
```




```
FAC6 = df6 %>%
  ACF(var = total_rain,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC")

FACP6 = df6 %>%
  ACF(var = total_rain,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP")

grid.arrange(FAC6, FACP6, nrow = 1)
```




```
aTSA::adf.test(ts(df6$total_rain), nlag = 10)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##       lag   ADF p.value
##  [1,]   0 -8.41  0.0100
##  [2,]   1 -6.25  0.0100
##  [3,]   2 -5.64  0.0100
##  [4,]   3 -5.47  0.0100
##  [5,]   4 -5.44  0.0100
##  [6,]   5 -4.48  0.0100
##  [7,]   6 -3.50  0.0100
##  [8,]   7 -2.96  0.0100
##  [9,]   8 -2.30  0.0219
## [10,]   9 -1.63  0.0989
## Type 2: with drift no trend
##       lag    ADF p.value
##  [1,]   0 -14.43    0.01
##  [2,]   1 -11.70    0.01
##  [3,]   2 -11.59    0.01
##  [4,]   3 -12.36    0.01
##  [5,]   4 -14.34    0.01
##  [6,]   5 -13.63    0.01
##  [7,]   6 -12.07    0.01
##  [8,]   7 -11.35    0.01
##  [9,]   8  -9.82    0.01
## [10,]   9  -7.63    0.01
## Type 3: with drift and trend
##       lag    ADF p.value
```

```
##  [1,]   0 -14.46    0.01
##  [2,]   1 -11.73    0.01
##  [3,]   2 -11.63    0.01
##  [4,]   3 -12.41    0.01
##  [5,]   4 -14.42    0.01
##  [6,]   5 -13.73    0.01
##  [7,]   6 -12.18    0.01
##  [8,]   7 -11.49    0.01
##  [9,]   8  -9.96    0.01
## [10,]   9  -7.75    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

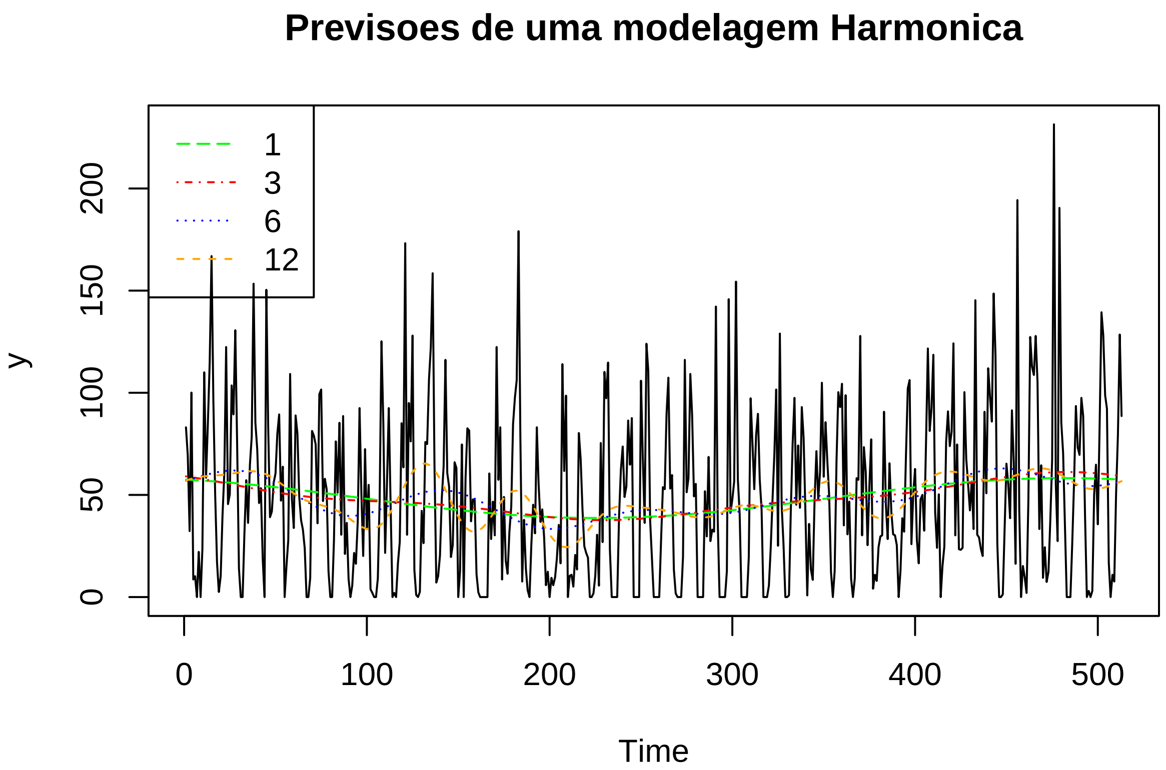
O teste de Dickey-Fuller sugere a estacionariedade da série

```
y <- ts(df6$total_rain)

fit1 <- haRmonics(y = y,
                  numFreq = 1,
                  delta = 0.1)
fit3 <- haRmonics(y = y,
                   numFreq = 3,
                   delta = 0.1)
fit6 <- haRmonics(y = y,
                    numFreq = 6,
                    delta = 0.1)
fit12 <- haRmonics(y = y,
                    numFreq = 12,
                    delta = 0.1)

x = ts(fit1$fitted)
w = ts(fit3$fitted)
z = ts(fit6$fitted)
v = ts(fit12$fitted)

plot(y, pch = 16, main = "Previsoes de uma modelagem Harmonica")
lines(x ,lty = 5, col = "green")
lines(w, lty = 4, col = "red")
lines(z, lty = 3, col = "blue")
lines(v, lty = 2, col = "orange")
legend("topleft", legend = c("1", "3", "6","12"),
       col = c("green", "red", "blue","orange"), lty = c(5, 4, 3,2))
```



A modelagem harmônica foi satisfatória em capturar todas as estruturas da série, deixando apenas ruido branco.

Visto que não foi testada a necessidade de diferenciação da série, farei a modelagem sarima com ordem de sazonalidade = 12, argumentos d = D = 0. Iniciei com grid de parâmetros p,P,q,Q,d,D no intervalo [0,2]; e ajustou-se o limite dos parâmetros até a convergência de um modelo ótimo sem parâmetros na fronteira.

```
modelos <- data.frame(p = integer(),
                      P = integer(),
                      q = integer(),
                      Q = integer(),
                      d = integer(),
                      D = integer(),
                      AIC = numeric(),
                      BIC = numeric())

tic()
for(p in 0:3){
  for(P in 0:2){
      for(q in 0:3){
          for(Q in 0:2){
              for(d in 0:0){
                for(D in 0:0){
            tryCatch({
              fit = astsa::sarima(y, details = FALSE, Model = FALSE,
                             p = p, d = d, q = q, P = P, D = D, Q = Q, S = 12)
              AIC = fit$ICs[1]
              BIC = fit$ICs[3]
              mod = c(p, P, q, Q,d,D, AIC, BIC)
              modelos = rbind(modelos, mod)
            }, error = function(e) {
            })
        }}}}}}
```

```r
toc()

colnames(modelos) = c("p","P","q","Q","d","D","AIC","BIC")

modelo = modelos %>%
  arrange(BIC, AIC) %>% head(1)
```

Logo, o melhor modelo (minimiza BIC) é o com os parâmetros:

```r
kable(modelo)
```

p	P	q	Q	d	D	AIC	BIC
2	1	2	1	0	0	9.810969	9.877094

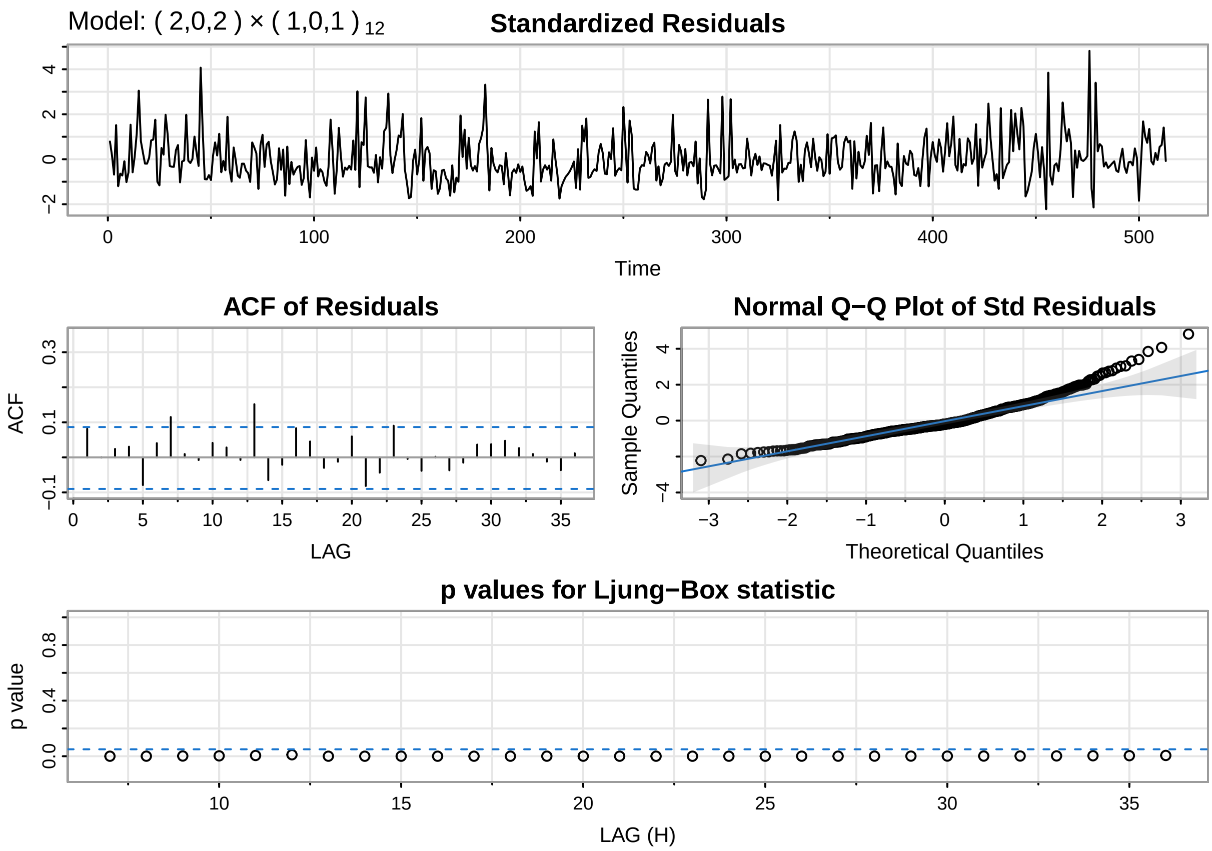
```r
fit = astsa::sarima(y,p = 2,P = 1, q = 2,Q = 1,d = 0, D = 0, S = 12)
```

```
## initial  value 3.717890
## iter   2 value 3.602397
## iter   3 value 3.583478
## iter   4 value 3.580737
## iter   5 value 3.565037
## iter   6 value 3.553340
## iter   7 value 3.546420
## iter   8 value 3.538025
## iter   9 value 3.526239
## iter  10 value 3.522587
## iter  11 value 3.513538
## iter  12 value 3.509796
## iter  13 value 3.509528
## iter  14 value 3.509456
## iter  15 value 3.509026
## iter  16 value 3.508243
## iter  17 value 3.507903
## iter  18 value 3.507821
## iter  19 value 3.507718
## iter  20 value 3.507668
## iter  21 value 3.507490
## iter  22 value 3.507078
## iter  23 value 3.506191
## iter  24 value 3.504985
## iter  25 value 3.503217
## iter  26 value 3.502402
## iter  27 value 3.501752
## iter  28 value 3.501253
## iter  29 value 3.500837
## iter  30 value 3.500393
## iter  31 value 3.500108
## iter  32 value 3.500072
## iter  33 value 3.500058
## iter  34 value 3.500014
## iter  35 value 3.499974
## iter  36 value 3.499859
## iter  37 value 3.499854
## iter  38 value 3.499607
## iter  39 value 3.499464
```

```
## iter  40 value 3.499396
## iter  41 value 3.499347
## iter  42 value 3.499335
## iter  43 value 3.499332
## iter  44 value 3.499321
## iter  45 value 3.499314
## iter  46 value 3.499310
## iter  47 value 3.499310
## iter  48 value 3.499309
## iter  49 value 3.499309
## iter  50 value 3.499309
## iter  51 value 3.499309
## iter  52 value 3.499309
## iter  53 value 3.499308
## iter  54 value 3.499307
## iter  55 value 3.499307
## iter  55 value 3.499307
## iter  55 value 3.499307
## final  value 3.499307
## converged
## initial  value 3.504813
## iter   2 value 3.504576
## iter   3 value 3.504529
## iter   4 value 3.503783
## iter   5 value 3.503099
## iter   6 value 3.502220
## iter   7 value 3.501431
## iter   8 value 3.501151
## iter   9 value 3.501058
## iter  10 value 3.500865
## iter  11 value 3.500556
## iter  12 value 3.500182
## iter  13 value 3.499465
## iter  14 value 3.499345
## iter  15 value 3.498956
## iter  16 value 3.498669
## iter  17 value 3.498624
## iter  18 value 3.498407
## iter  19 value 3.497199
## iter  20 value 3.496103
## iter  21 value 3.494407
## iter  22 value 3.493926
## iter  23 value 3.492606
## iter  24 value 3.490331
## iter  25 value 3.487443
## iter  26 value 3.486982
## iter  27 value 3.484444
## iter  28 value 3.482432
## iter  29 value 3.479829
## iter  30 value 3.479472
## iter  31 value 3.479214
## iter  32 value 3.478964
## iter  33 value 3.478752
## iter  34 value 3.478381
## iter  35 value 3.478209
## iter  36 value 3.477734
## iter  37 value 3.477114
## iter  38 value 3.475550
```

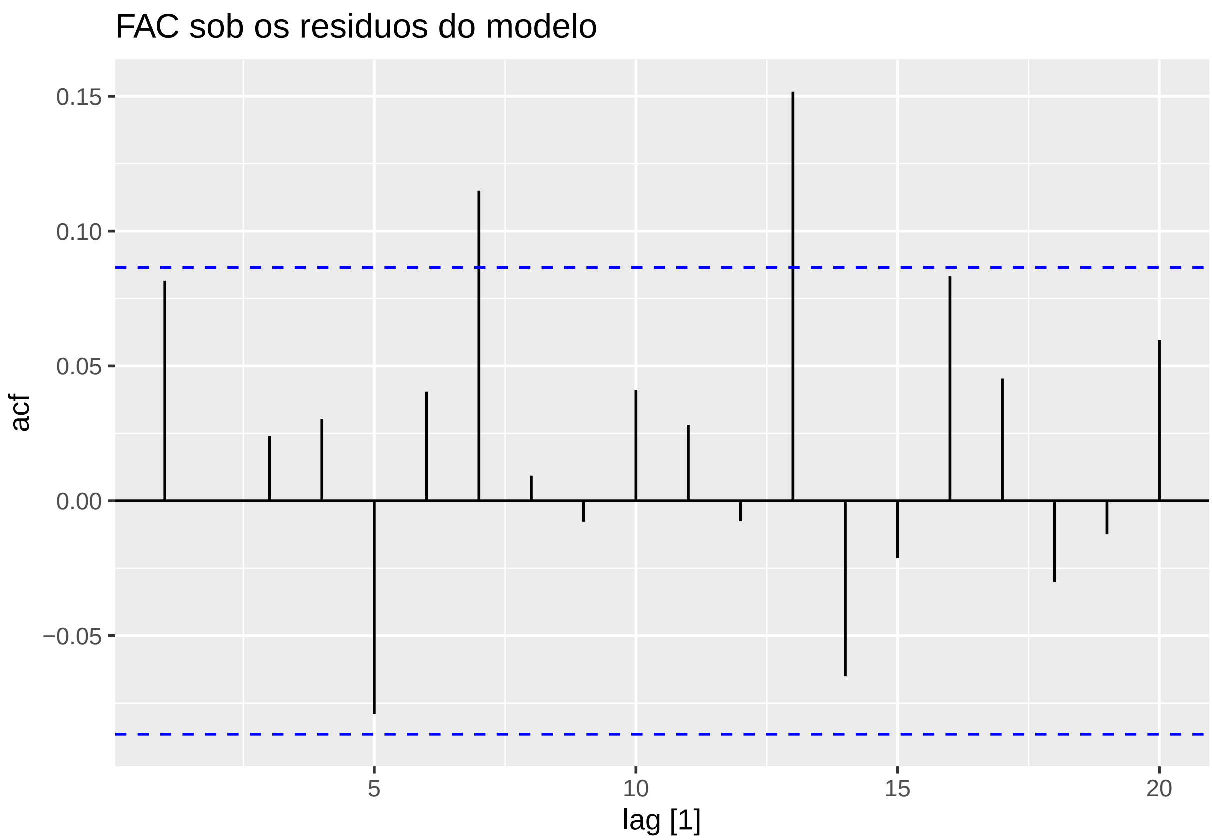
```
## iter  39 value 3.474286
## iter  40 value 3.472761
## iter  41 value 3.472366
## iter  42 value 3.472171
## iter  43 value 3.472095
## iter  44 value 3.472047
## iter  45 value 3.471895
## iter  46 value 3.471705
## iter  47 value 3.471687
## iter  48 value 3.471682
## iter  49 value 3.471677
## iter  50 value 3.471672
## iter  51 value 3.471574
## iter  52 value 3.471519
## iter  53 value 3.471419
## iter  54 value 3.471338
## iter  55 value 3.471282
## iter  56 value 3.471222
## iter  57 value 3.471152
## iter  58 value 3.471099
## iter  59 value 3.471078
## iter  60 value 3.471070
## iter  61 value 3.471064
## iter  62 value 3.471063
## iter  63 value 3.471063
## iter  64 value 3.471062
## iter  65 value 3.471054
## iter  66 value 3.471034
## iter  67 value 3.471021
## iter  68 value 3.471016
## iter  69 value 3.470995
## iter  70 value 3.470983
## iter  71 value 3.470977
## iter  72 value 3.470976
## iter  73 value 3.470974
## iter  74 value 3.470969
## iter  75 value 3.470961
## iter  76 value 3.470952
## iter  77 value 3.470952
## iter  77 value 3.470952
## iter  77 value 3.470952
## final  value 3.470952
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE   t.value p.value
## ar1     1.7247 0.0036  478.0919       0
## ar2    -0.9970 0.0033 -303.7063       0
## ma1    -1.6807 0.0179  -93.8226       0
## ma2     0.9550 0.0184   51.9577       0
## sar1    0.7972 0.0935    8.5284       0
## sma1   -0.6846 0.1091   -6.2767       0
## xmean  48.9337 2.1567   22.6896       0
##
## sigma^2 estimated as 1025.252 on 506 degrees of freedom
##
## AIC = 9.810969  AICc = 9.811402  BIC = 9.877094
```

##

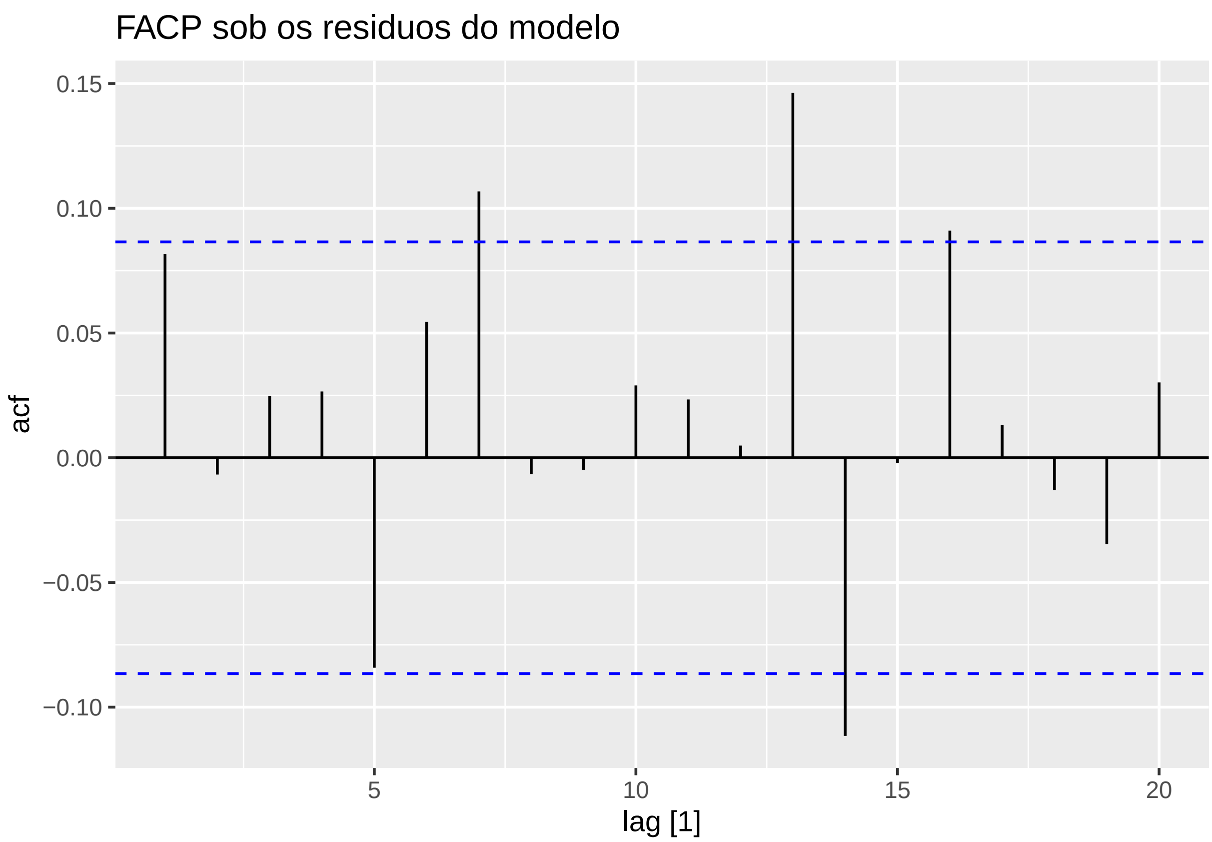



```
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```r
res %>%
  ACF(value,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```



```r
Box.test(fit[["fit"]][["residuals"]], lag = 20, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 38.532, df = 20, p-value = 0.007619
```

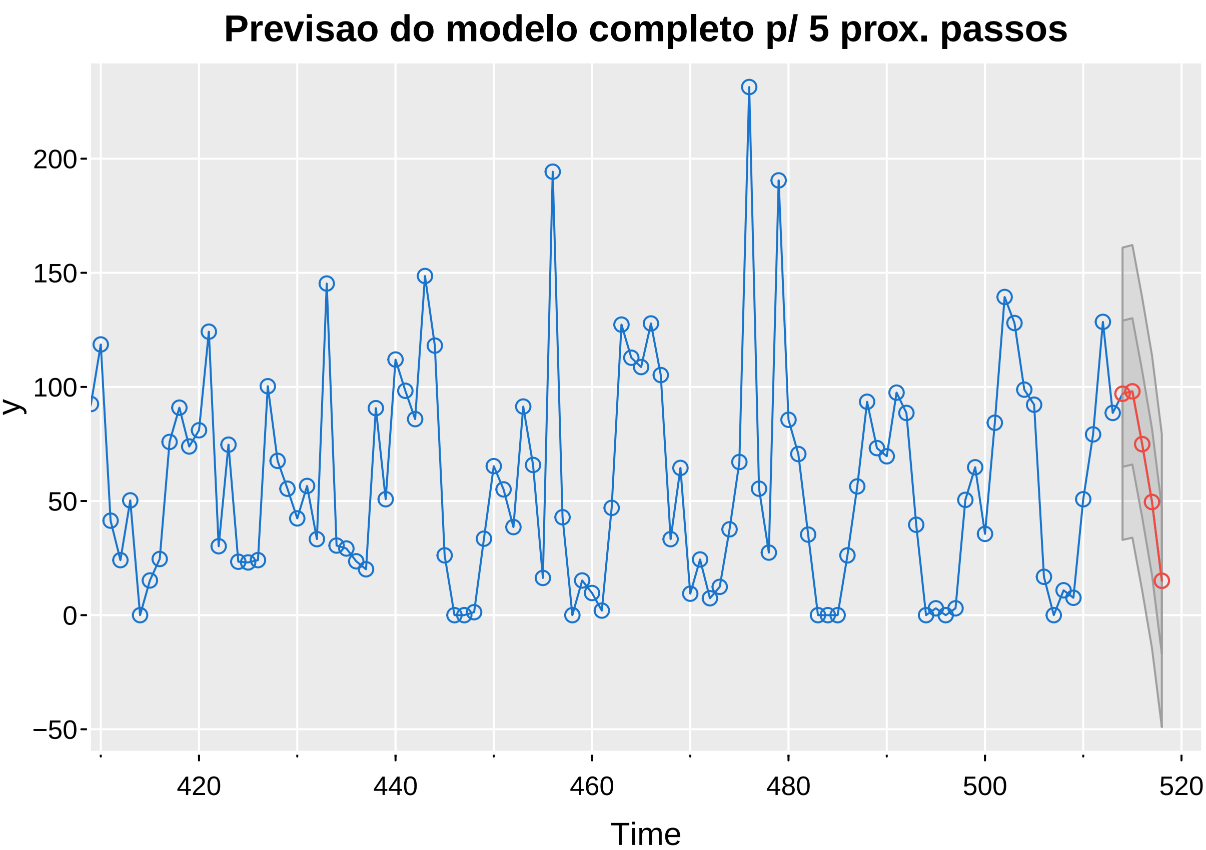
```r
shapiro.test(fit[["fit"]][["residuals"]])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.94604, p-value = 1.02e-12
```

Sob a hipótese nula de independência, o teste de Ljung-Box rejeita a hipótese nula. Portanto, não existem indícios de independência na série dos resíduos.

Avaliando as formas da FAC, FACP e resultado do teste de Ljung-Box, aparenta pouca ou nenhuma correlação nos resíduos, além da rejeição da normalidade pelo teste de Shapiro-Wilk, confirmando visualmente pelo Q-Q com uma forte fuga à normalidade. Como já foram feitos testes de diferenciação anteriormente, este é o melhor modelo que pode se obter com este método.

```r
prev = sarima.for(y,p = 2,P = 1,q = 2,Q = 1, d = 0, D = 0, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Previsao do modelo completo p/ 5 prox. passos")
```

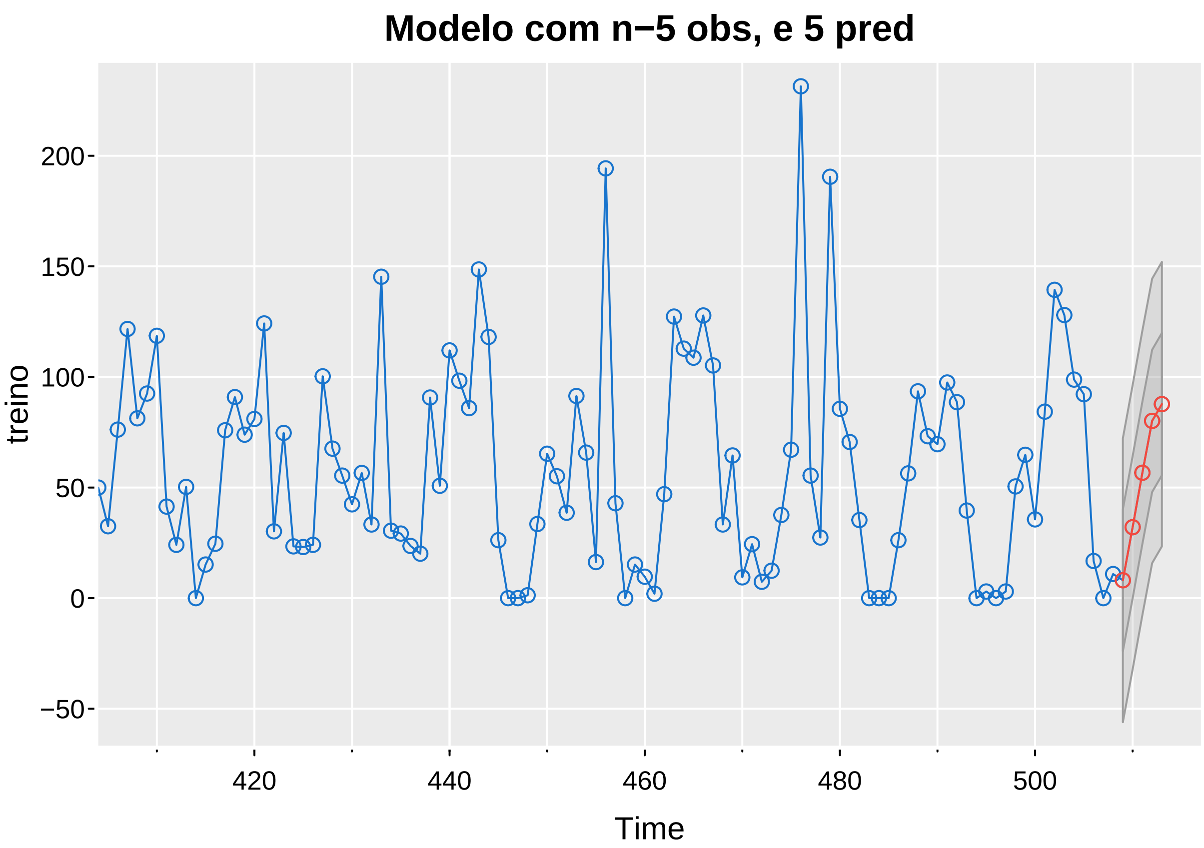


```r
prev
```

```
## $pred
## Time Series:
## Start = 514
## End = 518
## Frequency = 1
## [1] 97.00343 98.06032 74.93310 49.58118 15.07257
##
## $se
## Time Series:
## Start = 514
## End = 518
## Frequency = 1
## [1] 32.01955 32.05058 32.06908 32.07256 32.07371
```

```r
treino = ts(y[1:508])
teste = ts(y[509:513])
```

```r
prev = sarima.for(treino, p = 2, P = 1, q = 2,Q = 1, d = 0, D = 0, S = 12,
        n.ahead = 5, gg=TRUE, col=4,main="Modelo com n-5 obs, e 5 pred")
```




```r
prev$pred
```

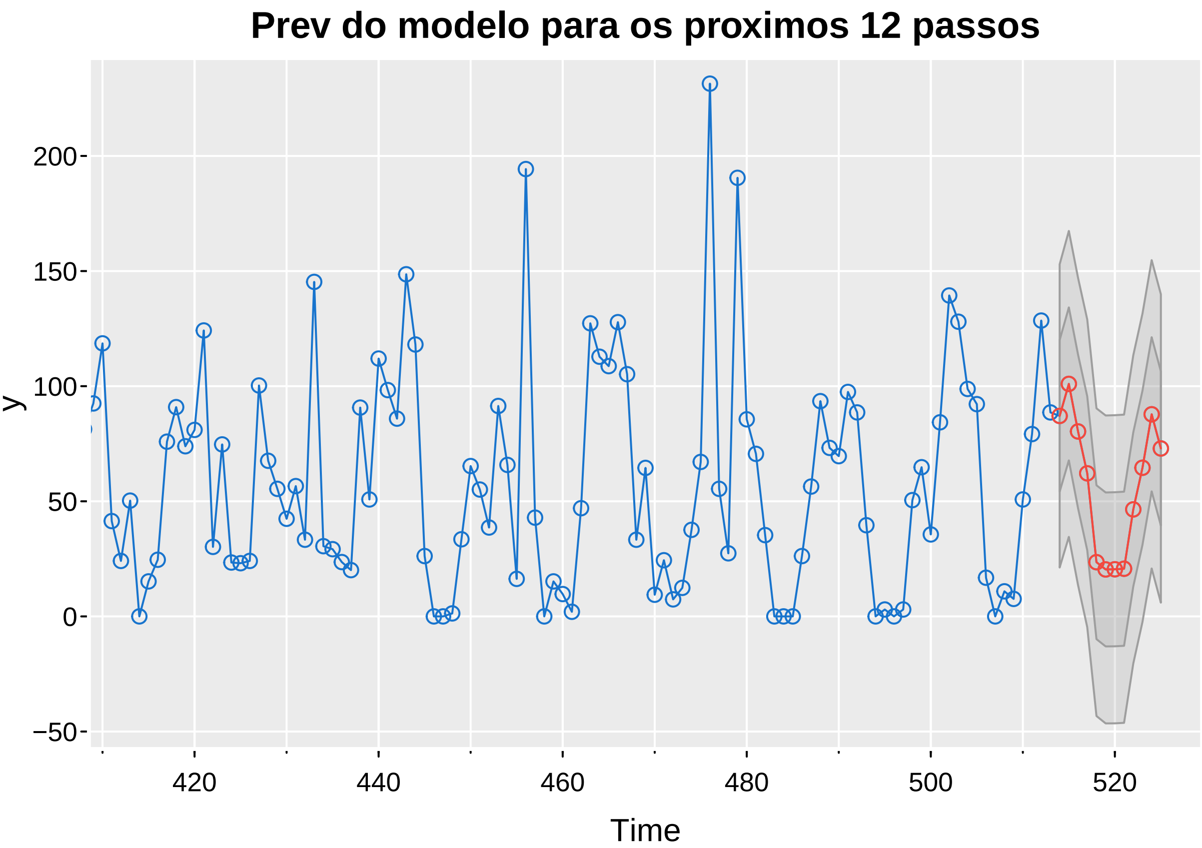
```
## Time Series:
## Start = 509
## End = 513
## Frequency = 1
## [1]  8.086115 32.079081 56.677676 80.134040 87.715625
```

```r
MAPE = MAPE(prev$pred, as.vector(teste))
MAPE
```

```
## [1] 0.2206455
```

Com um MAPE = 0.2206455, ou seja,entre 20 e 30%, podemos dizer que se trata de um bom modelo preditivo.

```r
prev = sarima.for(y, p = 2, P = 1, q = 3,Q = 1, d = 0, D = 0, S = 12,
          n.ahead = 12, gg=TRUE, col=4,main="Prev do modelo para os proximos 12 passos")
```



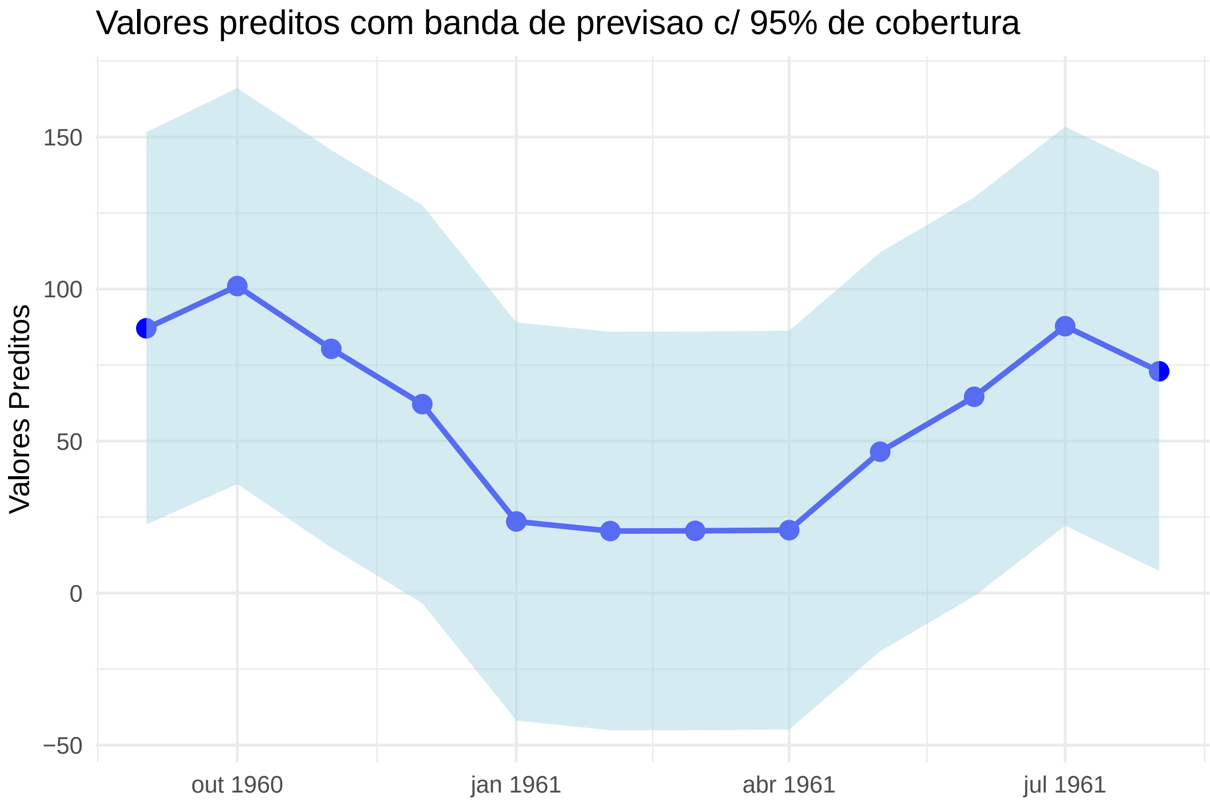

```r
datas <- seq.Date(from = as.Date("1960-09-01"), to = as.Date("1961-08-31"), by = "months")

previsoes = data.frame(datas,prev$pred,prev$se)
colnames(previsoes) = c("data","Valores_preditos","Erro_padrao")

previsoes = previsoes %>%
  mutate(Limite_superior = Valores_preditos + (1.96 * Erro_padrao),
         Limite_inferior = Valores_preditos - (1.96 * Erro_padrao))

ggplot(previsoes, aes(x = data)) +
  geom_line(aes(y = Valores_preditos), color = "blue", size = 1) +
  geom_point(aes(y = Valores_preditos), color = "blue", size = 3) +
  geom_ribbon(aes(ymin = Limite_inferior, ymax = Limite_superior), fill = "lightblue", alpha = 0.5)
  labs(title = "Valores preditos com banda de previsao c/ 95% de cobertura",
       x = "",
       y = "Valores Preditos") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



Grande parte dos pressupostos do modelo não foram atendidos. A fuga da normalidade é bastante acentuada, levando a banda de confiança não ser tão confiável assim. Ainda assim, o MAPE do modelo é relativamente baixo, e a estrutura visual da série comparada a suas previsões aparentam ser suaves. Logo, este pode ser um modelo útil para previsões, ainda que seus resultados devam ser observados com criticidade, visto que os pressupostos necessários não foram atendidos.

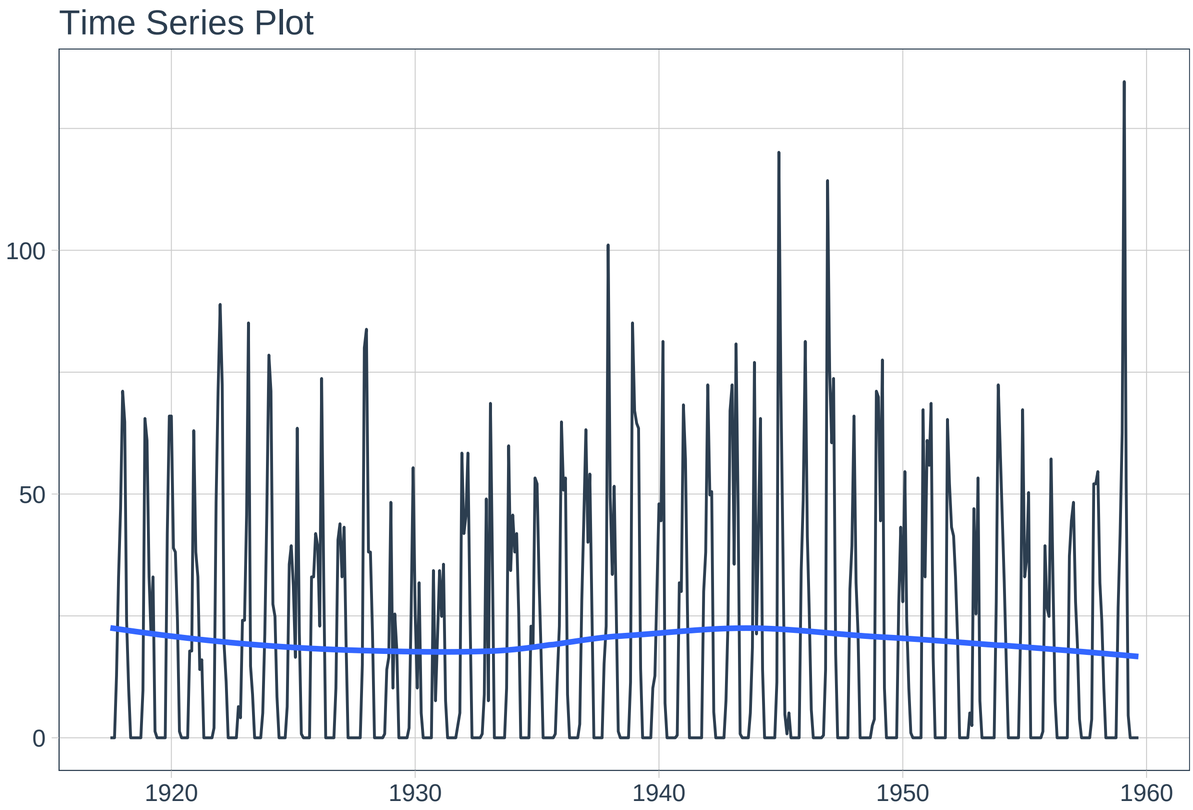
7. "Total Snow (cm)";

```
df7 = df[,c(5,20)]
df7 = df7[4:513,]
df7 = df7 %>% drop_na()
colnames(df7)[2]="total_snow"
```

```
df7$data = as.Date(df7$data)
df7 = tsibble::as_tsibble(df7,index=data,regular=F)
```

Visualizando a série com uma linha suavizada

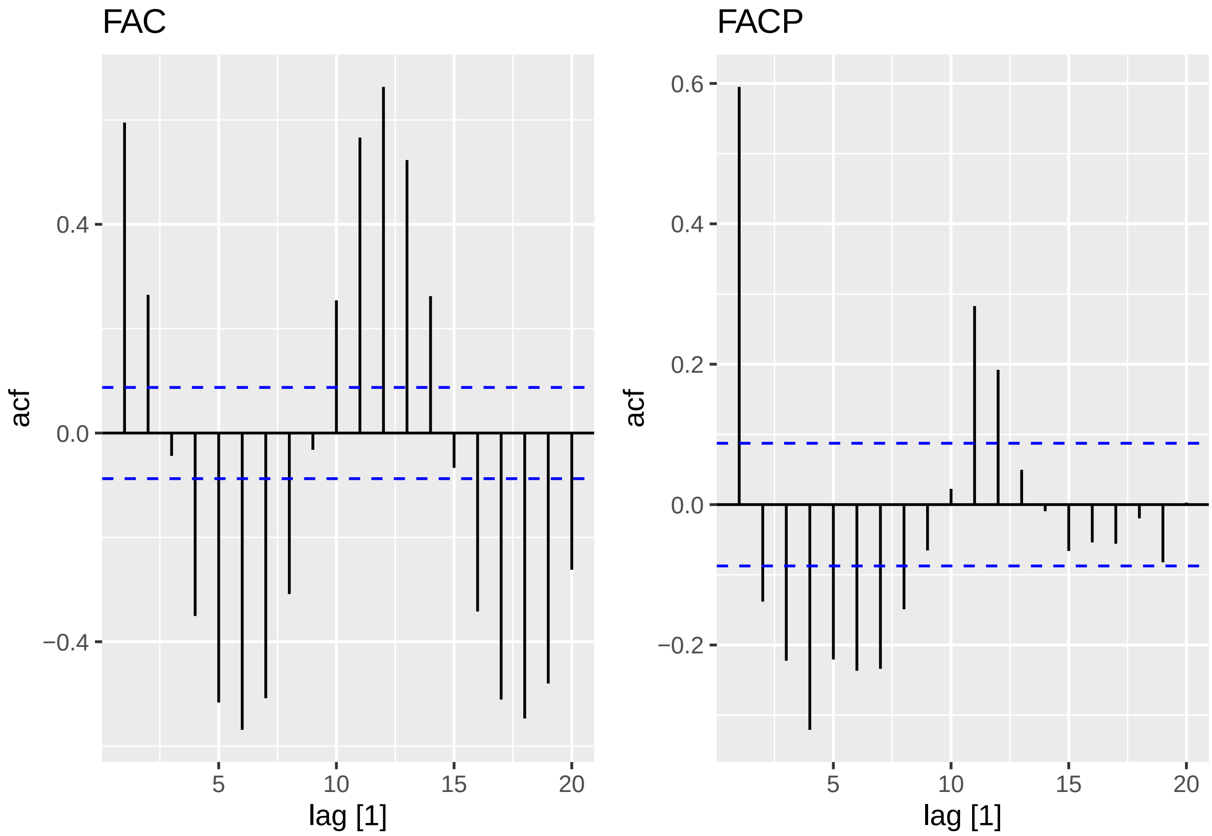
```
df7 %>%
  plot_time_series(data,total_snow, .interactive = F, .smooth = T)
```




```
FAC7 = df7 %>%
  ACF(var = total_snow,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC")

FACP7 = df7 %>%
  ACF(var = total_snow,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP")

grid.arrange(FAC7, FACP7, nrow = 1)
```




```r
aTSA::adf.test(ts(df7$total_snow), nlag = 20)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##        lag    ADF p.value
##  [1,]    0 -8.529  0.0100
##  [2,]    1 -8.272  0.0100
##  [3,]    2 -8.529  0.0100
##  [4,]    3 -9.139  0.0100
##  [5,]    4 -8.145  0.0100
##  [6,]    5 -7.144  0.0100
##  [7,]    6 -6.115  0.0100
##  [8,]    7 -4.699  0.0100
##  [9,]    8 -3.397  0.0100
## [10,]    9 -2.447  0.0157
## [11,]   10 -1.423  0.1702
## [12,]   11 -0.983  0.3275
## [13,]   12 -0.828  0.3827
## [14,]   13 -0.778  0.4008
## [15,]   14 -0.796  0.3942
## [16,]   15 -0.770  0.4035
## [17,]   16 -0.734  0.4164
## [18,]   17 -0.772  0.4028
## [19,]   18 -0.785  0.3983
## [20,]   19 -0.677  0.4367
## Type 2: with drift no trend
##        lag    ADF p.value
##  [1,]    0 -11.26    0.01
##  [2,]    1 -11.53    0.01
```

```
##   [3,]   2 -12.83    0.01
##   [4,]   3 -15.48    0.01
##   [5,]   4 -15.88    0.01
##   [6,]   5 -16.46    0.01
##   [7,]   6 -16.94    0.01
##   [8,]   7 -15.81    0.01
##   [9,]   8 -13.55    0.01
##  [10,]   9 -11.08    0.01
##  [11,]  10  -7.08    0.01
##  [12,]  11  -5.25    0.01
##  [13,]  12  -4.68    0.01
##  [14,]  13  -4.56    0.01
##  [15,]  14  -4.82    0.01
##  [16,]  15  -5.05    0.01
##  [17,]  16  -5.28    0.01
##  [18,]  17  -5.32    0.01
##  [19,]  18  -5.64    0.01
##  [20,]  19  -5.41    0.01
## Type 3: with drift and trend
##        lag    ADF p.value
##   [1,]   0 -11.24    0.01
##   [2,]   1 -11.52    0.01
##   [3,]   2 -12.82    0.01
##   [4,]   3 -15.46    0.01
##   [5,]   4 -15.86    0.01
##   [6,]   5 -16.44    0.01
##   [7,]   6 -16.93    0.01
##   [8,]   7 -15.79    0.01
##   [9,]   8 -13.54    0.01
##  [10,]   9 -11.07    0.01
##  [11,]  10  -7.06    0.01
##  [12,]  11  -5.24    0.01
##  [13,]  12  -4.66    0.01
##  [14,]  13  -4.54    0.01
##  [15,]  14  -4.80    0.01
##  [16,]  15  -5.03    0.01
##  [17,]  16  -5.27    0.01
##  [18,]  17  -5.30    0.01
##  [19,]  18  -5.62    0.01
##  [20,]  19  -5.40    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

O teste de Dickey-Fuller sugere a estacionariedade da série até um lag = 10.

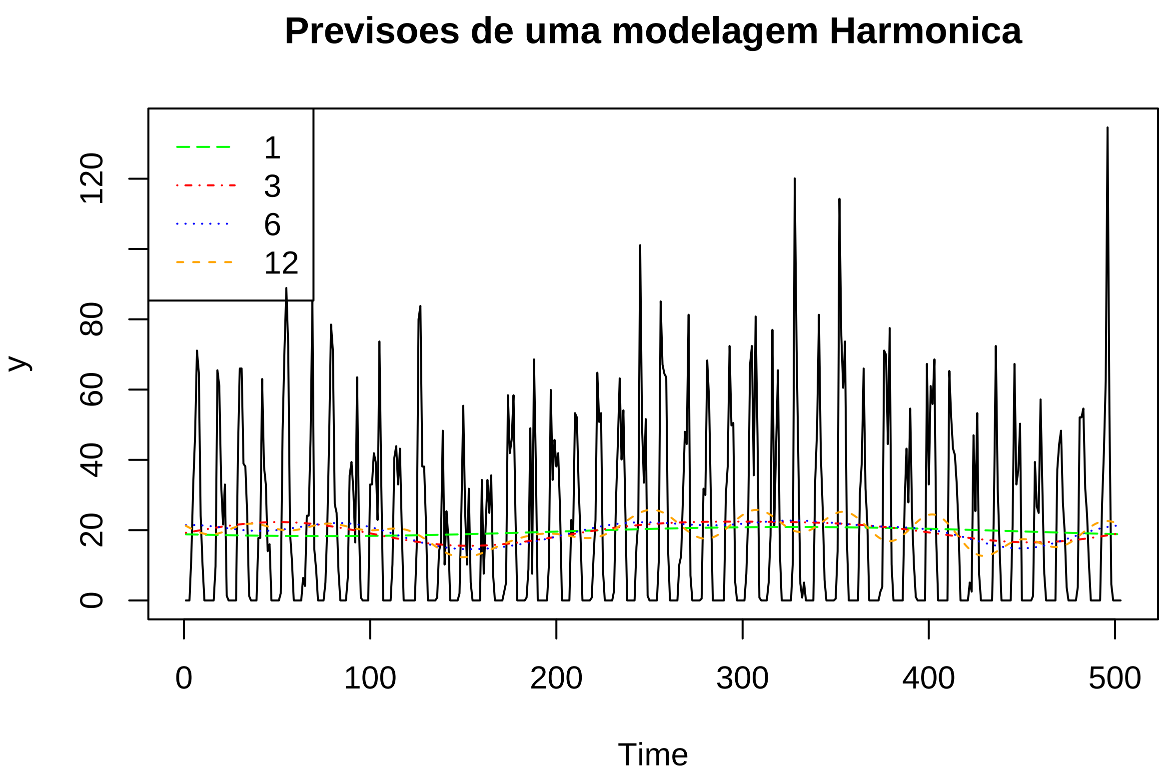
```r
y <- ts(df7$total_snow)

fit1 <- haRmonics(y = y,
                  numFreq = 1,
                  delta = 0.1)
fit3 <- haRmonics(y = y,
                   numFreq = 3,
                   delta = 0.1)
fit6 <- haRmonics(y = y,
                    numFreq = 6,
                    delta = 0.1)
fit12 <- haRmonics(y = y,
                    numFreq = 12,
```

```
                    delta = 0.1)

x = ts(fit1$fitted)
w = ts(fit3$fitted)
z = ts(fit6$fitted)
v = ts(fit12$fitted)

plot(y, pch = 16, main = "Previsoes de uma modelagem Harmonica")
lines(x ,lty = 5, col = "green")
lines(w, lty = 4, col = "red")
lines(z, lty = 3, col = "blue")
lines(v, lty = 2, col = "orange")
legend("topleft", legend = c("1", "3", "6","12"),
       col = c("green", "red", "blue","orange"), lty = c(5, 4, 3,2))
```



A modelagem harmônica foi satisfatória em capturar todas as estruturas da série, deixando apenas ruido branco.

Esta série parece ter sazonalidade semestral. Olhando os dados, a cada 6 meses o registro ou zera por 6 meses, ou é estritamente positivo por 6 meses, antes de zerar por 6 meses e assim por diante. Faz sentido quanto ao conjunto de dados: Se trata de precipitação de gelo, que deve ocorrer somente em duas estações do ano (Outono e Inverno?). Como os dados iniciam com 3 observações = 0, para depois ocorrer a semestralidade, resolvi remover as 3 primeiras observações, para evitar problemas na modelagem quanto ao reconhecimento do padrão semestral ante aos 3 primeiros dados (se fosse para chutar, os 3 meses anteriores aos primeiros dados também são = 0).

Visto que não foi testada a necessidade de diferenciação da série, farei a modelagem sarima com ordem de sazonalidade = 6, argumentos d = D = 0. Iniciei com grid de parâmetros p,P,q,Q,d,D no intervalo [0,2]; e ajustou-se o limite dos parâmetros até a convergência de um modelo ótimo sem parâmetros na fronteira.

```
modelos <- data.frame(p = integer(),
                      P = integer(),
                      q = integer(),
```

```r
                      Q = integer(),
                      d = integer(),
                      D = integer(),
                      AIC = numeric(),
                      BIC = numeric())

tic()
for(p in 0:6){
  for(P in 0:2){
      for(q in 0:6){
          for(Q in 0:2){
              for(d in 0:0){
                for(D in 0:0){
            tryCatch({
              fit = astsa::sarima(y, details = FALSE, Model = FALSE,
                          p = p, d = d, q = q, P = P, D = D, Q = Q, S = 6)
              AIC = fit$ICs[1]
              BIC = fit$ICs[3]
              mod = c(p, P, q, Q,d,D, AIC, BIC)
              modelos = rbind(modelos, mod)
            }, error = function(e) {
            })
        }}}}}}
toc()

colnames(modelos) = c("p","P","q","Q","d","D","AIC","BIC")

modelo = modelos %>%
  arrange(BIC, AIC) %>% head(1)
```

Logo, o melhor modelo (minimiza BIC) é o com os parâmetros:

```r
kable(modelo)
```

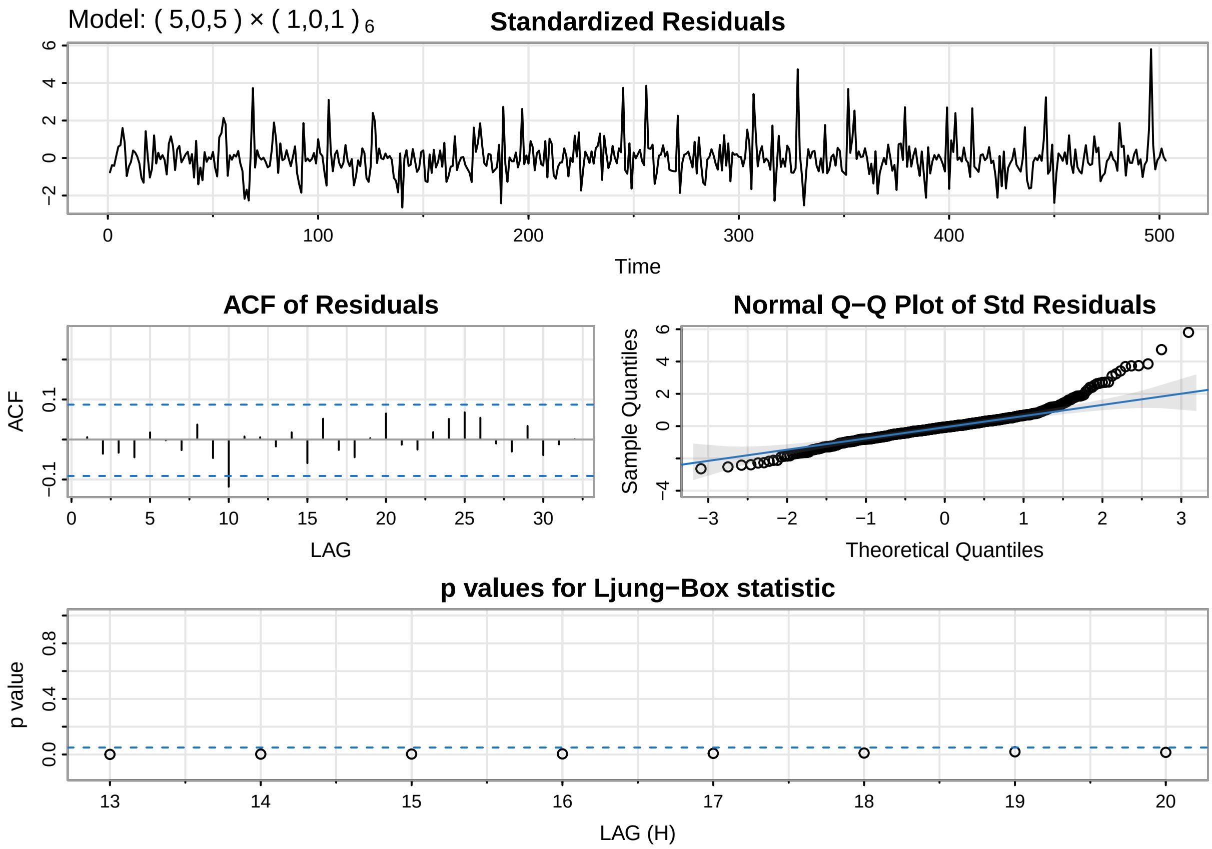
p	P	q	Q	d	D	AIC	BIC
5	1	5	1	0	0	8.299254	8.416726

```r
fit = astsa::sarima(y,p = 5,P = 1, q = 5,Q = 1,d = 0, D = 0, S = 6)
```

```
## initial  value 3.236838
## iter   2 value 2.963818
## iter   3 value 2.947299
## iter   4 value 2.910383
## iter   5 value 2.907158
## iter   6 value 2.898639
## iter   7 value 2.885020
## iter   8 value 2.869494
## iter   9 value 2.860713
## iter  10 value 2.848887
## iter  11 value 2.833590
## iter  12 value 2.820875
## iter  13 value 2.809486
## iter  14 value 2.794621
## iter  15 value 2.779764
## iter  16 value 2.774881
```

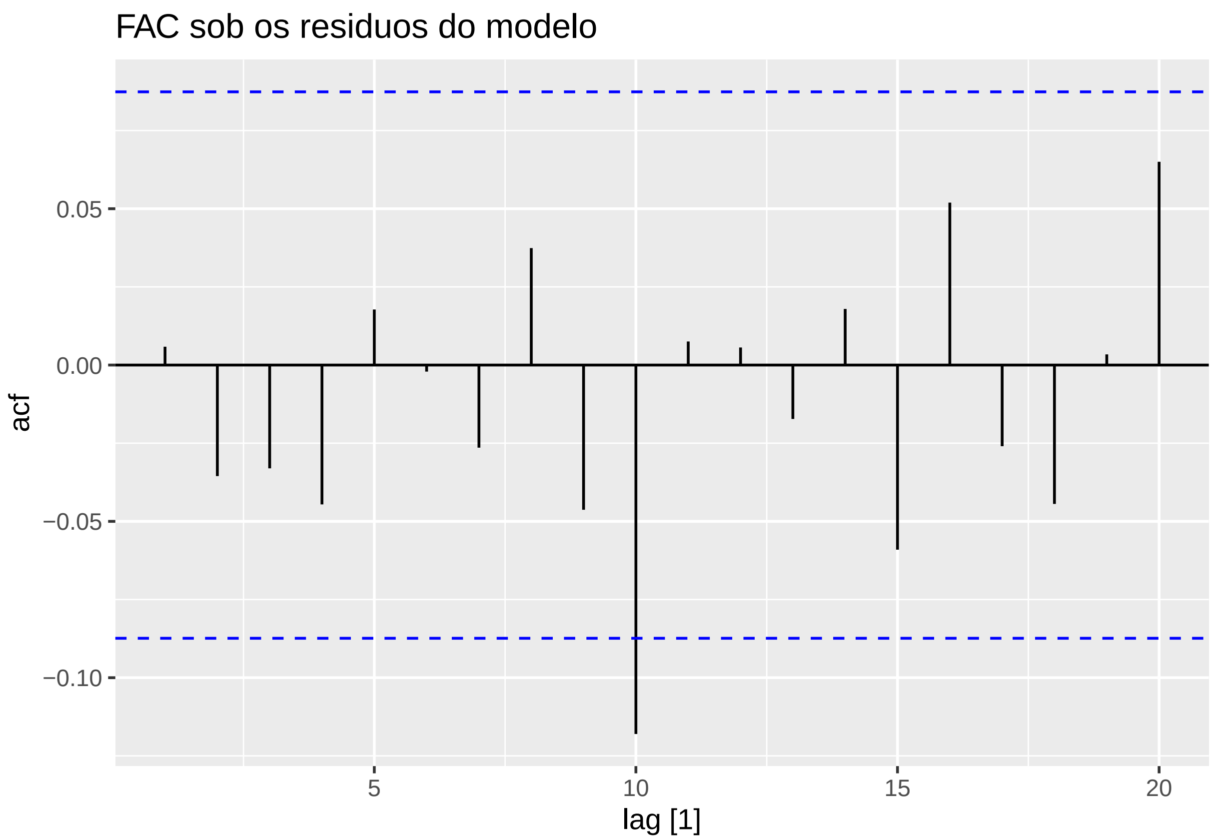
```
## iter  17 value 2.770168
## iter  18 value 2.768874
## iter  19 value 2.763205
## iter  20 value 2.759636
## iter  21 value 2.756665
## iter  22 value 2.749526
## iter  23 value 2.745024
## iter  24 value 2.738346
## iter  25 value 2.735268
## iter  26 value 2.731704
## iter  27 value 2.726167
## iter  28 value 2.724978
## iter  29 value 2.723312
## iter  30 value 2.721505
## iter  31 value 2.721077
## iter  32 value 2.720220
## iter  33 value 2.719479
## iter  34 value 2.718350
## iter  35 value 2.717279
## iter  36 value 2.716569
## iter  37 value 2.715790
## iter  38 value 2.714572
## iter  39 value 2.714192
## iter  40 value 2.713413
## iter  41 value 2.713385
## iter  42 value 2.713176
## iter  43 value 2.713041
## iter  44 value 2.712106
## iter  45 value 2.712077
## iter  46 value 2.712008
## iter  47 value 2.711973
## iter  48 value 2.711898
## iter  49 value 2.711823
## iter  50 value 2.711793
## iter  51 value 2.711694
## iter  52 value 2.711609
## iter  53 value 2.711590
## iter  54 value 2.711570
## iter  55 value 2.711562
## iter  56 value 2.711558
## iter  57 value 2.711558
## iter  58 value 2.711556
## iter  59 value 2.711556
## iter  60 value 2.711556
## iter  60 value 2.711556
## iter  60 value 2.711556
## final  value 2.711556
## converged
## initial  value 2.706978
## iter   2 value 2.706718
## iter   3 value 2.706241
## iter   4 value 2.705801
## iter   5 value 2.705484
## iter   6 value 2.705220
## iter   7 value 2.704842
## iter   8 value 2.704331
## iter   9 value 2.703851
## iter  10 value 2.703693
```

```
## iter  11 value 2.703593
## iter  12 value 2.703482
## iter  13 value 2.703283
## iter  14 value 2.702957
## iter  15 value 2.702875
## iter  16 value 2.702860
## iter  17 value 2.702858
## iter  18 value 2.702858
## iter  19 value 2.702857
## iter  20 value 2.702857
## iter  21 value 2.702857
## iter  22 value 2.702856
## iter  23 value 2.702856
## iter  24 value 2.702856
## iter  25 value 2.702856
## iter  26 value 2.702856
## iter  27 value 2.702856
## iter  28 value 2.702856
## iter  28 value 2.702856
## final  value 2.702856
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE  t.value p.value
## ar1     0.1716 0.0235   7.3029  0.0000
## ar2     0.1556 0.0259   5.9998  0.0000
## ar3     0.1866 0.0234   7.9655  0.0000
## ar4     0.0314 0.0252   1.2450  0.2137
## ar5    -0.9009 0.0231 -39.0090  0.0000
## ma1     0.0302 0.0226   1.3357  0.1823
## ma2    -0.1580 0.0221  -7.1439  0.0000
## ma3    -0.2229 0.0227  -9.8095  0.0000
## ma4    -0.1131 0.0248  -4.5556  0.0000
## ma5     0.9084 0.0250  36.3892  0.0000
## sar1    0.9302 0.0328  28.3894  0.0000
## sma1   -0.7896 0.0538 -14.6701  0.0000
## xmean  19.9878 1.9071  10.4805  0.0000
##
## sigma^2 estimated as 217.1407 on 490 degrees of freedom
##
## AIC = 8.299254  AICc = 8.300734  BIC = 8.416726
##
```

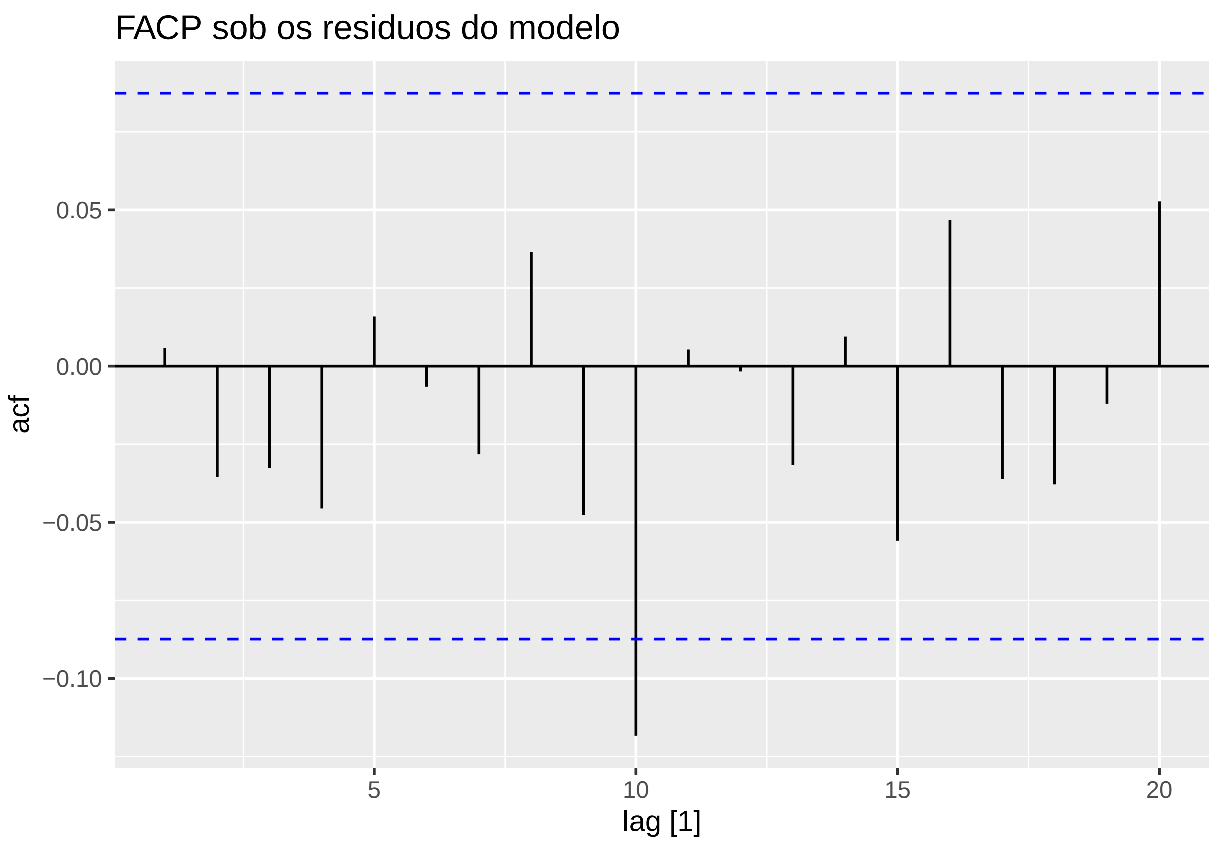



```
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```r
res %>%
  ACF(value,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```



```
Box.test(fit[["fit"]][["residuals"]], lag = 20, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 18.944, df = 20, p-value = 0.5255
```

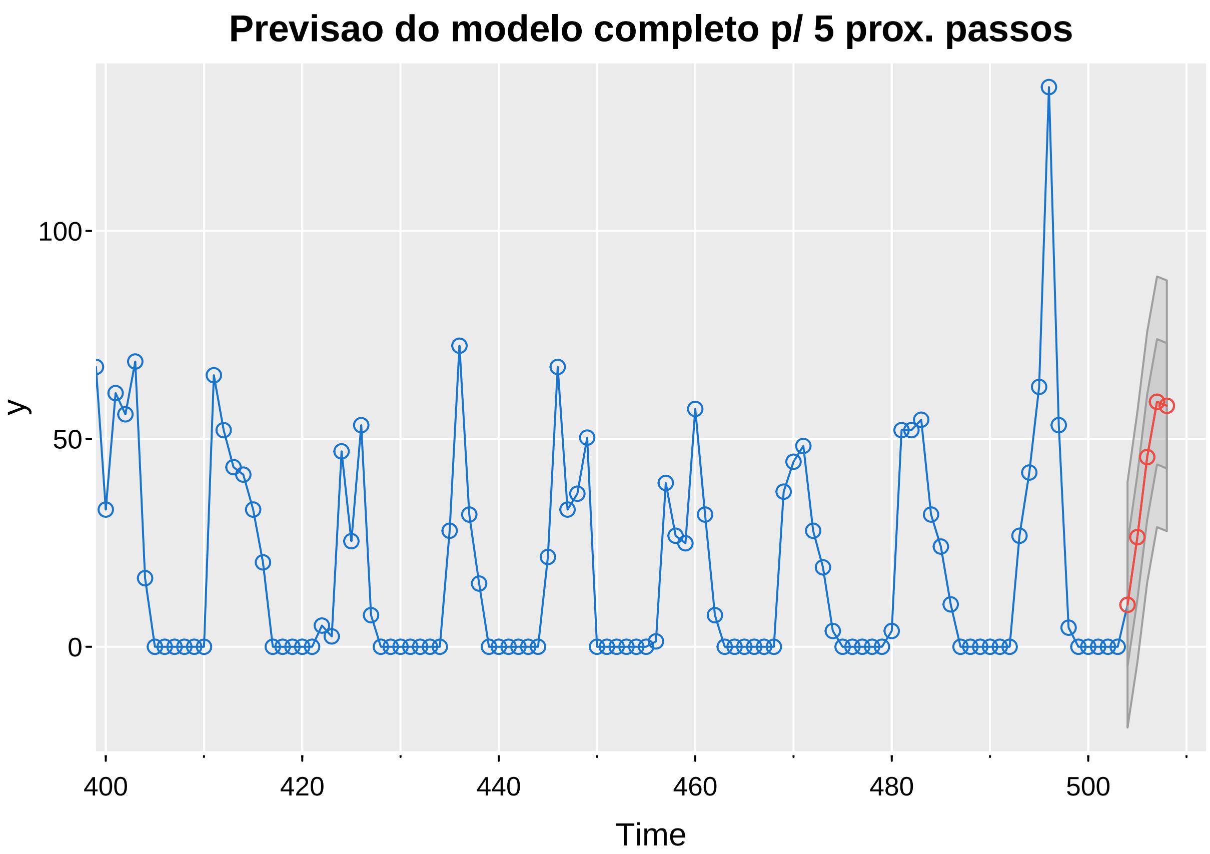
```
shapiro.test(fit[["fit"]][["residuals"]])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.91598, p-value = 4.216e-16
```

Sob a hipótese nula de independência, o teste de Ljung-Box rejeita não a hipótese nula. Portanto, existem indícios de independência na série dos resíduos.

Avaliando as formas da FAC, FACP e resultado do teste de Ljung-Box, não aparenta haver correlação nos resíduos, entretanto a rejeição da normalidade pelo teste de Shapiro-Wilk e gráfico Q-Q foi bastante acentuada. Como já foram feitos testes de diferenciação anteriormente, este é o melhor modelo que pode se obter com este método.

```
prev = sarima.for(y,p = 5,P = 1,q = 5,Q = 1, d = 0, D = 0, S = 6,
           n.ahead = 5, gg=TRUE, col=4,main="Previsao do modelo completo p/ 5 prox. passos")
```

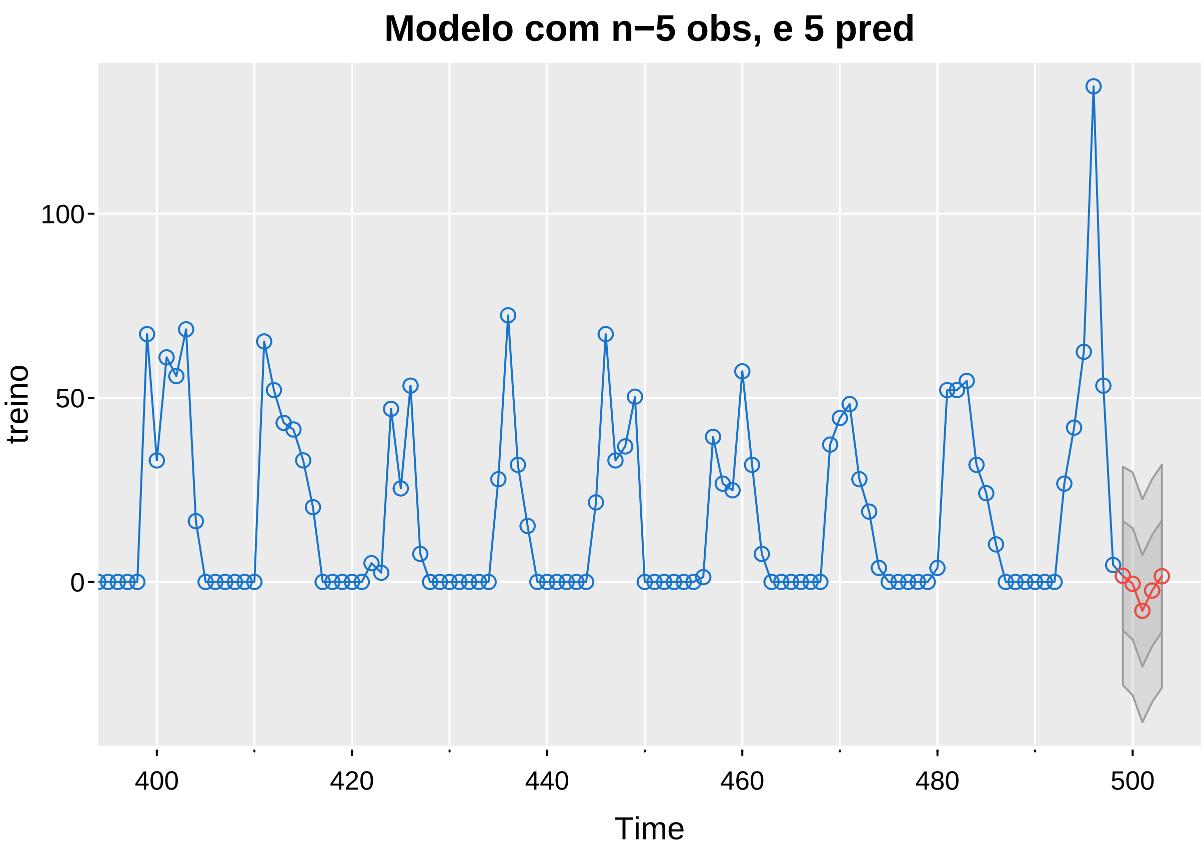


```r
prev
```

```
## $pred
## Time Series:
## Start = 504
## End = 508
## Frequency = 1
## [1] 10.09655 26.37653 45.62412 58.91603 57.95886
##
## $se
## Time Series:
## Start = 504
## End = 508
## Frequency = 1
## [1] 14.76400 15.05772 15.06385 15.06389 15.07407
```

```r
treino = ts(y[1:498])
teste = ts(y[499:503])
```

```r
prev = sarima.for(treino, p = 5, P = 1, q = 5,Q = 1, d = 0, D = 0, S = 6,
          n.ahead = 5, gg=TRUE, col=4,main="Modelo com n-5 obs, e 5 pred")
```




```r
prev$pred
```

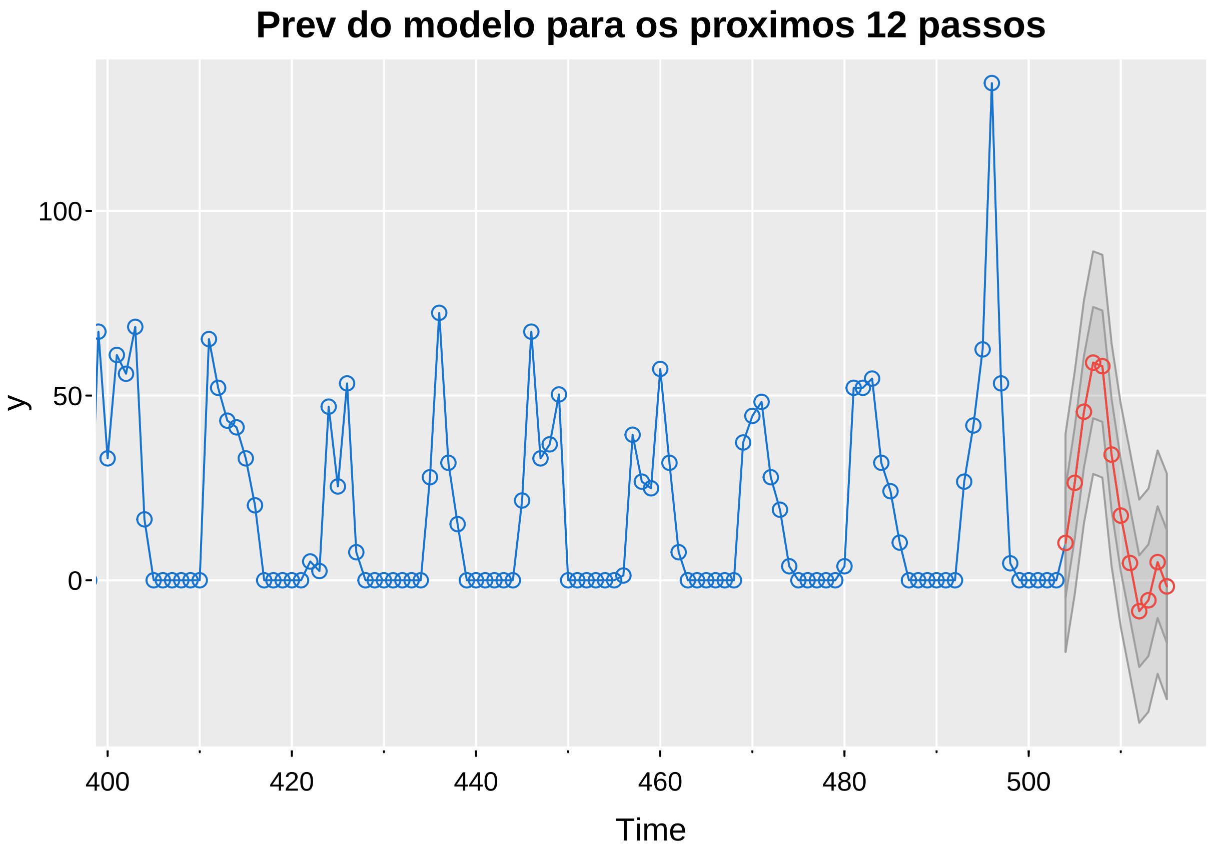
```
## Time Series:
## Start = 499
## End = 503
## Frequency = 1
## [1]  1.6465012 -0.4971798 -7.8313957 -2.3405954  1.5394575
```

```r
MAPE = MAPE(prev$pred, as.vector(teste))
MAPE
```

```
## [1] Inf
```

O MAPE retorna infinito, pois as previsões retornam valores negativos, o que não faz sentido com os dados. Estamos trabalhando com precipitação de neve, portanto o menor valor possível é zero. As previsões até capturam a estrutura da série, mas ultrapassando a barreira do zero, o que torna esses valores sem sentido.

```r
prev = sarima.for(y, p = 5, P = 1, q = 5,Q = 1, d = 0, D = 0, S = 6,
           n.ahead = 12, gg=TRUE, col=4,main="Prev do modelo para os proximos 12 passos")
```



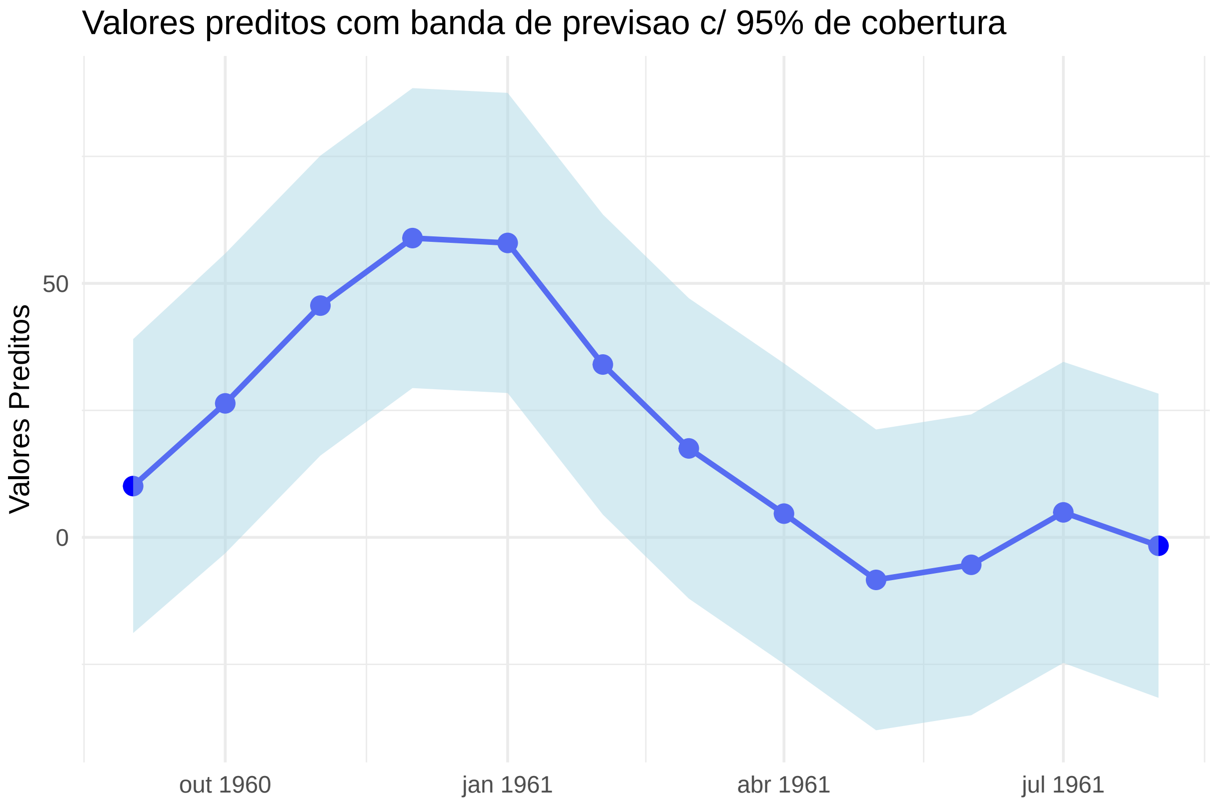

```r
datas <- seq.Date(from = as.Date("1960-09-01"), to = as.Date("1961-08-31"), by = "months")

previsoes = data.frame(datas,prev$pred,prev$se)
colnames(previsoes) = c("data","Valores_preditos","Erro_padrao")

previsoes = previsoes %>%
  mutate(Limite_superior = Valores_preditos + (1.96 * Erro_padrao),
         Limite_inferior = Valores_preditos - (1.96 * Erro_padrao))

ggplot(previsoes, aes(x = data)) +
  geom_line(aes(y = Valores_preditos), color = "blue", size = 1) +
  geom_point(aes(y = Valores_preditos), color = "blue", size = 3) +
  geom_ribbon(aes(ymin = Limite_inferior, ymax = Limite_superior), fill = "lightblue", alpha = 0.5)
  labs(title = "Valores preditos com banda de previsao c/ 95% de cobertura",
       x = "",
       y = "Valores Preditos") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



Este modelo deve ser explicado com bastante rigor. Os valores preditos não fazem sentido, visto que a precipitação de neve só poderia ocorrer no intervalo (0,+inf]; não fazendo sentido se falar em precipitação negativa. Este modelo faz previsões negativas, o que não tem interpretação na realidade. Se arredondarmos para zero todos os valores negativos, este modelo começa a fazer algum sentido, visto que captura a estrutura sazonal da série e para os meses de precipitação, aparenta fazer algum sentido os valores previstos. A normalidade dos resíduos utilizada para a banda de credibilidade também aparentam não a seguir por conta da inflação de zeros justamente, mas talvez tenham alguma serventia para os meses com precipitação positiva. Parece ser um modelo "melhor que nada", com alguma credibilidade para os níveis de precipitação nos meses que se espera precipitação. No entando, um outro modelo com distribuição de probabilidade com mais suporte no zero, e imagem no intervalo (0,+inf] (Poisson; Binomial negativa, Qui-quadrado...) Podem fazer mais sentido que este modelo com suposições de normalidade.

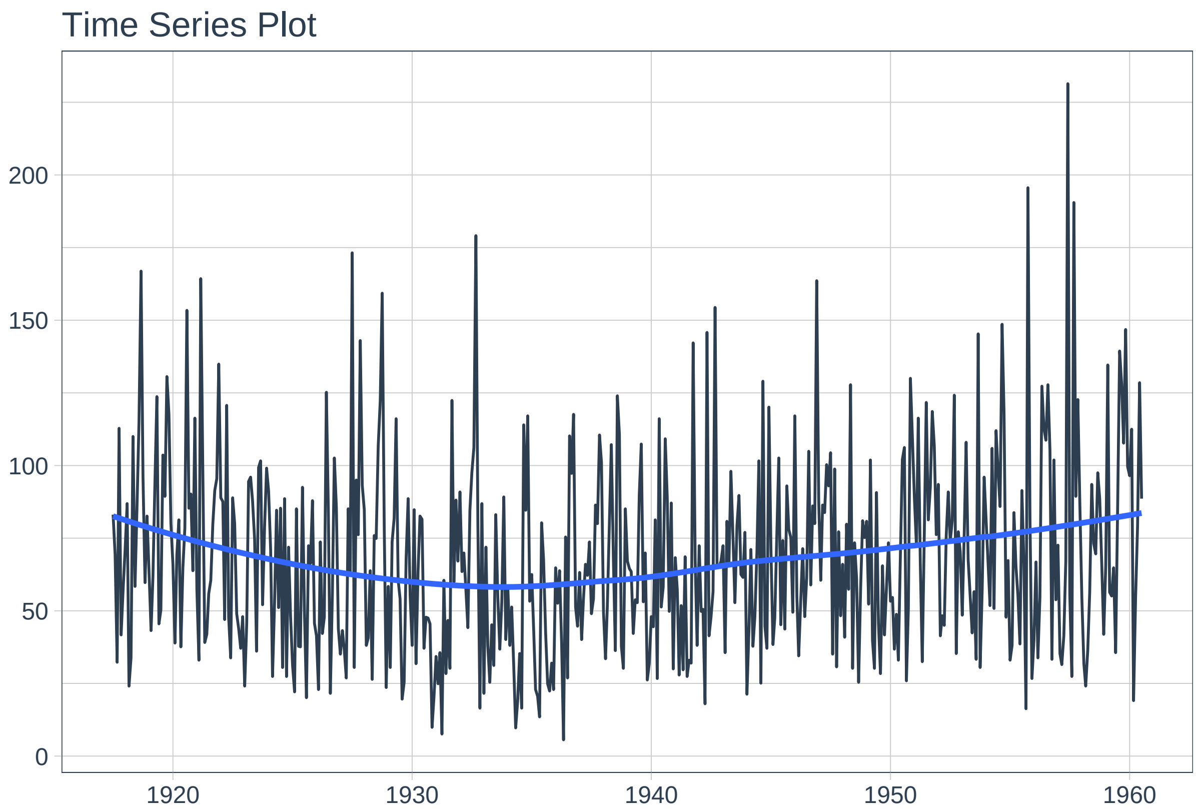
8. "Total Precip (mm)".

```
df8 = df[,c(5,22)]
df8 = df8 %>% drop_na()
colnames(df8)[2]="total_precip"
```

```
df8$data = as.Date(df8$data)
df8 = tsibble::as_tsibble(df8,index=data,regular=F)
```

Visualizando a série com uma linha suavizada

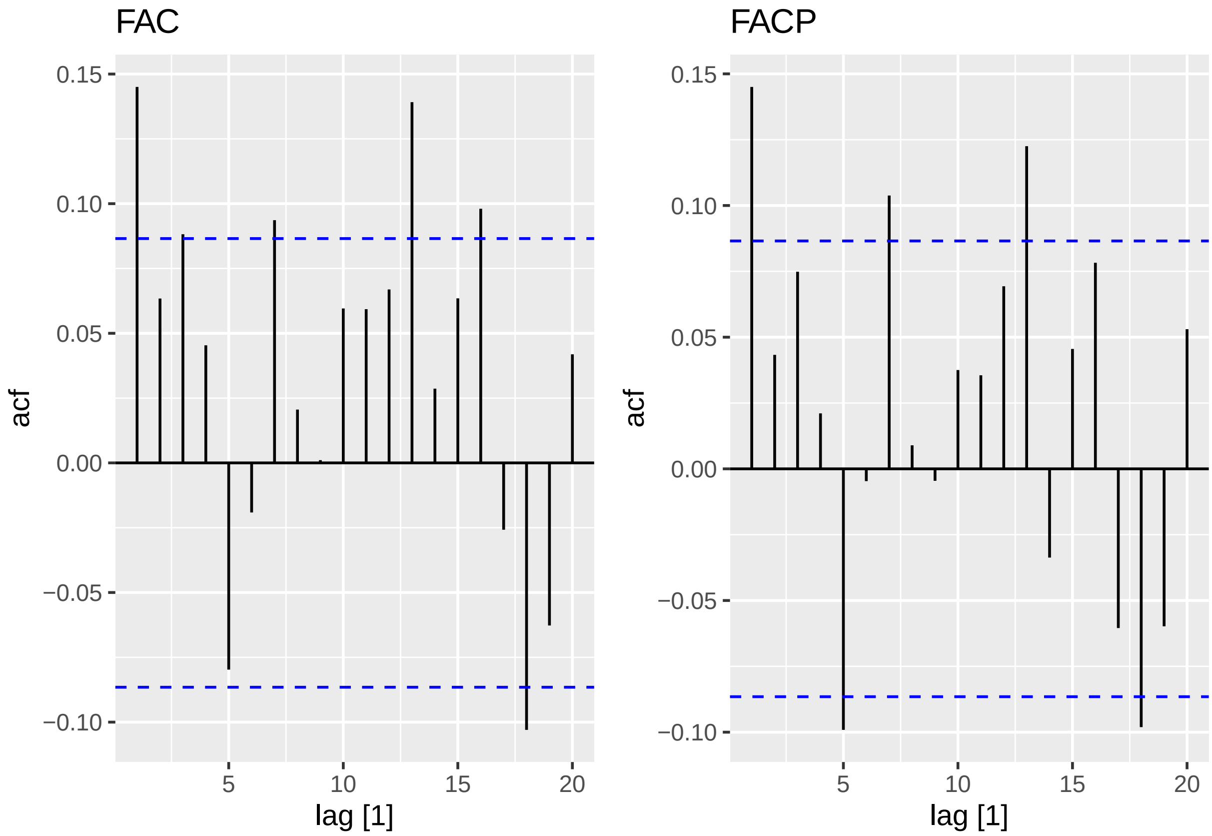
```
df8 %>%
  plot_time_series(data,total_precip, .interactive = F, .smooth = T)
```




```
FAC8 = df8 %>%
  ACF(var = total_precip,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC")

FACP8 = df8 %>%
  ACF(var = total_precip,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP")

grid.arrange(FAC8, FACP8, nrow = 1)
```




```
aTSA::adf.test(ts(df8$total_precip), nlag = 15)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##        lag    ADF p.value
##   [1,]   0 -6.834  0.0100
##   [2,]   1 -4.212  0.0100
##   [3,]   2 -2.914  0.0100
##   [4,]   3 -2.400  0.0177
##   [5,]   4 -2.199  0.0278
##   [6,]   5 -1.860  0.0636
##   [7,]   6 -1.442  0.1637
##   [8,]   7 -1.285  0.2196
##   [9,]   8 -1.097  0.2868
##  [10,]   9 -0.896  0.3585
##  [11,]  10 -0.801  0.3926
##  [12,]  11 -0.607  0.4617
##  [13,]  12 -0.425  0.5216
##  [14,]  13 -0.467  0.5094
##  [15,]  14 -0.505  0.4982
## Type 2: with drift no trend
##        lag    ADF p.value
##   [1,]   0 -19.51    0.01
##   [2,]   1 -14.06    0.01
##   [3,]   2 -11.02    0.01
##   [4,]   3  -9.66    0.01
##   [5,]   4  -9.80    0.01
##   [6,]   5  -8.99    0.01
##   [7,]   6  -7.46    0.01
```

```
## [8,]   7 -6.97    0.01
##  [9,]   8 -6.65    0.01
## [10,]   9 -6.09    0.01
## [11,]  10 -5.57    0.01
## [12,]  11 -4.93    0.01
## [13,]  12 -4.09    0.01
## [14,]  13 -4.15    0.01
## [15,]  14 -3.89    0.01
## Type 3: with drift and trend
##        lag    ADF p.value
##  [1,]   0 -19.60    0.01
##  [2,]   1 -14.17    0.01
##  [3,]   2 -11.11    0.01
##  [4,]   3  -9.78    0.01
##  [5,]   4  -9.92    0.01
##  [6,]   5  -9.13    0.01
##  [7,]   6  -7.60    0.01
##  [8,]   7  -7.12    0.01
##  [9,]   8  -6.78    0.01
## [10,]   9  -6.22    0.01
## [11,]  10  -5.72    0.01
## [12,]  11  -5.07    0.01
## [13,]  12  -4.25    0.01
## [14,]  13  -4.33    0.01
## [15,]  14  -4.12    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

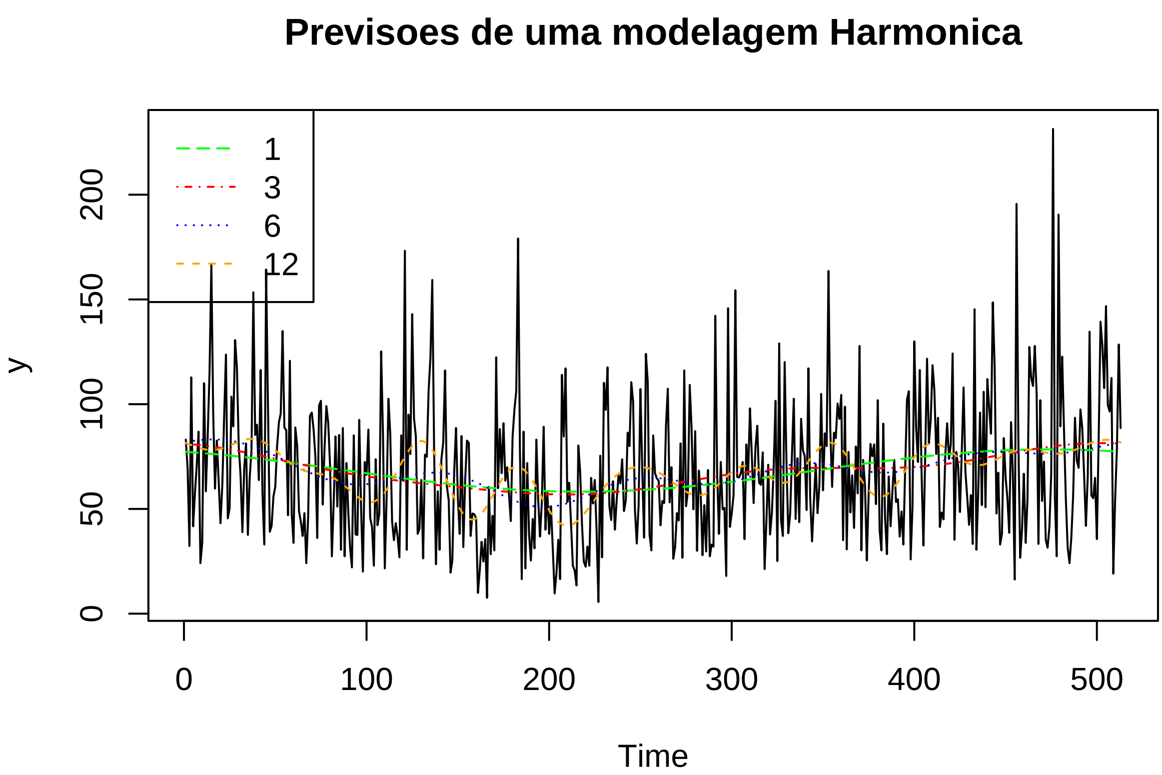
O teste de Dickey-Fuller sugere a estacionariedade da série até lag = 5.

```r
y <- ts(df8$total_precip)

fit1 <- haRmonics(y = y,
                  numFreq = 1,
                  delta = 0.1)
fit3 <- haRmonics(y = y,
                   numFreq = 3,
                   delta = 0.1)
fit6 <- haRmonics(y = y,
                    numFreq = 6,
                    delta = 0.1)
fit12 <- haRmonics(y = y,
                    numFreq = 12,
                    delta = 0.1)

x = ts(fit1$fitted)
w = ts(fit3$fitted)
z = ts(fit6$fitted)
v = ts(fit12$fitted)

plot(y, pch = 16, main = "Previsoes de uma modelagem Harmonica")
lines(x ,lty = 5, col = "green")
lines(w, lty = 4, col = "red")
lines(z, lty = 3, col = "blue")
lines(v, lty = 2, col = "orange")
legend("topleft", legend = c("1", "3", "6","12"),
       col = c("green", "red", "blue","orange"), lty = c(5, 4, 3,2))
```



A modelagem harmônica foi satisfatória em capturar todas as estruturas da série, deixando apenas ruido branco.

Visto que não foi testada a necessidade de diferenciação da série, farei a modelagem sarima com ordem de sazonalidade = 12, argumentos d = D = 0. Iniciei com grid de parâmetros p,P,q,Q,d,D no intervalo [0,2]; e ajustou-se o limite dos parâmetros até a convergência de um modelo ótimo sem parâmetros na fronteira.

```
modelos <- data.frame(p = integer(),
                      P = integer(),
                      q = integer(),
                      Q = integer(),
                      d = integer(),
                      D = integer(),
                      AIC = numeric(),
                      BIC = numeric())

tic()
for(p in 0:2){
  for(P in 0:2){
      for(q in 0:2){
          for(Q in 0:2){
              for(d in 0:2){
                for(D in 0:2){
            tryCatch({
              fit = astsa::sarima(y, details = FALSE, Model = FALSE,
                                  p = p, d = d, q = q, P = P, D = D, Q = Q, S = 12)
              AIC = fit$ICs[1]
              BIC = fit$ICs[3]
              mod = c(p, P, q, Q,d,D, AIC, BIC)
              modelos = rbind(modelos, mod)
            }, error = function(e) {
            })
        }}}}}}
```

```r
toc()

colnames(modelos) = c("p","P","q","Q","d","D","AIC","BIC")

modelo = modelos %>%
  arrange(BIC, AIC) %>% head(1)
```

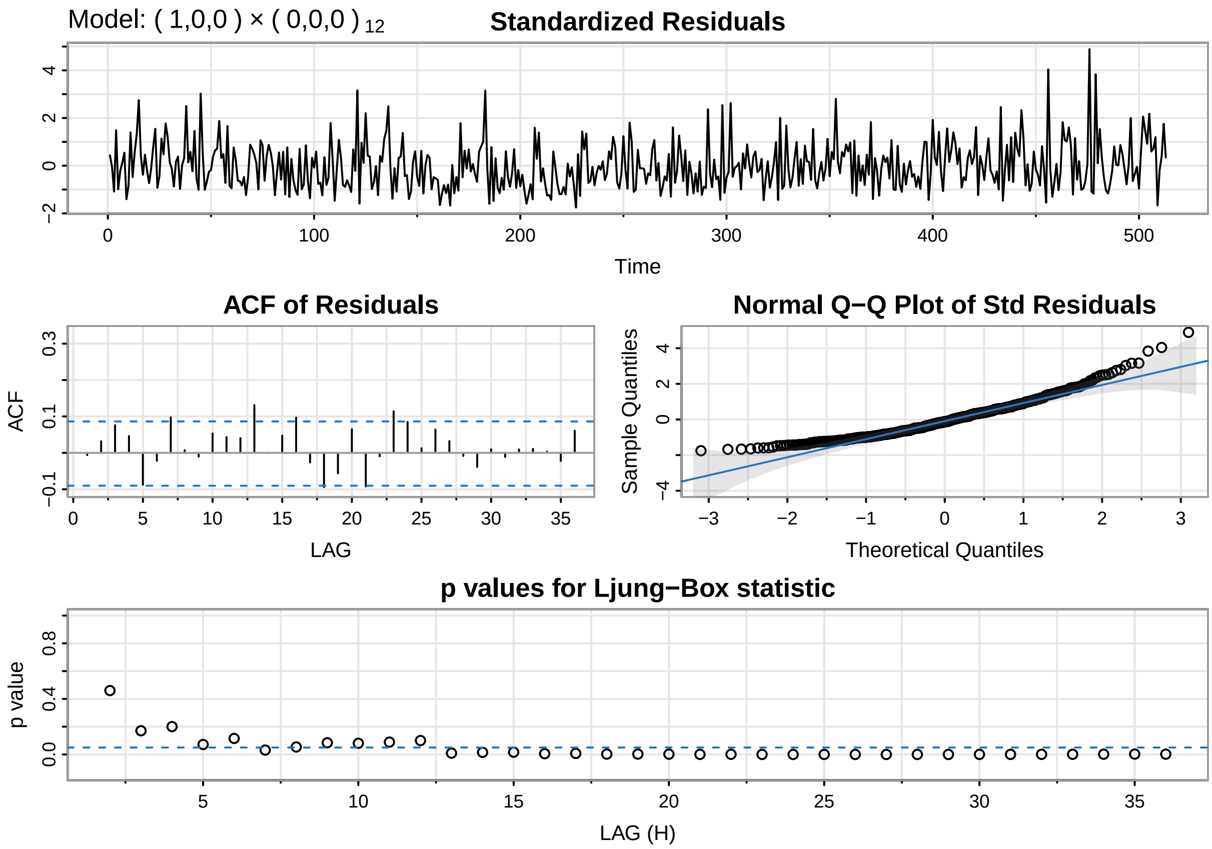
Logo, o melhor modelo (minimiza BIC) é o com os parâmetros:

```r
kable(modelo)
```

p	P	q	Q	d	D	AIC	BIC
1	0	0	0	0	0	9.864735	9.889532

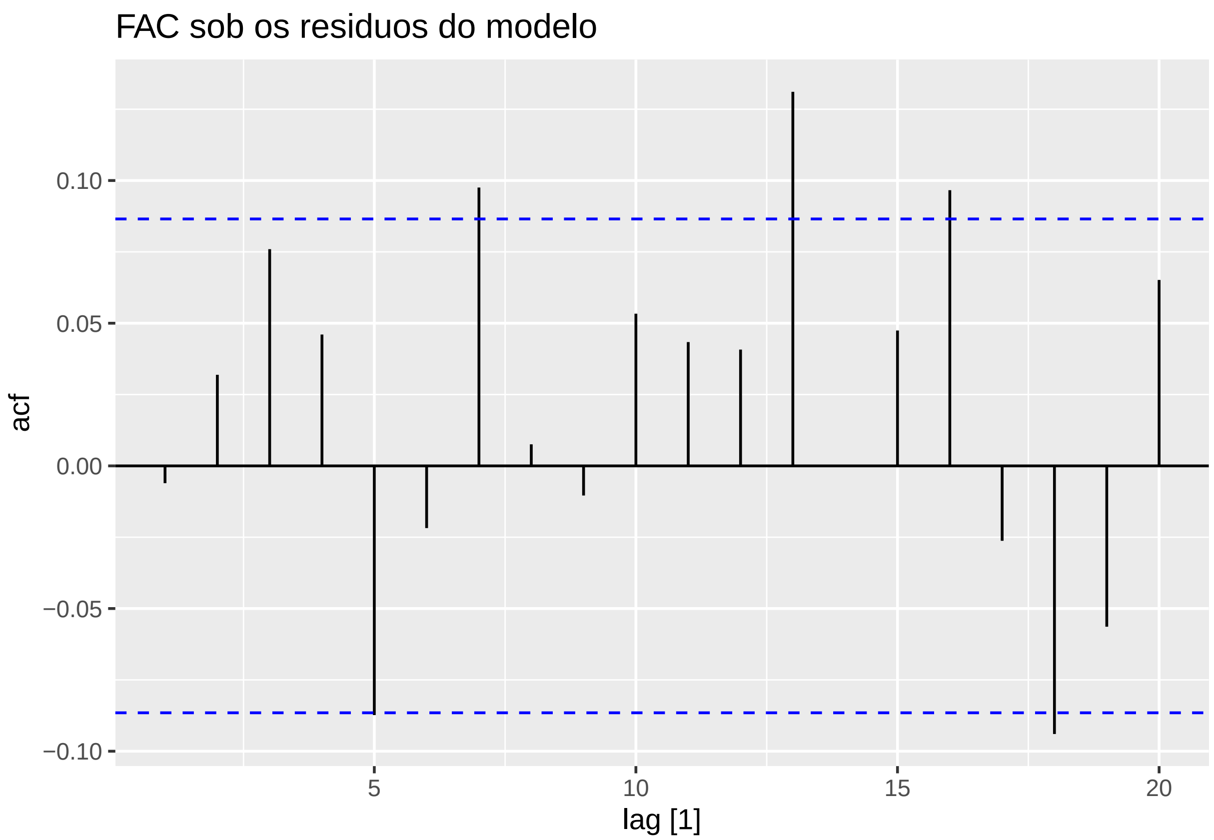
```r
fit = astsa::sarima(y,p = 1,P = 0, q = 0,Q = 0,d = 0, D = 0, S = 12)
```

```
## initial  value 3.518990
## iter   2 value 3.508349
## iter   2 value 3.508349
## iter   2 value 3.508349
## final  value 3.508349
## converged
## initial  value 3.507581
## iter   2 value 3.507581
## iter   3 value 3.507581
## iter   4 value 3.507581
## iter   4 value 3.507581
## iter   4 value 3.507581
## final  value 3.507581
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##       Estimate     SE t.value p.value
## ar1     0.1449 0.0437  3.3187   0.001
## xmean  68.3385 1.7222 39.6806   0.000
##
## sigma^2 estimated as 1113.341 on 511 degrees of freedom
##
## AIC = 9.864735  AICc = 9.864781  BIC = 9.889532
##
```

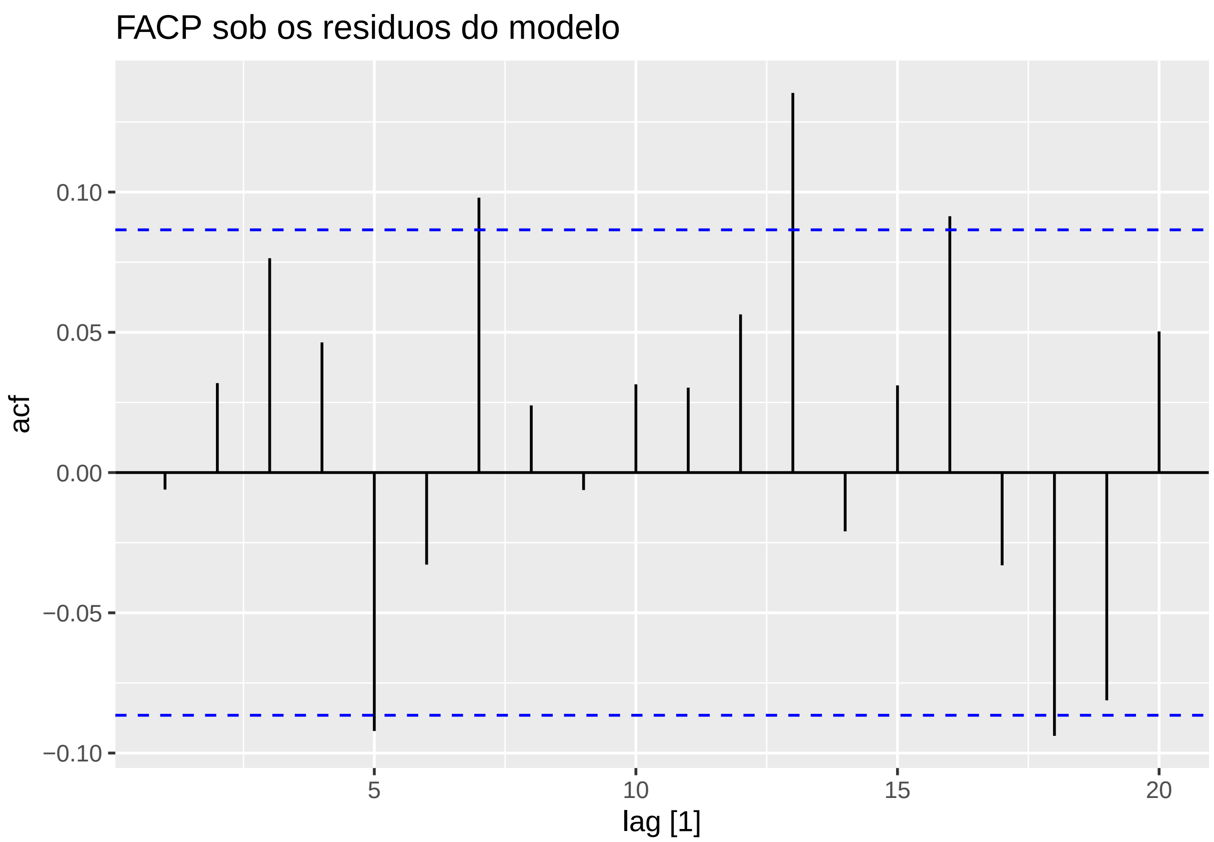



```r
res = as_tsibble(fit[["fit"]][["residuals"]])

res %>%
  ACF(value,lag_max = 20) %>%
  autoplot() +
  labs(title="FAC sob os residuos do modelo")
```




```r
res %>%
  ACF(value,lag_max = 20,type = "partial") %>%
  autoplot() +
  labs(title="FACP sob os residuos do modelo")
```



```r
Box.test(fit[["fit"]][["residuals"]], lag = 20, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  fit[["fit"]][["residuals"]]
## X-squared = 41.553, df = 20, p-value = 0.003161
```

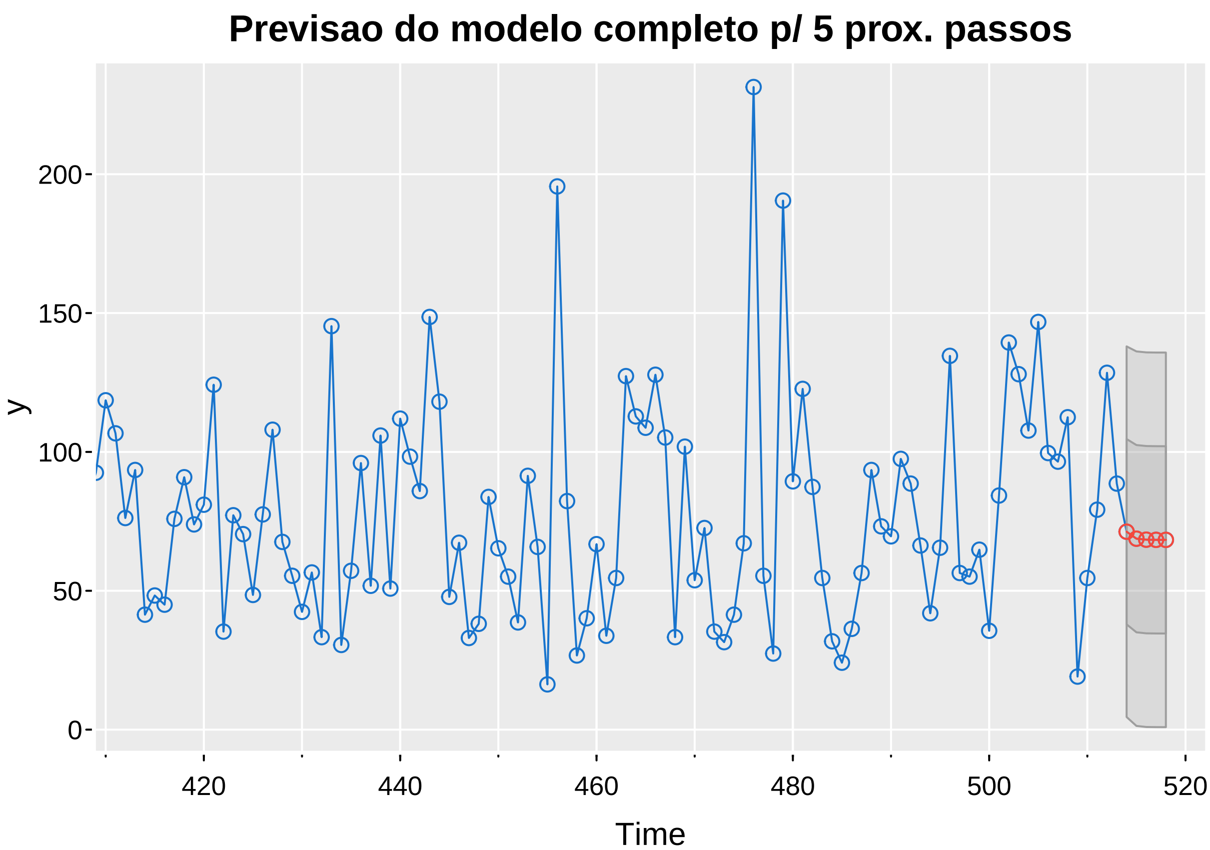
```r
shapiro.test(fit[["fit"]][["residuals"]])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  fit[["fit"]][["residuals"]]
## W = 0.94773, p-value = 1.756e-12
```

Sob a hipótese nula de independência, o teste de Ljung-Box rejeita a hipótese nula. Portanto, não existem fortes indícios de independência na série dos resíduos.

Avaliando as formas da FAC, FACP e resultado do teste de Ljung-Box, aparenta haver alguma correlação nos resíduos, além da forte rejeição da normalidade pelo teste de Shapiro-Wilk e gráfico Q-Q. Como já foram feitos testes de diferenciação anteriormente, este é o melhor modelo que pode se obter com este método.

```r
prev = sarima.for(y,p = 1,P = 0,q = 0,Q = 0, d = 0, D = 0, S = 12,
           n.ahead = 5, gg=TRUE, col=4,main="Previsao do modelo completo p/ 5 prox. passos")
```

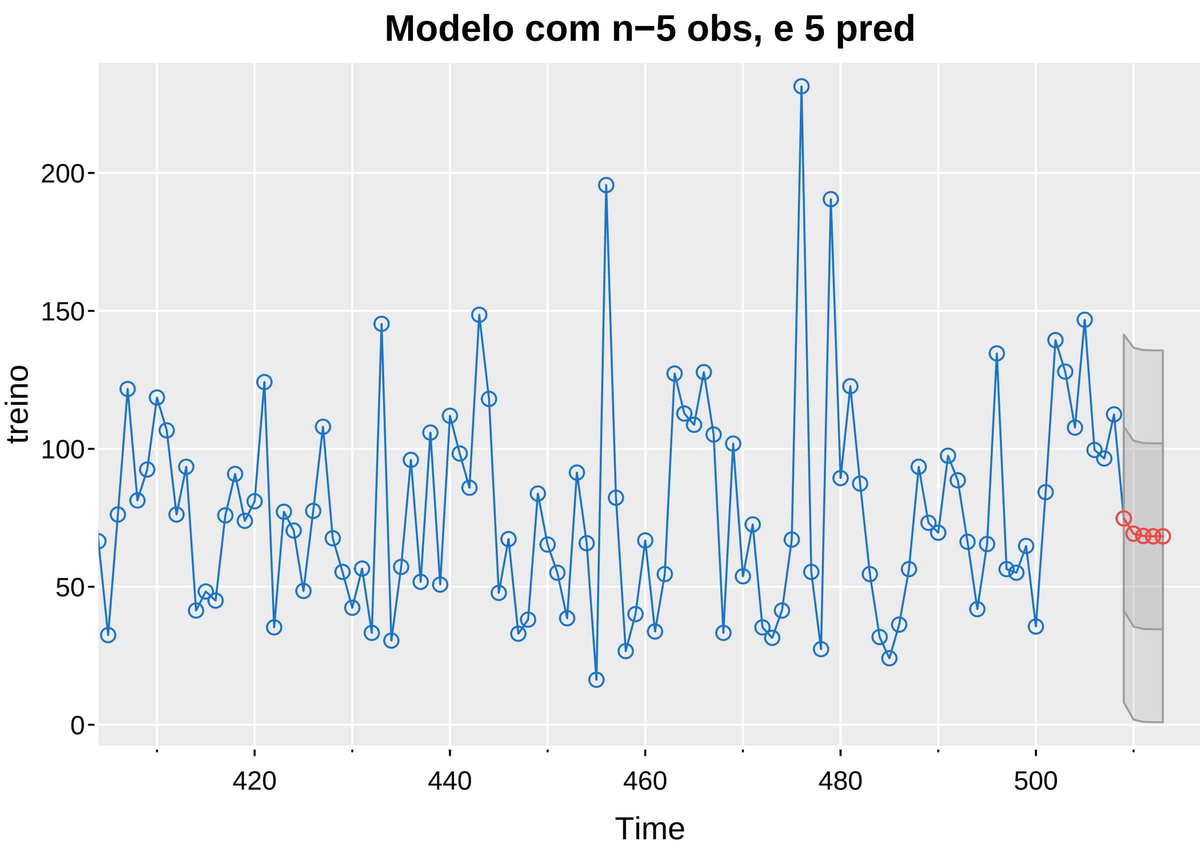


```
prev
```

```
## $pred
## Time Series:
## Start = 514
## End = 518
## Frequency = 1
## [1] 71.27399 68.76376 68.40008 68.34738 68.33975
##
## $se
## Time Series:
## Start = 514
## End = 518
## Frequency = 1
## [1] 33.36677 33.71515 33.72242 33.72258 33.72258
```

```
treino = ts(y[1:508])
teste = ts(y[509:513])
```

```
prev = sarima.for(treino, p = 1, P = 0, q = 0,Q = 0, d = 0, D = 0, S = 12,
          n.ahead = 5, gg=TRUE, col=4,main="Modelo com n-5 obs, e 5 pred")
```




```
prev$pred
```

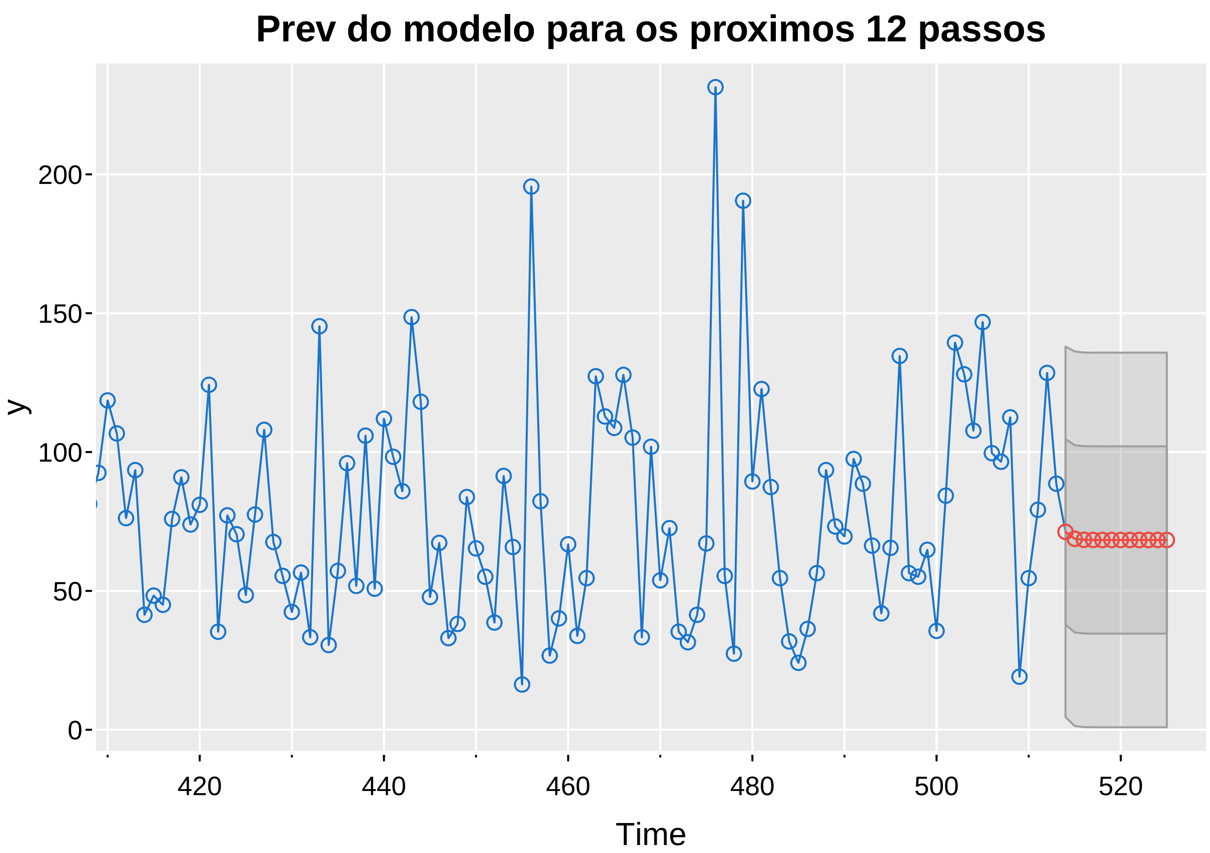
```
## Time Series:
## Start = 509
## End = 513
## Frequency = 1
## [1] 74.77160 69.24121 68.43055 68.31172 68.29430
```

```r
MAPE = MAPE(prev$pred, as.vector(teste))
MAPE
```

```
## [1] 0.8032901
```

Com um MAPE = 0.8032901, ou seja, acima de 80%, este é um modelo péssimo para previsões.

```r
prev = sarima.for(y, p = 1, P = 0, q = 0,Q = 0, d = 0, D = 0, S = 12,
        n.ahead = 12, gg=TRUE, col=4,main="Prev do modelo para os proximos 12 passos")
```



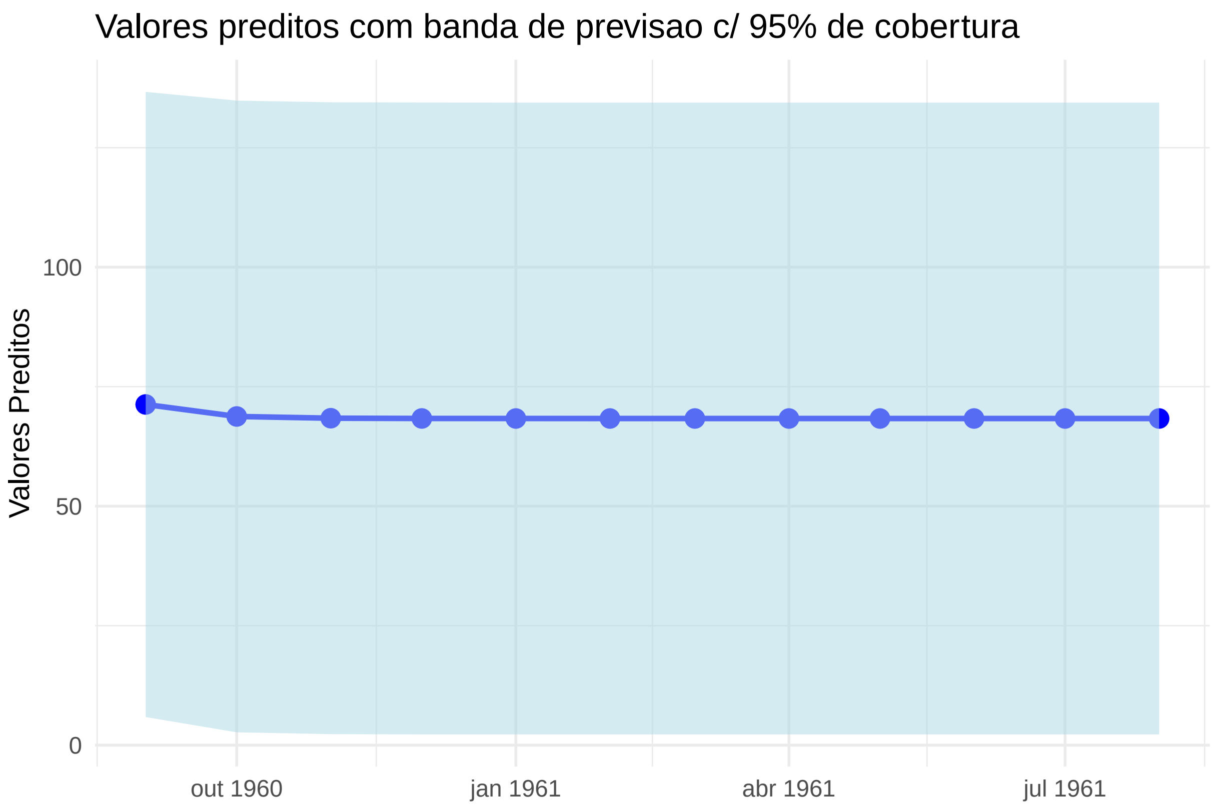

```r
datas <- seq.Date(from = as.Date("1960-09-01"), to = as.Date("1961-08-31"), by = "months")

previsoes = data.frame(datas,prev$pred,prev$se)
colnames(previsoes) = c("data","Valores_preditos","Erro_padrao")

previsoes = previsoes %>%
  mutate(Limite_superior = Valores_preditos + (1.96 * Erro_padrao),
         Limite_inferior = Valores_preditos - (1.96 * Erro_padrao))

ggplot(previsoes, aes(x = data)) +
  geom_line(aes(y = Valores_preditos), color = "blue", size = 1) +
  geom_point(aes(y = Valores_preditos), color = "blue", size = 3) +
  geom_ribbon(aes(ymin = Limite_inferior, ymax = Limite_superior), fill = "lightblue", alpha = 0.5)
  labs(title = "Valores preditos com banda de previsao c/ 95% de cobertura",
       x = "",
       y = "Valores Preditos") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



Este modelo não consegue diferenciar a série de um ruído branco, e só gera previsões em torno de uma média, utilizando de uma banda de credibilidade sem atendimento da normalidade dos resíduos. Em outras palavras, é um péssimo modelo para previsões, e que pode ser substituido por uma média da série para prever o próximo valor sem grandes diferenças. Portanto, necessita de um diagnóstico mais profunda, possivelmente utilizando outra técnica de modelagem de séries temporais para ajustar previsões coerentes com a realidade dos dados.