Lista 5 - Series

Davi Wentrick Feijó

2024-06-03

Considere a serie do consumo mensal de energia eletrica (ConsumoEnergiaEAgua_New.xlsx). Denotando X_t como o valor do consumo registrado no mes t e D_t como o número de dias de leitura, faca o que se pede a seguir.

```r
# Leitura dos dados
df <- read_excel("ConsumoEnergiaEAgua.xlsx") %>%
  dplyr::select(-c(2,5:11))

# Renomear colunas
colnames(df) <- c('Ano', 'Energia', 'Dias', 'NA')
df <- df[, c('Ano', 'Energia', 'Dias')]  # Selecionar apenas as colunas de interesse
```

1. Calcule o consumo médio diário $Y_t = \frac{x_t}{D_t}$ e explique o porquê dessa transformacao.

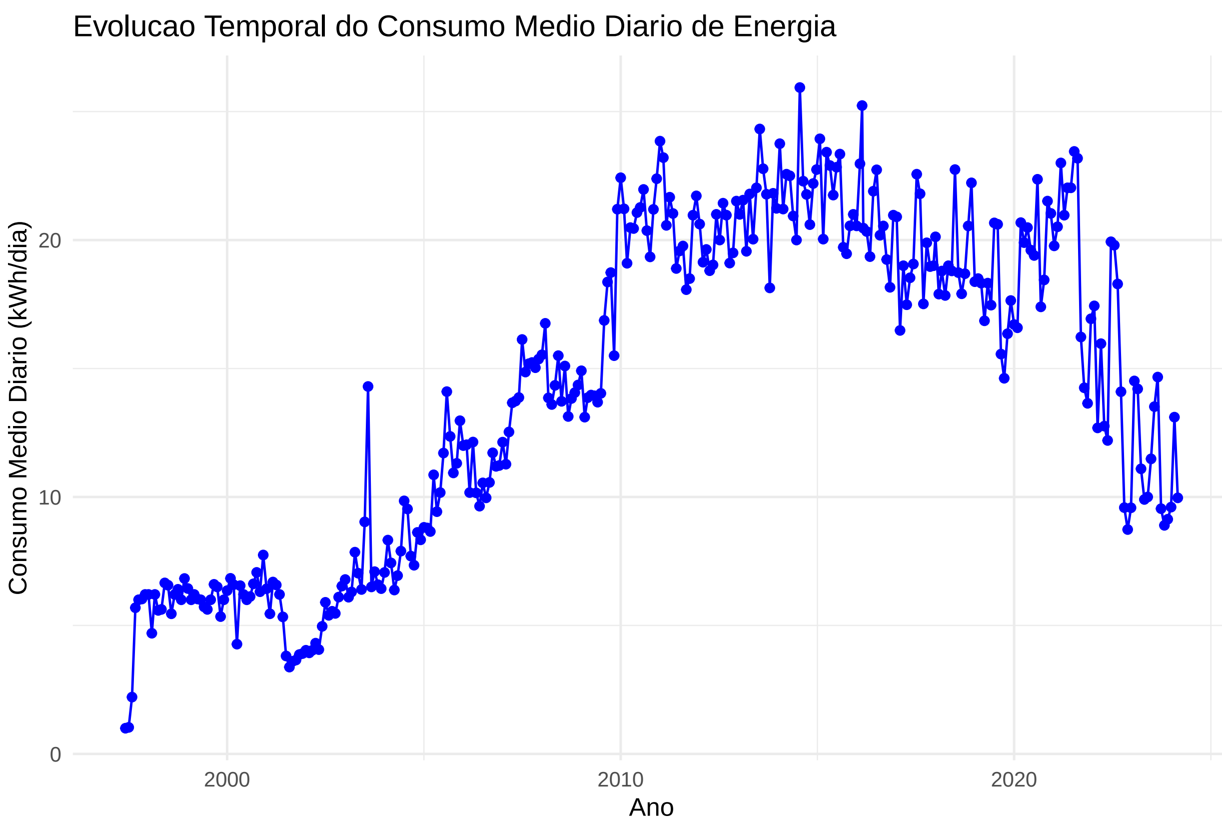
```r
# Converter a coluna 'Ano' para o tipo Date
df$Ano <- as.Date(df$Ano, format='%Y-%m-%d')

# Calcular o Consumo Médio Diário
df$Consumo_Medio_Diario <- df$Energia / df$Dias

# Remover linhas com valores NA na coluna 'Consumo_Medio_Diario'
df <- df[!is.na(df$Consumo_Medio_Diario), ]
```

```
## # A tibble: 321 x 4
##    Ano        Energia  Dias Consumo_Medio_Diario
##    <date>       <dbl> <dbl>                <dbl>
##  1 1997-06-01      42    42                 1
##  2 1997-07-01      31    30                 1.03
##  3 1997-08-01      73    33                 2.21
##  4 1997-09-01     165    29                 5.69
##  5 1997-10-01     180    30                 6
##  6 1997-11-01     199    33                 6.03
##  7 1997-12-01     180    29                 6.21
##  8 1998-01-01     180    29                 6.21
##  9 1998-02-01     155    33                 4.70
## 10 1998-03-01     180    29                 6.21
## # i 311 more rows
```

A transformação de $Y_t = \frac{x_t}{D_t}$ normaliza o consumo de energia conforme a variação no número de dias de cada mês. Isso permite comparações justas entre meses, eliminando o efeito de diferentes durações mensais.

Dividindo o consumo total de energia pelo número de dias do mês, obtém-se uma média diária que reflete melhor o padrão de consumo. Assim, é possível identificar tendências de consumo mais claramente. Dessa forma, a transformação facilita uma análise mais precisa e comparativa entre diferentes períodos.

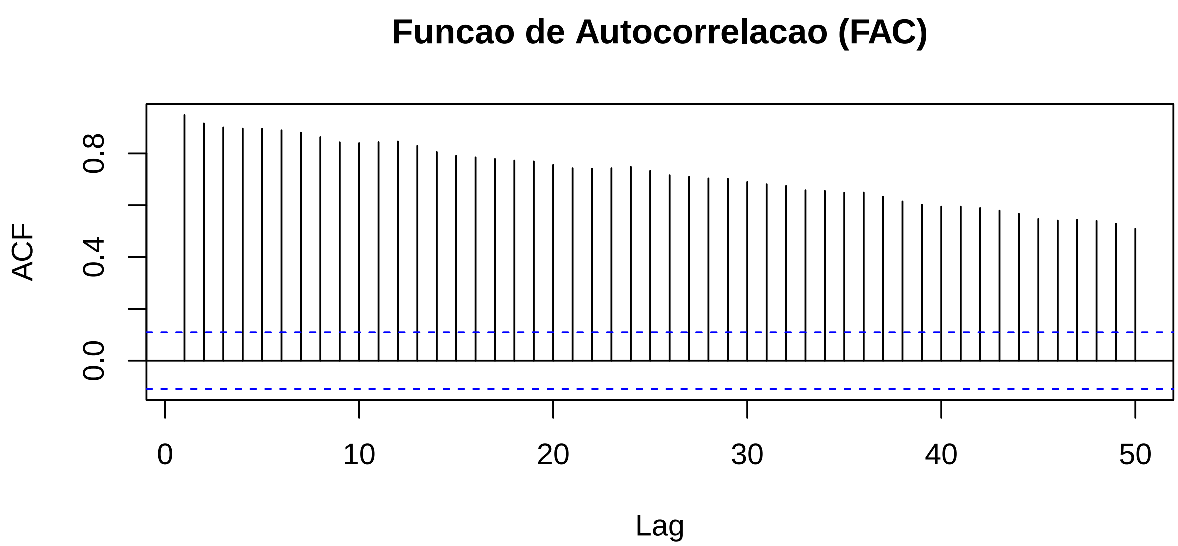
2. Apresente o gráfico da evolucao temporal de $\{Y_t\}$, e apresente sua descricao, contemplado elementos como o tamanho da serie e periodicidade dos dados.

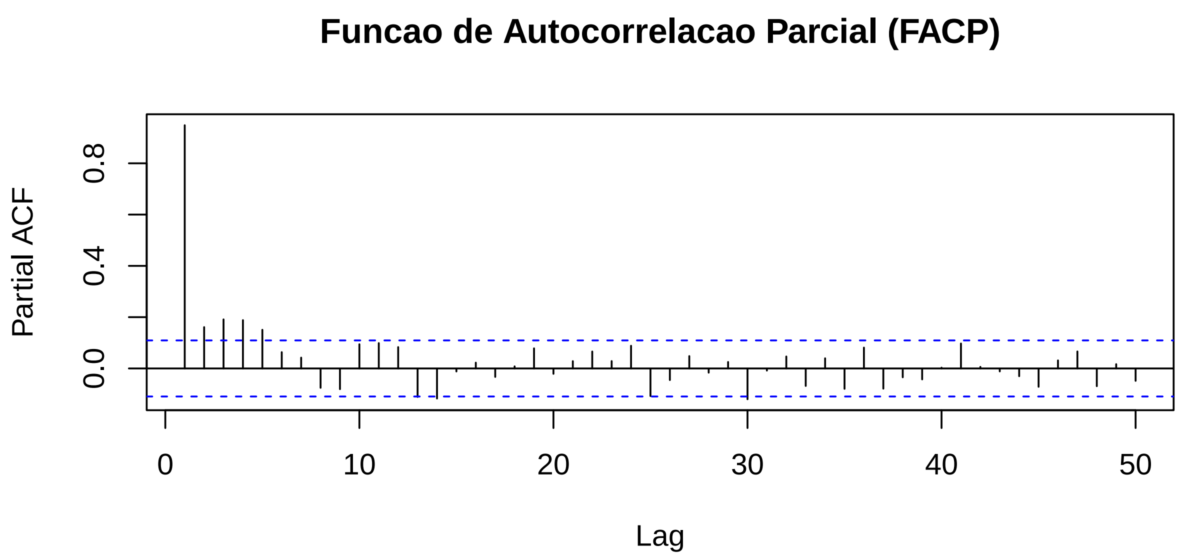


O gráfico mostra a evolução do consumo médio diário de energia de julho de 1997 a setembro de 2024, com uma tendência geral de aumento, embora com flutuações significativas. A série temporal, com 323 pontos de dados mensais, revela que o consumo começou baixo em 1997, aumentou gradualmente até 2004, subiu mais acentuadamente até 2012, e depois estabilizou com uma leve diminuição até 2020. Desde 2020, há uma queda clara no consumo, possivelmente devido a mudanças nos hábitos de consumo ou melhorias na eficiência energética. Variações sazonais são visíveis, associadas a mudanças climáticas ou eventos específicos, e picos notáveis em 2004 e 2012 indicam períodos de consumo anormalmente alto. O aumento gradual no consumo diário sugere crescimento populacional e maior uso de dispositivos elétricos, enquanto a queda pós-2020 pode refletir melhorias na eficiência energética ou impactos de crises globais, como a pandemia de COVID-19.

3. Apresente os gráficos da funcao de autocorrelacao (FAC) e da funcao de autocorrelacao parcial (FACP) de $\{Y_t\}$, considerando um número apropriado de defasagens (lag), incluindo a banda de 95% de confianca sob a hipótese nula de nao haver autocorrelacao serial. Em um parágrafo, descreva as formas da FAC e da FAPC, explicando o que se pode diagnosticar/sugerir com base nelas.

```
# Plotar a função de autocorrelação (ACF) e a função de autocorrelação parcial (PACF)
par(mfrow=c(2, 1))  # Configurar layout de 2 plots

acf(df$Consumo_Medio_Diario, lag.max = 50, main = "Funcao de Autocorrelacao (FAC)")
pacf(df$Consumo_Medio_Diario, lag.max = 50, main = "Funcao de Autocorrelacao Parcial (FACP)")
```





Os gráficos da Função de Autocorrelação (FAC) e Função de Autocorrelação Parcial (FACP) da série temporal

Y_t(Consumo Médio Diário), com 50 defasagens e uma banda de 95% de confiança, indicam a presença de autocorrelação significativa. A FAC mostra um decaimento lento das autocorrelações, típico de uma série não estacionária, enquanto a FACP apresenta autocorrelação significativa nas primeiras defasagens, sugerindo um modelo AR(1) ou AR(2). Para tornar a série estacionária, pode ser necessária uma transformação como a diferenciação. Após essa transformação, um modelo ARIMA com um componente AR de ordem 1 ou 2 pode ser adequado, sendo essencial a análise dos resíduos do modelo ajustado para confirmar a remoção da autocorrelação serial.

4. Aplique o teste aumentado de estacionariedade de Dickey-Fuller do pacote aTSA do R. Para a parte sazonal, faca a avaliacao por meio de um modelo de regressao com funcões harmonicas.

```r
# Converter a série 'Consumo_Medio_Diario' para um objeto ts
y <- ts(df$Consumo_Medio_Diario, frequency=12)

# Aplicar o teste aumentado de Dickey-Fuller
adf_result <- tseries::adf.test(y, alternative = "stationary", k =12)
```

```
## Warning in tseries::adf.test(y, alternative = "stationary", k = 12): p-value
## greater than printed p-value
```

```r
print(adf_result)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y
## Dickey-Fuller = 0.5756, Lag order = 12, p-value = 0.99
## alternative hypothesis: stationary
```

Agora vamos para o modelo de funcões harmonicas:

```r
# Criar funções harmônicas para modelagem sazonal
n <- length(y)
period <- 12
t <- 1:n

harmonics <- data.frame(
  sin1 = sin(2 * pi * t / period),
  cos1 = cos(2 * pi * t / period),
  sin2 = sin(4 * pi * t / period),
  cos2 = cos(4 * pi * t / period)
)

# Adicionar uma constante às funções harmônicas
X <- cbind(1, harmonics)

# Ajustar o modelo de regressão com funções harmônicas
harmonic_model <- lm(y ~ sin1 + cos1 + sin2 + cos2, data = harmonics)
summary(harmonic_model)
```

```
##
## Call:
## lm(formula = y ~ sin1 + cos1 + sin2 + cos2, data = harmonics)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.0703  -6.4166   0.6628   5.8432  10.8319
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 14.327436   0.357378  40.090   <2e-16 ***
## sin1         0.007594   0.504598   0.015    0.988
## cos1         0.219941   0.506204   0.434    0.664
## sin2         0.510700   0.505395   1.010    0.313
## cos2        -0.434643   0.505371  -0.860    0.390
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.402 on 316 degrees of freedom
## Multiple R-squared:  0.00617,    Adjusted R-squared:  -0.00641
## F-statistic: 0.4905 on 4 and 316 DF,  p-value: 0.7427
```

```r
# Extrair os resíduos do modelo
residuals <- residuals(harmonic_model)

# Aplicar o teste aumentado de Dickey-Fuller aos resíduos
adf_result_residuals <- tseries::adf.test(residuals,k = 12)
```

```
## Warning in tseries::adf.test(residuals, k = 12): p-value greater than printed
## p-value
```

```r
print(adf_result_residuals)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  residuals
## Dickey-Fuller = 0.47928, Lag order = 12, p-value = 0.99
## alternative hypothesis: stationary
```

O Teste Aumentado de Dickey-Fuller (ADF) indica que a série temporal original não é estacionária, e o modelo de regressão com funções harmônicas não captura bem a sazonalidade, como demonstrado pelo baixo R^2 e coeficientes estatisticamente insignificantes. Além disso, o teste ADF nos resíduos do modelo mostra que eles também não são estacionários, indicando que a sazonalidade não foi removida de forma eficaz.

5. Calcule a variacao do consumo $Z_t = Y_t - Y_{t-1}$, e explique o papel/significado dessa transformacao para a análise desses dados.

```r
# Calcular a diferença da série temporal
y_diff <- diff(y)

# Aplicar o teste aumentado de Dickey-Fuller à série diferenciada
adf_result_diff <- tseries::adf.test(y_diff,k=12)
```

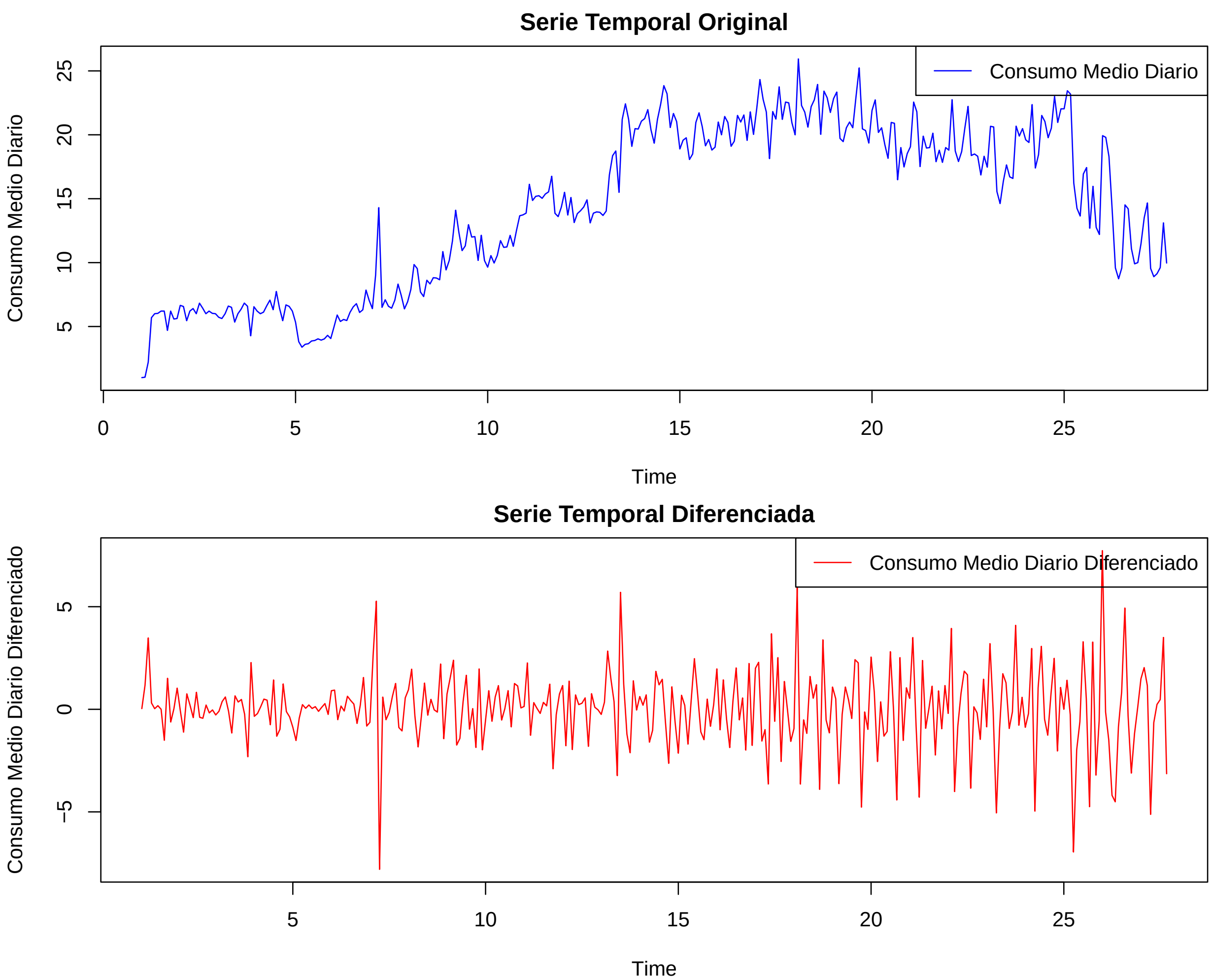
```
## Warning in tseries::adf.test(y_diff, k = 12): p-value smaller than printed
## p-value
```

```r
print(adf_result_diff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  y_diff
## Dickey-Fuller = -5.1886, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

A transformação para calcular a variação do consumo $Z_t = Y_t - Y_{t-1}$ é crucial para analisar séries temporais não estacionárias. A estatística ADF de -5.1886, sendo menor que todos os valores críticos, confirma que a série diferenciada é estacionária. A diferenciação foi eficaz, tornando a série adequada para modelos que assumem estacionaridade, como o ARIMA, que pode capturar tanto a parte autoregressiva quanto a média móvel.

6. Faca o gráfico da evolucao temporal de $\{Z_t\}$, e descreva em um parágrafo o aspecto dessa figura, comparando-a com a forma observada no item 2.

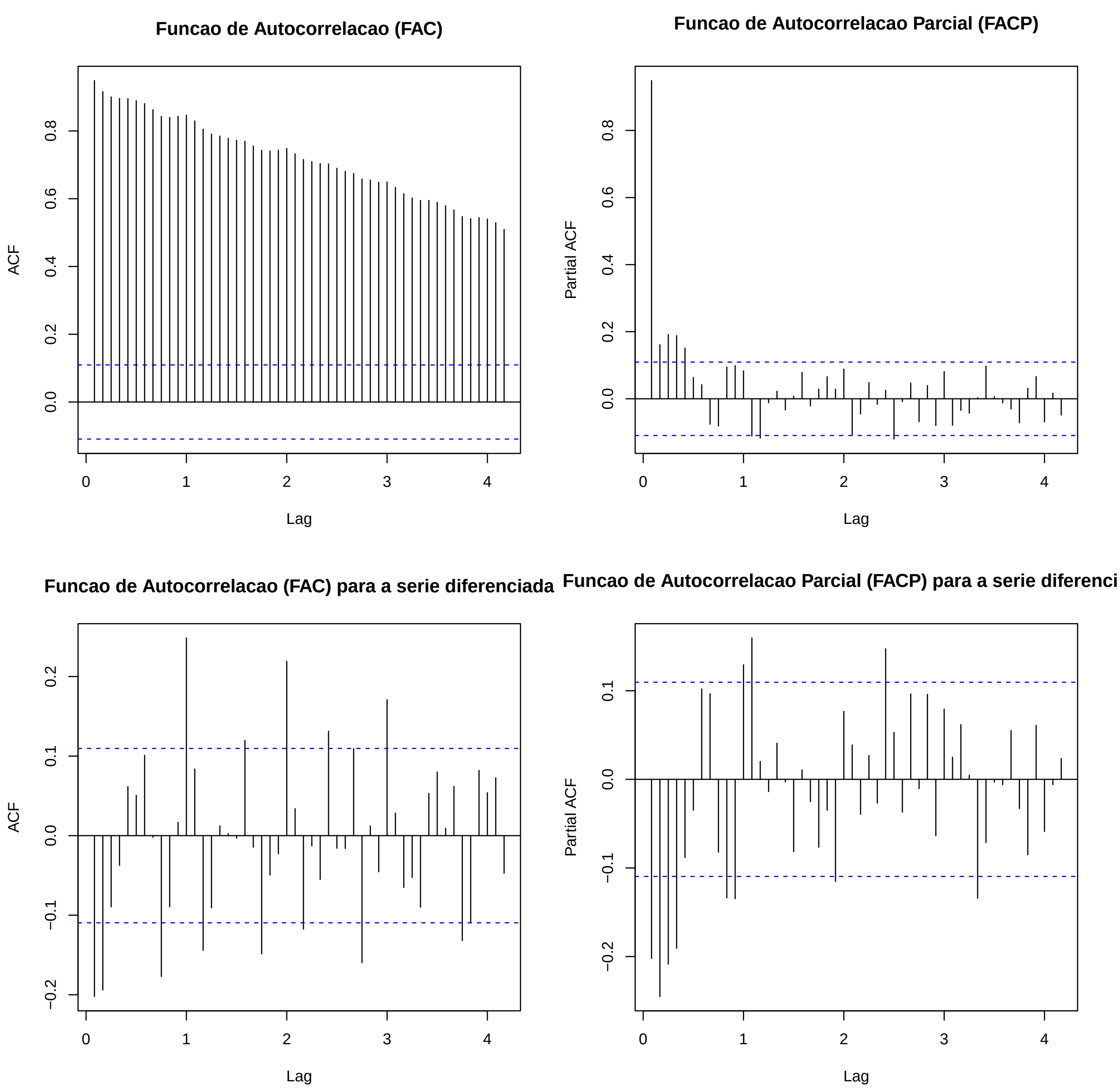


A série original exibe tendência e sazonalidade claras, com flutuações crescentes ao longo do tempo, indicando não estacionaridade. Após a diferenciação, a série mostra flutuações ao redor de uma média constante, sem tendência clara, indicando que se tornou estacionária. A série diferenciada perdeu a tendência de longo prazo e os padrões sazonais visíveis, resultando em um comportamento mais aleatório, como ruído branco. Isso confirma que a diferenciação foi bem-sucedida em remover a tendência e a sazonalidade, tornando a série adequada para modelos que assumem estacionaridade, como ARIMA.

7. Repita os passos 3 e 4, comparando os novos resultados com os anteriores.

Vamos comparar as FAC e FACP entre a serie original e a diferenciada:

```r
# Plotar as funções de autocorrelação (ACF) e autocorrelação parcial (PACF)
par(mfrow=c(2, 2))  # Configurar layout do plot

acf(y, lag.max=50, main='Funcao de Autocorrelacao (FAC)')
pacf(y, lag.max=50, main='Funcao de Autocorrelacao Parcial (FACP)')
acf(y_diff, lag.max=50, main='Funcao de Autocorrelacao (FAC) para a serie diferenciada')
pacf(y_diff, lag.max=50, main='Funcao de Autocorrelacao Parcial (FACP) para a serie diferenciada')
```



ChatGPT A Função de Autocorrelação (FAC) da série original mostra um decaimento lento, sugerindo uma tendência não estacionária, enquanto a Função de Autocorrelação Parcial (FACP) indica uma componente autoregressiva significativa. Após a diferenciação, a FAC da série diferenciada exibe um comportamento mais aleatório, com valores dentro dos limites de significância, indicando estacionaridade. A FACP da série diferenciada também mostra um comportamento aleatório, com apenas o primeiro lag significativamente diferente de zero. Isso confirma que a diferenciação removeu a tendência original, tornando a série adequada para modelagem ARIMA com pelo menos uma diferenciação.

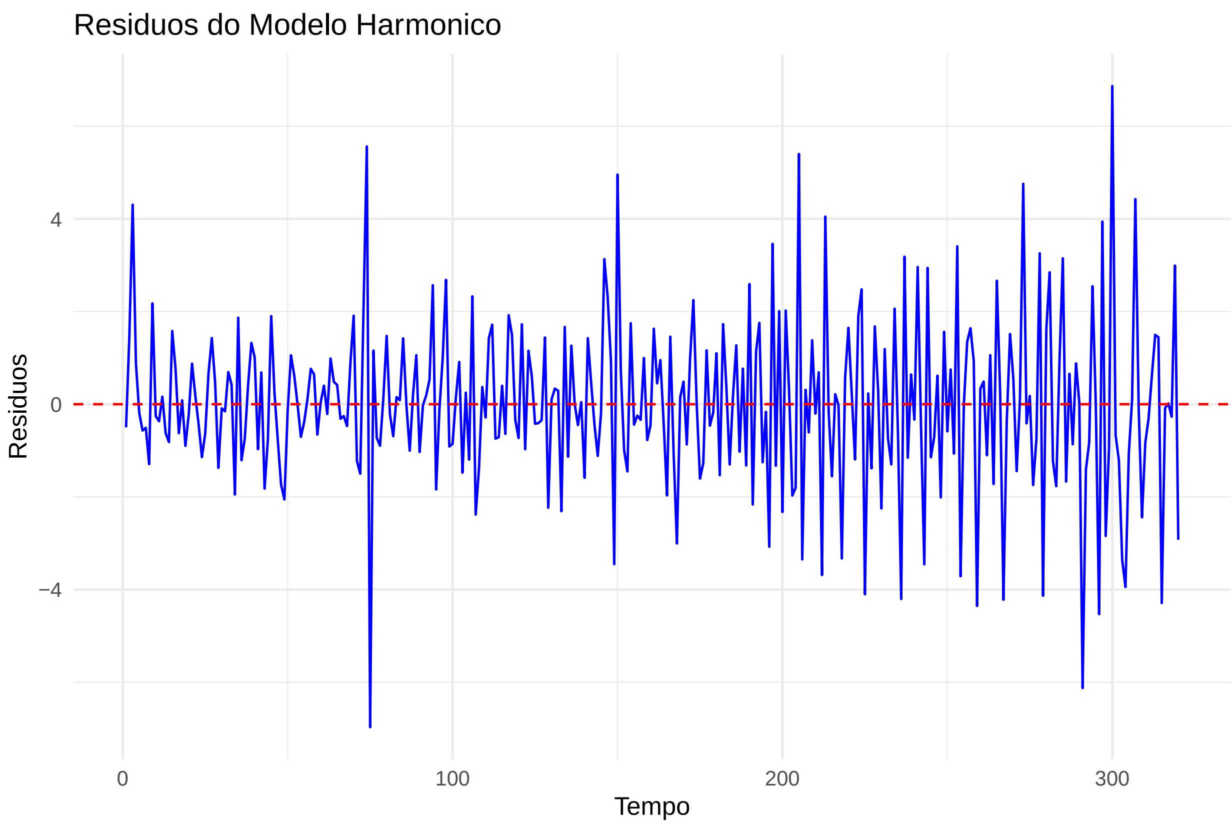
Em seguida vamos calcular o teste por meio das funcoes harmonicas:

```r
# Criar funções harmônicas para a série diferenciada
n_diff <- length(y_diff)
t_diff <- 1:n_diff

harmonics_diff <- data.frame(
  sin1 = sin(2 * pi * t_diff / period),
  cos1 = cos(2 * pi * t_diff / period),
  sin2 = sin(4 * pi * t_diff / period),
  cos2 = cos(4 * pi * t_diff / period)
)

# Adicionar uma constante às funções harmônicas
X_diff <- cbind(1, harmonics_diff)

# Ajustar o modelo de regressão com funções harmônicas para a série diferenciada
harmonic_model_diff <- lm(y_diff ~ sin1 + cos1 + sin2 + cos2, data = harmonics_diff)
summary(harmonic_model_diff)
```

```
##
## Call:
## lm(formula = y_diff ~ sin1 + cos1 + sin2 + cos2, data = harmonics_diff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9707 -0.9711 -0.0136  0.9964  6.8669
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02842    0.10175   0.279    0.780
## sin1        -0.08110    0.14390  -0.564    0.573
## cos1         0.06138    0.14390   0.427    0.670
## sin2         0.12033    0.14367   0.838    0.403
## cos2         0.77667    0.14411   5.389 1.39e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.82 on 315 degrees of freedom
## Multiple R-squared:  0.08767,    Adjusted R-squared:  0.07609
## F-statistic: 7.568 on 4 and 315 DF,  p-value: 7.839e-06
```

```r
# Extrair os resíduos do modelo
residuals_diff <- residuals(harmonic_model_diff)

# Aplicar o teste aumentado de Dickey-Fuller aos resíduos do modelo harmônico da série diferenciada
adf_result_residuals_diff <- adf.test(residuals_diff)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -23.4    0.01
## [2,]   1 -18.4    0.01
## [3,]   2 -15.0    0.01
## [4,]   3 -12.9    0.01
## [5,]   4 -11.0    0.01
## [6,]   5 -10.1    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -23.4    0.01
## [2,]   1 -18.4    0.01
## [3,]   2 -15.0    0.01
## [4,]   3 -12.9    0.01
## [5,]   4 -11.0    0.01
## [6,]   5 -10.0    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -23.5    0.01
## [2,]   1 -18.5    0.01
## [3,]   2 -15.1    0.01
## [4,]   3 -13.0    0.01
## [5,]   4 -11.2    0.01
## [6,]   5 -10.3    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```



A análise dos resíduos de um modelo de regressão com funções harmônicas indica que a sazonalidade foi capturada adequadamente.

Considere que $\hat{Y}_{t+h}$ representa a previsao no instante $t+h$ obtida com base nas informacões disponíveis até o tempo t; ou seja,

$$Y_1, \ldots, Y_t \to \hat{Y}_{t+h}$$

Separe a massa de dados em duas partes, conforme esquema abaixo:

Treinamento (modelagem)	validacao
$Y_1, \ldots, Y_m$	$Y_{m+1}, \ldots, Y_n$

Utilizando os dados de treinamento:

8. Considerando o modelo SARIMA$(p, d, q) \times (P, D, Q)s$ **para a série** Y_t**, defina um valor apropriado para a ordem sazonal** s **e as ordens de diferenciacões** d **e** D **com base nos passos anteriores.**

9. Defina uma malha de valores para as ordens autorregressivas p e P e de médias móveis q e Q e obtenha o valor do critério de informacao bayesiano de Schwarz (BIC) para cada combinacao $(p, d, q) \times (P, D, Q)$ por meio da funcao sarima do pacote astsa.

```r
# Funcao paralelizada (7.7sec) enquanto a normal demora
# 21sec Dividir os dados em treino e validação
train <- head(df$Consumo_Medio_Diario, round(0.8 * nrow(df)))
validation <- tail(df$Consumo_Medio_Diario, nrow(df) - length(train))
full_data <- c(train, validation)

# Definir as malhas de valores para os parâmetros p, d, q,
# P, D, Q, m
p <- c(0, 1, 2)
d <- c(1)
q <- c(0, 1, 2)
P <- c(0, 1, 2)
D <- c(1)
Q <- c(0, 1, 2)
m <- c(12)

# Criar combinações de pdq e PDQm
pdq_grid <- expand.grid(p = p, d = d, q = q)
PDQm_grid <- expand.grid(P = P, D = D, Q = Q, m = m)

# Configurar paralelismo
cores <- detectCores() - 1
cl <- makeCluster(cores)
registerDoParallel(cl)

# Função para calcular o BIC para cada combinação de pdq e
# PDQm
SARIMA_grid <- function(endog, pdq_grid, PDQm_grid) {
    # Pré-alocar data frame
    model_info <- data.frame(order = character(), seasonal_order = character(),
        MAPE = numeric(), RMSE = numeric(), AIC = numeric(),
        BIC = numeric(), stringsAsFactors = FALSE)

    # Usar foreach para paralelismo
    results <- foreach(i = 1:nrow(pdq_grid), .combine = rbind,
        .packages = c("forecast")) %dopar% {
        local_model_info <- data.frame(order = character(), seasonal_order = character(),
            MAPE = numeric(), RMSE = numeric(), AIC = numeric(),
            BIC = numeric(), stringsAsFactors = FALSE)

        for (j in 1:nrow(PDQm_grid)) {
            try({
                fit <- Arima(endog, order = as.numeric(pdq_grid[i,
                  1:3]), seasonal = list(order = as.numeric(PDQm_grid[j,
                  1:3]), period = PDQm_grid[j, 4]))
                pred <- fitted(fit)

                MAPE <- mean(abs((endog - pred)/endog), na.rm = TRUE)
                RMSE <- sqrt(mean((endog - pred)^2, na.rm = TRUE))
                AIC <- fit$aic
```

```r
                BIC <- BIC(fit)

                local_model_info <- rbind(local_model_info, data.frame(order = paste(as.numeric(pdq_gri
                  1:3]), collapse = ","), seasonal_order = paste(as.numeric(PDQm_grid[j,
                  1:4]), collapse = ","), MAPE = MAPE, RMSE = RMSE,
                  AIC = AIC, BIC = BIC, stringsAsFactors = FALSE))
            }, silent = TRUE)
        }

        return(local_model_info)
    }

    return(results)
}

# Medir o tempo de execução
start_time <- Sys.time()

# Calcular os modelos SARIMA
model_info <- SARIMA_grid(train, pdq_grid, PDQm_grid)

end_time <- Sys.time()
execution_time <- end_time - start_time
print(paste("Tempo necessário:", execution_time))
```

```
## [1] "Tempo necessário: 17.6855640411377"
```

```r
# Fechar cluster
stopCluster(cl)

# Obter os 10 melhores modelos com base nos critérios MAPE,
# RMSE, AIC e BIC
least_MAPE <- model_info %>%
    arrange(MAPE) %>%
    head(10)

least_RMSE <- model_info %>%
    arrange(RMSE) %>%
    head(10)

least_AIC <- model_info %>%
    arrange(AIC) %>%
    head(10)

least_BIC <- model_info %>%
    arrange(BIC) %>%
    head(10)
```

```
## [1] "Melhores modelos por MAPE:"

##    order seasonal_order       MAPE     RMSE      AIC      BIC
## 1  1,1,2       2,1,1,12 0.08931063 1.427524 905.8854 930.3656
## 2  0,1,2       2,1,1,12 0.08932665 1.427894 904.0454 925.0284
## 3  2,1,1       2,1,1,12 0.08936321 1.425696 905.0752 929.5554
## 4  0,1,2       2,1,2,12 0.08938787 1.427198 905.2428 929.7230
## 5  1,1,1       2,1,1,12 0.08939643 1.428318 904.1519 925.1349
## 6  2,1,2       2,1,1,12 0.08943931 1.425015 906.7949 934.7723
## 7  1,1,2       2,1,2,12 0.08944031 1.426653 907.0351 935.0124
## 8  1,1,2       1,1,2,12 0.08944357 1.428625 905.9620 930.4422
## 9  1,1,2       0,1,1,12 0.08944543 1.428407 902.4006 919.8864
## 10 0,1,2       0,1,1,12 0.08944570 1.428848 900.5666 914.5553

## [1] "Melhores modelos por RMSE:"

##    order seasonal_order       MAPE     RMSE      AIC      BIC
## 1  2,1,2       2,1,2,12 0.08986999 1.422732 907.6213 939.0958
## 2  2,1,1       2,1,2,12 0.08970899 1.423829 906.0019 933.9793
## 3  2,1,2       2,1,1,12 0.08943931 1.425015 906.7949 934.7723
## 4  2,1,2       0,1,1,12 0.08960846 1.425445 903.1934 924.1764
## 5  2,1,1       2,1,1,12 0.08936321 1.425696 905.0752 929.5554
## 6  2,1,2       1,1,2,12 0.08956530 1.425709 906.8309 934.8082
## 7  2,1,2       1,1,1,12 0.08958258 1.425780 904.8362 929.3164
## 8  2,1,2       0,1,2,12 0.08959984 1.425860 904.8536 929.3338
## 9  2,1,1       0,1,1,12 0.08959914 1.426291 901.5392 919.0251
## 10 2,1,1       1,1,2,12 0.08948140 1.426479 905.1212 929.6014

## [1] "Melhores modelos por AIC:"

##    order seasonal_order       MAPE     RMSE      AIC      BIC
## 1  0,1,1       0,1,1,12 0.08985144 1.430953 899.0621 909.5536
## 2  0,1,2       0,1,1,12 0.08944570 1.428848 900.5666 914.5553
## 3  1,1,1       0,1,1,12 0.08953324 1.429262 900.6692 914.6579
## 4  0,1,1       1,1,1,12 0.08983963 1.431275 900.6765 914.6652
## 5  0,1,1       0,1,2,12 0.08985964 1.431329 900.7014 914.6901
## 6  2,1,1       0,1,1,12 0.08959914 1.426291 901.5392 919.0251
## 7  0,1,2       1,1,1,12 0.08948017 1.429216 902.1429 919.6288
## 8  0,1,2       0,1,2,12 0.08949147 1.429281 902.1713 919.6572
## 9  1,1,1       1,1,1,12 0.08955842 1.429621 902.2522 919.7380
## 10 1,1,1       0,1,2,12 0.08957067 1.429681 902.2802 919.7660

## [1] "Melhores modelos por BIC:"

##    order seasonal_order       MAPE     RMSE      AIC      BIC
## 1  0,1,1       0,1,1,12 0.08985144 1.430953 899.0621 909.5536
## 2  0,1,2       0,1,1,12 0.08944570 1.428848 900.5666 914.5553
## 3  1,1,1       0,1,1,12 0.08953324 1.429262 900.6692 914.6579
## 4  0,1,1       1,1,1,12 0.08983963 1.431275 900.6765 914.6652
## 5  0,1,1       0,1,2,12 0.08985964 1.431329 900.7014 914.6901
## 6  2,1,0       0,1,1,12 0.09087644 1.443575 904.8032 918.7919
## 7  2,1,1       0,1,1,12 0.08959914 1.426291 901.5392 919.0251
## 8  0,1,2       1,1,1,12 0.08948017 1.429216 902.1429 919.6288
## 9  0,1,2       0,1,2,12 0.08949147 1.429281 902.1713 919.6572
## 10 1,1,1       1,1,1,12 0.08955842 1.429621 902.2522 919.7380
```

10. Liste os modelos com os menores BIC. Certifique-se que o melhor modelo nao possua uma ordem na extremidade da malha definida no item 9. Se houver, retorne para o passo 9 , ampliando a malha.

```r
# Listar os modelos com os menores BIC
least_BIC <- model_info %>% arrange(BIC) %>% head(10)
```

```
## [1] "Modelos com menores BIC:"
```

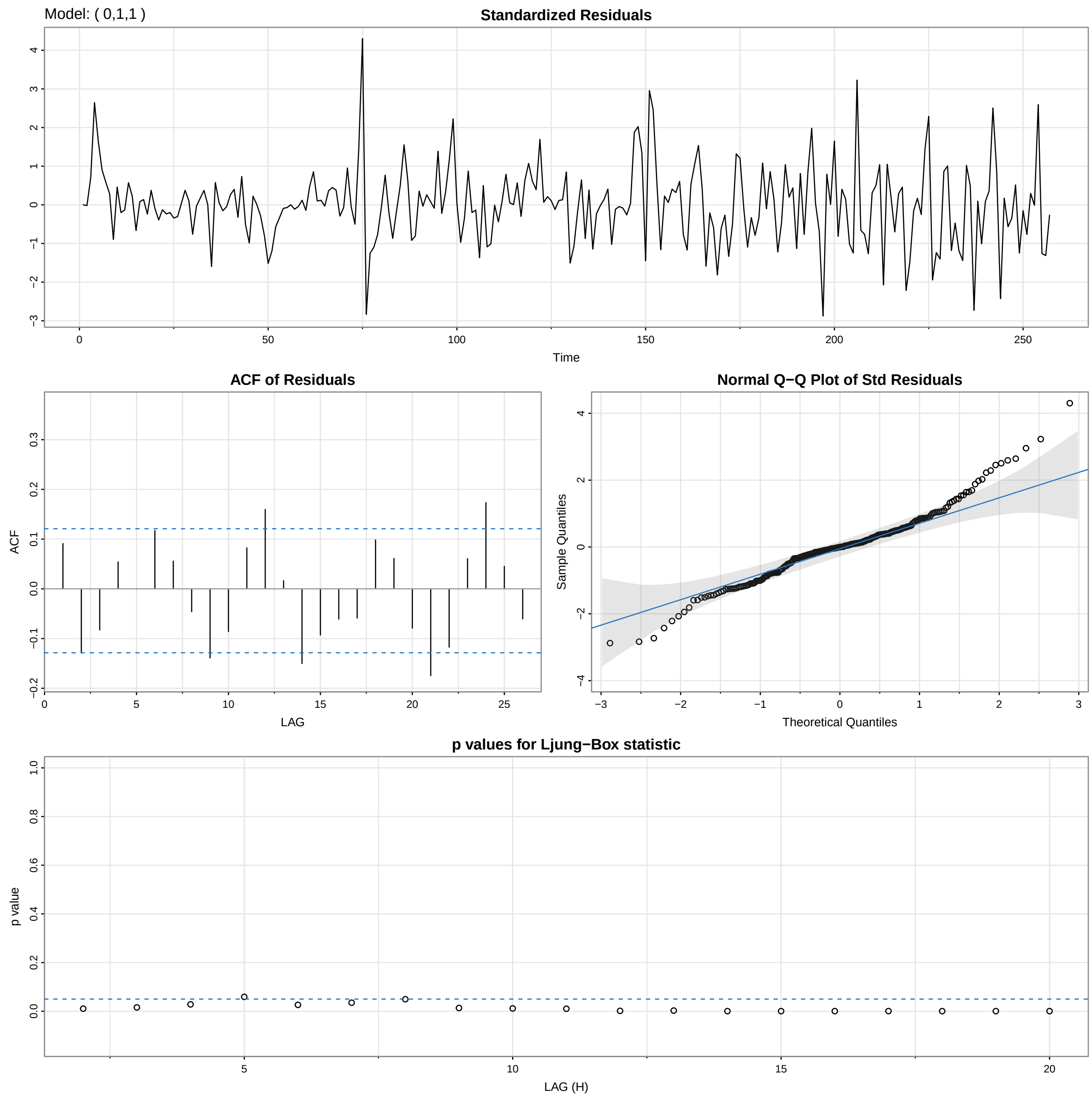
```
##    order seasonal_order       MAPE     RMSE      AIC      BIC
## 1  0,1,1       0,1,1,12 0.08985144 1.430953 899.0621 909.5536
## 2  0,1,2       0,1,1,12 0.08944570 1.428848 900.5666 914.5553
## 3  1,1,1       0,1,1,12 0.08953324 1.429262 900.6692 914.6579
## 4  0,1,1       1,1,1,12 0.08983963 1.431275 900.6765 914.6652
## 5  0,1,1       0,1,2,12 0.08985964 1.431329 900.7014 914.6901
## 6  2,1,0       0,1,1,12 0.09087644 1.443575 904.8032 918.7919
## 7  2,1,1       0,1,1,12 0.08959914 1.426291 901.5392 919.0251
## 8  0,1,2       1,1,1,12 0.08948017 1.429216 902.1429 919.6288
## 9  0,1,2       0,1,2,12 0.08949147 1.429281 902.1713 919.6572
## 10 1,1,1       1,1,1,12 0.08955842 1.429621 902.2522 919.7380
```

11. Inicie o diagnóstico com o modelo que apresenta o menor BIC:

11.1 Analise as estimativas dos parâmetros por meio da funcao sarima do pacote astsa.

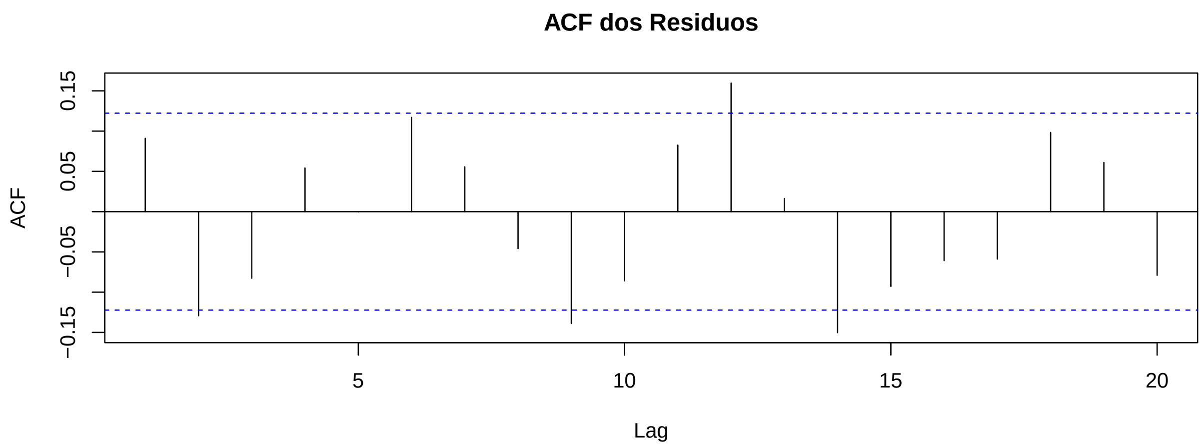
```r
# Analisar as estimativas dos parâmetros do melhor modelo
# Seleciona os parâmetros do melhor modelo com base no menor BIC
best_model_params <- least_BIC[1,1]
best_model_params_seasonal = least_BIC[1,2]

order <- as.integer(unlist(strsplit(best_model_params, ",")))
best_seasonal_order <- as.integer(unlist(strsplit(best_model_params_seasonal, ",")))
seasonal_order = list(order = best_seasonal_order, period = m)

# Ajusta o modelo SARIMA com os melhores parâmetros
best_model <- sarima(train,  p = order[1], d = order[2], q = order[3] , seasonal = seasonal_order)
```

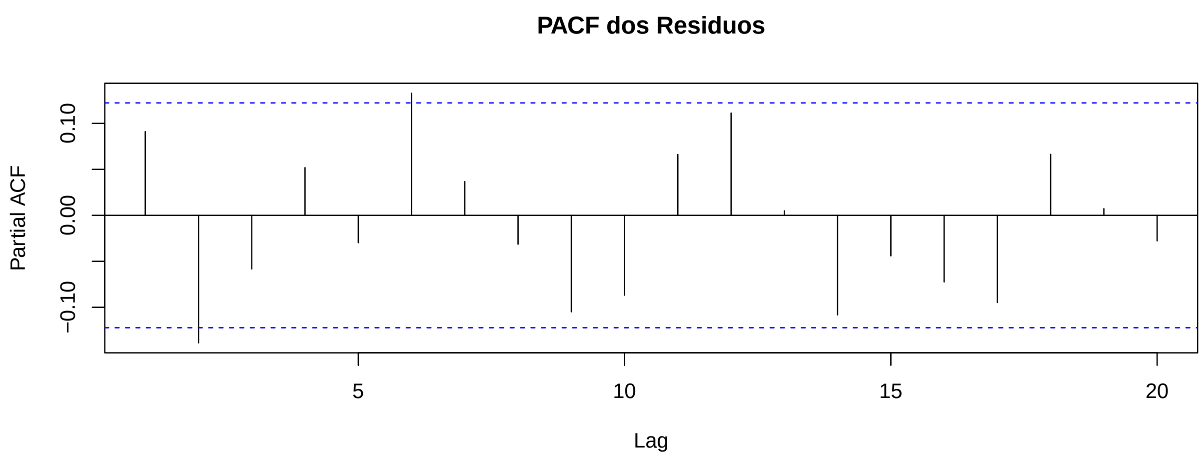
```
## initial  value 0.499087
## iter   2 value 0.426644
## iter   3 value 0.417404
## iter   4 value 0.406174
## iter   5 value 0.402059
## iter   6 value 0.401257
## iter   7 value 0.401168
## iter   8 value 0.401163
## iter   9 value 0.401163
## iter  10 value 0.401163
## iter  10 value 0.401163
## iter  10 value 0.401163
## final  value 0.401163
## converged
## initial  value 0.400774
## iter   2 value 0.400736
## iter   3 value 0.400736
## iter   4 value 0.400732
## iter   5 value 0.400732
## iter   5 value 0.400732
## iter   5 value 0.400732
## final  value 0.400732
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ma1       -0.5672 0.0590 -9.6149  0.0000
## constant   0.0662 0.0406  1.6308  0.1042
##
## sigma^2 estimated as 2.225425 on 254 degrees of freedom
##
## AIC = 3.662779  AICc = 3.662964  BIC = 3.704324
##
```


Model: (0,1,1)
Standardized Residuals
Time
ACF of Residuals
ACF
LAG
Normal Q−Q Plot of Std Residuals
Sample Quantiles
Theoretical Quantiles
p values for Ljung−Box statistic
p value
LAG (H)

11.2 Faca os gráficos da FAC e FACP residual, e aplique o teste de LjungBox.

```
# Obtém os resíduos do melhor modelo
residuals <- residuals(best_model$fit)

# Plota a ACF e PACF dos resíduos
par(mfrow = c(2, 1))

acf(residuals, main = "ACF dos Residuos", lag.max = 20)
pacf(residuals, main = "PACF dos Residuos", lag.max = 20)
```





```r
# Teste de Ljung-Box
ljung_box_result <- Box.test(residuals, lag = 10, type = "Ljung-Box", fitdf = length(order) - 1)
```

```
## [1] "Resultados para o décimo Lag"
```

```
##
##  Box-Ljung test
##
## data:  residuals
## X-squared = 21.276, df = 8, p-value = 0.006449
```

```
##
## Conclusão:
##  Há evidências de autocorrelação significativa nos resíduos no lag 10 (p < 0.05).
```

11.3 Teste a normalidade residual.

```r
# Teste de Shapiro-Wilk para normalidade dos resíduos
shapiro_test <- shapiro.test(residuals)
shapiro_p_value <- shapiro_test$p.value
```

```
## Resultado do Teste de Shapiro-Wilk (p-value): 8.069737e-06
```

```
## Os resíduos não parecem ser normalmente distribuídos. Considere investigar mais.
```

11.4 Caso haja problemas em 11.1 e 11.2 , repita a análise com os próximos modelos candidatos.

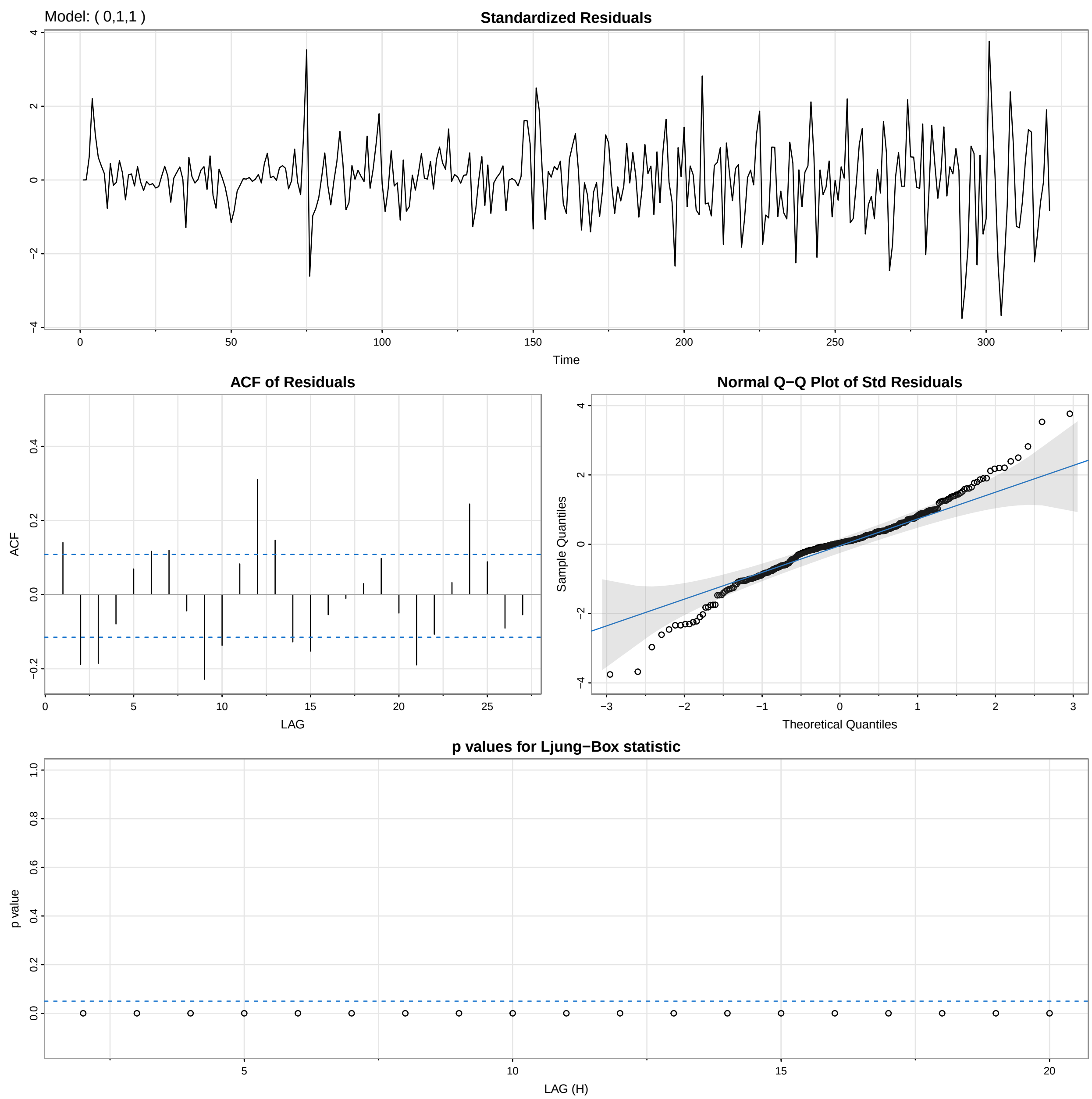
11.5 Caso nao seja possível encontrar um modelo adequado, será preciso redefinir o modelo no passo 8 . Se as ordens s, d, e D estiverem corretas, entao é possível que o modelo SARIMA näo seja apropriado. Utilizando os dados de validacao:

12. Como o método de estimacao é recursivo, a obtencao dos erros de previsao um passo à frente na massa de validacao pode ser realizada da seguinte forma:

12.1 Aplique o modelo sobre a base de dados completa, usando a funcao sarima do pacote astsa.

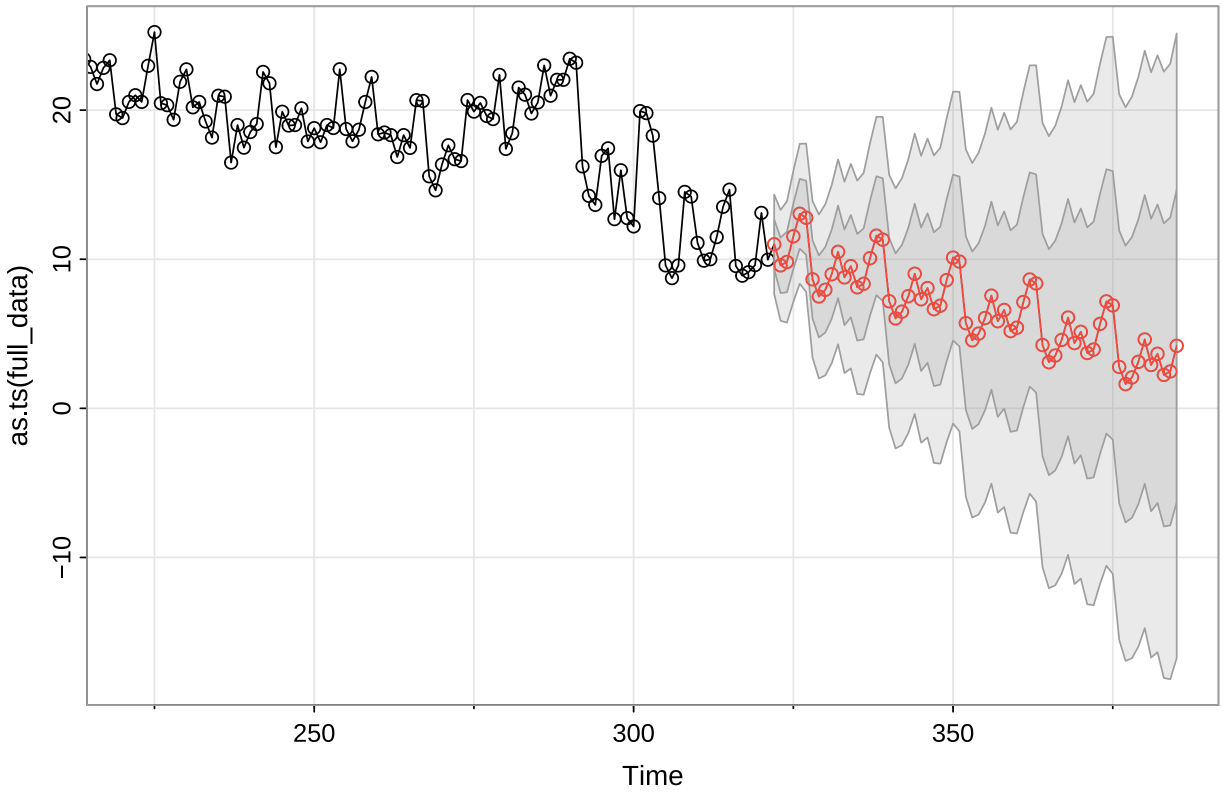
```
# Ajusta o modelo SARIMA na base de dados completa
fitted_model <- sarima(full_data, p = order[1], d = order[2], q = order[3] , seasonal = seasonal_order)
```

```
## initial  value 0.636798
## iter   2 value 0.605318
## iter   3 value 0.587562
## iter   4 value 0.587434
## iter   5 value 0.587394
## iter   6 value 0.587394
## iter   6 value 0.587394
## final  value 0.587394
## converged
## initial  value 0.587512
## iter   2 value 0.587509
## iter   3 value 0.587509
## iter   3 value 0.587509
## iter   3 value 0.587509
## final  value 0.587509
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ma1       -0.4935 0.0736 -6.7057  0.0000
## constant   0.0281 0.0511  0.5495  0.5831
##
## sigma^2 estimated as 3.235375 on 318 degrees of freedom
##
## AIC = 4.031644  AICc = 4.031763  BIC = 4.066972
##
```


Model: (0,1,1)
Standardized Residuals
Time
ACF of Residuals
ACF
LAG
Normal Q−Q Plot of Std Residuals
Sample Quantiles
Theoretical Quantiles
p values for Ljung−Box statistic
p value
LAG (H)

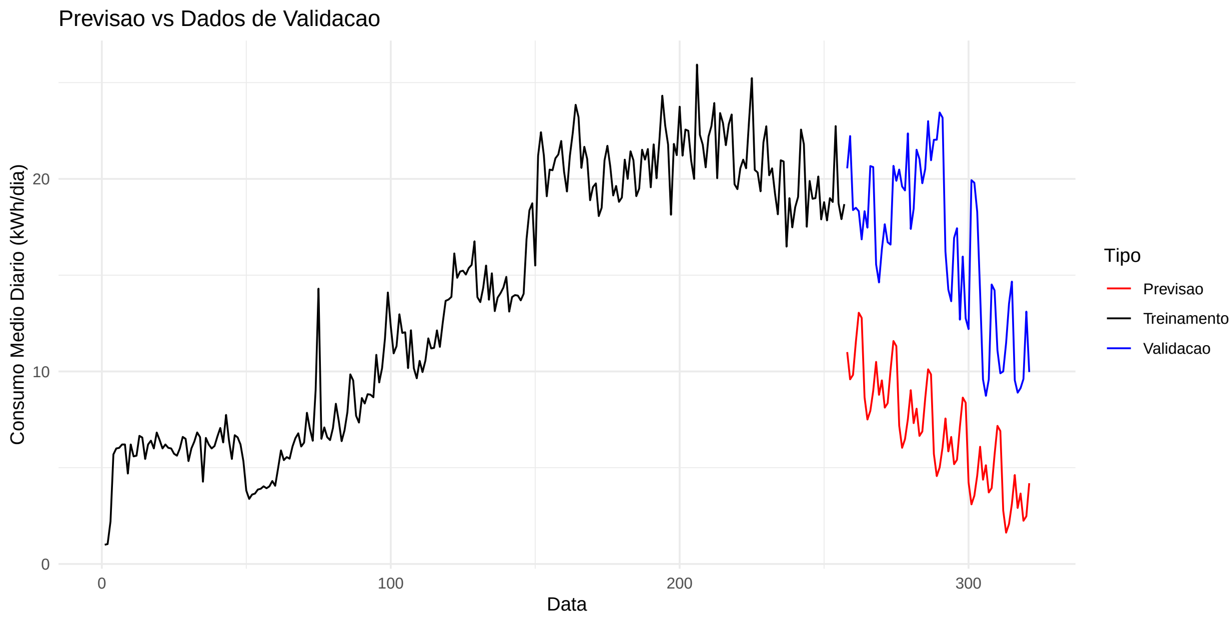
12.2 Obtenha os erros de previsão um passo à frente observados na parte da validação do modelo

```r
# Previsão na base de validação
forecast <- sarima.for(as.ts(full_data), n.ahead=length(validation),
                       p=order[1], d=order[2], q=order[3],
                       P=best_seasonal_order[1], D=best_seasonal_order[2], Q=best_seasonal_order[3], S=
```




```r
# Calcular os erros de previsão
mse <- mean((validation - forecast$pred)^2)
mae <- mean(abs(validation - forecast$pred))
```

```
##
## Erros de Previsão um passo à frente na massa de validação:

## MSE: 108.395056

## MAE: 9.684503
```



Previsao vs Dados de Validacao
Consumo Medio Diario (kWh/dia)
0
10
20
0
100
200
300
Data
Tipo
Previsao
Treinamento
Validacao

12.3 Calcule um índice de desempenho preditivo. Por exemplo, obtenha o MAPE (mean absolute percentage error)

```
forecast_mean <- forecast$pred

# Calcular o MAPE
mape <- mean(abs((validation - forecast_mean) / validation))
```

```
## O MAPE (Erro Médio Percentual Absoluto) é: 0.58%
```

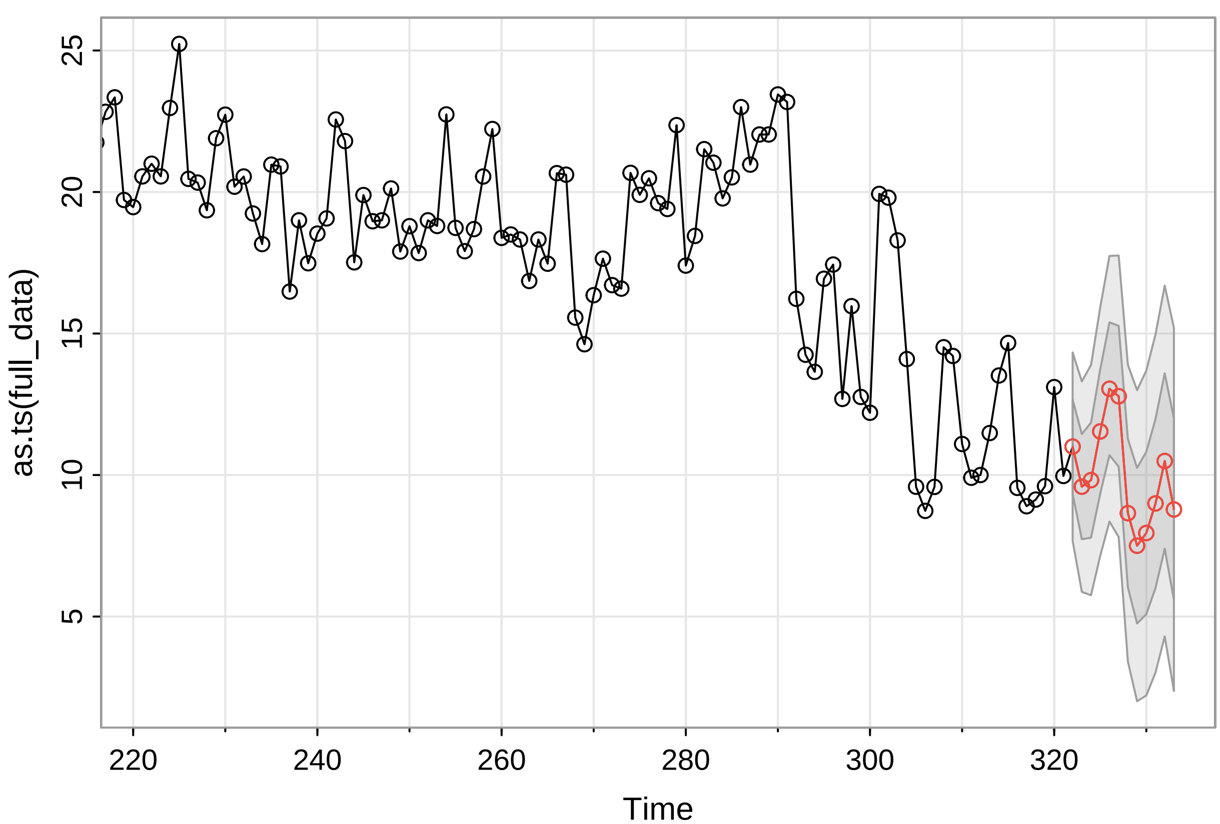
12.4 Como referência, modelos com MAPE inferiores a 10% geralmente são considerados muito bons. Entre 10% e 20% são bons modelos preditivos, e entre 20% e 50% são modelos razoáveis/aceitáveis.

```
if (mape < 10) {
  cat("Modelo muito bom.\n")
} else if (mape < 20) {
  cat("Bom modelo preditivo.\n")
} else if (mape < 50) {
  cat("Modelo razoável/aceitável.\n")
} else {
  cat("O modelo precisa ser melhorado.\n")
}
```

```
## Modelo muito bom.
```

13. Utilize a função sarima.for do pacote astsa para a obtenção de previsões para os próximos 12 meses (ou outro horizonte desejado) e a banda de previsão com 95% de cobertura. Discuta sobre as limitações dessas previsões, incluindo um insigh sobre como proceder se a hipótese de normalidade residual for descartada no passo 11.3.

```
horizon <- 12

last_date <- tail(df$Ano,n=1)

# Extrair o último mês e ano das datas
last_month <- as.numeric(format(last_date, "%m"))
last_year <- as.numeric(format(last_date, "%Y"))

# Gerar índices de datas para os próximos 12 meses
date_index <- seq(as.Date(paste0(last_year, "-", last_month, "-01")),
                  by = "month", length.out = horizon)

# Realizar a previsão usando o modelo ajustado
forecast <- sarima.for(as.ts(full_data), n.ahead = horizon,
                       p = order[1], d = order[2], q = order[3],
                       P = best_seasonal_order[1], D = best_seasonal_order[2], Q = best_seasonal_order[3
                       S = best_seasonal_order[4])
```



```r
# Extrair valores previstos e intervalos de confiança
forecast_values <- forecast$pred
forecast_conf_int <- forecast$pred + cbind(-1.96 * forecast$se, 1.96 * forecast$se)

# Criar dataframe de previsões
forecast_df <- data.frame(
  Mes = date_index,
  Previsões = forecast_values,
  Limite_Inferior = forecast_conf_int[, 1],
  Limite_Superior = forecast_conf_int[, 2]
)

# Converter coluna de meses para o formato ano-mês
forecast_df$Mes <- format(forecast_df$Mes, "%Y-%m")
```

```
##        Mes Previsões Limite_Inferior Limite_Superior
## 1  2024-02 11.004960        7.742508        14.26741
## 2  2024-03  9.589115        5.946415        13.23181
## 3  2024-04  9.819935        5.833091        13.80678
## 4  2024-05 11.539586        7.236030        15.84314
## 5  2024-06 13.048359        8.449853        17.64686
## 6  2024-07 12.782956        7.907311        17.65860
## 7  2024-08  8.652231        3.514373        13.79009
## 8  2024-09  7.502309        2.114986        12.88963
## 9  2024-10  7.950865        2.325129        13.57660
## 10 2024-11  8.995454        3.141005        14.84990
## 11 2024-12 10.495515        4.420958        16.57007
## 12 2025-01  8.783873        2.496910        15.07084
```

14. Redija um parágrafo concluindo o estudo (inclua uma recomendação sobre como o modelo deve ser atualizado à medida que novas informações estiverem disponíveis).