Lista Econometria 2

Shai Vaz

2023-08-30

Questão 1

Importar base

```
unemployment <- get_sidra(x = 6381,
                    variable = 4099,
                    period = "all",
                    classific = "all") %>%
  select(
    8,9,5
  ) %>%
  rename(
    "mes" = "Trimestre Móvel (Código)",
    "trimestre" = "Trimestre Móvel",
    "taxa" = "Valor"
  ) %>%
  mutate(
    mes = as_date(mes, format = "%Y%m")
  ) %>%
  filter(
   mes >= '2013-06-01'
  )
```

```
## Considering all categories once 'classific' was set to 'all' (default)
```

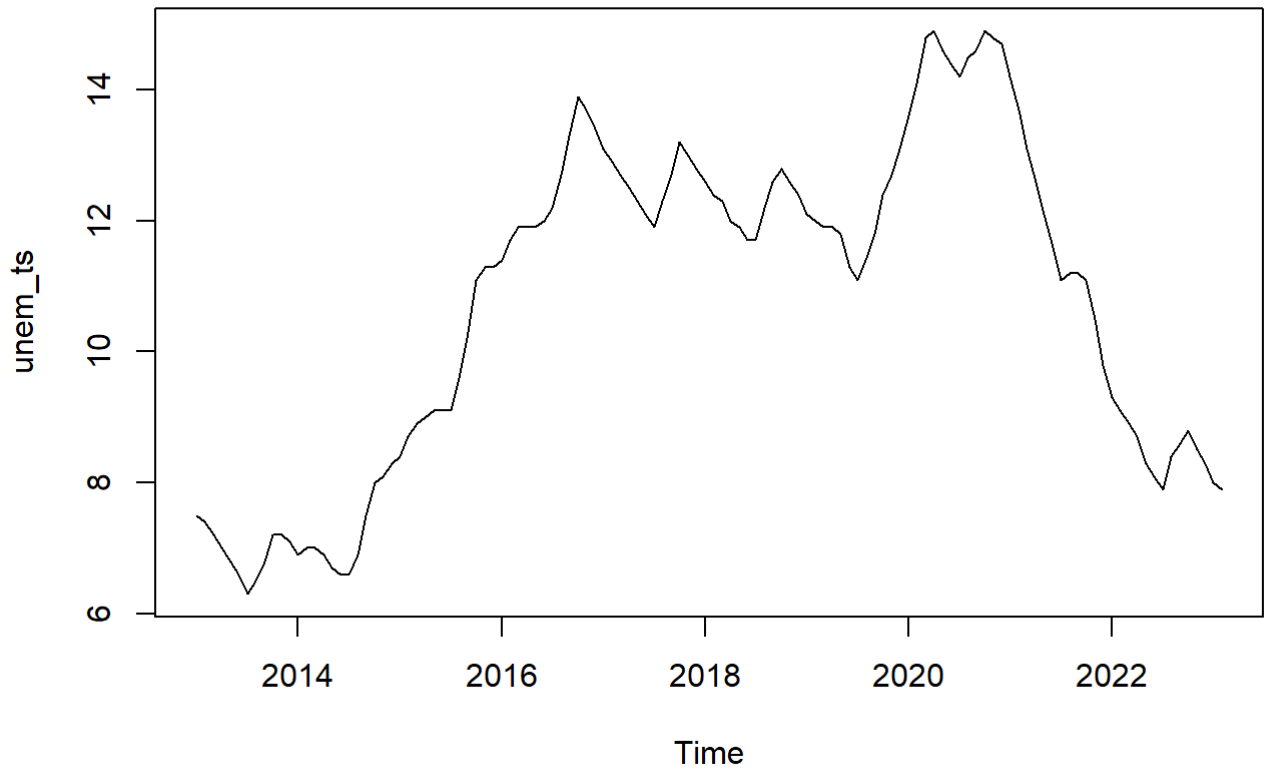
Time Series

Primeiro, a taxa de desemprego no período.

```
unem_ts <- ts(
  data = unemployment$taxa,
  start = c(2013,1),
  frequency = 12
    )
```

Plots

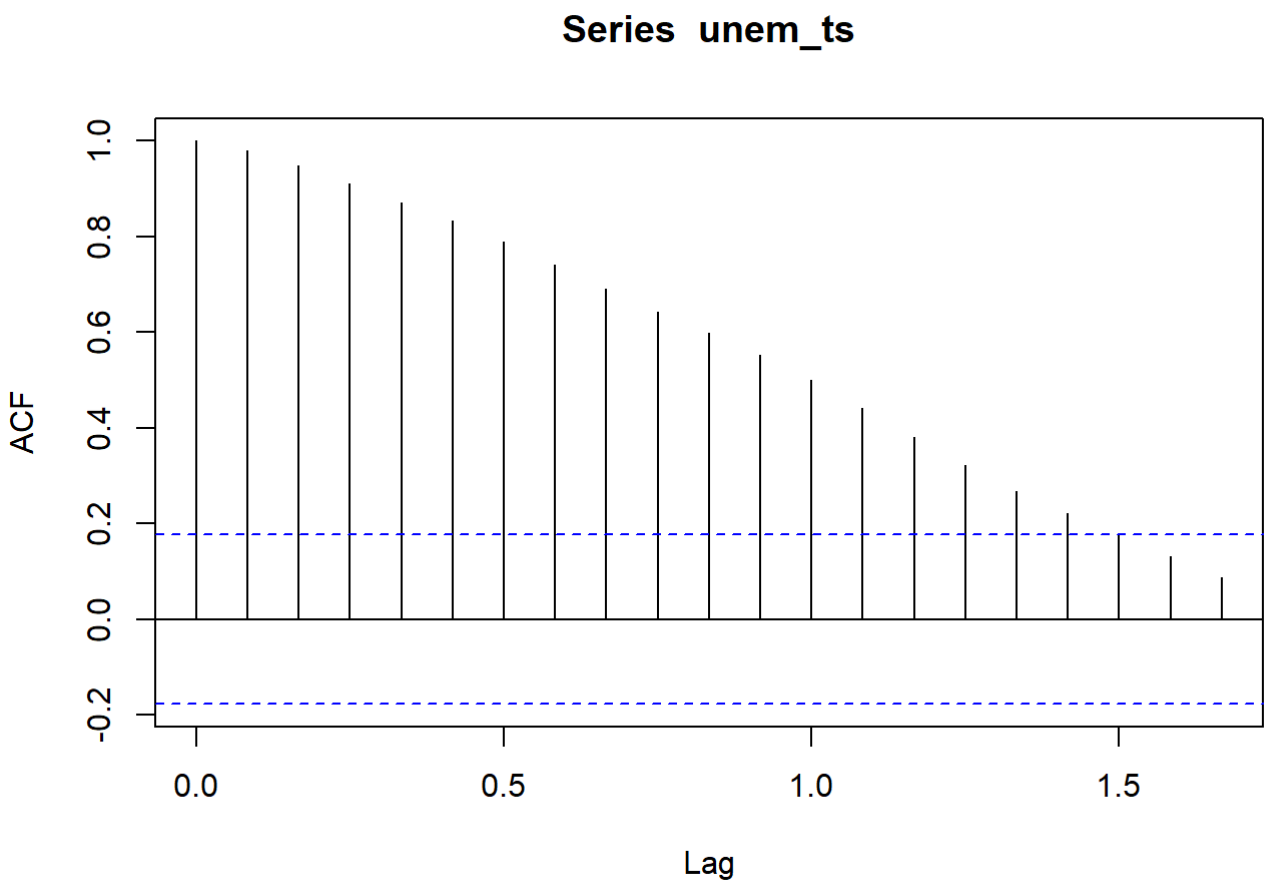
```
plot(unem_ts)
```



Funções de Autocorrelação

Autocorrelação

```
acf(unem_ts)
```



Esse padrão de ACF é um forte indicador de que a série é não estacionária. Para confirmar, vamos executar um teste de Dickey-Fuller aumentado que testa a hipótese nula da existência de raiz unitária na série temporal, ou seja, a série não é estacionária.

```
adf.test(unem_ts)
```

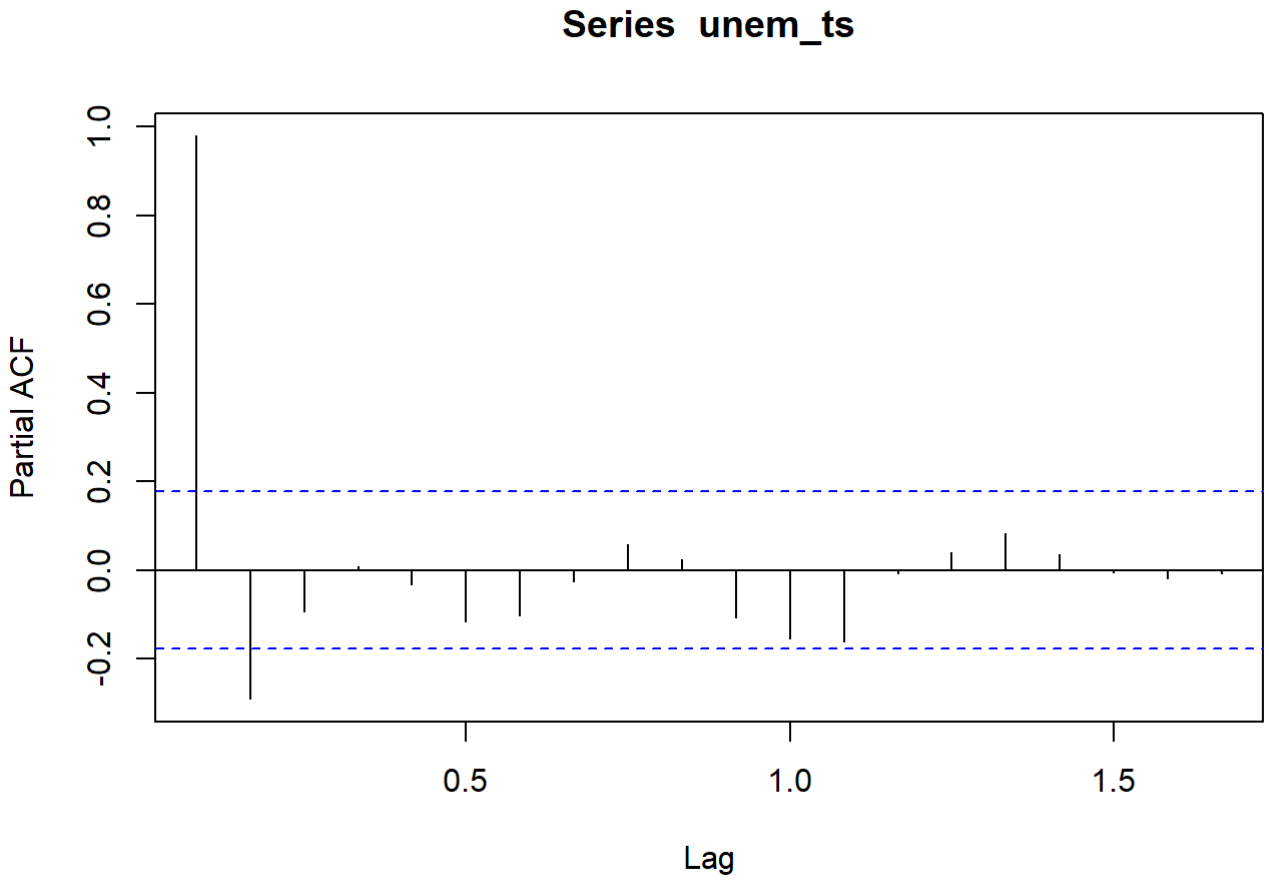
```
##
##  Augmented Dickey-Fuller Test
##
## data:  unem_ts
## Dickey-Fuller = -1.0359, Lag order = 4, p-value = 0.9291
## alternative hypothesis: stationary
```

O resultado é coerente, o p-value é maior que 0.05 e não violamos a hipótese nula de que a série não é estacionária. Afinal, taxas de desemprego estão sujeitas a flutuações cíclicas, mudanças estruturais e choques externos.

Naturalmente, precisamos estacionarizá-la. Caso não o fizermos, nosso modelo violará a condição de estacionariedade necessária no modelo, o que pode gerar relações espúrias. No entanto, iremos prosseguir com a série em nível, mas ao final da questão há modelos alternativos.

Autocorrelação Parcial

```
pacf(unem_ts)
```



Conclusão das Autocorrelações

Por que os padrões de correlação estão um tanto estranhos, e parece difícil discernir qual o número correto de p e q?

Primeiramente, a série não é estacionária.

Além disso, o dado que importamos não é, de fato, mensal, mas sim uma média móvel trimestral terminando no mês selecionado. Isso implica que 2/3 da taxa de determinado mês está repetido na taxa do mês seguinte. Portanto, há um fator de correlação sendo inserido exógenamente aos dados, não sendo relacionado ao processo gerador, mas à própria construção da base.

Contudo, se forçarmos uma análise meramente visual dos gráficos, podemos propor um modelo AR(2), tendo em vista a persistência do ACF e o cut-off em 2 do PACF.

Hipótese de modelo

Testaremos uma hipótese, utilizando a série em nível, de componente AR de ordem 2 e componente MA de ordem 0.

```
summary(
  Arima(
      unem_ts,
      order = c(2,
                0,
                0),
      method = "ML"
      )
)
```

```
## Series: unem_ts
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2    mean
##       1.6766  -0.6898  9.9366
## s.e.  0.0641   0.0646  1.4182
##
## sigma^2 = 0.05464:  log likelihood = 3
## AIC=1.99   AICc=2.34   BIC=13.21
##
## Training set error measures:
##                      ME      RMSE       MAE        MPE   MAPE      MASE
## Training set 0.01012313 0.2308641 0.1769599 0.04207688 1.7124 0.1011725
##                    ACF1
## Training set 0.1201889
```

O modelo apresenta uma log-likelihood baixa, o ajuste do modelo aos dados não parece ser muito bom. Vamos, então, iterar testes de ordens para o modelo arma e, com base no critério da informação, decidir quais são os valores de p e q mais adequados.

Critério da Informação

```
# Valores possíveis para "p" e "q"
p_values <- 1:2
q_values <- 1:12
```

```
# Inicializar vetores para armazenar valores de AIC e BIC
aic_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
bic_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
aicc_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
```

```
# Loop para calcular AIC e BIC para diferentes combinações de p e q

for (i in p_values){
  for (j in q_values){

    modelo <- Arima(
      unem_ts,
      order = c(i, 0, j),
      method = "ML"
      )

    aic_values[i,j] <- AIC(modelo)
    bic_values[i,j] <- BIC(modelo)
    aicc_values[i,j] <- modelo$aicc

    }
  }
```

```
# Encontrar as posições mínimas de AIC e BIC
min_aic_pos <- which(aic_values == min(aic_values), arr.ind = TRUE)
min_bic_pos <- which(bic_values == min(bic_values), arr.ind = TRUE)
min_aicc_pos <- which(aicc_values == min(aicc_values), arr.ind = TRUE)

print(min_aic_pos)
```

```
##      row col
## [1,]   2   5
```

```
print(min_bic_pos)
```

```
##      row col
## [1,]   1   2
```

```
print(min_aicc_pos)
```

```
##      row col
## [1,]   2   5
```

Realizaremos, agora os dois modelos encontrados.

AIC Minimizador:

```
summary(
  Arima(
      unem_ts,
      order = c(min_aic_pos[1,1],
                0,
                min_aic_pos[1,2]),
      method = "ML"
      )
)
```

```
## Series: unem_ts
## ARIMA(2,0,5) with non-zero mean
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produzidos
```

```
##         ar1     ar2     ma1     ma2     ma3     ma4      ma5     mean
##       2e-04  0.9998  2.0037  1.8228  0.9874  0.0178  -0.1549  10.7633
## s.e.  1e-04  0.0001  0.0892  0.1908  0.2267  0.1843   0.0909      NaN
##
## sigma^2 = 0.03941:  log likelihood = 25.46
## AIC=-32.91   AICc=-31.3   BIC=-7.68
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE    MAPE       MASE
## Training set 0.001082506 0.1918935 0.1496278 0.02672955 1.41769 0.08554604
##                     ACF1
## Training set -0.00741971
```

O modelo acima usa p = 2 e q = 5. Além disso, não apresenta log-likelihood negativa como o anterior.

BIC Minimizador:

```
summary(
  Arima(
      unem_ts,
      order = c(min_bic_pos[1,1],
                0,
                min_bic_pos[1,2]),
      method = "ML"
      )
)
```

```
## Series: unem_ts
## ARIMA(1,0,2) with non-zero mean
##
## Coefficients:
##          ar1     ma1     ma2    mean
##       0.9801  0.8046  0.7488  9.3472
## s.e.  0.0148  0.0652  0.0592  1.8495
##
## sigma^2 = 0.04282:  log likelihood = 17.63
## AIC=-25.27   AICc=-24.75   BIC=-11.25
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE       MASE
## Training set 0.01137891 0.2035195 0.1587408 0.07502154 1.527831 0.09075617
##                     ACF1
## Training set 0.08988894
```

O modelo acima, embora apresente uma log-likelihood menor, usa p = 1 e q = 2, sendo, então, consideravelmente mais parcimonioso que o primeiro.

AICc Minimizador:

```
summary(
  Arima(
      unem_ts,
      order = c(min_aicc_pos[1,1],
                0,
                min_aicc_pos[1,2]),
      method = "ML"
      )
)
```

```
## Series: unem_ts
## ARIMA(2,0,5) with non-zero mean
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produzidos
```

```
##         ar1     ar2     ma1     ma2     ma3     ma4      ma5     mean
##       2e-04  0.9998  2.0037  1.8228  0.9874  0.0178  -0.1549  10.7633
## s.e.  1e-04  0.0001  0.0892  0.1908  0.2267  0.1843   0.0909      NaN
##
## sigma^2 = 0.03941:  log likelihood = 25.46
## AIC=-32.91   AICc=-31.3   BIC=-7.68
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE    MAPE       MASE
## Training set 0.001082506 0.1918935 0.1496278 0.02672955 1.41769 0.08554604
##                     ACF1
## Training set -0.00741971
```

O modelo acima é o mesmo que o AIC Minimizador.

Por Auto Arima

A biblioteca forecast possui esta função “auto.arima” que estima o melhor modelo para a série temporal. Na prática, o código executa um método chamado Box Jenkins que também se baseia no critério da informação para eleger os melhores valores de p e q para o modelo.

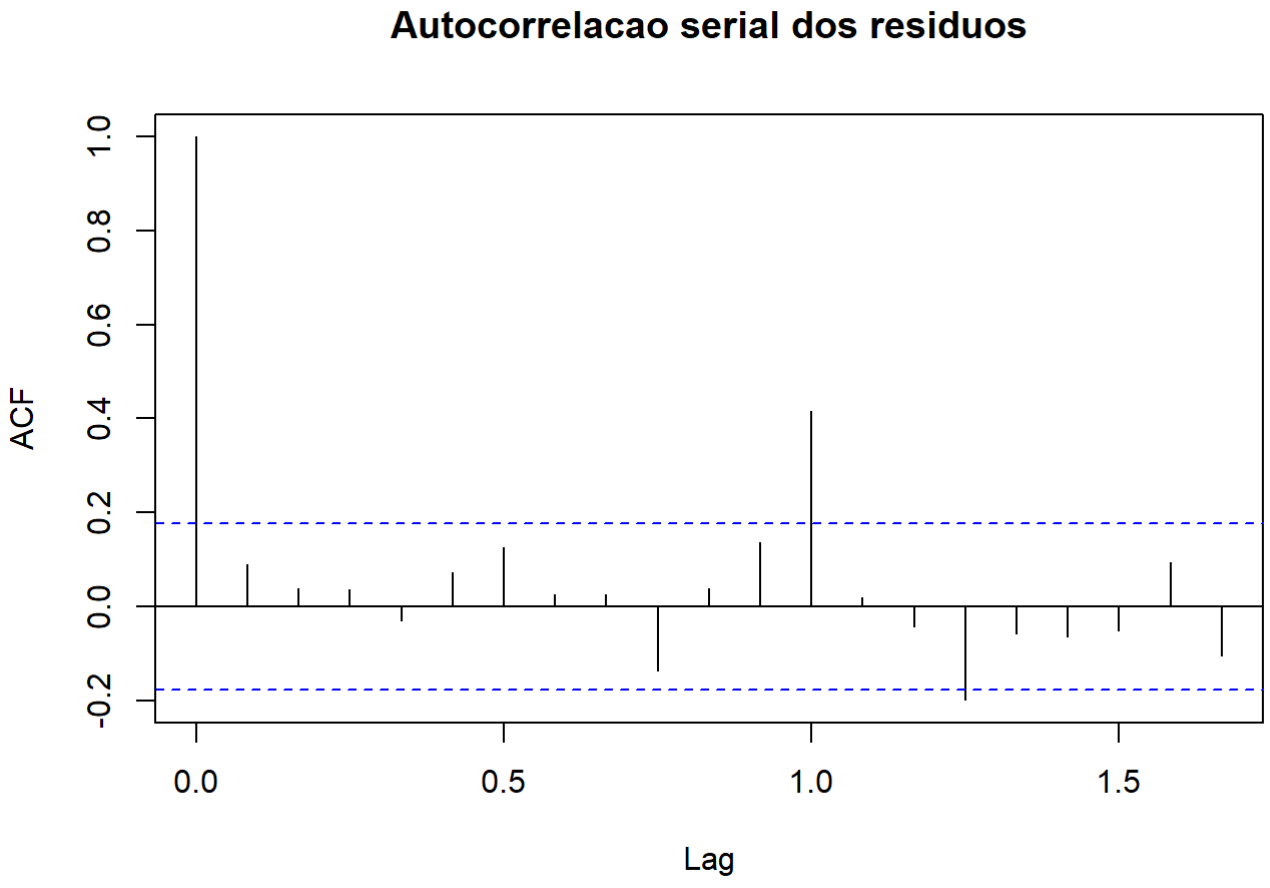
```
modelo_arma_auto <- auto.arima(
  unem_ts,
  d = 0,
  seasonal = FALSE
)

print(modelo_arma_auto)
```

```
## Series: unem_ts
## ARIMA(1,0,2) with non-zero mean
##
## Coefficients:
##          ar1     ma1     ma2    mean
##       0.9801  0.8046  0.7488  9.3409
## s.e.  0.0148  0.0652  0.0592  1.8524
##
## sigma^2 = 0.04282:  log likelihood = 17.63
## AIC=-25.27   AICc=-24.75   BIC=-11.25
```

É muito interessante observar que o resultado é igual ao modelo que obtivemos manualmente antes e julgamos ser o mais parcimonioso. Dessa forma, prosseguiremos com esse modelo.

Teste de diagnóstico pela autocorrelação serial do resíduo

```
# Resíduos do modelo ARMA
residuos <- residuals(modelo_arma_auto)

# Função de autocorrelação dos reíduos (ACF)
acf_residuos <- acf(residuos, plot = FALSE)

# Plot da ACF dos resíduos
plot(acf_residuos, main = "Autocorrelacao serial dos residuos")
```



Alguns erros com defasagens longas são significativamente correlacionados no modelo. No entanto, dadas as considereções que fizemos anteriormente quanto à estacionariedade da série, é possível que isso esteja afetando o modelo.

Análise de tendência central

Nível

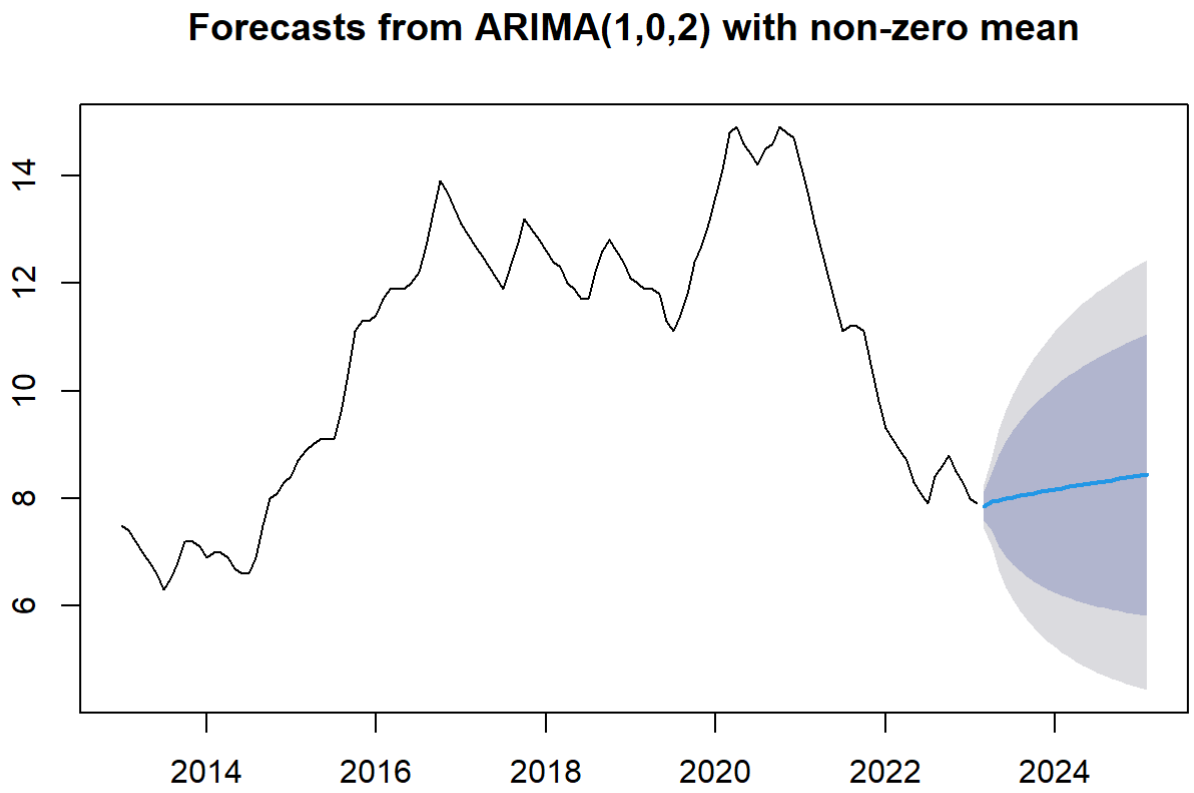
```
summary(unem_ts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.30    8.40   11.65   10.76   12.60   14.90
```

A maioria dos dados está concentrada entre aproximadamente 8.20 e 12.60, com a mediana próxima ao centro desse intervalo. A média é um pouco menor do que a mediana, sugerindo uma leve assimetria à esquerda, o que significa que há valores ligeiramente mais baixos puxando a média para baixo. No entanto, essa assimetria é leve, já que a diferença entre a média e a mediana não é significativa.

Previsão

```
plot(forecast(modelo_arma_auto))
```

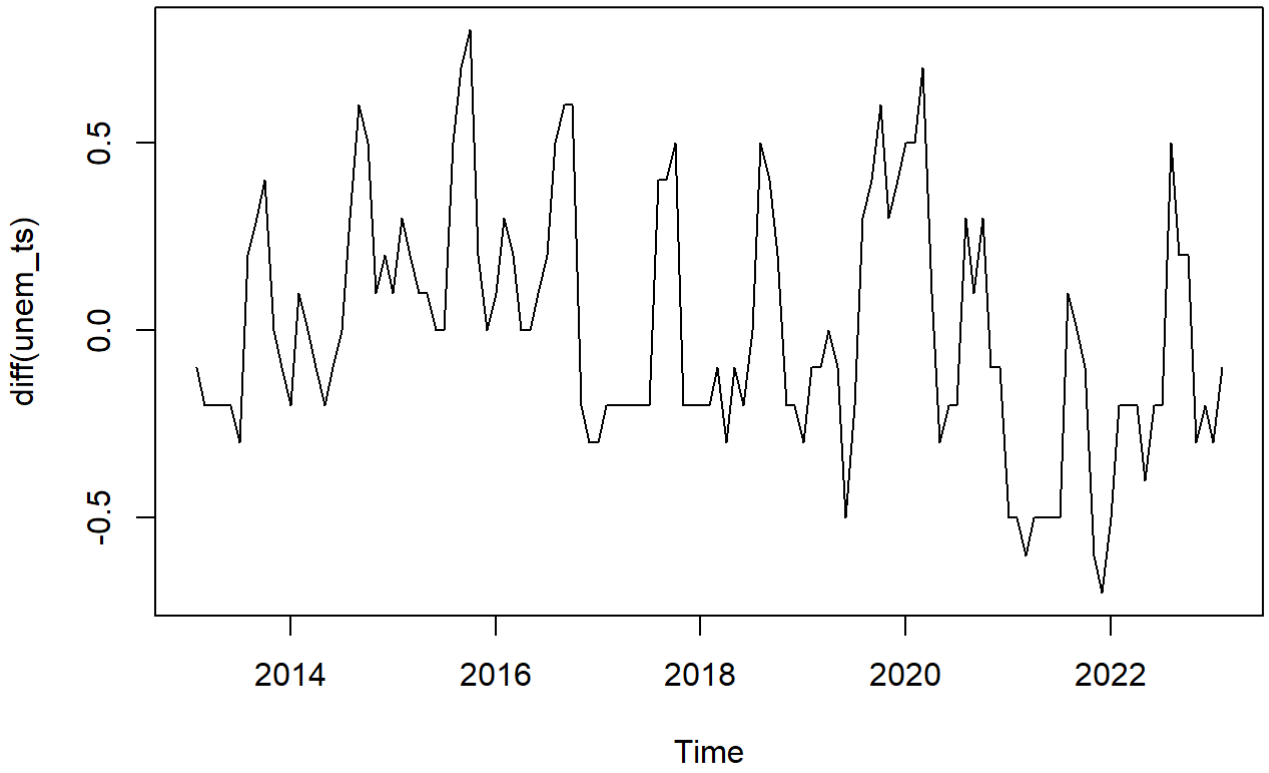


Extra

Primeira Diferença

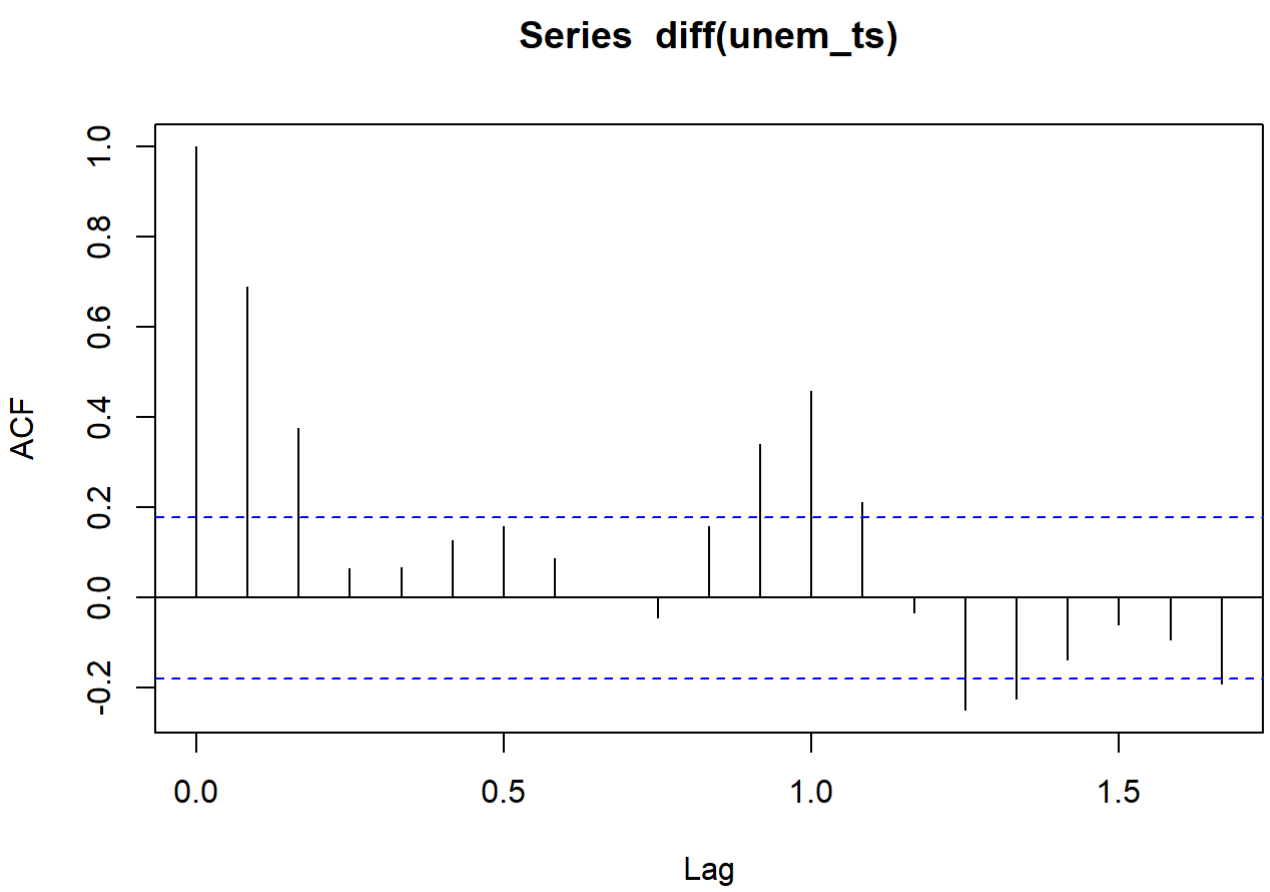
Plot

```
plot(diff(unem_ts))
```

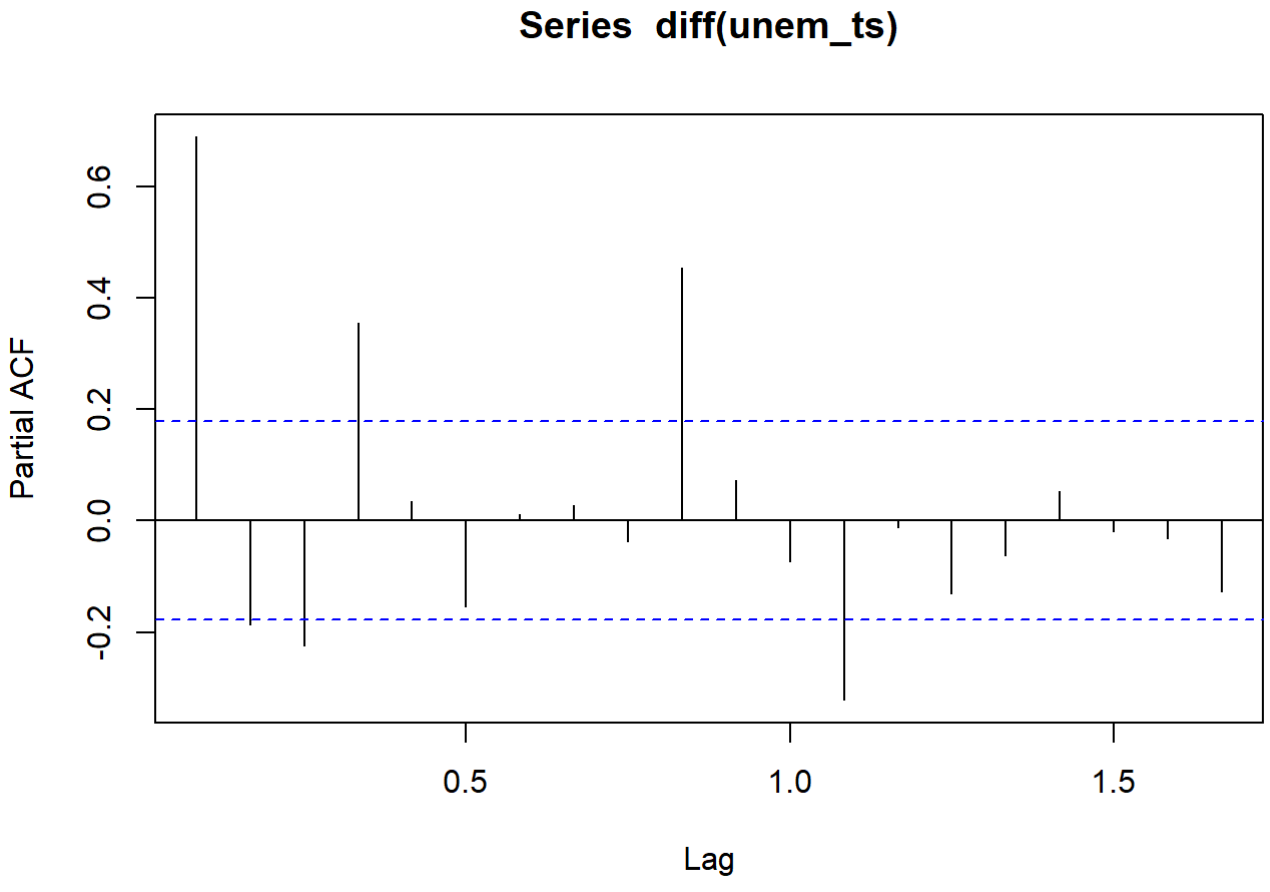


Autocorrelation Functions

```
acf(diff(unem_ts))
```




```
pacf(diff(unem_ts))
```



Critério da informação

```
# Valores possíveis para "p" e "q"
p_values <- 1:6
q_values <- 1:12
```

```
# Inicializar vetores para armazenar valores de AIC e BIC
aic_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
bic_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
aicc_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
```

```
# Loop para calcular AIC e BIC para diferentes combinações de p e q

for (i in p_values){
  for (j in q_values){

    modelo <- Arima(
      diff(unem_ts, differences = 1),
      order = c(i, 0, j),
      method = "ML"
      )

    aic_values[i,j] <- AIC(modelo)
    bic_values[i,j] <- BIC(modelo)
    aicc_values[i,j] <- modelo$aicc

    }
  }
```

```
# Encontrar as posições mínimas de AIC e BIC
min_aic_pos <- which(aic_values == min(aic_values), arr.ind = TRUE)
min_bic_pos <- which(bic_values == min(bic_values), arr.ind = TRUE)
min_aicc_pos <- which(aicc_values == min(aicc_values), arr.ind = TRUE)

print(min_aic_pos)
```

```
##      row col
## [1,]   3   9
```

```
print(min_bic_pos)
```

```
##      row col
## [1,]   3   9
```

```
print(min_aicc_pos)
```

```
##      row col
## [1,]   3   9
```

Todos valores de p e q foram iguais, ou seja, p = 3 e q = 9. Realizaremos, agora os dois modelos encontrados.

AIC e BIC Minimizador:

```
summary(
  modelo_diff <- Arima(
      diff(unem_ts, differences = 1),
      order = c(min_aic_pos[1,1],
                0,
                min_aic_pos[1,2]),
      method = "ML"
      )
)
```

```
## Series: diff(unem_ts, differences = 1)
## ARIMA(3,0,9) with non-zero mean
##
## Coefficients:
```

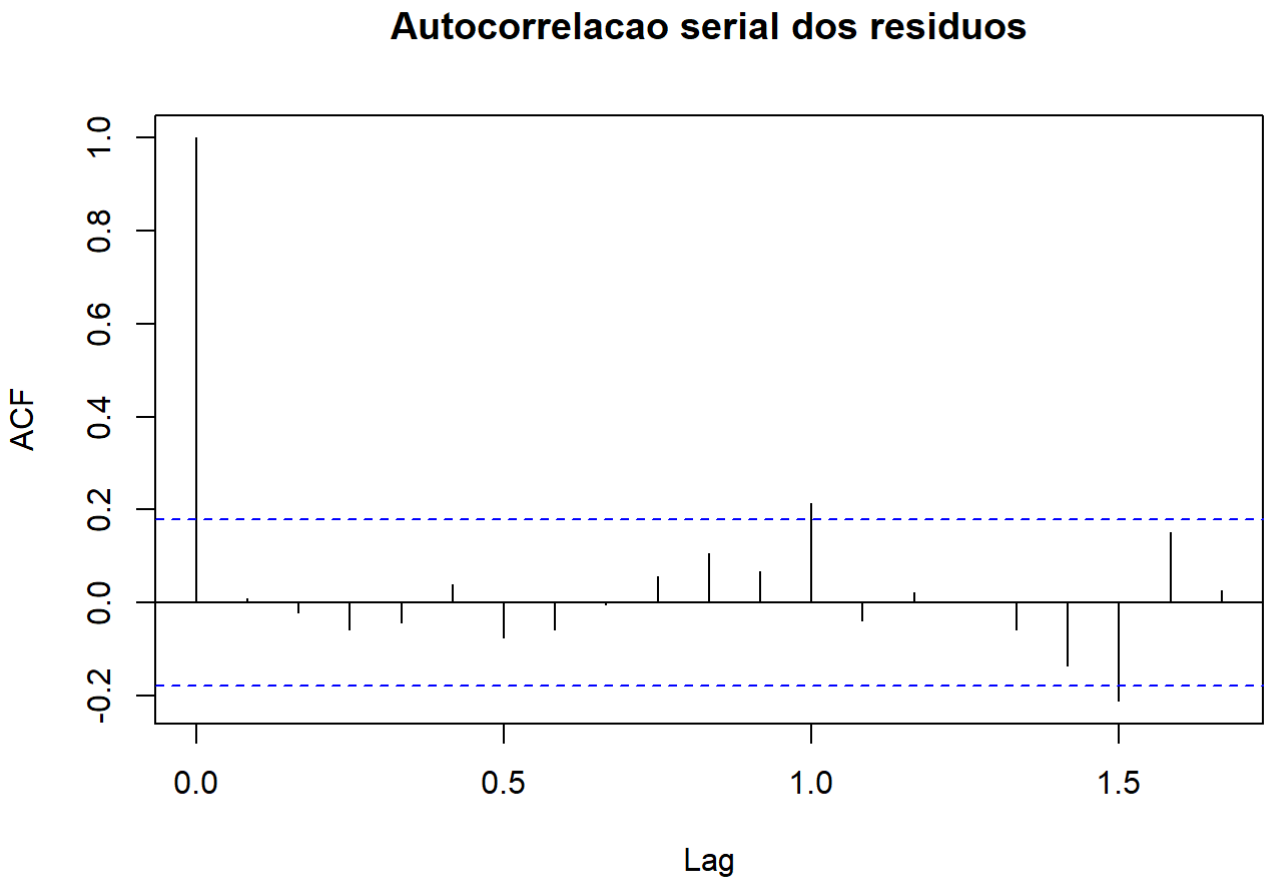
```
## Warning in sqrt(diag(x$var.coef)): NaNs produzidos
```

```
##           ar1      ar2  ar3     ma1     ma2     ma3     ma4     ma5     ma6
##       -0.0036  -0.0036   -1  0.7287  0.8769  1.8378  1.5824  1.5255  0.8635
## s.e.   0.0012   0.0013  NaN  0.0805  0.1052  0.1333  0.1948  0.2086  0.2239
##          ma7     ma8     ma9     mean
##       0.8610  0.6554  0.0235  -0.0071
## s.e.  0.1184  0.1232  0.1126   0.0642
##
## sigma^2 = 0.02358:  log likelihood = 50.48
## AIC=-72.96   AICc=-69   BIC=-33.82
##
## Training set error measures:
##                       ME      RMSE       MAE MPE MAPE      MASE        ACF1
## Training set 0.001794394 0.1450777 0.1143757 NaN  Inf 0.4686823 0.009335916
```

Teste de diagnóstico pela autocorrelação serial do

resíduo

```
# Resíduos do modelo ARMA
residuos <- residuals(modelo_diff)

# Função de autocorrelação dos reíduos (ACF)
acf_residuos <- acf(residuos, plot = FALSE)

# Plot da ACF dos resíduos
plot(acf_residuos, main = "Autocorrelacao serial dos residuos")
```



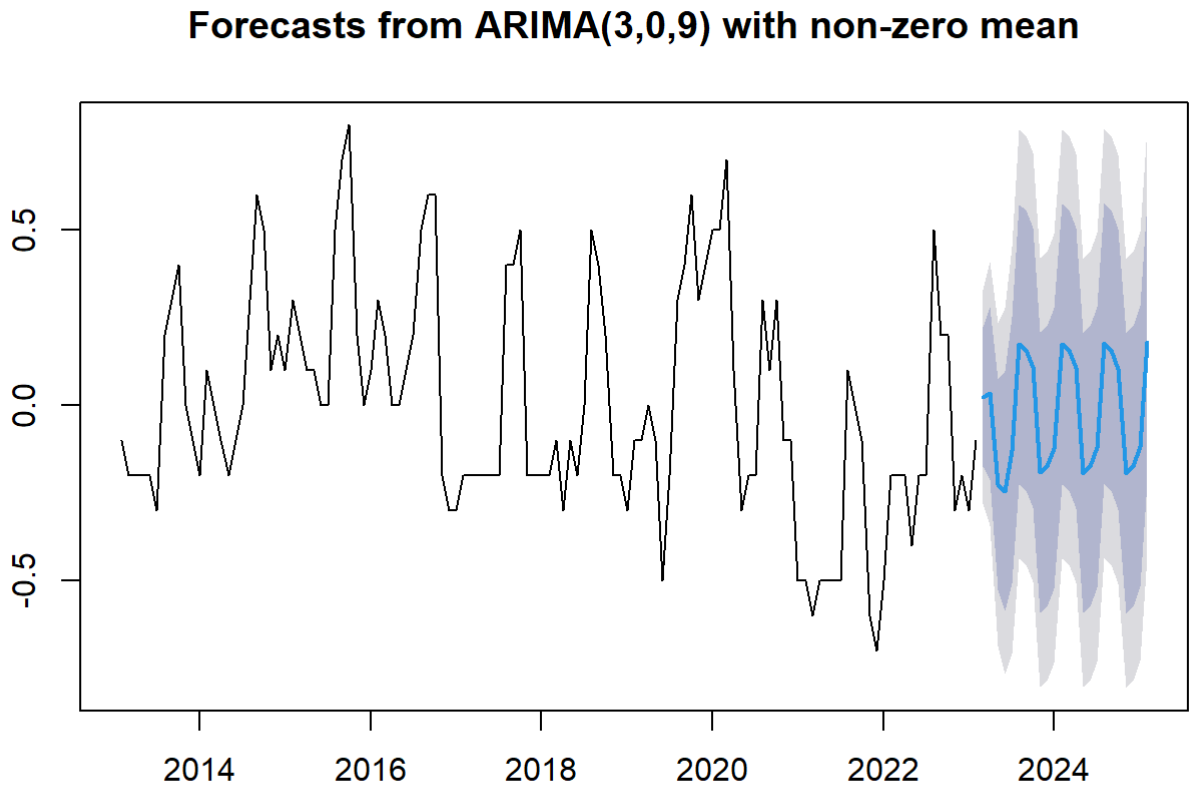
Análise de tendência central

```
summary(diff(unem_ts))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.700000 -0.200000 -0.100000  0.003306  0.200000  0.800000
```

Previsão

```
plot(forecast(modelo_diff))
```



Base Trimestral

Importação da Base

```
unemployment_tri <- get_sidra(x = 6468,
                    variable = 4099,
                    period = "all",
                    classific = "all")  %>%
  select(
    8,9,5
  ) %>%
  rename(
    "tri_cod" = "Trimestre (Código)",
    "trimestre" = "Trimestre",
    "taxa" = "Valor"
  ) %>%
  filter(row_number(.)>4) #pula dados de 2012
```

```
## Considering all categories once 'classific' was set to 'all' (default)
```

Time Series

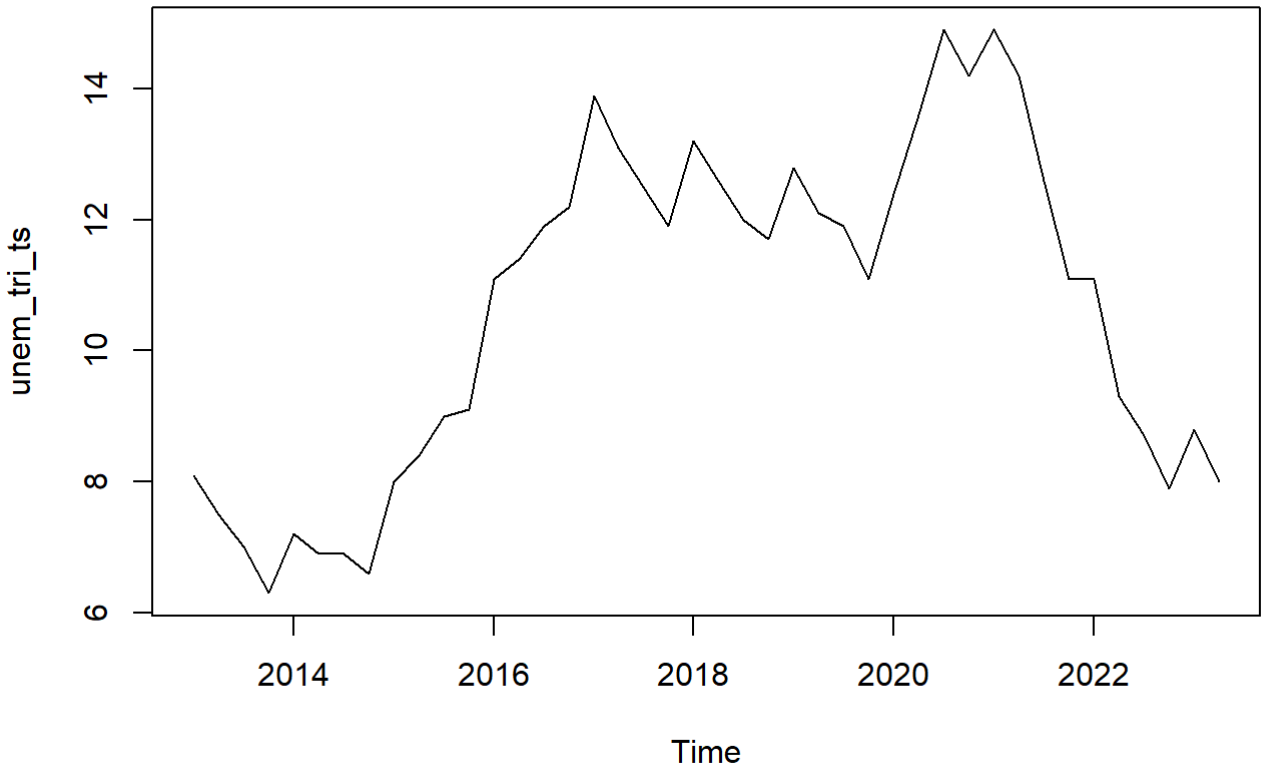
Primeiro, a taxa de desemprego no período.

```
unem_tri_ts <- ts(
  data = unemployment_tri$taxa,
  start = c(2013,1),
  frequency = 4
    )
```

Com a série trimestral temos poucos graus de liberdade, o que pode afetar a qualidade dos modelos.

Plots

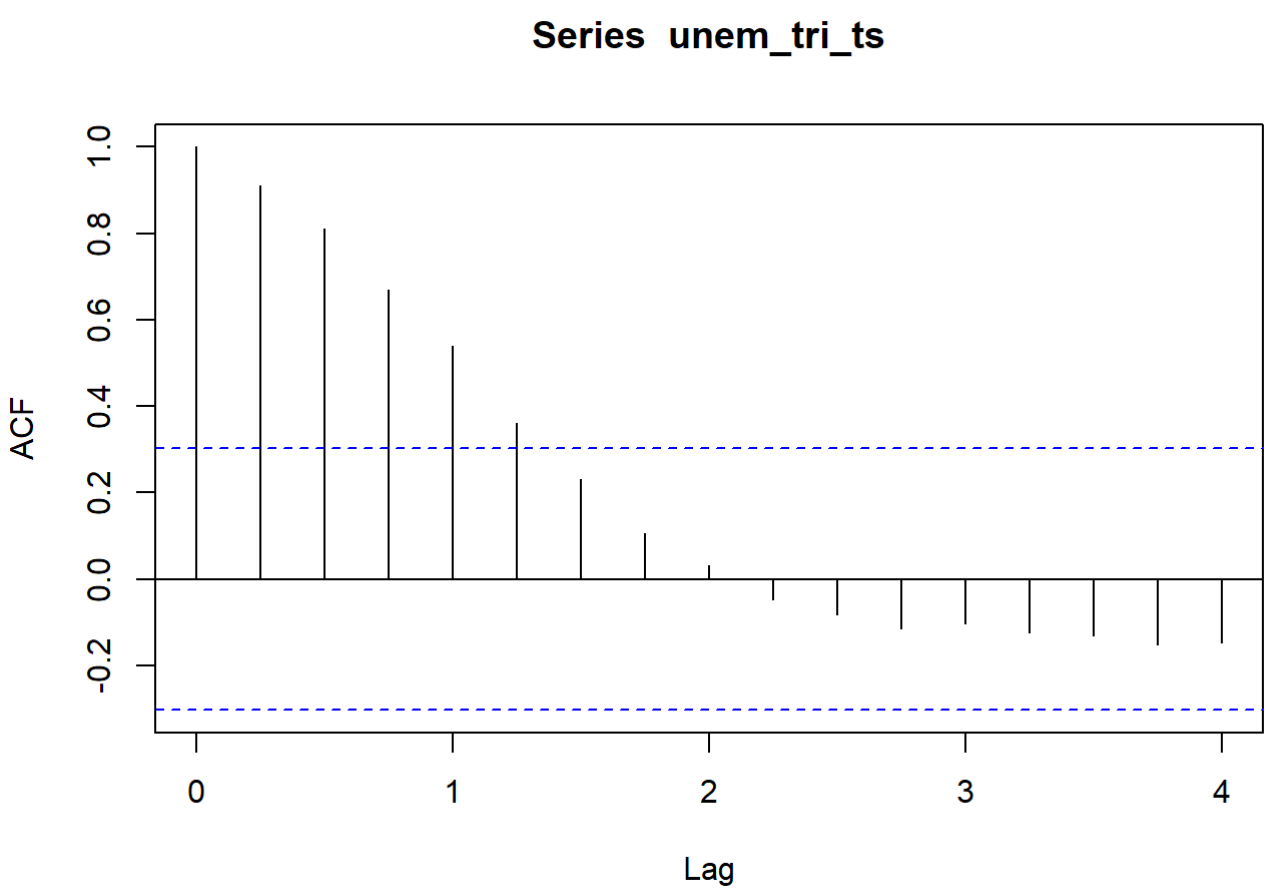
```
plot(
  unem_tri_ts
)
```



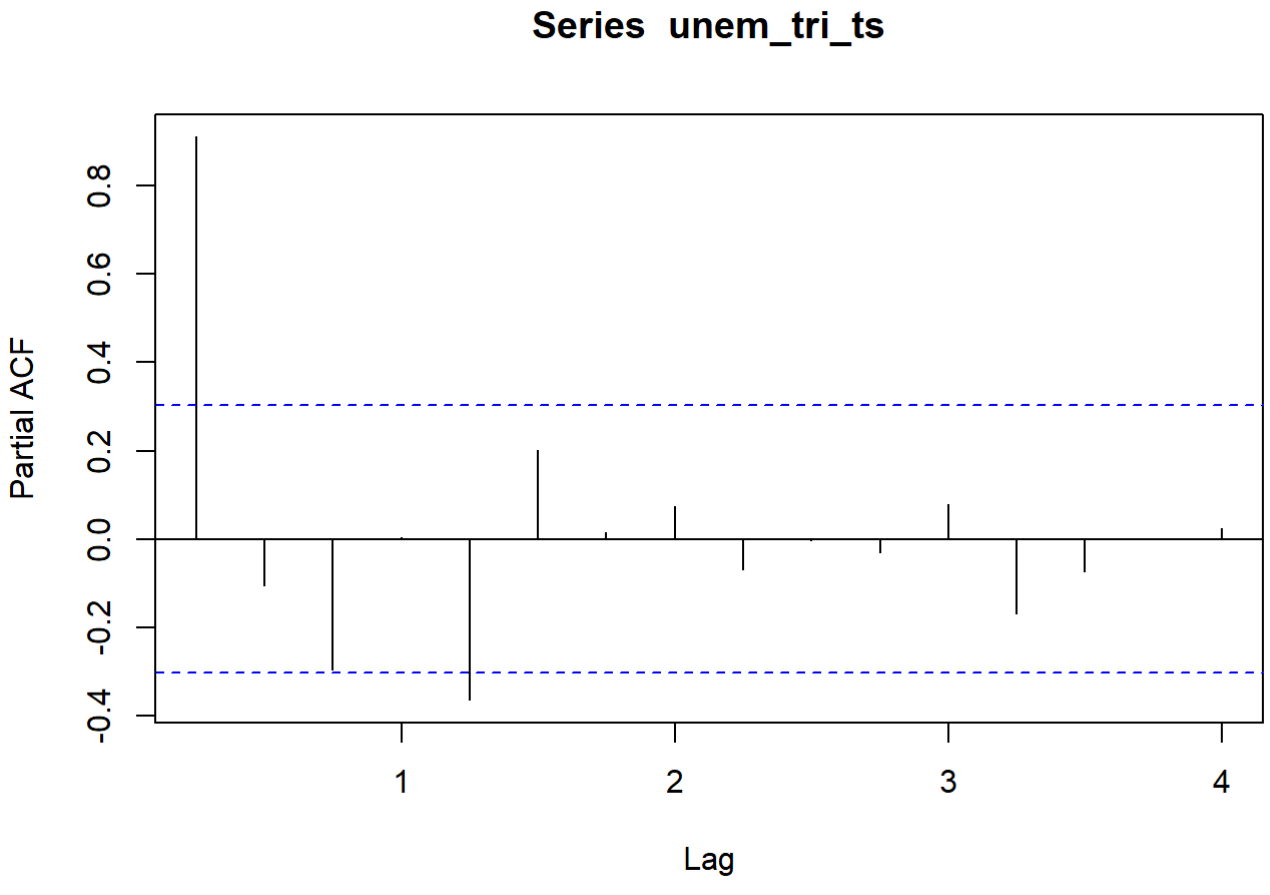
Autocorrelation Functions

Série Inicial

```
acf(unem_tri_ts)
```




```
pacf(unem_tri_ts)
```



Hipótese de Modelo ARMA na Série Nível

```
ordem_ar <- 1
ordem_ma <- 0
```

```
modelo_arma <- Arima(
  unem_tri_ts,
  order = c(ordem_ar, 0, ordem_ma),
  method = "ML"
  )

summary(modelo_arma)
```

```
## Series: unem_tri_ts
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.9331  9.6237
## s.e.  0.0470  1.6841
##
## sigma^2 = 0.858:  log likelihood = -56.38
## AIC=118.76   AICc=119.39   BIC=123.97
##
## Training set error measures:
##                      ME      RMSE       MAE         MPE     MAPE     MASE
## Training set 0.05709093 0.9039857 0.7515579 -0.2512439 7.253507 0.434691
##                    ACF1
## Training set 0.1296252
```

Série em Nível: Melhor Modelo

```
# Valores possíveis para "p" e "q"
p_values <- 1:6
q_values <- 1:6
```

```
# Inicializar vetores para armazenar valores de AIC e BIC
aic_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
bic_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
aicc_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
```

```
# Loop para calcular AIC e BIC para diferentes combinações de p e q

for (i in p_values){
  for (j in q_values){

    modelo <- Arima(
      unem_tri_ts,
      order = c(i, 0, j),
      method = "ML")

    aic_values[i,j] <- AIC(modelo)
    bic_values[i,j] <- BIC(modelo)
    aicc_values[i,j] <- modelo$aicc

    }
  }
```

```
# Encontrar as posições mínimas de AIC e BIC
min_aic_pos <- which(aic_values == min(aic_values), arr.ind = TRUE)
min_bic_pos <- which(bic_values == min(bic_values), arr.ind = TRUE)
min_aicc_pos <- which(aicc_values == min(aicc_values), arr.ind = TRUE)

print(min_aic_pos)
```

```
##      row col
## [1,]   5   5
```

```
print(min_bic_pos)
```

```
##      row col
## [1,]   5   1
```

```
print(min_aicc_pos)
```

```
##      row col
## [1,]   5   1
```

Realizaremos, agora os dois modelos encontrados.

AIC Minimizador:

```
summary(
  Arima(
      unem_tri_ts,
      order = c(min_aic_pos[1,1],
                0,
                min_aic_pos[1,2]),
      method = "ML"
      )
)
```

```
## Series: unem_tri_ts
## ARIMA(5,0,5) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ar5     ma1      ma2     ma3
##       0.9380  -0.0243  -0.0256  0.9755  -0.9612  0.9589  -0.0290  0.0293
## s.e.  0.0436   0.0294   0.0312  0.0318   0.0311  0.3120   0.2037  0.2176
##           ma4     ma5     mean
##       -0.9622  -0.986  10.5801
## s.e.   0.2004   0.341   0.4665
##
## sigma^2 = 0.1868:  log likelihood = -29.98
## AIC=83.96   AICc=94.72   BIC=104.81
##
## Training set error measures:
##                       ME      RMSE       MAE         MPE     MAPE      MASE
## Training set -0.01363945 0.3713536 0.2764788 -0.3306049 2.588897 0.1599116
##                     ACF1
## Training set 0.02151828
```

BIC Minimizador:

```
summary(
  Arima(
      unem_tri_ts,
      order = c(min_bic_pos[1,1],
                0,
                min_bic_pos[1,2]),
      method = "ML"
      )
)
```

```
## Series: unem_tri_ts
## ARIMA(5,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ar2      ar3     ar4      ar5     ma1     mean
##       0.7597  0.2414  -0.1904  0.6693  -0.6586  0.9894  10.5065
## s.e.  0.1155  0.1312   0.1300  0.1331   0.1148  0.0328   0.8693
##
## sigma^2 = 0.2876:  log likelihood = -35.01
## AIC=86.03   AICc=90.39   BIC=99.93
##
## Training set error measures:
##                       ME      RMSE       MAE         MPE     MAPE      MASE
## Training set 0.009392715 0.4895708 0.3881569 -0.2352118 3.745277 0.2245047
##                     ACF1
## Training set 0.06456771
```

AICc Minimizador

```
summary(
  Arima(
      unem_tri_ts,
      order = c(min_aicc_pos[1,1],
                0,
                min_aicc_pos[1,2]),
      method = "ML"
      )
)
```

```
## Series: unem_tri_ts
## ARIMA(5,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ar2      ar3     ar4      ar5     ma1     mean
##       0.7597  0.2414  -0.1904  0.6693  -0.6586  0.9894  10.5065
## s.e.  0.1155  0.1312   0.1300  0.1331   0.1148  0.0328   0.8693
##
## sigma^2 = 0.2876:  log likelihood = -35.01
## AIC=86.03   AICc=90.39   BIC=99.93
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set 0.009392715 0.4895708 0.3881569 -0.2352118 3.745277 0.2245047
##                     ACF1
## Training set 0.06456771
```

Melhor Modelo Minimizador:

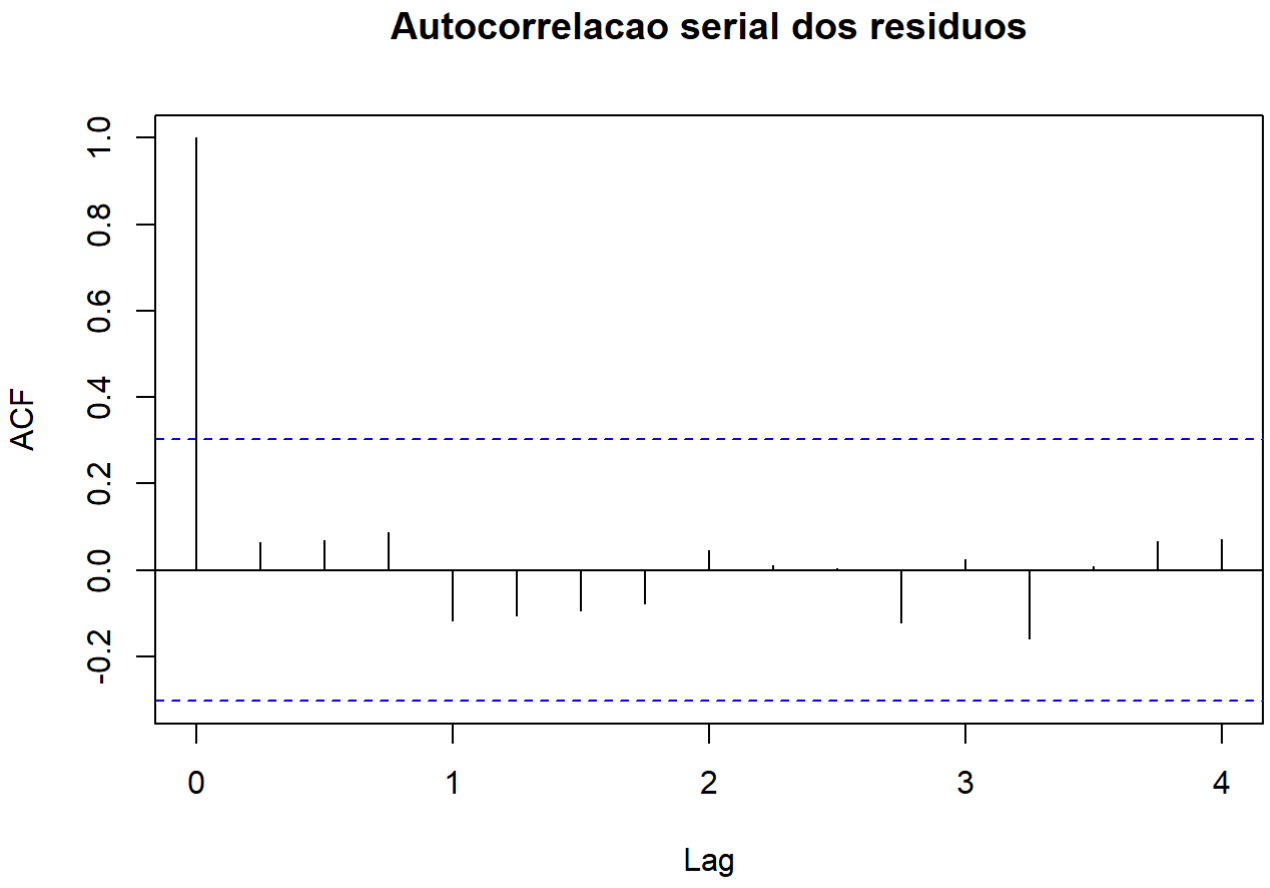
Salvamos o modelo minimizador escolhido pelo critério da parcimômia.

Modelo minimizador

```
modelo_arma_min <- Arima(
      unem_tri_ts,
      order = c(5,
                0,
                1),
      method = "ML"
      )

summary(modelo_arma_min)
```

```
## Series: unem_tri_ts
## ARIMA(5,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ar2      ar3     ar4      ar5     ma1     mean
##       0.7597  0.2414  -0.1904  0.6693  -0.6586  0.9894  10.5065
## s.e.  0.1155  0.1312   0.1300  0.1331   0.1148  0.0328   0.8693
##
## sigma^2 = 0.2876:  log likelihood = -35.01
## AIC=86.03   AICc=90.39   BIC=99.93
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set 0.009392715 0.4895708 0.3881569 -0.2352118 3.745277 0.2245047
##                    ACF1
## Training set 0.06456771
```

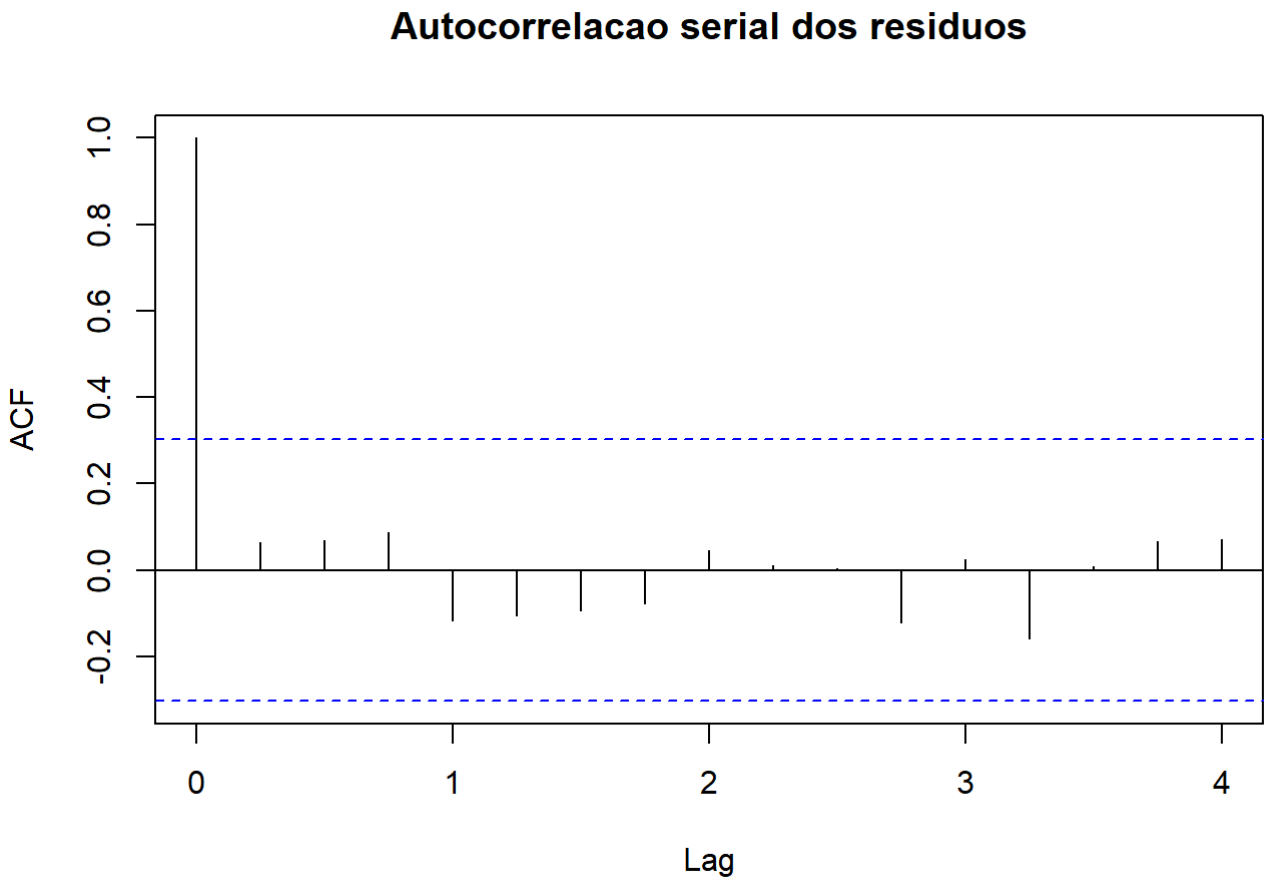
```
# Resíduos do modelo ARMA
residuos <- residuals(modelo_arma_min)

# Função de autocorrelação dos reíduos (ACF)
acf_residuos <- acf(residuos, plot = FALSE)

# Plot da ACF dos resíduos
plot(acf_residuos, main = "Autocorrelacao serial dos residuos")
```



Teste de diagnóstico pela autocorrelação serial do resíduo

Primeira Diferença

```
# Resíduos do modelo ARMA
residuos <- residuals(modelo_arma_min)

# Função de autocorrelação dos reíduos (ACF)
acf_residuos <- acf(residuos, plot = FALSE)

# Plot da ACF dos resíduos
plot(acf_residuos, main = "Autocorrelacao serial dos residuos")
```



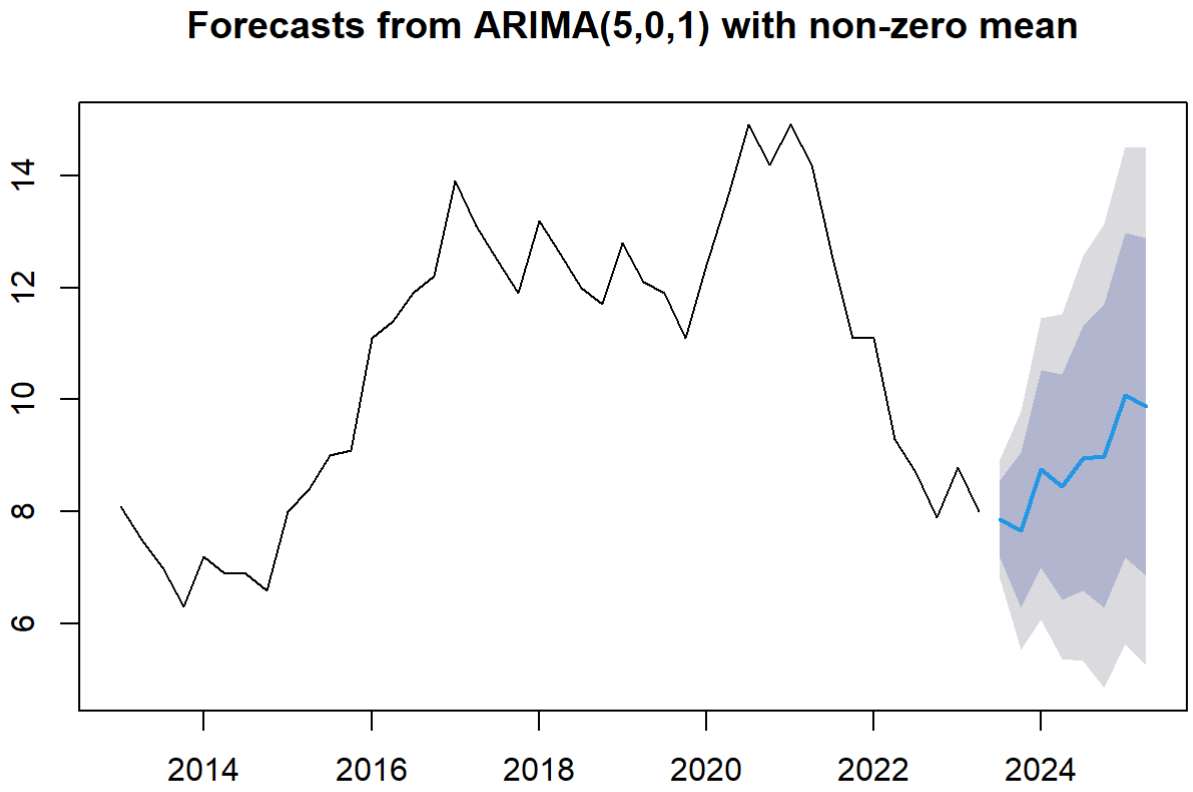
Análise de tendência central

```
summary(unem_tri_ts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   6.300   8.175  11.250  10.669  12.575  14.900
```

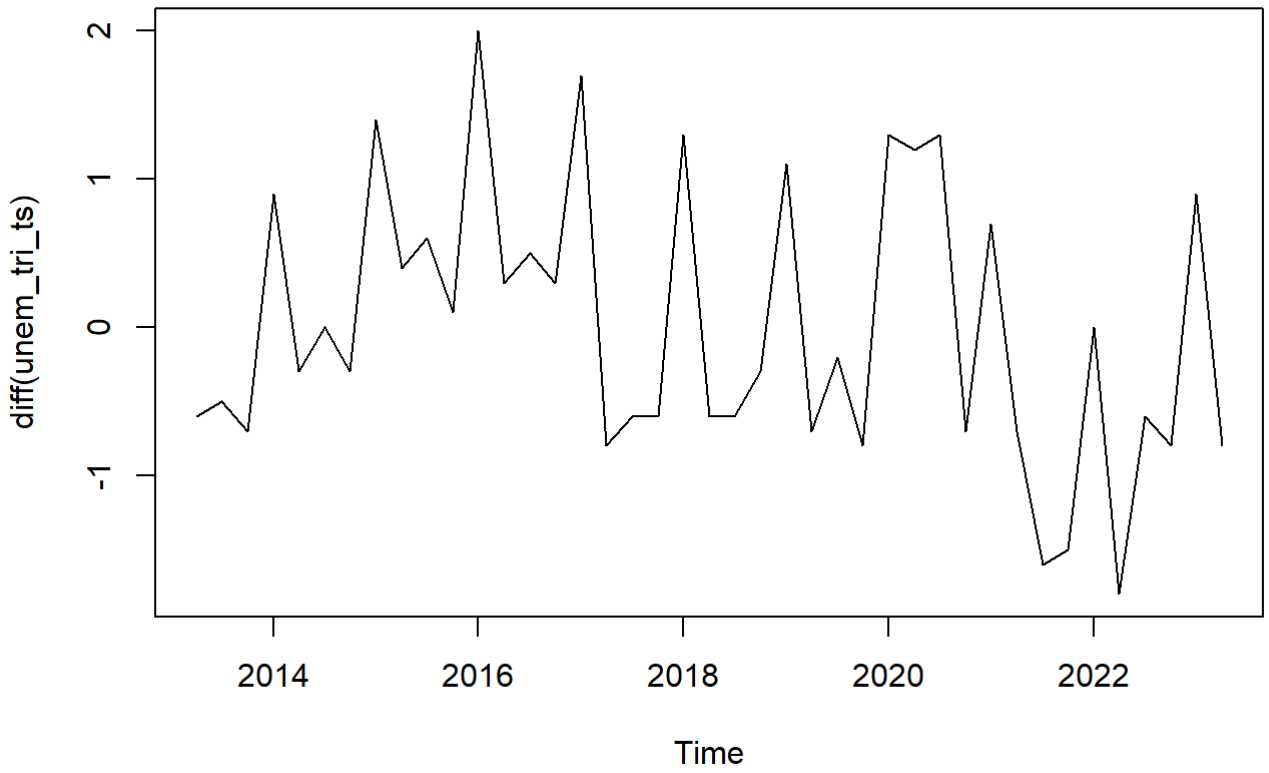
Previsão

```
plot(forecast(modelo_arma_min))
```

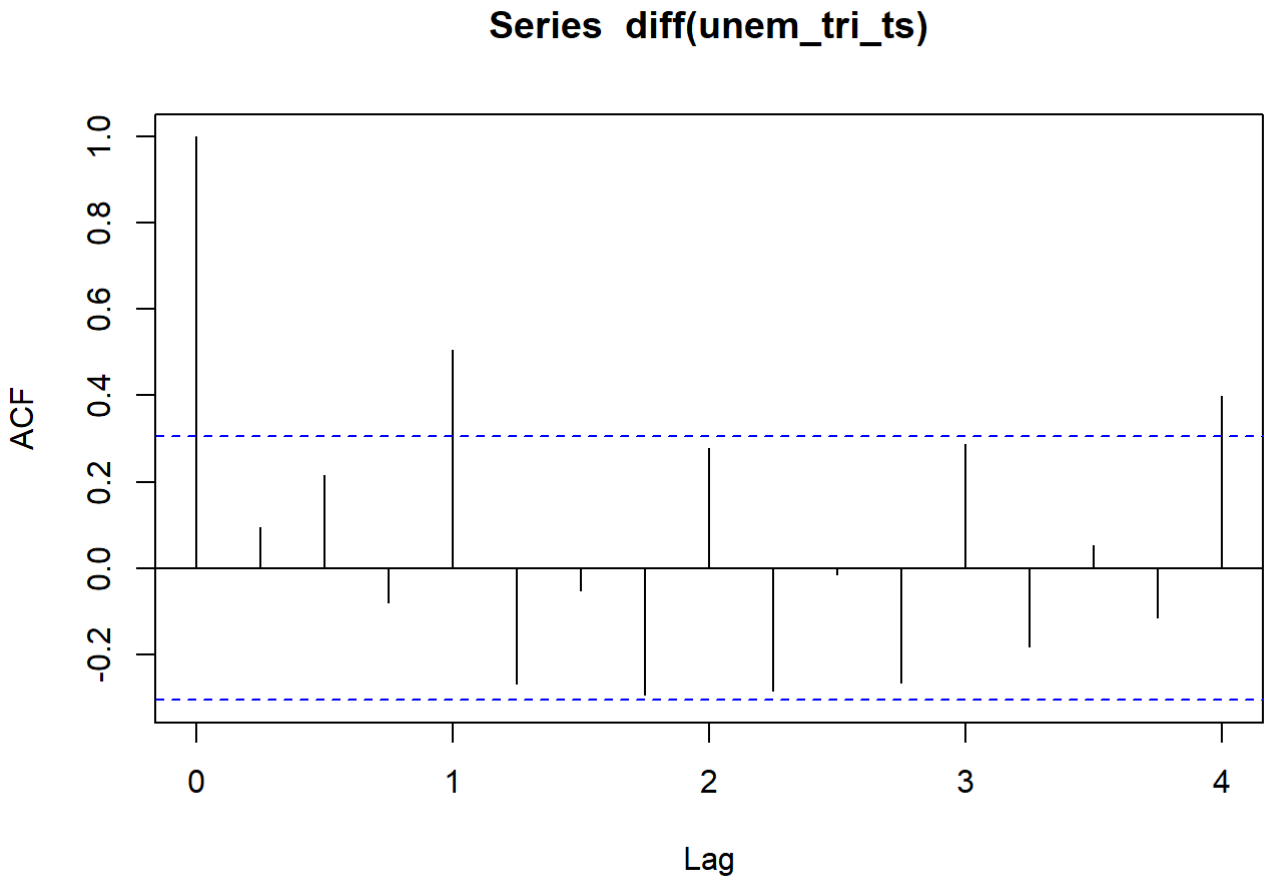


Série da Primeira diferença:

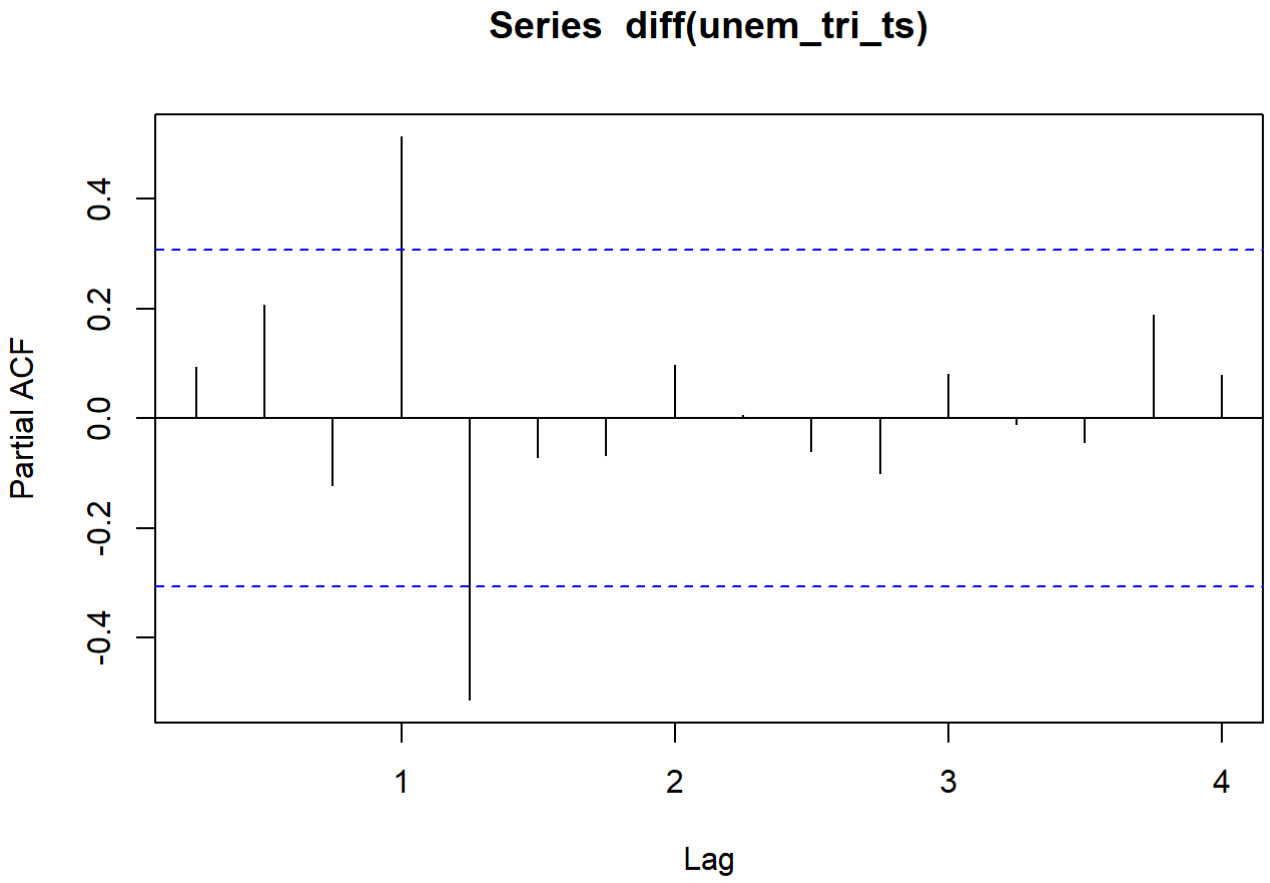
```
plot(
  diff(unem_tri_ts)
)
```




```
acf(diff(unem_tri_ts))
```



```
pacf(diff(unem_tri_ts))
```



Conclusão das Autocorrelações

Parece que a série da primeira diferença gera resultados melhores. O pacote forecast pode nos ajudar a decidir o número ideal de diferenças.

```
ndiffs(x=unem_tri_ts)
```

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

Como esperávamos.

Hipótese de Modelo ARMA na Primeira Diferença

```
ordem_ar <- 5
ordem_ma <- 4
```

```
modelo_arma_diff <- Arima(
  diff(unem_tri_ts),
  order = c(ordem_ar, 0, ordem_ma),
  method = "ML"
  )

summary(modelo_arma_diff)
```

```
## Series: diff(unem_tri_ts)
## ARIMA(5,0,4) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ar5     ma1     ma2     ma3     ma4
##       0.0710  -0.1404  -0.4874  0.4532  -0.2708  0.8688  0.0817  0.8820  0.6784
## s.e.  0.2689   0.1618   0.1900  0.1789   0.2345  0.3087  0.2074  0.3554  0.3651
##         mean
##       0.0081
## s.e.  0.1832
##
## sigma^2 = 0.289:  log likelihood = -32.45
## AIC=86.91   AICc=96.01   BIC=105.76
##
## Training set error measures:
##                        ME      RMSE   MAE MPE MAPE     MASE       ACF1
## Training set -0.005426039 0.4674677 0.382 NaN  Inf 0.560873 0.01268473
```

Primeira Diferença: Melhor modelo

```
# Valores possíveis para "p" e "q"
p_values <- 1:8
q_values <- 1:8
```

```
# Inicializar vetores para armazenar valores de AIC e BIC
aic_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
bic_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
aicc_values <- matrix(NA,
                     nrow = length(p_values),
                     ncol = length(q_values))
```

```
# Loop para calcular AIC e BIC para diferentes combinações de p e q

for (i in p_values){
  for (j in q_values){

    modelo <- Arima(
      diff(unem_tri_ts),
      order = c(i, 0, j),
      method = "ML"
      )

    aic_values[i,j] <- AIC(modelo)
    bic_values[i,j] <- BIC(modelo)
    aicc_values[i,j] <- modelo$aicc

    }
  }
```

```
# Encontrar as posições mínimas de AIC e BIC
min_aic_pos <- which(aic_values == min(aic_values), arr.ind = TRUE)
min_bic_pos <- which(bic_values == min(bic_values), arr.ind = TRUE)
min_aicc_pos <- which(aicc_values == min(aicc_values), arr.ind = TRUE)

print(min_aic_pos)
```

```
##      row col
## [1,]   4   4
```

```
print(min_bic_pos)
```

```
##      row col
## [1,]   4   1
```

```
print(min_aicc_pos)
```

```
##      row col
## [1,]   4   1
```

Realizaremos, agora os três modelos encontrados.

AIC Minimizador:

```
summary(
  Arima(
      diff(unem_ts, differences = 1),
      order = c(min_aic_pos[1,1],
                0,
                min_aic_pos[1,2]),
      method = "ML"
      )
)
```

```
## Series: diff(unem_ts, differences = 1)
## ARIMA(4,0,4) with non-zero mean
##
## Coefficients:
##          ar1     ar2      ar3     ar4     ma1     ma2      ma3      ma4
##       0.1357  0.6467  -0.2609  0.2277  0.8088  0.0521  -0.3284  -0.5626
## s.e.  0.2388  0.2037   0.1336  0.1273  0.2279  0.3712   0.3076   0.1580
##          mean
##       -0.0017
## s.e.   0.0667
##
## sigma^2 = 0.0408:  log likelihood = 24.49
## AIC=-28.99   AICc=-26.99   BIC=-1.03
##
## Training set error measures:
##                        ME      RMSE     MAE MPE MAPE      MASE       ACF1
## Training set 0.0005002799 0.1943416 0.14688 NaN  Inf 0.6018767 0.01450573
```

BIC Minimizador:

```
summary(
  Arima(
      diff(unem_ts, differences = 1),
      order = c(min_bic_pos[1,1],
                0,
                min_bic_pos[1,2]),
      xreg = 1:length(diff(unem_ts, differences = 1)),
      method = "ML"
      )
)
```

```
## Series: diff(unem_ts, differences = 1)
## Regression with ARIMA(4,0,1) errors
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ma1  intercept     xreg
##       0.8351  -0.0037  -0.4961  0.3299  -0.0103     0.1411  -0.0023
## s.e.  0.1927   0.1655   0.1042  0.0953   0.1941     0.1082   0.0015
##
## sigma^2 = 0.04545:  log likelihood = 18.27
## AIC=-20.54   AICc=-19.26   BIC=1.82
##
## Training set error measures:
##                       ME      RMSE       MAE MPE MAPE      MASE        ACF1
## Training set 0.001707879 0.2069355 0.1575759 NaN  Inf 0.6457059 0.004214391
```

AICc Minimizador

```
summary(
  Arima(
      diff(unem_ts, differences = 1),
      order = c(min_aicc_pos[1,1],
                0,
                min_aicc_pos[1,2]),
      xreg = 1:length(diff(unem_ts, differences = 1)),
      method = "ML"
      )
)
```

```
## Series: diff(unem_ts, differences = 1)
## Regression with ARIMA(4,0,1) errors
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ma1  intercept     xreg
##       0.8351  -0.0037  -0.4961  0.3299  -0.0103     0.1411  -0.0023
## s.e.  0.1927   0.1655   0.1042  0.0953   0.1941     0.1082   0.0015
##
## sigma^2 = 0.04545:  log likelihood = 18.27
## AIC=-20.54   AICc=-19.26   BIC=1.82
##
## Training set error measures:
##                       ME      RMSE       MAE MPE MAPE      MASE        ACF1
## Training set 0.001707879 0.2069355 0.1575759 NaN  Inf 0.6457059 0.004214391
```

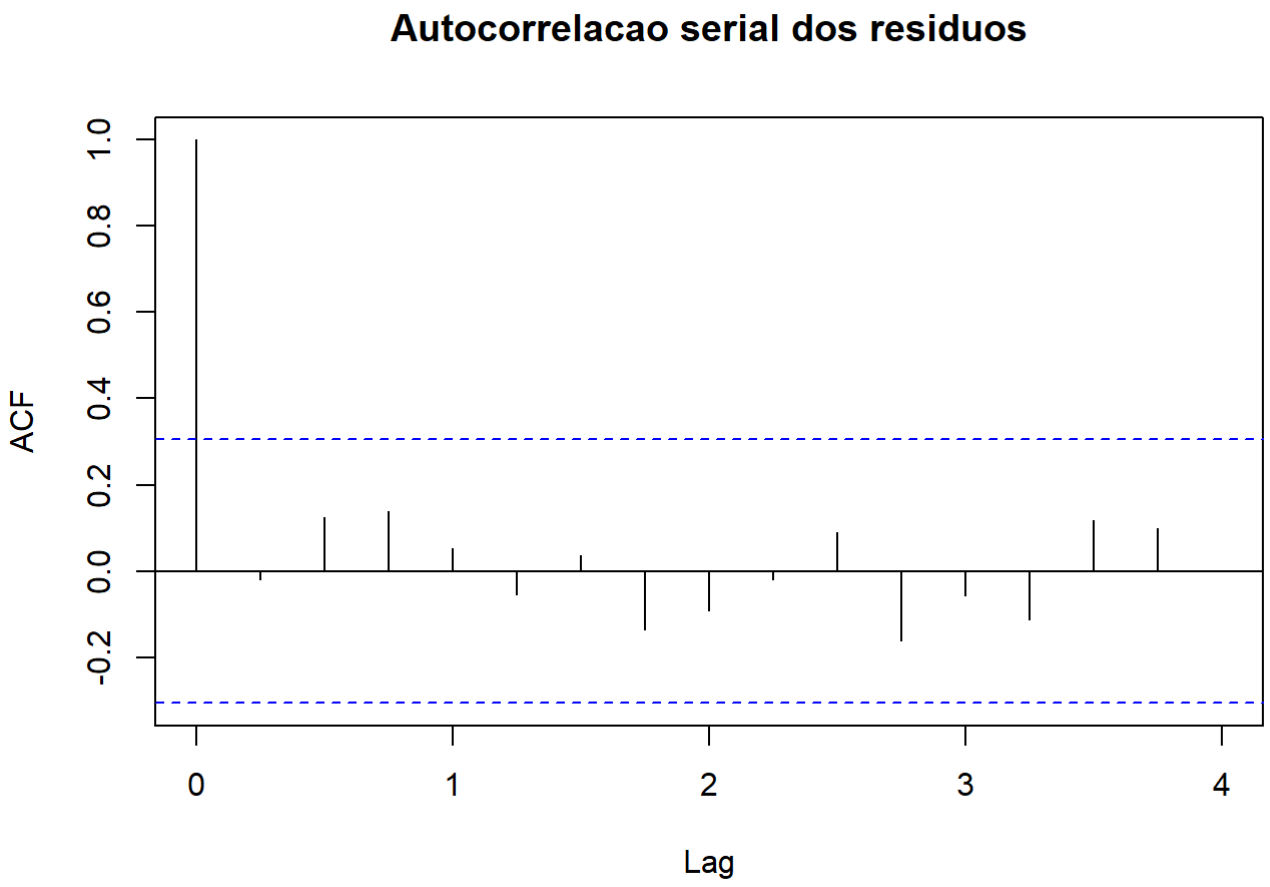
Melhor Modelo Minimizador:

Salvamos o modelo minimizador escolhido pelo critério da parcimômia.

```
modelo_arma_diff_min <- Arima(
      diff(unem_tri_ts),
      order = c(4,
                0,
                4),
      method = "ML"
      )

summary(modelo_arma_diff_min)
```

```
## Series: diff(unem_tri_ts)
## ARIMA(4,0,4) with non-zero mean
##
## Coefficients:
##           ar1      ar2      ar3     ar4     ma1     ma2     ma3     ma4
##       -0.7083  -0.7085  -0.7168  0.2834  1.8027  1.7512  1.8101  0.8657
## s.e.   0.2260   0.2245   0.2145  0.2150  0.2874  0.3543  0.3291  0.2705
##          mean
##       -0.0045
## s.e.   0.1683
##
## sigma^2 = 0.2392:  log likelihood = -31.27
## AIC=82.53   AICc=89.87   BIC=99.67
##
## Training set error measures:
##                        ME      RMSE       MAE MPE MAPE      MASE        ACF1
## Training set -0.006956624 0.4320483 0.3329619 NaN  Inf 0.4888726 -0.01989377
```

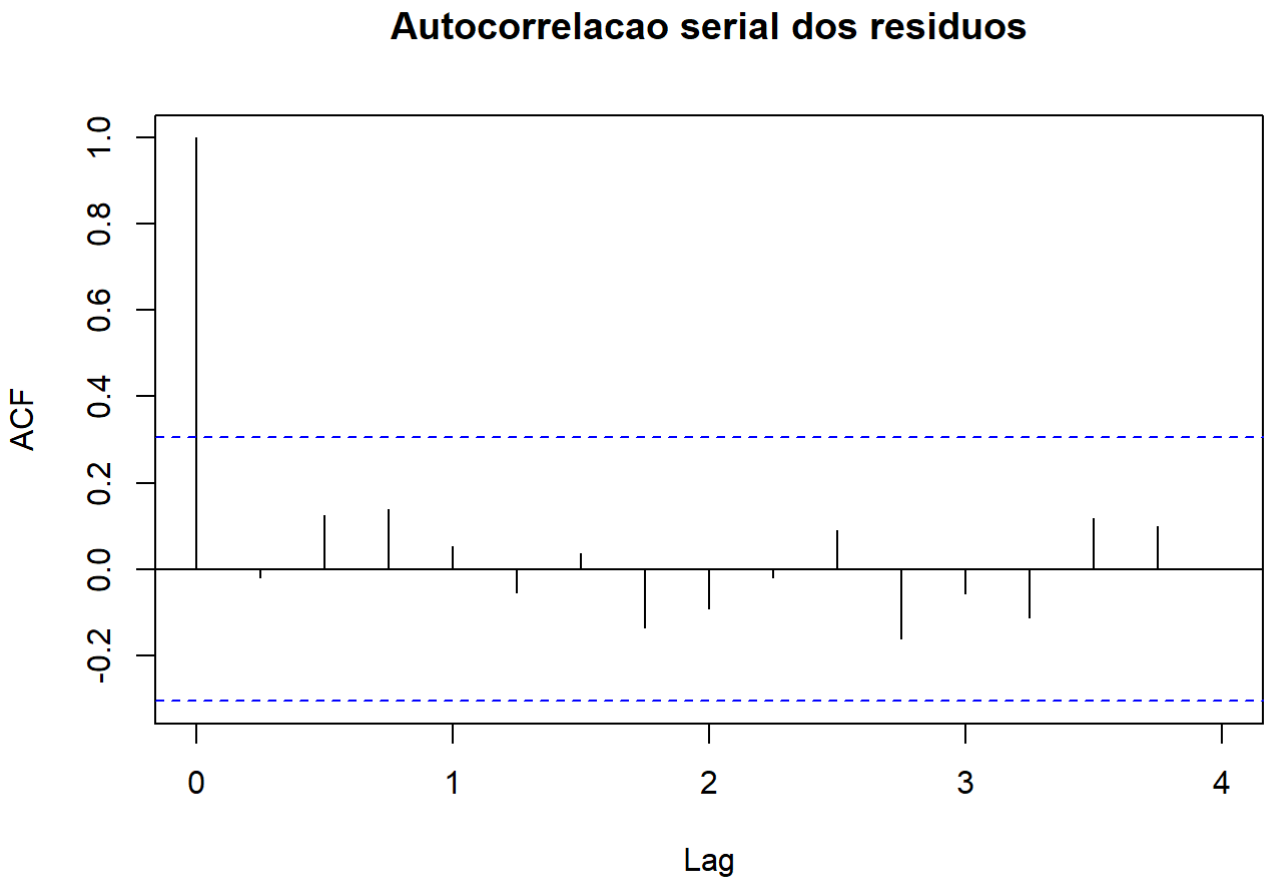
```
# Resíduos do modelo ARMA
residuos <- residuals(modelo_arma_diff_min)

# Função de autocorrelação dos reíduos (ACF)
acf_residuos <- acf(residuos, plot = FALSE)

# Plot da ACF dos resíduos
plot(acf_residuos, main = "Autocorrelacao serial dos residuos")
```



Teste de diagnóstico pela autocorrelação serial do resíduo

Primeira Diferença

```
# Resíduos do modelo ARMA
residuos <- residuals(modelo_arma_diff_min)

# Função de autocorrelação dos reíduos (ACF)
acf_residuos <- acf(residuos, plot = FALSE)

# Plot da ACF dos resíduos
plot(acf_residuos, main = "Autocorrelacao serial dos residuos")
```



Análise de tendência central

```
summary(diff(unem_tri_ts))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.800000 -0.700000 -0.300000 -0.002439  0.700000  2.000000
```

Previsão

```
plot(forecast(modelo_arma_diff_min))
```