Homework3

Daniel de Souza Sobrinho Macedo RA: 813524

Inicialização das bibliotecas

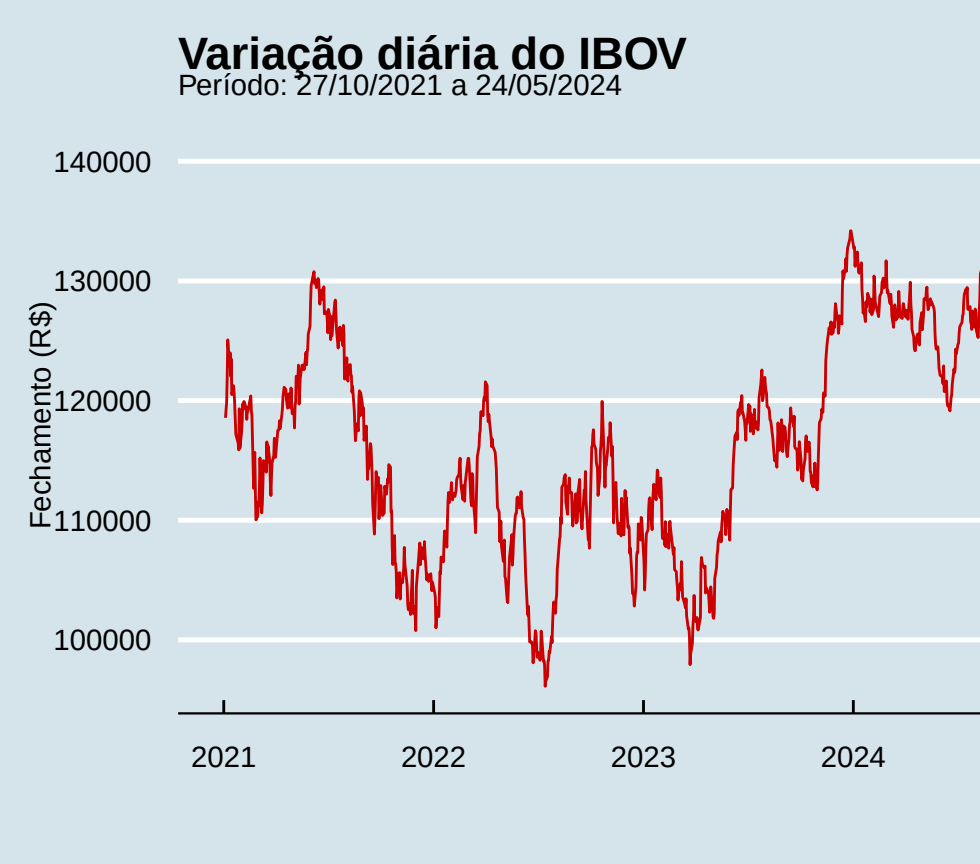
```
#install.packages("lmtest")

library(BatchGetSymbols)
library(tidyverse)
library(ggthemes)
library(FinTS)
library(WriteXLS)
library(xtable)
library(tbl2xts)
library(forecast)
library(tseries)
library(timeSeries)
```

Análise inicial dos dados:

Coleta e preparação dos dados do IBOVESPA desde 01/01/2021:

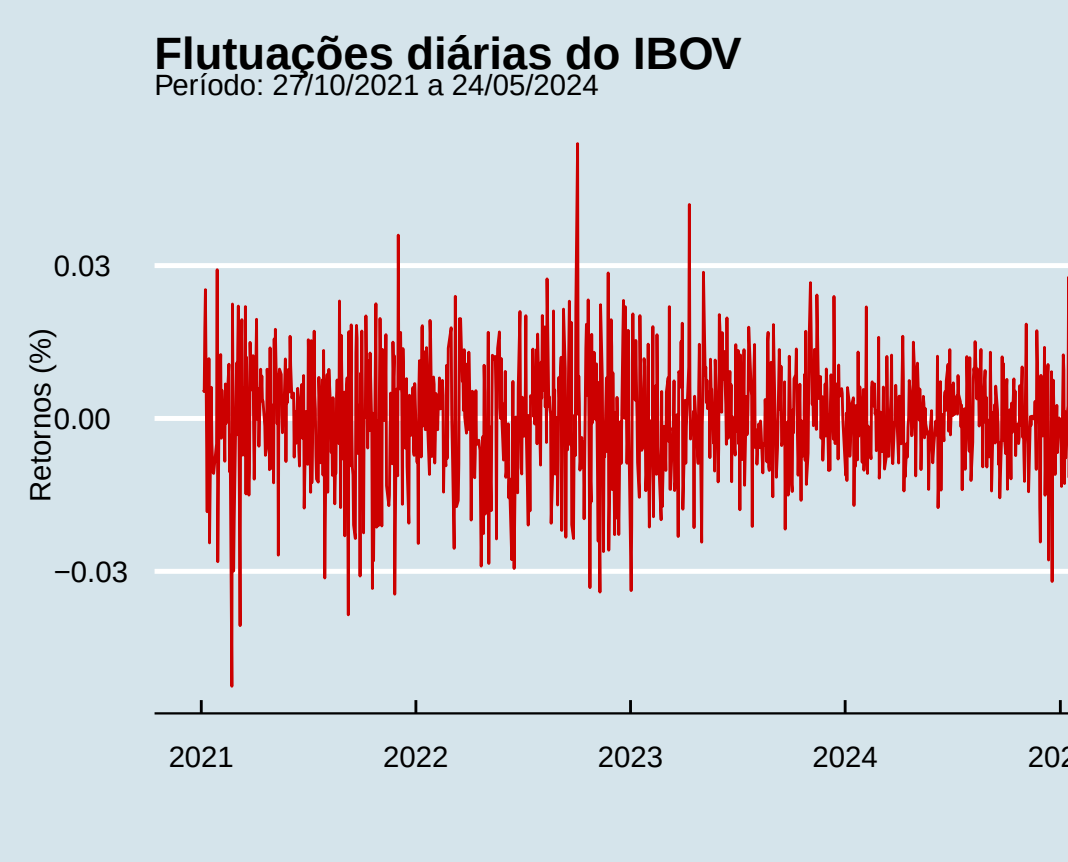
```
dados_ibov <- BatchGetSymbols("^BVSP",
                              first.date = '2021-01-01',
                              last.date = Sys.time(),
                              type.return = "log",
                              freq.data = "daily")[[2]]

serie_retornos <- dados_ibov %>%
  select(ref.date, ticker, ret.closing.prices) %>%
  select(ret.closing.prices) %>%
  slice(-1)

serie_retornos <- as.ts(serie_retornos)
```

```
grafico_precos <- ggplot(dados_ibov, aes(ref.date, price.close)) +
  geom_line(color = '#CC0000') +
  labs(x = "", y = 'Fechamento (R$)', title = "Variação diária do IBOV",
       subtitle = "Período: 27/10/2021 a 24/05/2024", caption = "Fonte: B3") +
  theme_economist()
grafico_precos
```



Evolução dos preços da Ibovespa:

```
grafico_retornos <- ggplot(dados_ibov, aes(ref.date, ret.closing.prices)) +
  geom_line(color = '#CC0000') +
  labs(x = "", y = 'Retornos (%)', title = "Flutuações diárias do IBOV",
       subtitle = "Período: 27/10/2021 a 24/05/2024", caption = "Fonte: B3") +
  theme_economist()
grafico_retornos
```



Variação percentual diária

Indentificação de modelos:

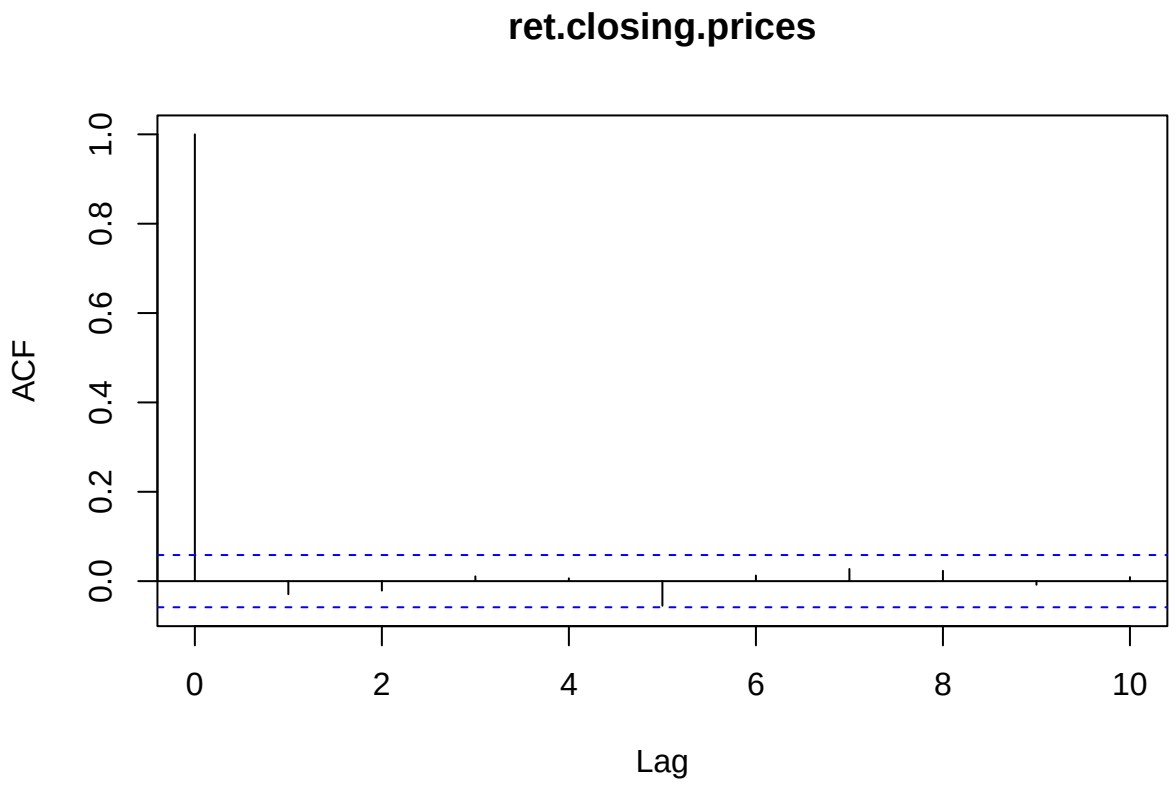
Utilizando auto.arima para sugestão automática de modelo

```
auto.arima(serie_retornos)
```

```
## Series: serie_retornos
## ARIMA(0,0,0) with zero mean
##
## sigma^2 = 0.00013:  log likelihood = 3439.93
## AIC=-6877.85   AICc=-6877.85   BIC=-6872.83
```

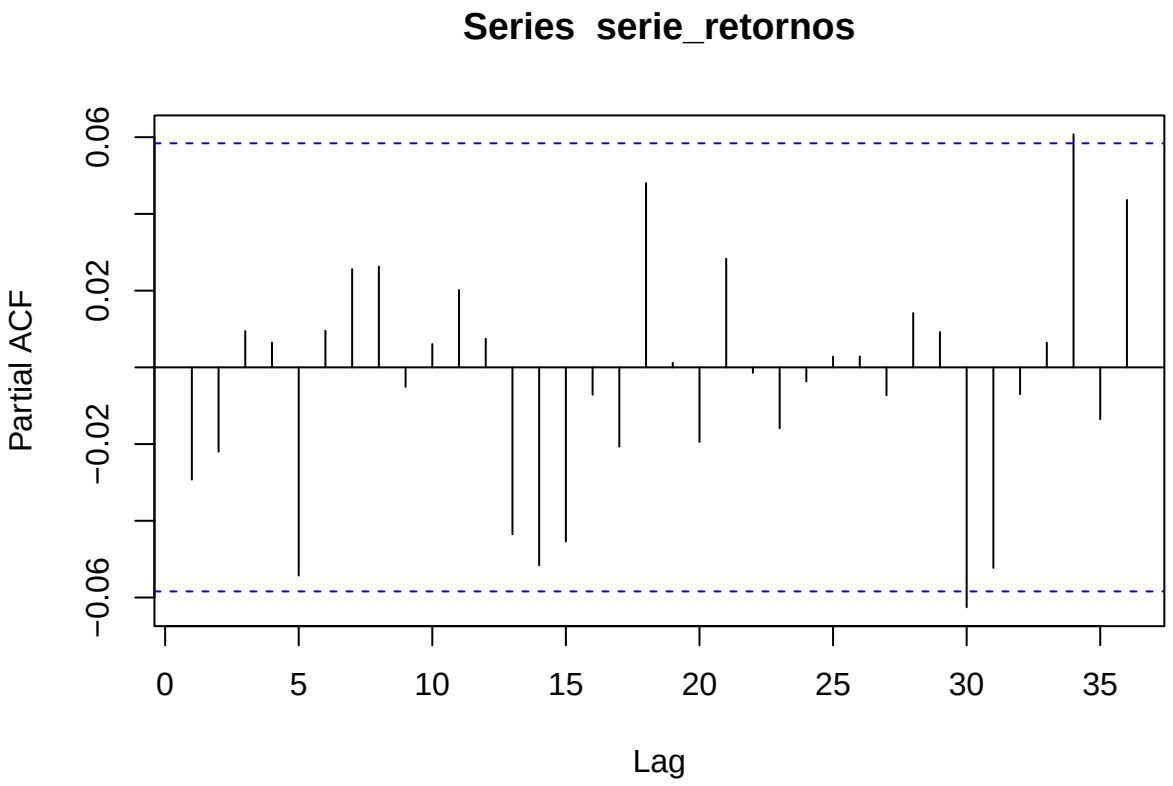
Análise de autocorrelação (FAC)

```
acf(serie_retornos, lag.max = 10)
```



Análise parcial de autocorrelação (FACP)

```
pacf(serie_retornos, lag.max = 36)
```



Verificação via AIC e BIC (modelo ARMA)

```
melhor_aic <- Inf
melhor_bic <- Inf
melhor_arma_aic <- c(0, 0)
melhor_arma_bic <- c(0, 0)

for (p in 0:5) {
  for (q in 0:5) {
    tryCatch({
      modelo_temp <- Arima(serie_retornos, order = c(p, 0, q))
      if (AIC(modelo_temp) < melhor_aic) {
        melhor_aic <- AIC(modelo_temp)
        melhor_arma_aic <- c(p, 0, q)
      }
      if (BIC(modelo_temp) < melhor_bic) {
        melhor_bic <- BIC(modelo_temp)
        melhor_arma_bic <- c(p, 0, q)
      }
    }, error = function(e) NULL)
  }
}

cat("Modelo ótimo segundo AIC: ", melhor_arma_aic, "\n")
```

```
## Modelo ótimo segundo AIC:  3 0 3
```

```
cat("Modelo ótimo segundo BIC: ", melhor_arma_bic, "\n")
```

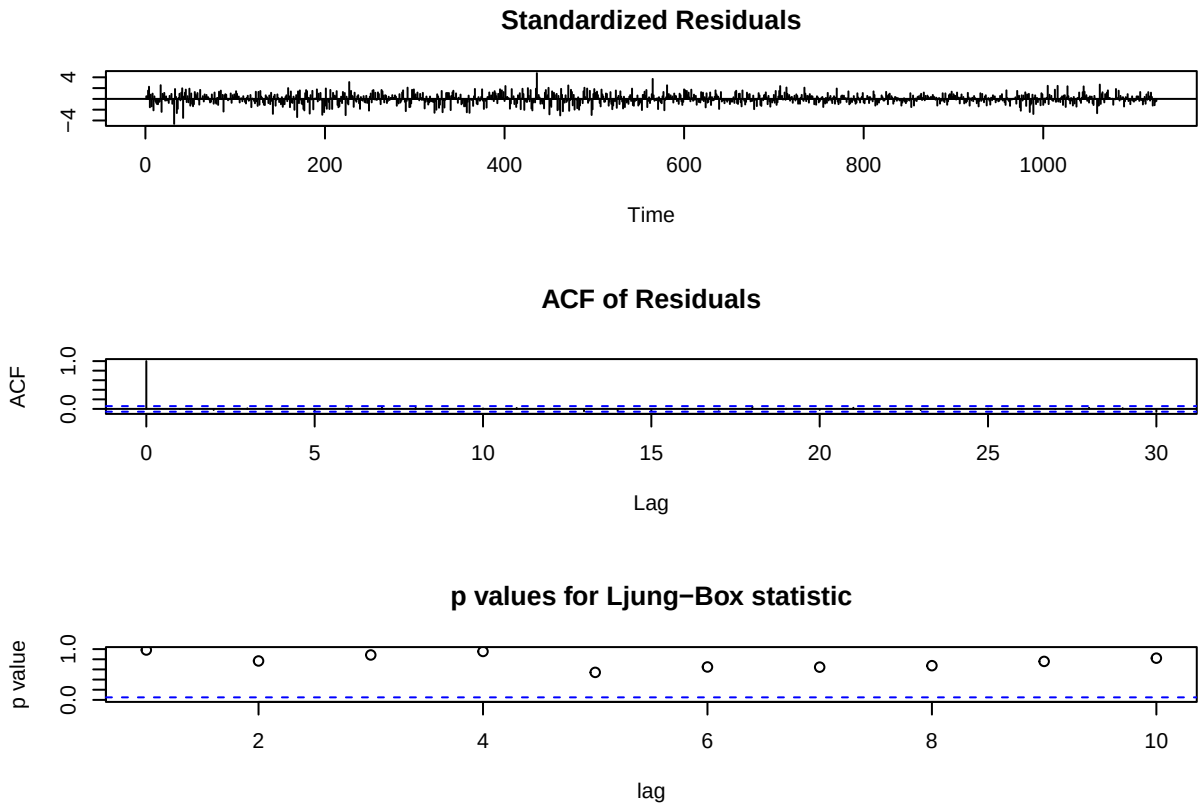
```
## Modelo ótimo segundo BIC:  0 0 0
```

Modelo AR(1) e diagnóstico dos resíduos

```
modelo_ar1 <- arima(serie_retornos, order = c(1, 0, 0))
```

Testes do Box-Pierce e Ljung-Box

```
tsdiag(modelo_ar1)
```




```
Box.test(modelo_ar1$residuals, lag = 1)
```

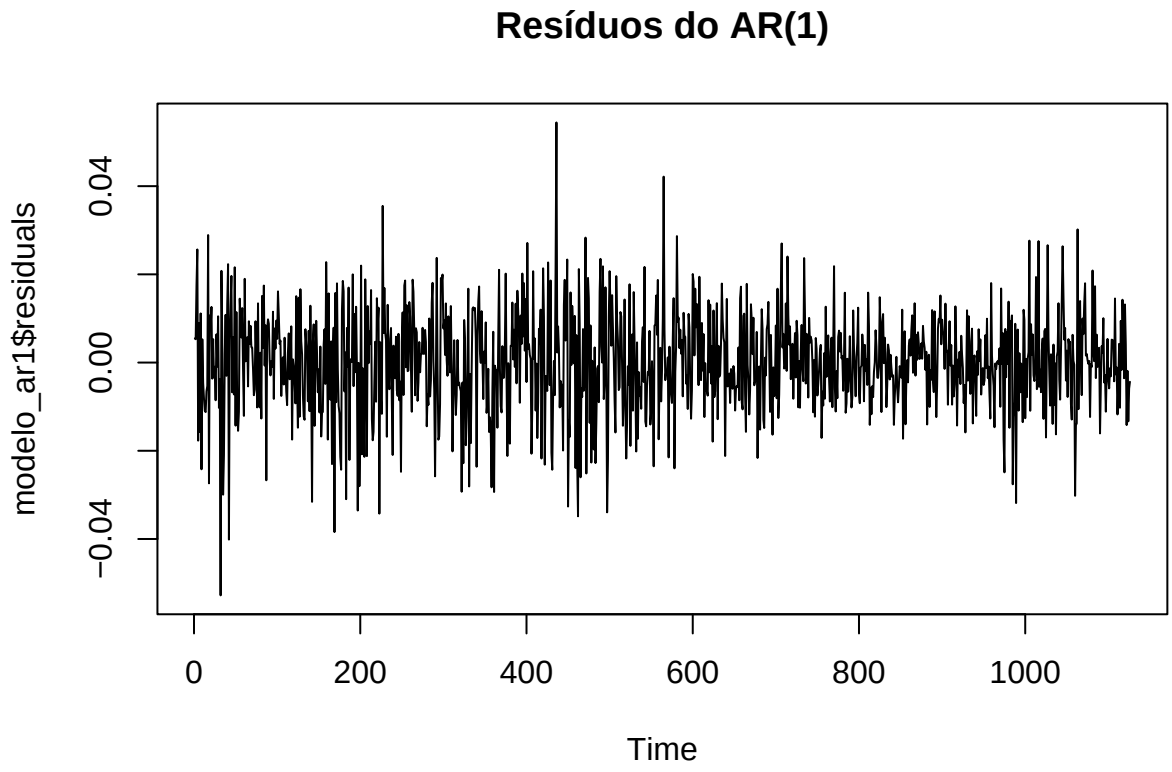
```
##
##  Box-Pierce test
##
## data:  modelo_ar1$residuals
## X-squared = 0.00049438, df = 1, p-value = 0.9823
```

```r
Box.test(residuals(modelo_ar1), type = "Ljung")
```

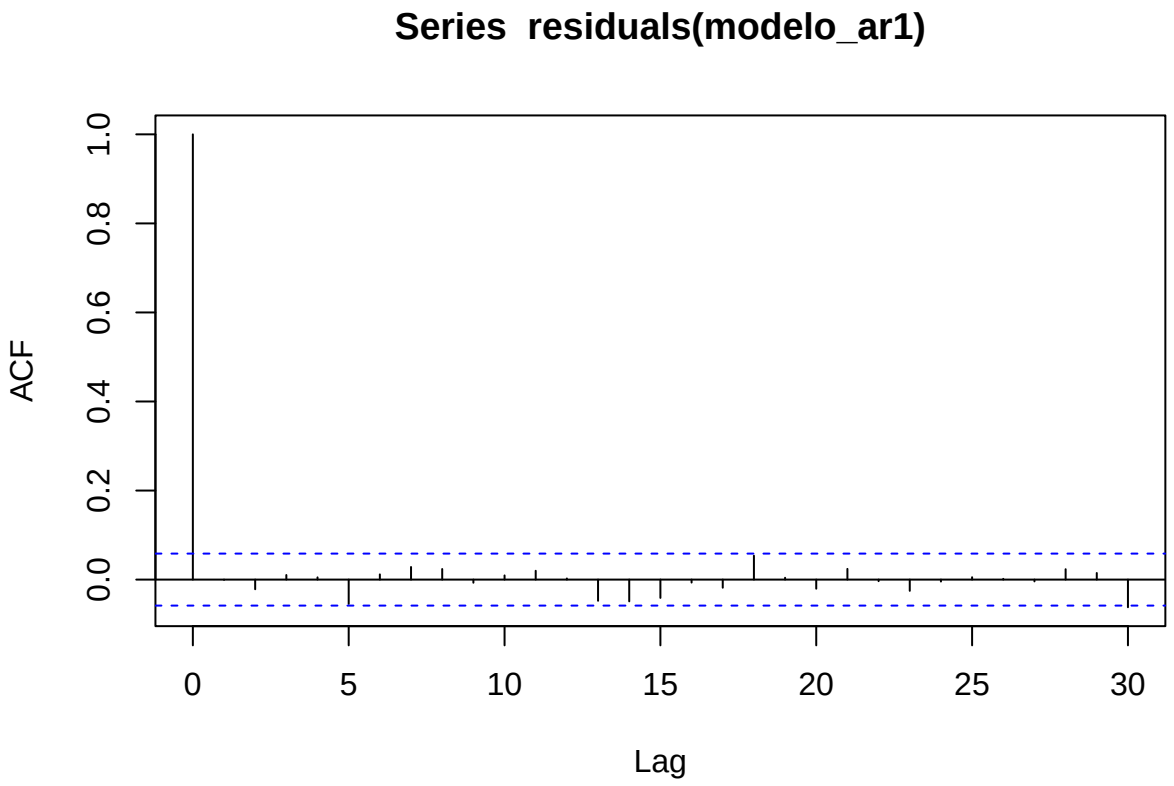
```
##
##  Box-Ljung test
##
## data:  residuals(modelo_ar1)
## X-squared = 0.0004957, df = 1, p-value = 0.9822
```

O gráfico dos resíduos e das FAC e FACP dos resíduos

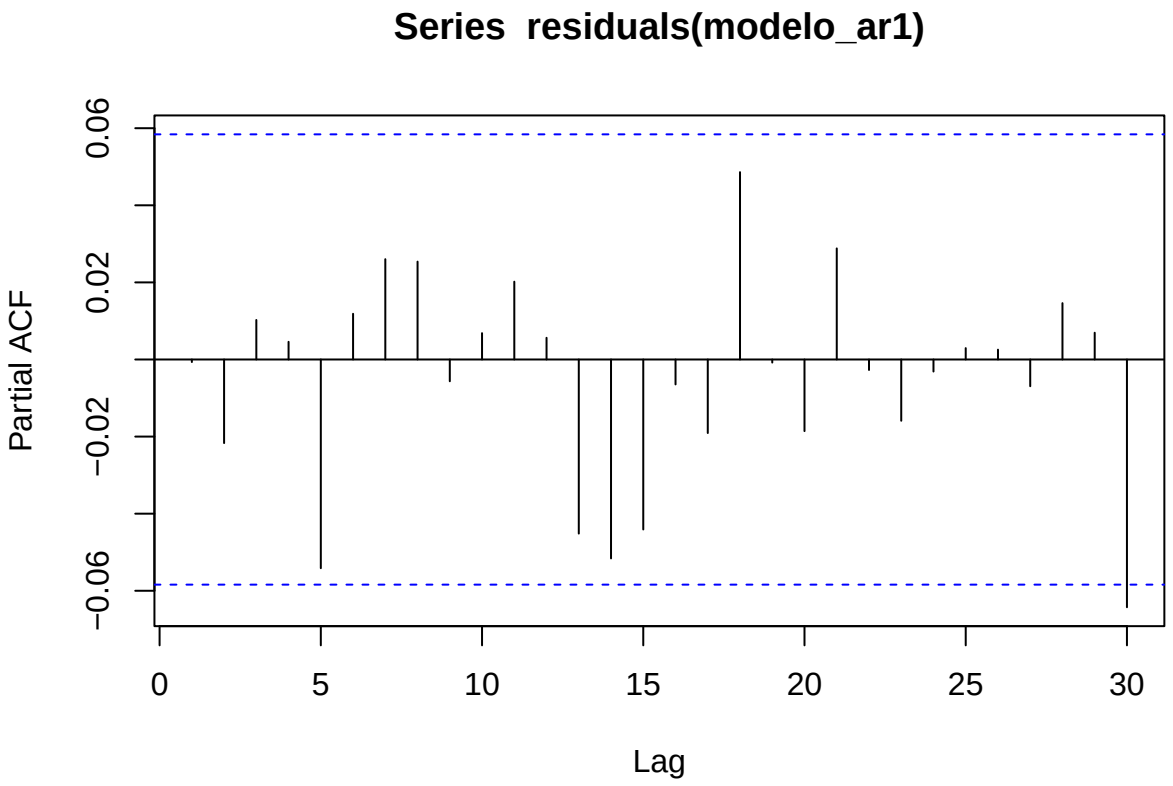
```r
plot.ts(modelo_ar1$residuals, main = "Resíduos do AR(1)")
```




```r
acf(residuals(modelo_ar1))
```




```
pacf(residuals(modelo_ar1))
```



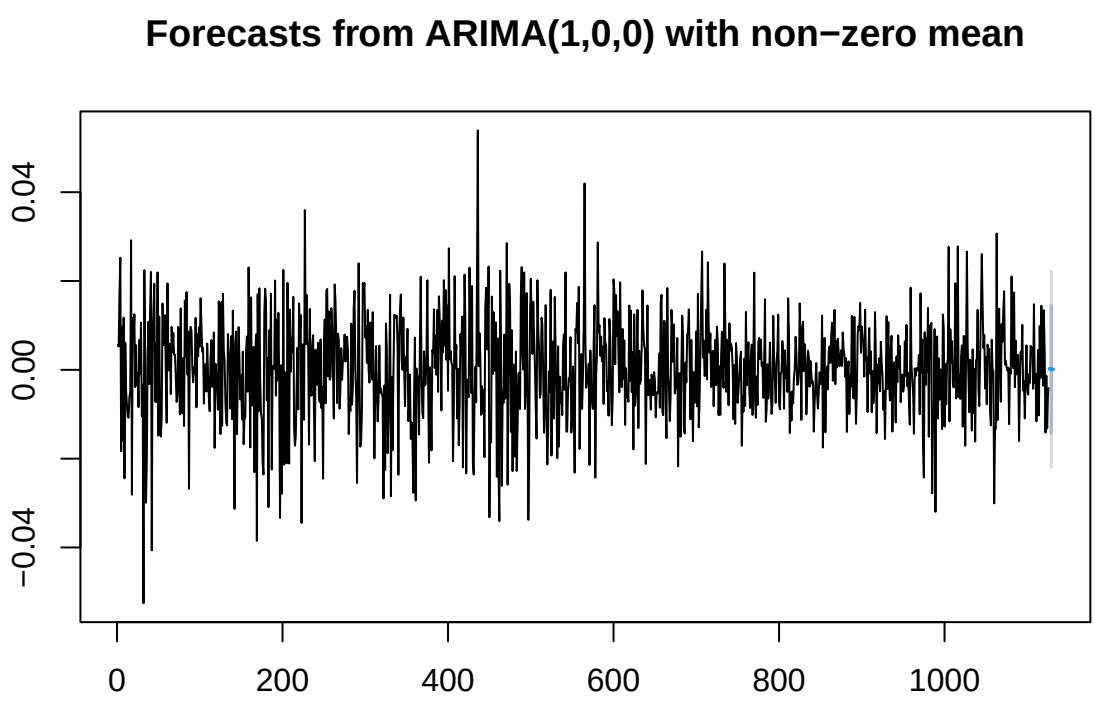
Nota-se que os resíduos não apresentam um comportamento típico de ruído branco, parecendo-se mais com a própria série temporal. Dessa forma, podemos concluir que os modelos ARMA não são suficientemente precisos para representar séries temporais financeiras

Previsões futuras com AR(1) dos próximos 5 tempos:

```
prev <- forecast(modelo_ar1, h = 5)
prev
```

```
##      Point Forecast       Lo 80      Hi 80       Lo 95      Hi 95
## 1127   0.0002446729 -0.01436092 0.01485026 -0.02209265 0.02258200
## 1128   0.0001184200 -0.01449341 0.01473025 -0.02222845 0.02246529
## 1129   0.0001221106 -0.01448973 0.01473395 -0.02222476 0.02246898
## 1130   0.0001220027 -0.01448983 0.01473384 -0.02222487 0.02246888
## 1131   0.0001220059 -0.01448983 0.01473384 -0.02222487 0.02246888
```

```
plot(prev)
```



Modelos para as ações da GetSP500Stocks

Coleta dos tickers

```
df_empresas <- GetSP500Stocks()
lista_tickers <- head(df_empresas %>% select(Tickers) %>% slice(-1), 50)
```

Obtivemos as informações de cada ticker e inserimos como uma linha no nosso data.frame. Nesse data.frame, a primeira coluna corresponde ao nome da empresa e a segunda traz a série temporal dos retornos associados a ela.

```
dados_acoes <- data.frame()

for (ticker in lista_tickers$Tickers) {
  tmp <- BatchGetSymbols(ticker,
                         first.date = '2019-01-01',
                         last.date = Sys.time(),
                         type.return = "log",
                         freq.data = "daily")[[2]]
  if (nrow(tmp) > 0) {
    dados_acoes <- rbind(dados_acoes, data.frame(
      "Empresa" = ticker,
```

```
      "Serie" = I(list(na.omit(as.ts(tmp$ret.closing.prices))))
    ))
  }
}
head(dados_acoes, 5)
```

```
##   Empresa        Serie
## 1     AOS -0.02445....
## 2     ABT -0.04834....
## 3    ABBV -0.03350....
## 4     ACN -0.03473....
## 5    ADBE -0.04029....
```

Com base na série temporal, podemos utilizar a função auto.arima para identificar o modelo mais adequado para cada uma delas.

```
modelos <- c()

for (i in 1:nrow(dados_acoes)) {
  modelo_i <- auto.arima(dados_acoes$Serie[[i]])
  modelo_i$series <- dados_acoes$Empresa[[i]]
  modelos <- c(modelos, I(list(modelo_i)))
}
dados_acoes$Modelo <- modelos
```

Por fim, o modelo obtido anteriormente pode ser utilizado para estimar o valor no tempo t+1.

```
previsoes_t1 <- c()

for (i in 1:nrow(dados_acoes)) {
  pred <- predict(dados_acoes$Modelo[[i]], n.ahead = 1)
  previsoes_t1 <- c(previsoes_t1, pred$pred[1])
}

dados_acoes$Prev_T1 <- previsoes_t1
dados_acoes <- dados_acoes[order(-dados_acoes$Prev_T1), ]
head(data.frame(dados_acoes$Empresa, dados_acoes$Prev_T1), 5)
```

```
##   dados_acoes.Empresa dados_acoes.Prev_T1
## 1                ACGL         0.005171326
## 2                 AMP         0.002910074
## 3                ALLE         0.002317148
## 4                AMAT         0.002205528
## 5                AMGN         0.002160119
```

Separação em estimativa e previsão (80/20)

```r
serie_amostra <- dados_acoes[1, ]
serie <- serie_amostra$Serie[[1]]
serie_vetor <- as.vector(serie)
serie_limpa <- ts(serie_vetor[!sapply(serie_vetor, is.null)])

n_total <- length(serie_limpa)
n_estimativa <- floor(0.8 * n_total)
serie_estimativa <- serie_limpa[1:n_estimativa]
serie_prev <- serie_limpa[(n_estimativa + 1):n_total]
```

```r
modelo_final <- auto.arima(serie_estimativa)
previsoes <- forecast(modelo_final, h = length(serie_prev))

resultado <- data.frame(
  Tempo = time(serie_prev),
  Observado = as.numeric(serie_prev),
  Previsto = as.numeric(previsoes$mean),
  Inferior = as.numeric(previsoes$lower[,2]),
  Superior = as.numeric(previsoes$upper[,2])
)

erro <- resultado$Observado - resultado$Previsto
mae <- mean(abs(erro))
mse <- mean(erro^2)

head(resultado, 5)
```

Treinamento e avaliação do modelo

```
##   Tempo     Observado      Previsto    Inferior   Superior
## 1     1  0.0060903416 -2.018721e-03 -0.04178634 0.03774890
## 2     2 -0.0168383693  3.367197e-03 -0.03647427 0.04320866
## 3     3 -0.0001102954  5.665715e-04 -0.03981089 0.04094403
## 4     4  0.0034127579 -6.180767e-05 -0.04065327 0.04052966
## 5     5 -0.0044057956  2.014699e-03 -0.03857698 0.04260638
```

```r
print(paste("Erro absoluto médio (MAE):", mae))
```

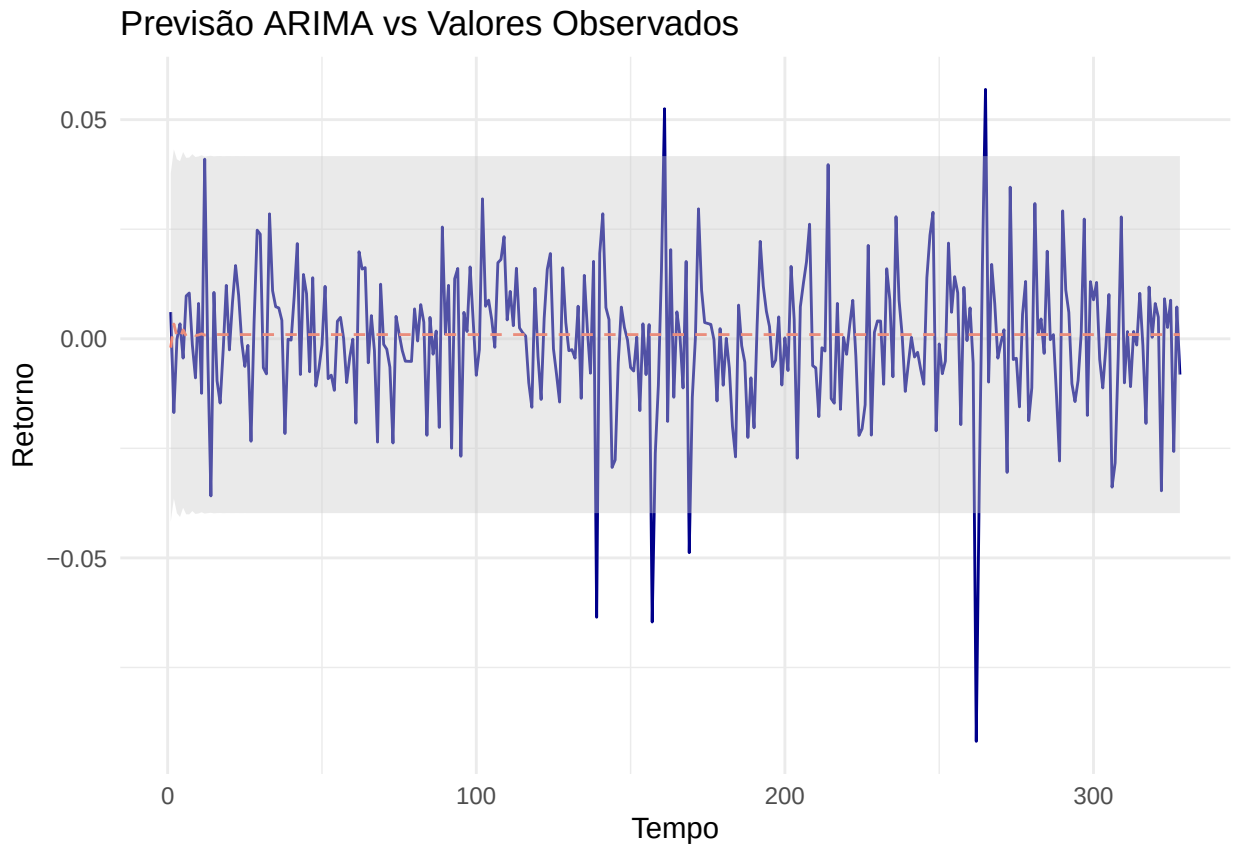
```
## [1] "Erro absoluto médio (MAE): 0.0117695493066049"
```

```r
print(paste("Erro quadrático médio (MSE):", mse))
```

```
## [1] "Erro quadrático médio (MSE): 0.000266644112680996"
```

```r
# Plotar os resultados

grafico_final <- ggplot(resultado, aes(x = Tempo)) +
  geom_line(aes(y = Observado), color = "darkblue") +
```

```
  geom_line(aes(y = Previsto), color = "tomato", linetype = "dashed") +
  geom_ribbon(aes(ymin = Inferior, ymax = Superior), fill = "gray80", alpha = 0.4) +
  labs(title = "Previsão ARIMA vs Valores Observados",
       x = "Tempo", y = "Retorno") +
  theme_minimal()

grafico_final
```



Como podemos ver, as previsões geradas pelo modelo ARIMA não apresentam alta confiabilidade quando aplicadas a séries temporais financeiras