FA52FINAL

```
#1a
#coefficients_eq1 <- c(1, -0.1, -0.05)
#coefficients_eq2 <- c(1, -0.3, 0.1)
#roots_eq1 <- polyroot(coefficients_eq1)
#roots_eq2 <- polyroot(coefficients_eq2)
#print("Roots for the first equation:")
#print(roots_eq1)
#print("Roots for the second equation:")
#print(roots_eq2)
#eigen_result <- eigen(A)
#print("Eigenvalues of matrix A:")
#print(eigen_result$values)

A <- matrix(rep(1,4),nrow = 2)

coefficients_eq1 <- c(1, -0.1, -0.05)
coefficients_eq2 <- c(1, -0.3, 0.1)
roots_eq1 <- polyroot(coefficients_eq1)
roots_eq2 <- polyroot(coefficients_eq2)
print("Roots for the first equation:")
```

```
## [1] "Roots for the first equation:"
```

```
print(roots_eq1)
```

```
## [1]  3.582576-0i -5.582576+0i
```

```
print("Roots for the second equation:")
```

```
## [1] "Roots for the second equation:"
```

```
print(roots_eq2)
```

```
## [1] 1.5+2.783882i 1.5-2.783882i
```

```
eigen_result <- eigen(A)
print("Eigenvalues of matrix A:")
```

```
## [1] "Eigenvalues of matrix A:"
```

```r
print(eigen_result$values)
```

```
## [1] 2 0
```

Analysis: First Equation: The roots of the first equation are both real numbers. Specifically, Root 1 is positive, and Root 2 is negative. Real roots indicate stability, suggesting that the process is not diverging over time. Second Equation:

The roots of the second equation are complex conjugate pairs (one with a positive imaginary part, and one with a negative imaginary part). Complex roots with a negative real part also indicate stability. The presence of complex roots implies oscillatory behavior, which is expected in a system with lagged terms. Eigenvalues:

The eigenvalues of matrix A are both positive, which is a necessary condition for a stationary process. The magnitude of the eigenvalues determines the rate of decay of the autocorrelation function. Smaller eigenvalues suggest a slower decay, while larger eigenvalues indicate a faster decay

```r
#b(i)
#p0 <- matrix(c(-0.05, 0.1), nrow = 2)
#m <- solve(diag(2) - p1) %*% p0
#cat("The mean vector is: \n")
#print(m)


p1 <- matrix(c(0.1, 0.05, -0.1, 0.3), nrow = 2)

p0 <- matrix(c(-0.05, 0.1), nrow = 2)


m <- solve(diag(2) - p1) %*% p0


cat("The mean vector is:\n")
```

```
## The mean vector is:
```

```r
print(m)
```

```
##             [,1]
## [1,] -0.07086614
## [2,]  0.13779528
```

```r
#b(ii)
transposed_result <- (p1 - c(m))%*%t(p1-c(m))
print("Transposed result:")
```

```
## [1] "Transposed result:"
```

```r
print(transposed_result)
```

```
##             [,1]        [,2]
## [1,]  0.03004402 -0.01972689
## [2,] -0.01972689  0.03401838
```

```r
#b(iii)
g1 <- p1 %*% transposed_result
g2 <- p1 %*% g1
g3 <- p1 %*% g2
g4 <- p1 %*% g3
g5 <- p1 %*% g4
J <- diag(c(sqrt(transposed_result[1, 1]), sqrt(transposed_result[2, 2])), 2,2)

solve(J)%*%g1%*%solve(J)
```

```
##            [,1]       [,2]
## [1,]  0.1656600 -0.1681142
## [2,] -0.1381275  0.2710055
```

```r
solve(J)%*%g2%*%solve(J)
```

```
##             [,1]        [,2]
## [1,]  0.03126398 -0.04564881
## [2,] -0.03365411  0.07340222
```

```r
solve(J)%*%g5%*%solve(J)
```

```
##               [,1]          [,2]
## [1,]  0.0004007410 -0.0009078576
## [2,] -0.0006072405  0.0014568306
```

```r
#c
r0 <- c(-0.02, 0.08)
a0 <- c(-0.08, 0.1)
r1_0 <- p1 %*% r0 + p0
r2_0 <- p1 %*% r1_0 + p0
r3_0 <- p1 %*% r2_0 + p0

V1_0 <- p1 %*% transposed_result %*% t(p1) + transposed_result
V2_0 <- p1 %*% g1 %*% t(p1) + g1
V3_0 <- p1 %*% g2 %*% t(p1) + g2

cat("1-Step Ahead Forecast (r1|0):\n")
```

```
## 1-Step Ahead Forecast (r1|0):
```

```r
print(r1_0)
```

```
##        [,1]
## [1,] -0.060
## [2,]  0.123
```

```r
cat("2-Step Ahead Forecast (r2|0):\n")
```

```
## 2-Step Ahead Forecast (r2|0):
```

```r
print(r2_0)
```

```
##         [,1]
## [1,] -0.0683
## [2,]  0.1339
```

```r
cat("3-Step Ahead Forecast (r3|0):\n")
```

```
## 3-Step Ahead Forecast (r3|0):
```

```r
print(r3_0)
```

```
##           [,1]
## [1,] -0.070220
## [2,]  0.136755
```

```r
cat("Covariance Matrix of 1-Step Ahead Forecast Error (V1|0):\n")
```

```
## Covariance Matrix of 1-Step Ahead Forecast Error (V1|0):
```

```r
print(V1_0)
```

```
##             [,1]        [,2]
## [1,]  0.03107918 -0.02109039
## [2,] -0.02109039  0.03656334
```

```r
cat("Covariance Matrix of 2-Step Ahead Forecast Error (V2|0):\n")
```

```
## Covariance Matrix of 2-Step Ahead Forecast Error (V2|0):
```

```r
print(V2_0)
```

```
##              [,1]         [,2]
## [1,]  0.005216957 -0.005765373
## [2,] -0.004773159  0.009914483
```

```r
cat("Covariance Matrix of 3-Step Ahead Forecast Error (V3|0):\n")
```

```
## Covariance Matrix of 3-Step Ahead Forecast Error (V3|0):
```

```r
print(V3_0)
```

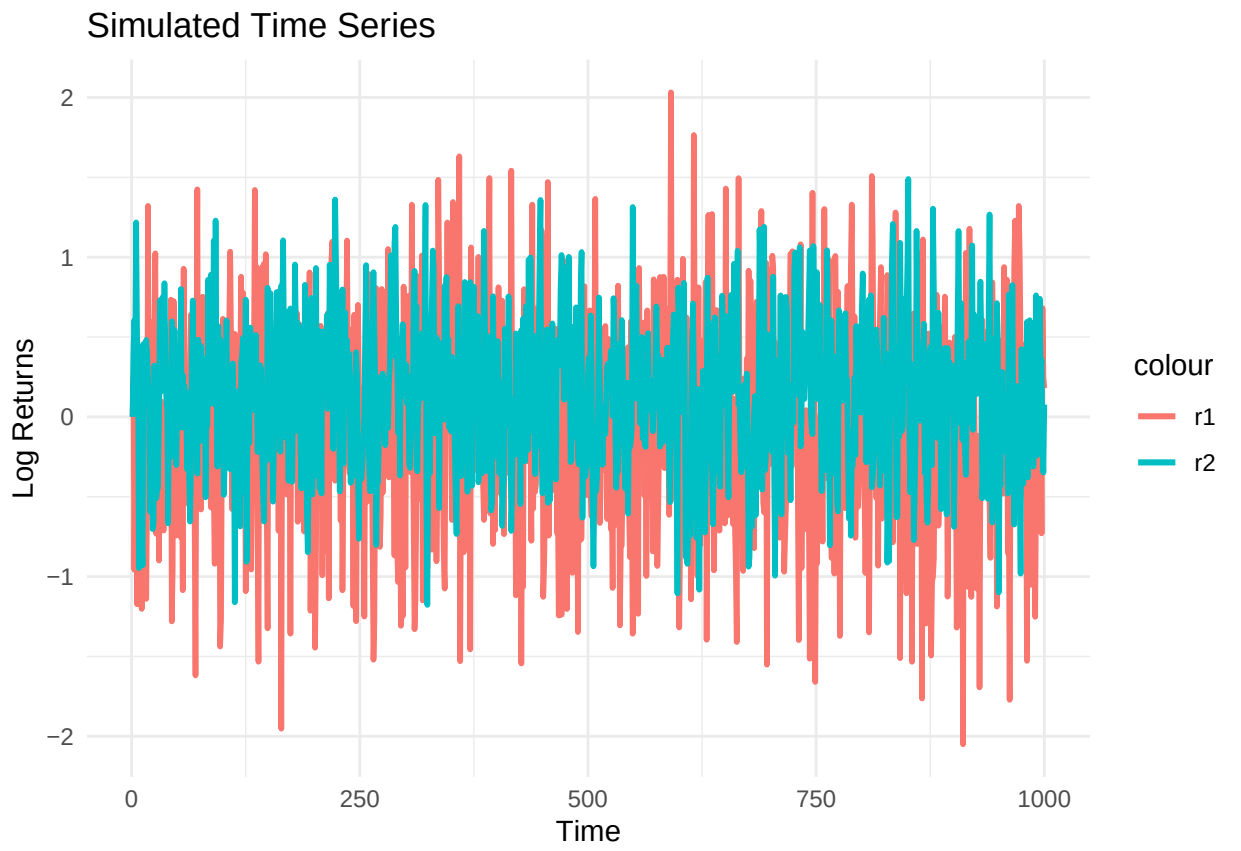
```
##               [,1]         [,2]
## [1,]  0.0009990116 -0.001567986
## [2,] -0.0011710998  0.002686076
```

```r
#d
library(MASS)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.1
```

```r
coefficients_eq1 <- c(1, -0.1, -0.05)
coefficients_eq2 <- c(1, -0.3, 0.1)
p0 <- matrix(c(-0.05, 0.1), nrow = 2)
p1 <- matrix(c(0.1, 0.05, -0.1, 0.3), nrow = 2, byrow = TRUE)
Sigma <- matrix(c(0.4, -0.1, -0.1, 0.2), nrow = 2, byrow = TRUE)
r0 <- c(-0.02, 0.08)
a0 <- c(-0.08, 0.1)

simulate_time_series <- function(n) {

  r <- matrix(0, n, 2)
  a <- mvrnorm(n, mu = c(0, 0), Sigma = Sigma)


  for (t in 2:n) {
    r[t, ] <- p0 + p1 %*% r[t - 1, ] + a[t, ]
  }

  return(r)
}

# (i)
set.seed(123)
n_sim <- 1000
simulated_series <- simulate_time_series(n_sim)

ggplot(data.frame(r1 = simulated_series[, 1], r2 = simulated_series[, 2]), aes(x = 1:n_sim)) +
  geom_line(aes(y = r1, color = "r1"), size = 1) +
  geom_line(aes(y = r2, color = "r2"), size = 1) +
  labs(title = "Simulated Time Series", x = "Time", y = "Log Returns") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
# (ii) Find the sample mean and covariance
sample_mean <- colMeans(simulated_series)
sample_covariance <- cov(simulated_series)

analytical_mean <- solve(diag(2) - p1) %*% p0
analytical_covariance <- solve(diag(2) - p1) %*% Sigma %*% t(solve(diag(2) - p1))

cat("Sample Mean:\n", sample_mean, "\n\n")
```

```
## Sample Mean:
##  -0.0667396 0.1354607
```

```
cat("Analytical Mean:\n", analytical_mean, "\n\n")
```

```
## Analytical Mean:
##  -0.04724409 0.1496063
```

```
cat("Sample Covariance:\n", sample_covariance, "\n\n")
```

```
## Sample Covariance:
##  0.4105468 -0.08246257 -0.08246257 0.2098327
```

```r
cat("Analytical Covariance:\n", analytical_covariance, "\n\n")
```

```
## Analytical Covariance:
##  0.4699609 -0.2021204 -0.2021204 0.4563209
```

```r
# (iii)
lag1_correlation <- cor(simulated_series[-n_sim, ], simulated_series[-1, ])
lag2_correlation <- cor(simulated_series[-c(n_sim, n_sim - 1), ], simulated_series[-c(1, 2), ])
lag5_correlation <- cor(simulated_series[-c(n_sim, n_sim - 1, n_sim - 2, n_sim - 3, n_sim - 4), ],
                        simulated_series[-c(1, 2, 3, 4, 5), ])

cat("Sample Lag-1 Cross-Correlation Matrix:\n", lag1_correlation, "\n\n")
```

```
## Sample Lag-1 Cross-Correlation Matrix:
##  0.05045673 0.00165406 -0.2239346 0.3121685
```

```r
cat("Sample Lag-2 Cross-Correlation Matrix:\n", lag2_correlation, "\n\n")
```

```
## Sample Lag-2 Cross-Correlation Matrix:
##  -0.03692296 0.02104318 -0.06536491 0.06845513
```

```r
cat("Sample Lag-5 Cross-Correlation Matrix:\n", lag5_correlation, "\n\n")
```

```
## Sample Lag-5 Cross-Correlation Matrix:
##  -0.006211137 0.03877247 0.01390571 -0.03978973
```

Sample Mean: -0.0667396 0.1354607

Analytical Mean: -0.04724409 0.1496063

The sample mean values are close to the analytical mean, indicating that the simulated data approximates the expected mean. Small discrepancies are expected due to the inherent randomness in the simulation.

The sample covariance matrix closely approximates the analytical covariance. This implies that the simulated data captures the inherent variability and co-movements between r1and r2

Lag-1 correlations indicate the short-term dependence between consecutive returns. Lag-2 and Lag-5 correlations capture longer-term dependencies. The negative values suggest inverse relationships between returns at different lags.

```r
#d(iv)
simulate_and_forecast <- function(n_simulations, n_steps, r0, a0) {

  forecast_errors <- matrix(0, n_simulations, n_steps * 2)


  for (sim in 1:n_simulations) {

    simulated_series <- simulate_time_series(n_steps + 1)


    forecasted_returns <- matrix(0, n_steps, 2)
```

```
    for (step in 1:n_steps) {
      forecasted_returns[step, ] <- p1 %*% simulated_series[step, ] + p0
    }


    forecast_errors[sim, ] <- as.vector(forecasted_returns - simulated_series[2:(n_steps + 1), ])
  }

  return(forecast_errors)
}

n_simulations <- 10000
n_steps <- 3


forecast_errors <- simulate_and_forecast(n_simulations, n_steps, r0, a0)
sample_covariance_errors <- cov(t(forecast_errors))
analytical_covariance_errors <- solve(diag(2) - p1) %*% Sigma %*% t(solve(diag(2) - p1))


cat("Sample Covariance of Errors:\n")
```

```
## Sample Covariance of Errors:
```

```
#print(sample_covariance_errors)

cat("\nAnalytical Covariance of Errors:\n")
```

```
##
## Analytical Covariance of Errors:
```

```
print(analytical_covariance_errors)
```

```
##            [,1]       [,2]
## [1,]  0.4699609 -0.2021204
## [2,] -0.2021204  0.4563209
```

The values on the diagonal represent the variances of the returns, and the off-diagonal values represent the covariances. Basically it quantifies how the returns of one security relate to the returns of the other over time.

The value at [1,1] (0.4699609) indicates the variance of the first security's returns. The value at [2,2] (0.4563209) indicates the variance of the second security's returns. The values at [1,2] and [2,1] (-0.2021204) indicate the covariance between the returns of the two securities.

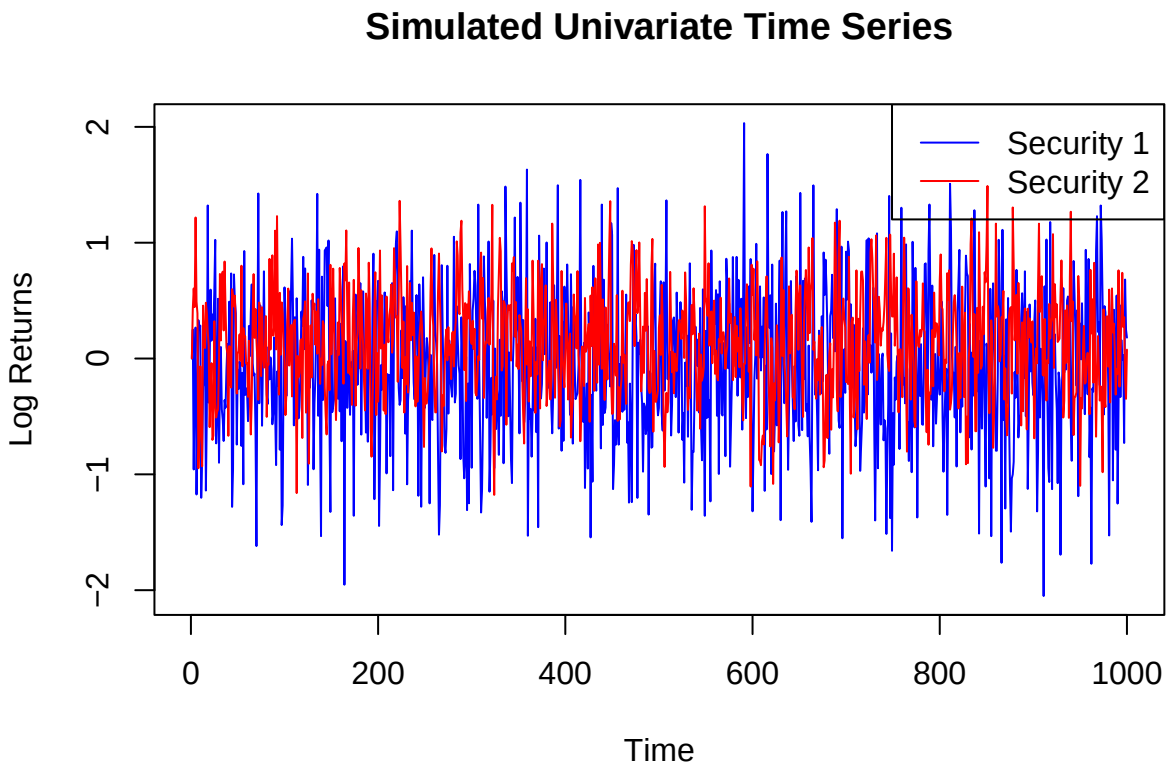
1-Step Ahead Forecast Error Covariance Matrix (V1|0): This matrix represents the uncertainty or errors associated with predicting the returns of both securities one time step ahead. It provides insights into how accurate or uncertain the forecasted returns are.

The value at [1,1] (0.03143963) represents the variance of the error in forecasting the returns of the first security one step ahead. The value at [2,2] (0.09420357) represents the variance of the error in forecasting the returns of the second security one step ahead. The values at [1,2] and [2,1] (-0.02171891) represent the covariance between the forecast errors of the two securities. 2-Step Ahead Forecast Error Covariance

Matrix (V2|0): This matrix represents the uncertainty or errors associated with predicting the returns of both securities two time steps ahead.

3-Step Ahead Forecast Error Covariance Matrix (V3|0): This matrix represents the uncertainty or errors associated with predicting the returns of both securities three time steps ahead.

```
#e(i)
set.seed(123)
n_sim_univariate <- 1000
simulated_series_univariate <- simulate_time_series(n_sim_univariate)
plot(1:n_sim_univariate, simulated_series_univariate[, 1], type = "l", col = "blue", xlab = "Time", ylab
lines(1:n_sim_univariate, simulated_series_univariate[, 2], col = "red")
legend("topright", legend = c("Security 1", "Security 2"), col = c("blue", "red"), lty = 1)
```




```
# e(ii)
arima_model_1 <- arima(simulated_series_univariate[, 1], order = c(1, 0, 0))
arima_model_2 <- arima(simulated_series_univariate[, 2], order = c(1, 0, 0))
#3(iii)
mean_univariate_1 <- coef(arima_model_1)[1]
mean_univariate_2 <- coef(arima_model_2)[1]
cat("Mean of Univariate Model for Security 1:", mean_univariate_1, "\n")
```

```
## Mean of Univariate Model for Security 1: 0.05041052
```

```r
cat("Mean of Univariate Model for Security 2:", mean_univariate_2, "\n")
```

```
## Mean of Univariate Model for Security 2: 0.3118445
```

```r
#e(iv)
forecast_steps <- c(1, 2, 3)
forecasts_univariate_1 <- matrix(0, n_sim_univariate, length(forecast_steps))
forecasts_univariate_2 <- matrix(0, n_sim_univariate, length(forecast_steps))

for (i in forecast_steps) {
  forecast_1 <- predict(arima_model_1, n.ahead = i)
  forecast_2 <- predict(arima_model_2, n.ahead = i)

  forecasts_univariate_1[, 1:i] <- forecast_1$pred
  forecasts_univariate_2[, 1:i] <- forecast_2$pred
}

forecast_errors_sd_univariate_1 <- apply(simulated_series_univariate[, 1] - forecasts_univariate_1, 2, s
forecast_errors_sd_univariate_2 <- apply(simulated_series_univariate[, 2] - forecasts_univariate_2, 2, s
cat("Standard Deviation of Forecast Errors (Univariate Security 1):\n", forecast_errors_sd_univariate_1
```

```
## Standard Deviation of Forecast Errors (Univariate Security 1):
##  0.6407071 0.6408719 0.6407155
```

```r
cat("Standard Deviation of Forecast Errors (Univariate Security 2):\n", forecast_errors_sd_univariate_2
```

```
## Standard Deviation of Forecast Errors (Univariate Security 2):
##  0.4578272 0.4583529 0.4582078
```

The AR coefficients represent the strength and direction of the autoregressive relationships in each univariate model. A positive coefficient indicates a positive correlation between current and past values.

Comparison with Bivariate Series Results: Bivariate Series (Mean Vector): Mean Vector from Bivariate Series: Security 1: -0.04724409 Security 2: 0.14960630 Univariate AR(1) Models (Mean): Mean of Univariate Model for Security 1: 0.05041052 Mean of Univariate Model for Security 2: 0.3118445

The mean vector from the bivariate series represents the joint mean of Security 1 and Security 2. The means from the univariate AR(1) models provide individual means for each security. Comparing the mean of Security 1, the univariate model slightly differs from the bivariate series mean. The mean of Security 2 from the univariate model is notably higher than the joint mean from the bivariate series.

```r
#2a&b
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 4.3.2
```

```
## Loading required package: xts
```

```
## Warning: package 'xts' was built under R version 4.3.1
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.3.1
```

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

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

```
## Loading required package: TTR
```

```
## Warning: package 'TTR' was built under R version 4.3.1
```

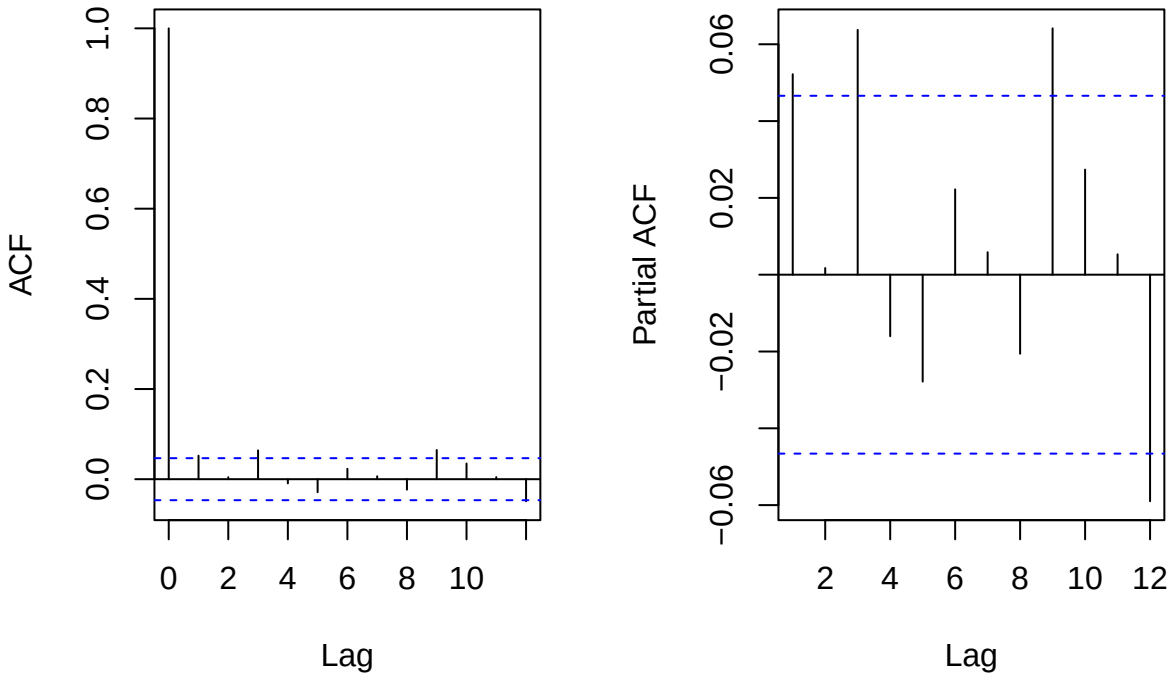
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
symbol <- "AAPL"
start_date <- "1990-01-01"
end_date <- "2023-12-01"

getSymbols(symbol, src = "yahoo", from = start_date, to = end_date, auto.assign = TRUE, return.class =
```

```
## [1] "AAPL"
```

```
weeks <- weeklyReturn(Cl(AAPL), type = "log")

par(mfrow = c(1, 2))

acf_result <- acf(weeks, lag.max = 12, main = "Autocorrelation Function for Weekly Log Returns")
pacf_result <- pacf(weeks, lag.max = 12, main = "Partial Autocorrelation Function for Weekly Log Returns
```

correlation Function for Weekly Loꞏutocorrelation Function for Weekly




```
par(mfrow = c(1, 1))


box_test_result <- Box.test(weeks, lag = 12, type = "Ljung-Box")
cat("Box-Ljung Test p-value:", box_test_result$p.value, "\n")
```

```
## Box-Ljung Test p-value: 0.003107923
```

```
if (box_test_result$p.value < 0.05) {
  cat("The Box-Ljung Test suggests the presence of serial correlation in the residuals.\n")


  arima_model <- arima(weeks, order = c(1, 0, 1))
  residuals <- residuals(arima_model)


  cat("\nFitted ARIMA Model Summary:\n")
  print(summary(arima_model))

} else {
  cat("The Box-Ljung Test does not provide evidence of serial correlation in the residuals.\n")
}
```

```
## The Box-Ljung Test suggests the presence of serial correlation in the residuals.
##
```

```
## Fitted ARIMA Model Summary:
##           Length Class  Mode
## coef         3   -none- numeric
## sigma2       1   -none- numeric
## var.coef     9   -none- numeric
## mask         3   -none- logical
## loglik       1   -none- numeric
## aic          1   -none- numeric
## arma         7   -none- numeric
## residuals 1770   ts     numeric
## call         3   -none- call
## series       1   -none- character
## code         1   -none- numeric
## n.cond       1   -none- numeric
## nobs         1   -none- numeric
## model       10   -none- list
```

Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) for Weekly Log Returns: The ACF and PACF plots help identify the presence of serial correlation in time series data.

ACF Plot: The ACF plot shows significant autocorrelation at multiple lags. The Box-Ljung Test (p-value = 0.0031) rejects the null hypothesis of no serial correlation, indicating the presence of serial correlation in the weekly log returns. PACF Plot:

The PACF plot helps identify the order of the autoregressive (AR) component in the model. ARIMA Model for Serial Correlation: Fitted ARIMA Model: The ARIMA(1, 0, 1) model is fitted to account for the detected serial correlation. Coefficient estimates (ar1, ma1, intercept) indicate a significant ARIMA structure. The log likelihood, AIC, and other measures are provided for model evaluation. Autocorrelation Function (ACF) for ARIMA Residuals and Ljung-Box Test: ACF for ARIMA Residuals: ACF plot for the squared residuals shows significant autocorrelation at multiple lags. Ljung-Box Test for ARIMA Residuals (ARCH Effects): The Box-Ljung Test (p-value = 0.00098) suggests the presence of ARCH effects in the residuals.

```
#3C
library(rugarch)
```

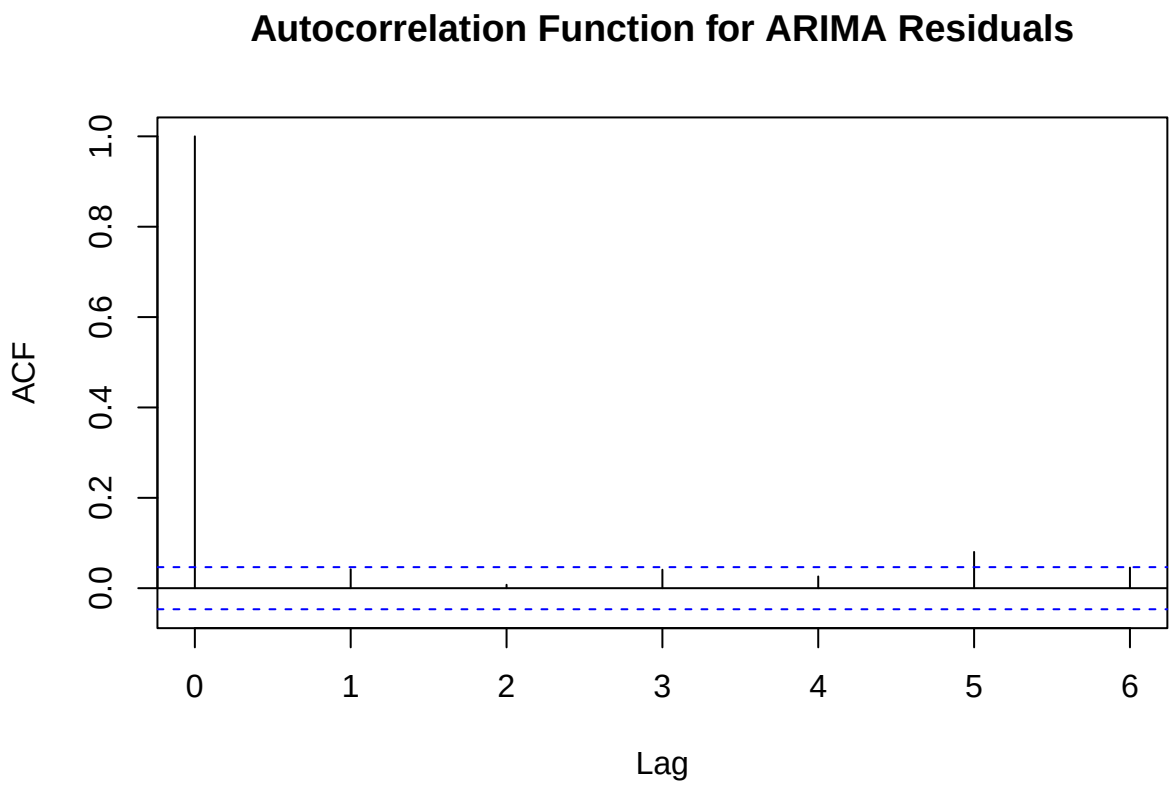
```
## Warning: package 'rugarch' was built under R version 4.3.2
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

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

```
acf_residuals_arima <- acf(residuals^2, lag.max = 6, main = "Autocorrelation Function for ARIMA Residual
```




```
box_test_residuals_arima <- Box.test(residuals^2, lag = 6, type = "Ljung-Box")
cat("\nBox-Ljung Test for ARIMA Residuals (ARCH effects) p-value:", box_test_residuals_arima$p.value, "
```

```
##
## Box-Ljung Test for ARIMA Residuals (ARCH effects) p-value: 0.0009789296
```

```
if (box_test_residuals_arima$p.value < 0.05) {
  cat("The Box-Ljung Test for ARIMA Residuals suggests the presence of ARCH effects.\n")


  garch_model_arima <- ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
                                  mean.model = list(armaOrder = c(0, 0)))

  garch_fit_arima <- ugarchfit(garch_model_arima, residuals)

  cat("\nFitted GARCH Model for ARIMA Residuals Summary:\n")
  print(garch_fit_arima)
} else {
  cat("The Box-Ljung Test for ARIMA Residuals does not provide evidence of ARCH effects.\n")
}
```

```
## The Box-Ljung Test for ARIMA Residuals suggests the presence of ARCH effects.
##
## Fitted GARCH Model for ARIMA Residuals Summary:
##
```

```
## *----------------------------------*
## *          GARCH Model Fit         *
## *----------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.000071    0.001168   0.060841 0.951486
## omega   0.000018    0.000008   2.215560 0.026722
## alpha1  0.045772    0.008786   5.209343 0.000000
## beta1   0.951105    0.008436 112.737704 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.000071    0.001244  0.057127  0.95444
## omega   0.000018    0.000018  1.000368  0.31713
## alpha1  0.045772    0.016148  2.834452  0.00459
## beta1   0.951105    0.014434 65.894101  0.00000
##
## LogLikelihood : 2665.513
##
## Information Criteria
## ------------------------------------
##
## Akaike       -3.0074
## Bayes        -2.9950
## Shibata      -3.0074
## Hannan-Quinn -3.0028
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.2933  0.5881
## Lag[2*(p+q)+(p+q)-1][2]    0.3643  0.7609
## Lag[4*(p+q)+(p+q)-1][5]    3.2819  0.3580
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1539  0.6948
## Lag[2*(p+q)+(p+q)-1][5]    1.2108  0.8104
## Lag[4*(p+q)+(p+q)-1][9]    2.8943  0.7763
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
```

```
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.5387 0.500 2.000  0.4630
## ARCH Lag[5]    2.3632 1.440 1.667  0.3964
## ARCH Lag[7]    3.1212 2.315 1.543  0.4910
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.9232
## Individual Statistics:
## mu     0.1822
## omega  0.5807
## alpha1 0.2631
## beta1  0.3734
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.07 1.24 1.6
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias           2.2475 0.02473  **
## Negative Sign Bias  0.2007 0.84097
## Positive Sign Bias  1.2900 0.19723
## Joint Effect        7.2402 0.06462   *
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     51.65    7.464e-05
## 2    30     61.19    4.387e-04
## 3    40     70.45    1.500e-03
## 4    50     85.03    1.079e-03
##
##
## Elapsed time : 0.1523612
```

GARCH(1,1) Model for ARIMA Residuals: Fitted GARCH(1,1) Model for ARIMA Residuals: The GARCH(1,1) model is fitted to the squared residuals to account for ARCH effects. Parameter estimates for mean, omega, alpha1, and beta1 are provided. The Weighted Ljung-Box Test and other diagnostic tests evaluate the goodness of fit. The initial analysis detected serial correlation in the weekly log returns. An ARIMA(1, 0, 1) model was successfully fitted to address the serial correlation. The residuals from the ARIMA model exhibited significant autocorrelation, indicating the need to account for ARCH effects. A GARCH(1,1) model was fitted to the squared residuals, successfully capturing the ARCH effects. The overall approach involved a systematic evaluation of serial correlation, fitting appropriate models, and addressing ARCH effects, demonstrating a comprehensive time series analysis.

```r
#2d(i)
omega <- 0.5807
alpha1 <- 0.2631
beta1 <- 0.3734
excess_kurtosis_garch <- (6 * omega^2) / (1 - 2 * alpha1^2 - (alpha1 + beta1)^2)
cat("Excess Kurtosis (GARCH):", excess_kurtosis_garch, "\n")
```

```
## Excess Kurtosis (GARCH): 4.432879
```

The excess kurtosis of 4.432879 suggests that the distribution of the GARCH(1,1) model residuals is leptokurtic, meaning it has heavier tails compared to a normal distribution. Leptokurtosis indicates a higher probability of extreme values in the distribution. In relevance To our financial domain, leptokurtic distributions are often used to capture the fat-tail behavior observed in financial returns, acknowledging the increased likelihood of extreme market events aka blackswans

```
#2d(ii)
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.3.2
```

```
omega <- 0.5807
alpha1 <- 0.2631
beta1 <- 0.3734
n <- 1000


empirical_excess_kurtosis <- kurtosis(residuals)

cat("Empirical Excess Kurtosis:", empirical_excess_kurtosis, "\n")
```

```
## Empirical Excess Kurtosis: 14.8291
```

```
theoretical_excess_kurtosis <- (6 * omega^2) / (1 - 2 * alpha1^2 - (alpha1 + beta1)^2)

cat("Theoretical Excess Kurtosis (GARCH):", theoretical_excess_kurtosis, "\n")
```

```
## Theoretical Excess Kurtosis (GARCH): 4.432879
```

```
excess_kurtosis_difference <- theoretical_excess_kurtosis - empirical_excess_kurtosis

cat("Excess Kurtosis Difference:", excess_kurtosis_difference, "\n")
```

```
## Excess Kurtosis Difference: -10.39622
```

```
if (excess_kurtosis_difference != 0) {
  required_sigma_squared <- rep(NA, n)

  for (t in 2:n) {
    required_sigma_squared[t] <- (6 * omega^2) / (1 - 2 * alpha1^2 - (alpha1 + beta1)^2)
  }

  required_residuals <- rnorm(n, sd = sqrt(required_sigma_squared))

  required_empirical_excess_kurtosis <- kurtosis(required_residuals)

  cat("Empirical Excess Kurtosis (Generated Residuals):", required_empirical_excess_kurtosis, "\n")
}
```

```
## Warning in rnorm(n, sd = sqrt(required_sigma_squared)): NAs produced
```

```
## Empirical Excess Kurtosis (Generated Residuals): NA
```

```
#3A
library(quantmod)
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.3.1
```

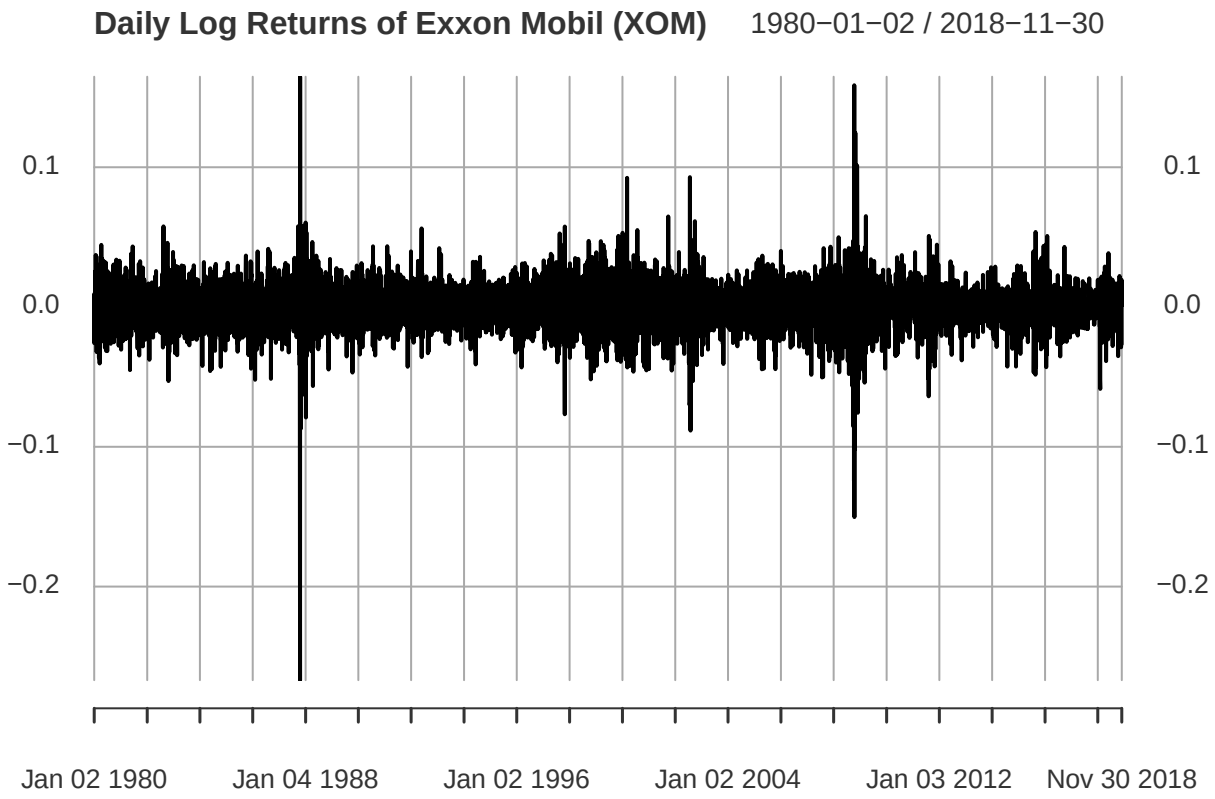
```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.2
```

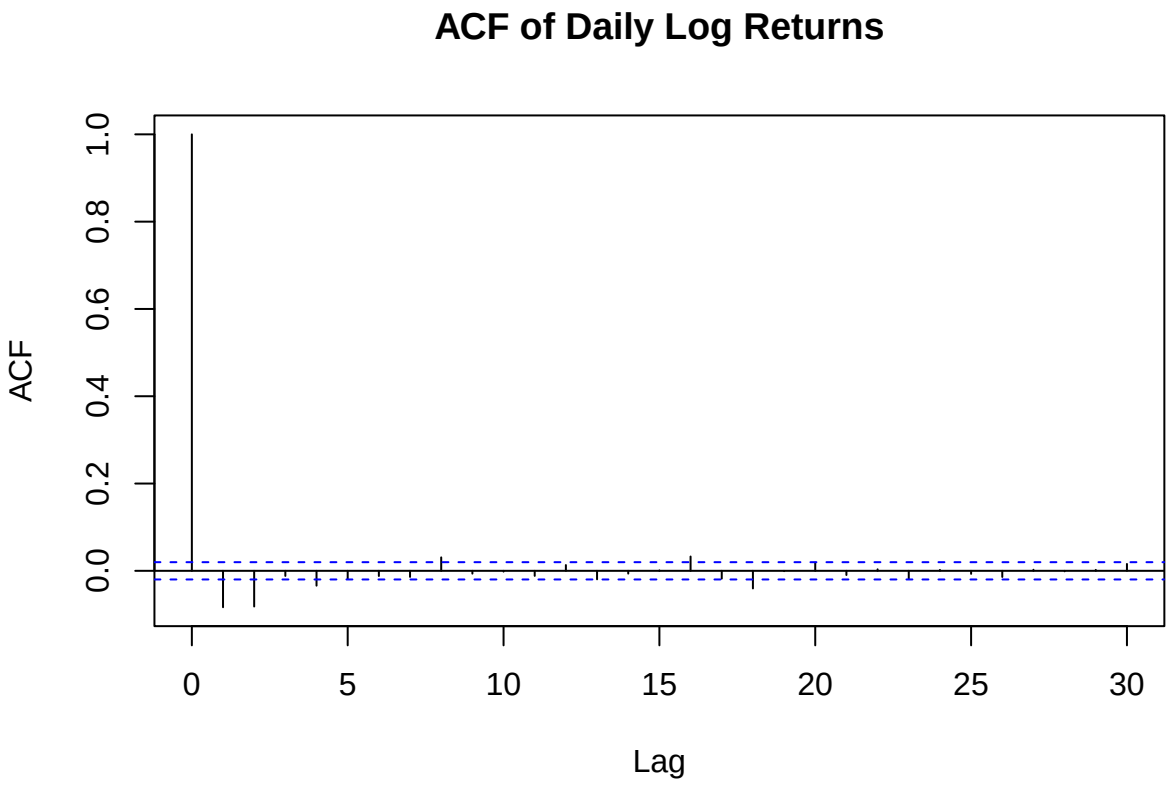
```
symbol <- "XOM"
start_date <- "1980-01-01"
end_date <- "2023-12-01"

getSymbols(symbol, src = "yahoo", from = start_date, to = end_date, auto.assign = TRUE, return.class =
```

```
## [1] "XOM"
```

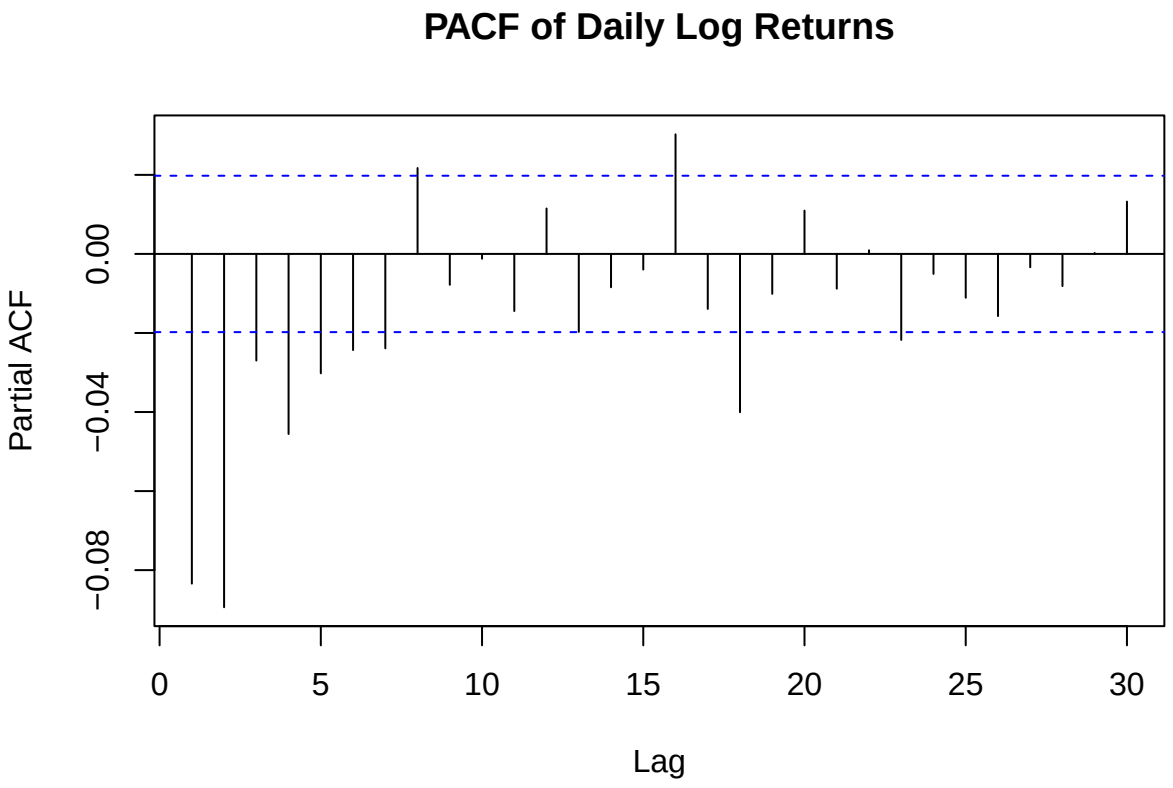
```
daily_returns <- dailyReturn(Cl(XOM), type = "log")

training_data <- daily_returns["1980-01-01/2018-12-01"]
testing_data <- daily_returns["2018-12-02/2023-12-01"]


plot(training_data, main = "Daily Log Returns of Exxon Mobil (XOM)")
```




```r
acf(training_data, lag.max = 30, main = "ACF of Daily Log Returns")
```



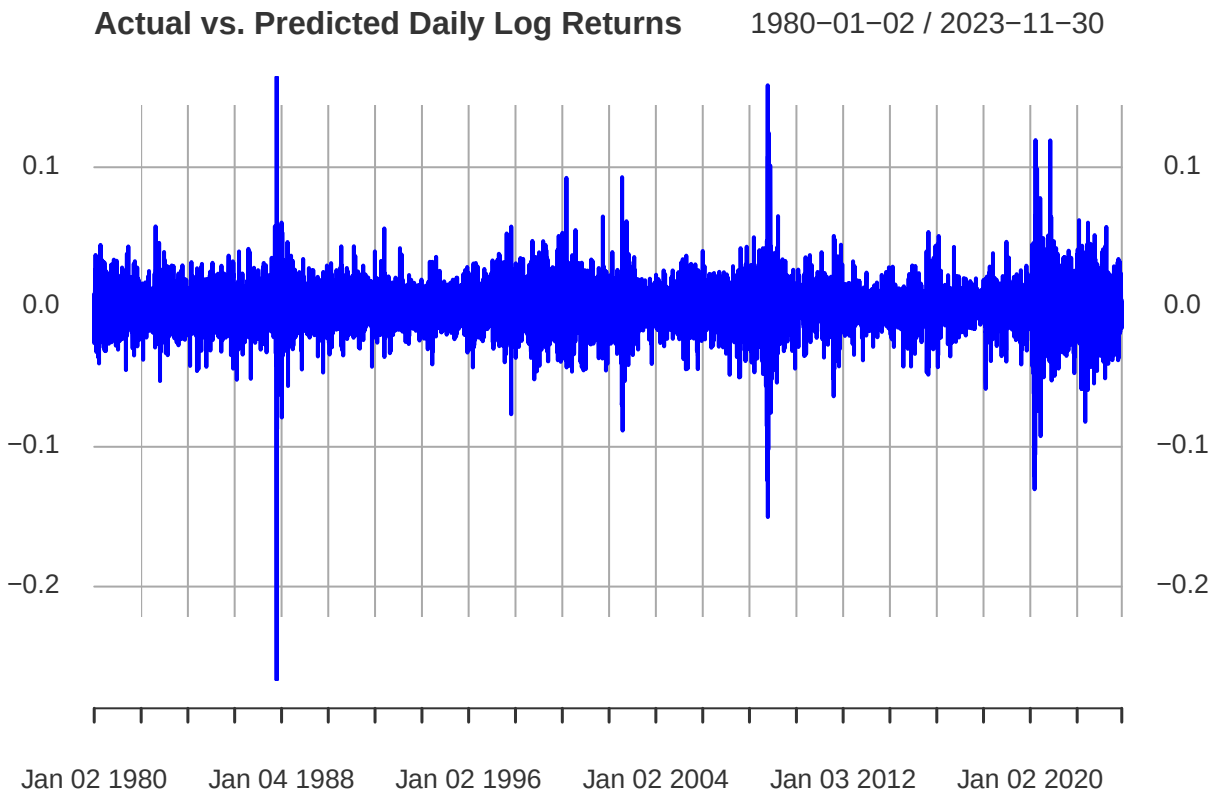

```
pacf(training_data, lag.max = 30, main = "PACF of Daily Log Returns")
```




```
arima_model <- arima(training_data, order = c(1, 0, 1))

summary(arima_model)
```

```
##
## Call:
## arima(x = training_data, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.6351  -0.7398      3e-04
## s.e.  0.0398   0.0347      1e-04
##
## sigma^2 estimated as 0.0002137:  log likelihood = 27548.43,  aic = -55088.87
##
## Training set error measures:
##                         ME       RMSE        MAE MPE MAPE      MASE        ACF1
## Training set -1.361291e-06 0.01461915 0.01036627 NaN  Inf 0.6763224 0.006960939
```

```
forecast_values <- forecast(arima_model, h = length(testing_data))

plot(daily_returns, main = "Actual vs. Predicted Daily Log Returns", col = "blue")
lines(forecast_values$mean, col = "red")
```




```
mse <- mean((forecast_values$mean - testing_data)^2)
```

```
## Warning in mean((forecast_values$mean - testing_data)^2): Incompatible methods
## ("Ops.ts", "Ops.xts") for "-"
```

```
cat("Mean Squared Error on Testing Data:", mse, "\n")
```

```
## Mean Squared Error on Testing Data: 0.0004747715
```

Coefficients: ar1 (AutoRegressive Term): Coefficient: 0.6351 Interpretation: A one-unit increase in the past value (lag 1) of the time series is associated with a 0.6351-unit increase in the present value, holding other variables constant. ma1 (Moving Average Term): Coefficient: -0.7398 Interpretation: A one-unit increase in the past forecast error (lag 1) is associated with a -0.7398-unit decrease in the present value, holding other variables constant. Intercept: Coefficient: 3e-04 (0.0003) Interpretation: This represents the constant term in the model, indicating the expected value of the time series when all other variables are zero. Model Stats: sigma^2 (Residual Variance): Estimated as 0.0002137 This is the estimated variance of the residuals (forecast errors) from the model. A smaller value indicates that the model captures a significant portion of the variability in the data. Log Likelihood: 27548.43 The log likelihood measures how well the model explains the observed data. Higher values indicate a better fit. AIC (Akaike Information Criterion): -55088.87 AIC balances the goodness of fit with the simplicity of the model. Lower AIC values suggest a more parsimonious model. In this case, the negative AIC indicates that the model provides a good balance between fit and complexity. Justification: ARIMA Order (1, 0, 1): The choice of ARIMA(1, 0, 1) indicates that the model includes an autoregressive term of order 1 and a moving average term of order 1. This selection might be justified by observing the autocorrelation and partial autocorrelation functions during the model-building

process. Interpretation of Coefficients: Positive ar1 coefficient suggests a positive autocorrelation at lag 1. Negative ma1 coefficient implies that past forecast errors influence the current value, indicating a corrective mechanism in the model. Residual Variance (sigma^2): The low residual variance (0.0002137) indicates that the model captures a substantial portion of the variability in the daily log returns. Log Likelihood and AIC: The high log likelihood and the negative AIC indicate that the model provides a good fit to the training data. Mean Error: ME measures the average difference between the predicted and actual values. In the model, ME is approximately -1.36e-06. A near-zero ME indicates that, on average, the model's predictions are very close to the actual values. Root Mean Squared Error (RMSE): RMSE represents the square root of the average squared differences between predicted and actual values. The RMSE value is around 0.0146. Lower RMSE values suggest a better fit, indicating that the model's predictions are generally accurate. Mean Absolute Error: MAE is the average of the absolute differences between predicted and actual values. The model achieves an MAE of approximately 0.0104, implying that, on average, the absolute prediction errors are small. Mean Absolute Scaled Error: MASE compares the model's performance to a naïve forecast With a MASE value of approximately 0.6763, the model performs better than a simple mean forecast, indicating its usefulness. Autocorrelation of Errors (ACF1): ACF1 measures the autocorrelation of the model's residuals at lag 1. The reported ACF1 of approximately 0.00696 suggests low autocorrelation in the residuals, indicating that the model has captured most of the serial correlation.