Assignment 3: Time Series Modeling

LIU Liangjie

2025-04-12

Load Required Packages

```
library(tseries)
```

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

```
library(FinTS)
```

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

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

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

```
library(rugarch)
```

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

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

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

```
library(forecast)
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:FinTS':
##
##     Acf
```

```
library(ggplot2)
setwd("/Users/liuliangjie/Desktop/MSDM/5053/4:10")
```

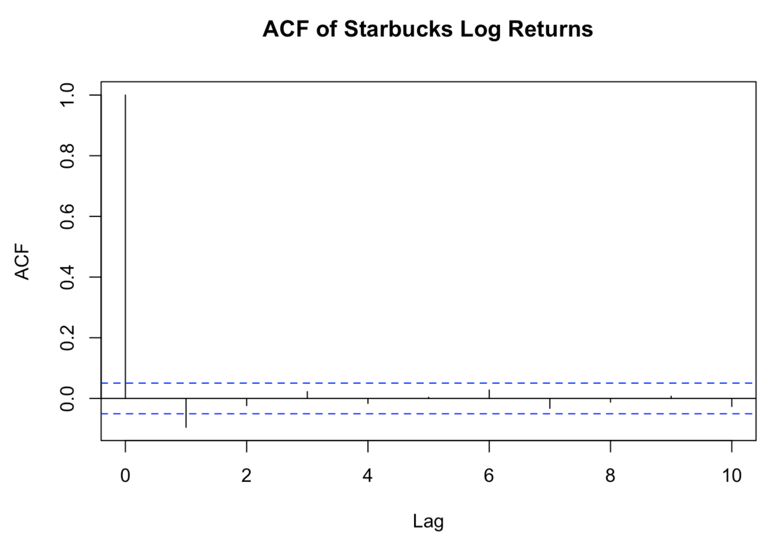
Problem 1: Starbucks and S&P500 Returns

Data Loading and Transformation

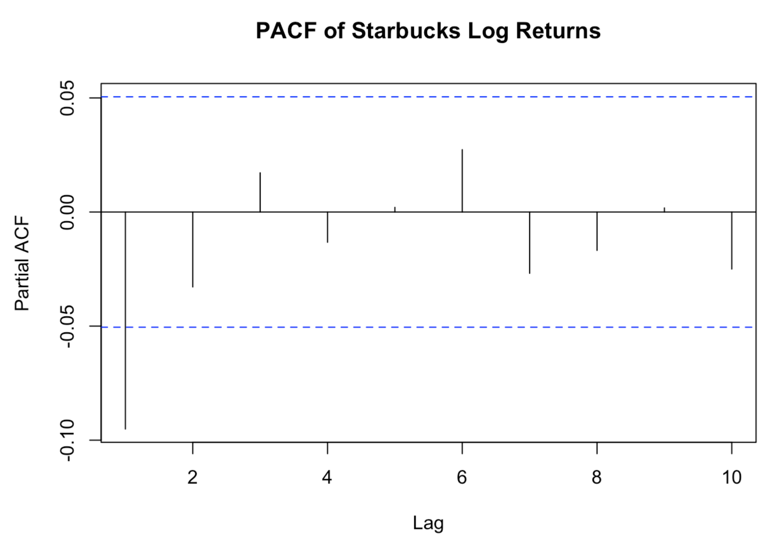
```
# Step 1: Load and preprocess the data
# The file contains three columns: date, SBUX simple return, and S&P500 simple return
df <- read.table("d-sbuxsp0106.txt", header = FALSE)
colnames(df) <- c("Date", "SBUX_simple", "SP500_simple")

# Convert the date column to Date format and sort by time
df$Date <- as.Date(as.character(df$Date), format = "%Y%m%d")
df <- df[order(df$Date), ]
# Step 2: Convert to percentage log returns
df$SBUX_log <- 100 * log(1 + df$SBUX_simple)
df$SP500_log <- 100 * log(1 + df$SP500_simple)
```

a. Serial Correlation in SBUX Log Returns

```
# Step 3: Plot ACF and PACF for SBUX log return
sbux_log <- na.omit(df$SBUX_log)

# ACF plot (up to lag 10)
acf(sbux_log, lag.max = 10, main = "ACF of Starbucks Log Returns")
```




```
# PACF plot (up to lag 10)
pacf(sbux_log, lag.max = 10, main = "PACF of Starbucks Log Returns")
```




```
# Step 4: Perform Ljung-Box test for autocorrelation
ljung_result <- Box.test(sbux_log, lag = 10, type = "Ljung-Box")

cat("Ljung-Box Test Result (lag = 10):\n")
```

```
## Ljung-Box Test Result (lag = 10):
```

```
print(ljung_result)
```

```
##
##  Box-Ljung test
##
## data:  sbux_log
## X-squared = 19.823, df = 10, p-value = 0.03098
```

```
if (ljung_result$p.value < 0.05) {
  cat("Significant autocorrelation detected (reject white noise hypothesis)\n")
} else {
  cat("No significant autocorrelation found\n")
}
```

```
## Significant autocorrelation detected (reject white noise hypothesis)
```

b. ARCH Effect in SBUX Log Returns

```
# Step 2: Perform ARCH-LM test (lag = 10)
# The function ArchTest() in FinTS package tests for ARCH effects

arch_test <- ArchTest(sbux_log, lags = 10)

cat("ARCH-LM Test Result (lags = 10):\n")
```

```
## ARCH-LM Test Result (lags = 10):
```

```
cat(sprintf("LM Statistic: %.4f\n", arch_test$statistic))
```

```
## LM Statistic: 39.3433
```

```
cat(sprintf("LM p-value  : %.4f\n", arch_test$p.value))
```

```
## LM p-value  : 0.0000
```

```
# Step 3: Interpretation of the ARCH-LM test result

if (arch_test$p.value < 0.05) {
  cat("Significant ARCH effect detected (reject null hypothesis of homoskedasticity)\n")
} else {
  cat("No significant ARCH effect found (fail to reject null hypothesis of homoskedasticity)\n")
}
```

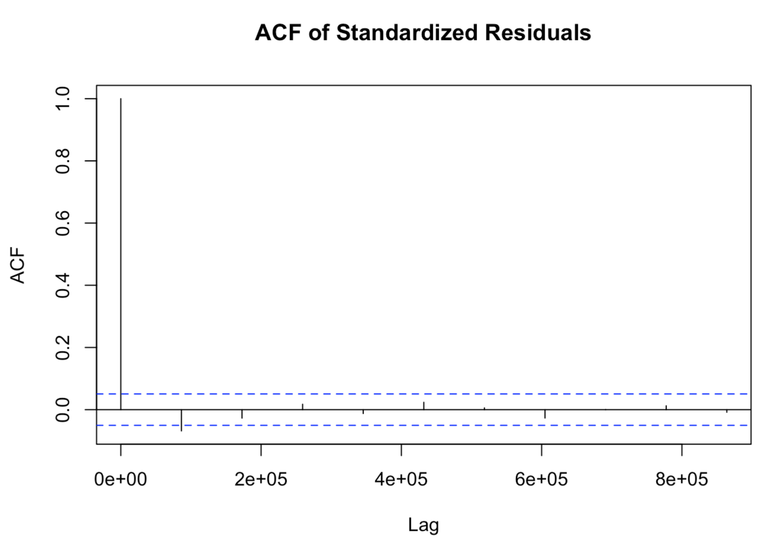
```
## Significant ARCH effect detected (reject null hypothesis of homoskedasticity)
```

c. Fit GARCH(1,1) Model

```
# Step 2: Fit GARCH(1,1) model with normal distribution assumption
# Specify the model: GARCH(1,1) with normal errors
garch_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "norm"
)

# Fit the model to sbux_log
garch_fit <- ugarchfit(spec = garch_spec, data = sbux_log)

# Show summary of the fitted model
show(garch_fit)
```

```
##
## *---------------------------------*
## *          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      0.123262    0.047561   2.5917 0.009551
## omega   0.015955    0.005040   3.1657 0.001547
## alpha1  0.019625    0.001670  11.7513 0.000000
## beta1   0.976124    0.001297 752.3759 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.123262    0.047535    2.5930 0.009513
## omega   0.015955    0.008011    1.9916 0.046412
## alpha1  0.019625    0.003246    6.0467 0.000000
## beta1   0.976124    0.000804 1214.7278 0.000000
##
## LogLikelihood : -3155.673
##
## Information Criteria
## ------------------------------------
##
## Akaike       4.1933
## Bayes        4.2074
## Shibata      4.1933
## Hannan-Quinn 4.1986
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic  p-value
## Lag[1]                      7.059 0.007888
## Lag[2*(p+q)+(p+q)-1][2]     7.615 0.008182
## Lag[4*(p+q)+(p+q)-1][5]     8.488 0.022264
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1035  0.7477
## Lag[2*(p+q)+(p+q)-1][5]    0.8612  0.8902
## Lag[4*(p+q)+(p+q)-1][9]    2.1799  0.8825
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.05276 0.500 2.000  0.8183
## ARCH Lag[5]   1.85436 1.440 1.667  0.5043
## ARCH Lag[7]   2.29708 2.315 1.543  0.6546
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.5078
## Individual Statistics:
## mu     0.06836
## omega  0.07240
## alpha1 0.15786
## beta1  0.11581
##
## 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           1.2120 0.2257
## Negative Sign Bias  0.2165 0.8286
## Positive Sign Bias  1.0692 0.2852
## Joint Effect        1.8253 0.6094
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     59.17    5.230e-06
## 2    30     78.10    2.188e-06
## 3    40     89.31    8.094e-06
## 4    50     99.54    2.683e-05
##
##
## Elapsed time : 0.056633
```

```
# Step 3: Residual diagnostics

# Extract standardized residuals
std_resid <- residuals(garch_fit, standardize = TRUE)
std_resid <- na.omit(std_resid)

# ACF plot of standardized residuals
acf(std_resid, lag.max = 10, main = "ACF of Standardized Residuals")
```




```
# Perform ARCH-LM test on squared residuals
arch_test_resid <- ArchTest(std_resid, lags = 10)

cat("\nARCH-LM test on standardized residuals:\n")
```

```
##
## ARCH-LM test on standardized residuals:
```

```
cat(sprintf("LM Statistic: %.4f, p-value: %.4f\n", arch_test_resid$statistic, arch_test_resid$p.value))
```

```
## LM Statistic: 4.3735, p-value: 0.9289
```

```
if (arch_test_resid$p.value < 0.05) {
  cat("ARCH effect still present in residuals → Model may be insufficient\n")
} else {
  cat("No ARCH effect found in residuals → Model fits well\n")
}
```

```
## No ARCH effect found in residuals → Model fits well
```

```
# Step 4: Write out the fitted GARCH(1,1) model

params <- coef(garch_fit)
mu <- params["mu"]
omega <- params["omega"]
alpha <- params["alpha1"]
beta <- params["beta1"]

cat("\nThe fitted GARCH(1,1) model is:\n")
```

```
##
## The fitted GARCH(1,1) model is:
```

```
cat(sprintf("r_t = %.4f + ε_t\n", mu))
```

```
## r_t = 0.1233 + ε_t
```

```
cat("ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)\n")
```

```
## ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)
```

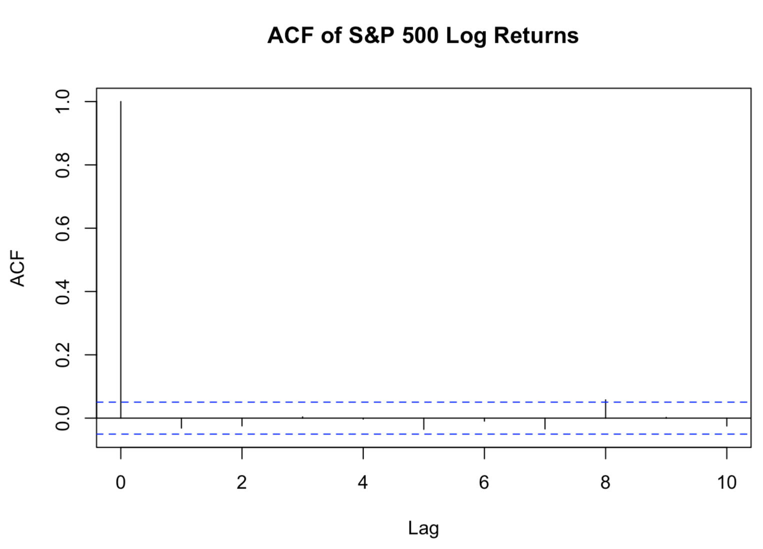
```
cat(sprintf("h_t = %.4f + %.4f * ε_(t-1)^2 + %.4f * h_(t-1)\n", omega, alpha, beta))
```

```
## h_t = 0.0160 + 0.0196 * ε_(t-1)^2 + 0.9761 * h_(t-1)
```

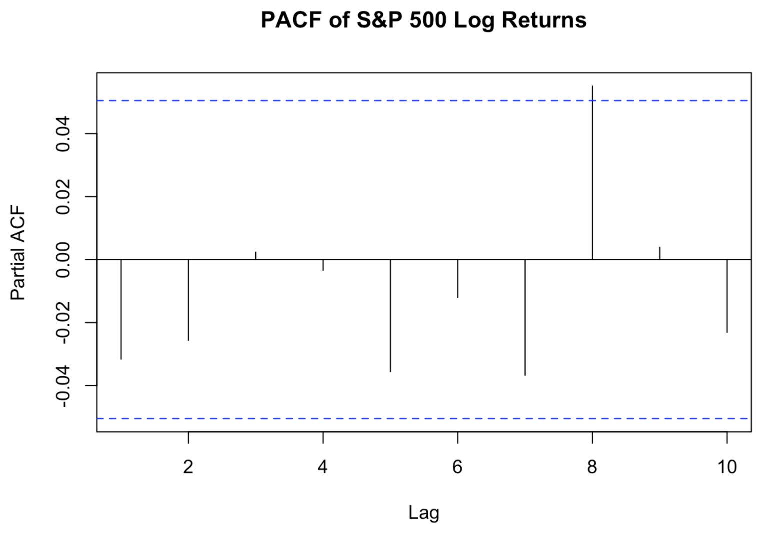
Problem 2: S&P500 Log Returns

a. Serial Correlation Test

```
# Step 1: Extract SP500 log return series
sp_log <- na.omit(df$SP500_log)
# Step 2: ACF and PACF plots for SP500 log returns

acf(sp_log, lag.max = 10, main = "ACF of S&P 500 Log Returns")
```




```
pacf(sp_log, lag.max = 10, main = "PACF of S&P 500 Log Returns")
```




```
# Step 3: Ljung-Box test for autocorrelation (lag = 10)

sp_ljung <- Box.test(sp_log, lag = 10, type = "Ljung-Box")

cat("Ljung-Box Test Result (lag = 10) for SP500 log return:\n")
```

```
## Ljung-Box Test Result (lag = 10) for SP500 log return:
```

```
print(sp_ljung)
```

```
##
##  Box-Ljung test
##
## data:  sp_log
## X-squared = 12.253, df = 10, p-value = 0.2685
```

```
if (sp_ljung$p.value < 0.05) {
  cat("Significant autocorrelation detected in SP500 log returns\n")
} else {
  cat("No significant autocorrelation found in SP500 log returns\n")
}
```

```
## No significant autocorrelation found in SP500 log returns
```

b. ARCH Effect Test

```
# Step 4: Perform ARCH-LM test on SP500 log returns (lag = 10)

arch_test_sp <- ArchTest(sp_log, lags = 10)

cat("\nARCH-LM Test Result (SP500 log return, lags = 10):\n")
```

```
##
## ARCH-LM Test Result (SP500 log return, lags = 10):
```

```
cat(sprintf("LM Statistic: %.4f\n", arch_test_sp$statistic))
```

```
## LM Statistic: 330.2333
```

```
cat(sprintf("LM p-value  : %.4f\n", arch_test_sp$p.value))
```

```
## LM p-value  : 0.0000
```

```
if (arch_test_sp$p.value < 0.05) {
  cat("Significant ARCH effect detected in SP500 log returns\n")
} else {
  cat("No significant ARCH effect found in SP500 log returns\n")
}
```

```
## Significant ARCH effect detected in SP500 log returns
```

c. Fit IGARCH(1,1) Model

```
# Step 2: Fit IGARCH(1,1) model (normal distribution)

# In IGARCH, omega = 0 and alpha + beta = 1 is imposed
igarch_spec <- ugarchspec(
  variance.model = list(model = "iGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "norm"
)

# Fit the model to SP500 log returns
igarch_fit <- ugarchfit(spec = igarch_spec, data = sp_log)

# Step 3: Show model estimation results

show(igarch_fit)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : iGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.039700    0.020102   1.9749 0.048283
## omega   0.003201    0.001920   1.6671 0.095485
## alpha1  0.067541    0.013261   5.0933 0.000000
## beta1   0.932459          NA       NA       NA
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.039700    0.020512   1.9354 0.052938
## omega   0.003201    0.002431   1.3168 0.187912
## alpha1  0.067541    0.017749   3.8053 0.000142
## beta1   0.932459          NA       NA       NA
##
## LogLikelihood : -2007.41
##
## Information Criteria
## ------------------------------------
##
## Akaike       2.6681
## Bayes        2.6787
## Shibata      2.6681
## Hannan-Quinn 2.6720
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.933 0.16440
## Lag[2*(p+q)+(p+q)-1][2]     3.588 0.09731
## Lag[4*(p+q)+(p+q)-1][5]     4.898 0.16142
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.171  0.2793
## Lag[2*(p+q)+(p+q)-1][5]     3.114  0.3867
## Lag[4*(p+q)+(p+q)-1][9]     4.048  0.5808
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]     1.454 0.500 2.000  0.2279
## ARCH Lag[5]     1.943 1.440 1.667  0.4840
## ARCH Lag[7]     2.370 2.315 1.543  0.6393
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.5758
## Individual Statistics:
## mu     0.13820
## omega  0.06809
## alpha1 0.09692
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          0.846 1.01 1.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value     prob sig
## Sign Bias           0.1060 0.915608
## Negative Sign Bias  0.5657 0.571670
## Positive Sign Bias  2.8767 0.004076 ***
## Joint Effect       13.6418 0.003436 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     36.90     0.008162
## 2    30     44.94     0.029837
## 3    40     57.46     0.028571
## 4    50     62.38     0.094972
##
##
## Elapsed time : 0.02325916
```

```
# Extract estimated parameters
params <- coef(igarch_fit)
mu <- params["mu"]
alpha <- params["alpha1"]
beta <- params["beta1"]
alpha_plus_beta <- alpha + beta

# Write out the fitted model expression
cat("\nThe fitted IGARCH(1,1) model is:\n")
```

```
##
## The fitted IGARCH(1,1) model is:
```

```
cat(sprintf("r_t = %.4f + ε_t\n", mu))
```

```
## r_t = 0.0397 + ε_t
```

```
cat("ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)\n")
```

```
## ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)
```

```
cat(sprintf("h_t = %.4f * ε_(t-1)^2 + %.4f * h_(t-1)\n", alpha, beta))
```

```
## h_t = 0.0675 * ε_(t-1)^2 + 0.9325 * h_(t-1)
```

```
cat(sprintf("where α + β = %.4f ≈ 1\n", alpha_plus_beta))
```

```
## where α + β = 1.0000 ≈ 1
```

d. Forecast 1 to 4 Steps Ahead

```
# Step 1: Use the IGARCH model to forecast 1~4 steps ahead

# Forecast horizon = 4
igarch_forecast <- ugarchforecast(igarch_fit, n.ahead = 4)
```

```
## Warning in `setfixed<-`(`*tmp*`, value = as.list(pars)): Unrecognized Parameter
## in Fixed Values: beta1...Ignored
```

```
# Extract forecasted mean and variance
mean_forecast <- fitted(igarch_forecast)
var_forecast <- sigma(igarch_forecast)^2
std_forecast <- sqrt(var_forecast)

# Print forecasted mean and std
cat("Forecast of log returns for S&P500:\n")
```

```
## Forecast of log returns for S&P500:
```

```
for (i in 1:4) {
  cat(sprintf("Step %d: Mean = %.4f, Std = %.4f\n",
              i, mean_forecast[i], std_forecast[i]))
}
```

```
## Step 1: Mean = 0.0397, Std = 0.5003
## Step 2: Mean = 0.0397, Std = 0.5035
## Step 3: Mean = 0.0397, Std = 0.5066
## Step 4: Mean = 0.0397, Std = 0.5098
```

```
# Step 2: Compute 95% confidence interval for 1-step ahead forecast

z_critical <- qnorm(0.975)  # 95% z critical value
mean_1 <- mean_forecast[1]
std_1 <- std_forecast[1]

lower <- mean_1 - z_critical * std_1
upper <- mean_1 + z_critical * std_1

cat("\n1-step ahead forecast interval (95% CI):\n")
```

```
##
## 1-step ahead forecast interval (95% CI):
```

```
cat(sprintf("Mean: %.4f\n", mean_1))
```

```
## Mean: 0.0397
```

```
cat(sprintf("95%% CI: [%.4f, %.4f]\n", lower, upper))
```

```
## 95% CI: [-0.9408, 1.0202]
```

Problem 3: Extensions on SBUX

a. Fit GARCH(1,1)-M Model

```
# Step 2: Fit GARCH(1,1)-M model using normal distribution

# Specify GARCH-M(1,1) model with volatility-in-mean term
garchm_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE,
                    archm = TRUE, archpow = 1),  # use sqrt(h_t)
  distribution.model = "norm"
)

# Fit the model
garchm_fit <- ugarchfit(spec = garchm_spec, data = df$SBUX_log)

# Step 3: Print summary of fitted model

show(garchm_fit)
```

```
##
## *---------------------------------*
## *          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      0.368078    0.129344    2.8457 0.004431
## archm  -0.134049    0.065042   -2.0610 0.039308
## omega   0.016216    0.005560    2.9167 0.003537
## alpha1  0.019815    0.001788   11.0837 0.000000
## beta1   0.975863    0.000659 1480.5805 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error   t value Pr(>|t|)
## mu      0.368078    0.112147    3.2821 0.001030
## archm  -0.134049    0.055782   -2.4031 0.016258
## omega   0.016216    0.008874    1.8273 0.067661
## alpha1  0.019815    0.003081    6.4313 0.000000
## beta1   0.975863    0.000601 1624.3544 0.000000
##
## LogLikelihood : -3155.157
##
## Information Criteria
## ------------------------------------
##
## Akaike       4.1940
## Bayes        4.2116
## Shibata      4.1939
## Hannan-Quinn 4.2005
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic  p-value
## Lag[1]                      7.136 0.007554
## Lag[2*(p+q)+(p+q)-1][2]     7.705 0.007744
## Lag[4*(p+q)+(p+q)-1][5]     8.576 0.021163
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.1239  0.7248
## Lag[2*(p+q)+(p+q)-1][5]    0.9203  0.8774
## Lag[4*(p+q)+(p+q)-1][9]    2.2299  0.8759
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.05828 0.500 2.000  0.8092
## ARCH Lag[5]   1.84526 1.440 1.667  0.5065
## ARCH Lag[7]   2.27736 2.315 1.543  0.6587
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.5369
## Individual Statistics:
## mu     0.08532
## archm  0.07706
## omega  0.07730
## alpha1 0.15608
## beta1  0.11843
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.28 1.47 1.88
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           1.2002 0.2302
## Negative Sign Bias  0.1065 0.9152
## Positive Sign Bias  1.0626 0.2881
## Joint Effect        1.8936 0.5948
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     60.58    3.134e-06
## 2    30     82.52    4.922e-07
## 3    40     97.17    7.207e-07
## 4    50    101.66    1.499e-05
##
##
## Elapsed time : 0.1634262
```

```
# Extract parameters
params <- coef(garchm_fit)
mu <- params["mu"]
lambda <- params["archm"]
omega <- params["omega"]
alpha <- params["alpha1"]
beta <- params["beta1"]
# Step 4: Write out the fitted model expression

cat("\nThe fitted GARCH(1,1)-M model is:\n")
```

```
##
## The fitted GARCH(1,1)-M model is:
```

```
cat(sprintf("r_t = %.4f + %.4f * h_t + ε_t\n", mu, lambda))
```

```
## r_t = 0.3681 + -0.1340 * h_t + ε_t
```

```
cat("ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)\n")
```

```
## ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)
```

```
cat(sprintf("h_t = %.4f + %.4f * ε_(t-1)^2 + %.4f * h_(t-1)\n", omega, alpha, beta))
```

```
## h_t = 0.0162 + 0.0198 * ε_(t-1)^2 + 0.9759 * h_(t-1)
```

b. Check Significance of ARCH-in-Mean Term

```
# Step 1: Extract parameter estimates and p-values from GARCH-M model

# coef(garchm_fit) gives point estimates
# garchm_fit@fit$matcoef contains: Estimate, Std. Error, t-value, and p-value
param_table <- as.data.frame(garchm_fit@fit$matcoef)

# Rename columns for clarity
colnames(param_table) <- c("Estimate", "Std.Error", "t.value", "p.value")

# Print the parameter table
cat("\nParameter estimates and p-values:\n")
```

```
##
## Parameter estimates and p-values:
```

```
print(round(param_table, 4))
```

```
##        Estimate Std.Error   t.value p.value
## mu       0.3681    0.1293    2.8457  0.0044
## archm   -0.1340    0.0650   -2.0609  0.0393
## omega    0.0162    0.0056    2.9167  0.0035
## alpha1   0.0198    0.0018   11.0837  0.0000
## beta1    0.9759    0.0007 1480.5805  0.0000
```

```
# Step 2: Check significance of ARCH-in-mean parameter (lambda)

lambda_pval <- param_table["archm", "p.value"]

if (lambda_pval < 0.05) {
  cat(sprintf("\nARCH-in-mean parameter is significant (p = %.4f < 0.05)\n", lambda_pval))
} else {
  cat(sprintf("\nARCH-in-mean parameter is not significant (p = %.4f ≥ 0.05)\n", lambda_pval))
}
```

```
##
## ARCH-in-mean parameter is significant (p = 0.0393 < 0.05)
```

c. Fit EGARCH(1,1) Model

```
# Step 1: Fit EGARCH(1,1) model with normal distribution

egarch_spec <- ugarchspec(
  variance.model = list(model = "eGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "norm"
)

egarch_fit <- ugarchfit(spec = egarch_spec, data = sbux_log)

# Step 2: Display model estimation results

show(egarch_fit)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : eGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error     t value Pr(>|t|)
## mu      0.091660    0.048790      1.8787 0.060291
## omega   0.010882    0.001546      7.0395 0.000000
## alpha1 -0.039487    0.008608     -4.5872 0.000004
## beta1   0.993387    0.000003 357183.0235 0.000000
## gamma1  0.047908    0.002278     21.0315 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error     t value Pr(>|t|)
## mu      0.091660    0.057037      1.6070  0.10805
## omega   0.010882    0.001995      5.4554  0.00000
## alpha1 -0.039487    0.013822     -2.8568  0.00428
## beta1   0.993387    0.000005 212622.6891  0.00000
## gamma1  0.047908    0.003084     15.5363  0.00000
##
## LogLikelihood : -3136.98
##
## Information Criteria
## ------------------------------------
##
## Akaike       4.1698
## Bayes        4.1875
## Shibata      4.1698
## Hannan-Quinn 4.1764
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic  p-value
## Lag[1]                      7.071 0.007832
## Lag[2*(p+q)+(p+q)-1][2]     7.494 0.008814
## Lag[4*(p+q)+(p+q)-1][5]     8.259 0.025401
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.07892  0.7788
## Lag[2*(p+q)+(p+q)-1][5]   0.82551  0.8977
## Lag[4*(p+q)+(p+q)-1][9]   2.16006  0.8851
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.2468 0.500 2.000  0.6193
## ARCH Lag[5]    1.7998 1.440 1.667  0.5172
## ARCH Lag[7]    2.1994 2.315 1.543  0.6751
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.6498
## Individual Statistics:
## mu     0.02710
## omega  0.08806
## alpha1 0.27486
## beta1  0.14059
## gamma1 0.09408
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:         1.28 1.47 1.88
## Individual Statistic:    0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value   prob sig
## Sign Bias           1.2353 0.2169
## Negative Sign Bias  0.7896 0.4299
## Positive Sign Bias  1.4541 0.1461
## Joint Effect        2.7562 0.4308
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     58.51    6.648e-06
## 2    30     71.41    1.937e-05
## 3    40     85.06    2.837e-05
## 4    50    104.65    6.508e-06
##
##
## Elapsed time : 0.1472561
```

```
# Extract parameters
eg_params <- coef(egarch_fit)
mu <- eg_params["mu"]
omega <- eg_params["omega"]
alpha <- eg_params["alpha1"]
beta <- eg_params["beta1"]
gamma <- eg_params["gamma1"]

# Step 3: Write out the EGARCH(1,1) model expression

cat("\nThe fitted EGARCH(1,1) model is:\n")
```

```
##
## The fitted EGARCH(1,1) model is:
```

```
cat(sprintf("r_t = %.4f + ε_t\n", mu))
```

```
## r_t = 0.0917 + ε_t
```

```
cat("ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)\n")
```

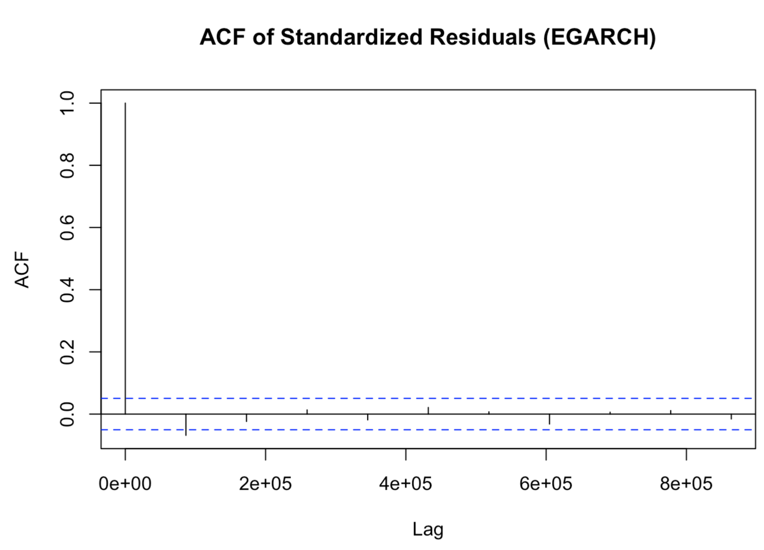
```
## ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)
```

```
cat(sprintf("log(h_t) = %.4f + %.4f * |ε_(t-1)/√h_(t-1)| + %.4f * (ε_(t-1)/√h_(t-1)) + %.4f * log(h_(t-1))\n",
            omega, alpha, gamma, beta))
```

```
## log(h_t) = 0.0109 + -0.0395 * |ε_(t-1)/√h_(t-1)| + 0.0479 * (ε_(t-1)/√h_(t-1)) + 0.9934 * log(h_(t-1))
```

d. Check Significance of Leverage Term

```
# Step 4: Residual autocorrelation and ARCH effect check

# Standardized residuals
resid_std <- residuals(egarch_fit, standardize = TRUE)
resid_std <- na.omit(resid_std)

# ACF plot of standardized residuals
acf(resid_std, lag.max = 10, main = "ACF of Standardized Residuals (EGARCH)")
```




```
# ARCH-LM test for residuals
arch_test <- ArchTest(resid_std, lags = 10)

cat(sprintf("\nARCH-LM Test on EGARCH residuals: LM Statistic = %.4f, p-value = %.4f\n",
            arch_test$statistic, arch_test$p.value))
```

```
##
## ARCH-LM Test on EGARCH residuals: LM Statistic = 4.9075, p-value = 0.8973
```

```
if (arch_test$p.value < 0.05) {
  cat("ARCH effect still present in residuals → Model may be insufficient\n")
} else {
  cat("No significant ARCH effect in residuals → Model fits well\n")
}
```

```
## No significant ARCH effect in residuals → Model fits well
```

```
# Step: Extract EGARCH(1,1) parameter table and check significance of gamma

# garch@fit$matcoef contains the coefficient matrix: Estimate, Std. Error, t-value, and p-value
egarch_table <- as.data.frame(egarch_fit@fit$matcoef)
colnames(egarch_table) <- c("Estimate", "Std.Error", "t.value", "p.value")

# Print parameter table
cat("\nParameter estimates for EGARCH(1,1):\n")
```

```
##
## Parameter estimates for EGARCH(1,1):
```

```
print(round(egarch_table, 4))
```

```
##        Estimate Std.Error     t.value p.value
## mu       0.0917    0.0488      1.8787  0.0603
## omega    0.0109    0.0015      7.0395  0.0000
## alpha1  -0.0395    0.0086     -4.5872  0.0000
## beta1    0.9934    0.0000 357183.0235  0.0000
## gamma1   0.0479    0.0023     21.0315  0.0000
```

```
# Step: Check significance of gamma (leverage effect)

gamma_pval <- egarch_table["gamma1", "p.value"]

if (gamma_pval < 0.05) {
  cat(sprintf("\nGamma parameter is significant (p = %.4f < 0.05) → Leverage effect detected\n", gamma_pval))
} else {
  cat(sprintf("\nGamma parameter is not significant (p = %.4f ≥ 0.05) → No strong evidence of leverage effect\n",
gamma_pval))
}
```

```
##
## Gamma parameter is significant (p = 0.0000 < 0.05) → Leverage effect detected
```

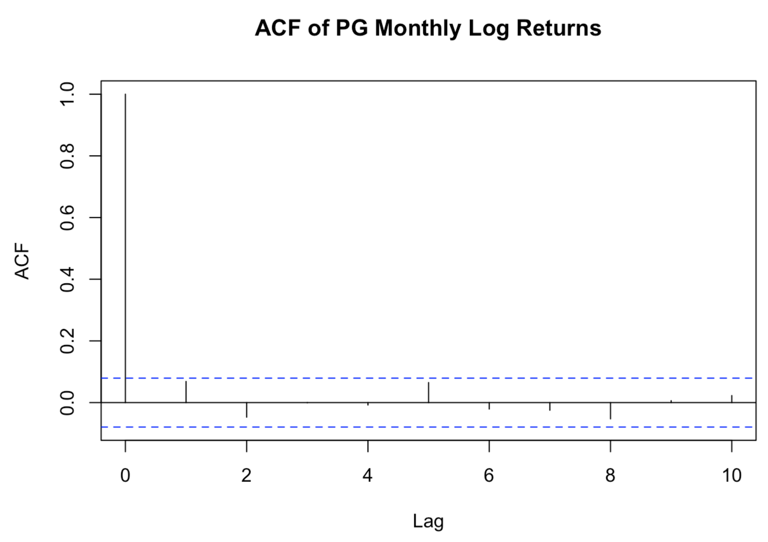
Problem 4: PG Monthly Returns

Data Loading and Log Return Calculation

```
# Step 1: Read PG monthly return data

# The file contains two columns: Date and PG simple return
pg_df <- read.table("m-pg5606.txt", header = FALSE)
colnames(pg_df) <- c("Date", "PG_simple")

# Parse date and sort by time
pg_df$Date <- as.Date(as.character(pg_df$Date), format = "%Y%m%d")
pg_df <- pg_df[order(pg_df$Date), ]
# Step 2: Convert to percentage log returns

pg_df$PG_log <- 100 * log(1 + pg_df$PG_simple)
```

a. Serial Correlation Test

```
# Step 3: ACF plot and Ljung-Box test for PG log returns

pg_log <- na.omit(pg_df$PG_log)

# ACF plot (up to lag 10)
acf(pg_log, lag.max = 10, main = "ACF of PG Monthly Log Returns")
```




```
# Ljung-Box test (up to lag 10)
ljung_pg <- Box.test(pg_log, lag = 10, type = "Ljung-Box")

cat("Ljung-Box Test Result (lag = 10):\n")
```

```
## Ljung-Box Test Result (lag = 10):
```

```
print(ljung_pg)
```

```
##
##  Box-Ljung test
##
## data:  pg_log
## X-squared = 9.65, df = 10, p-value = 0.4717
```

```
if (ljung_pg$p.value < 0.05) {
  cat("Significant autocorrelation detected in PG monthly log returns\n")
} else {
  cat("No significant autocorrelation found in PG monthly log returns\n")
}
```

```
## No significant autocorrelation found in PG monthly log returns
```

b. Fit GARCH(1,1)

```
# Step 1: Prepare PG log return series

pg_log <- na.omit(pg_df$PG_log)

# Step 2: Fit GARCH(1,1) model with normal distribution

library(rugarch)

pg_spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0), include.mean = TRUE),
  distribution.model = "norm"
)

pg_fit <- ugarchfit(spec = pg_spec, data = pg_log)

# Step 3: Show model estimation results

show(pg_fit)
```

```
##
## *---------------------------------*
## *          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      0.856293    0.158010   5.4192 0.000000
## omega   0.853384    0.393121   2.1708 0.029947
## alpha1  0.096377    0.027015   3.5675 0.000360
## beta1   0.862390    0.031428  27.4398 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.856293    0.173354   4.9396 0.000001
## omega   0.853384    0.477145   1.7885 0.073692
## alpha1  0.096377    0.032547   2.9611 0.003065
## beta1   0.862390    0.038132  22.6160 0.000000
##
## LogLikelihood : -1743.945
##
## Information Criteria
## ------------------------------------
##
## Akaike       5.7122
## Bayes        5.7411
## Shibata      5.7122
## Hannan-Quinn 5.7235
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.995  0.1578
## Lag[2*(p+q)+(p+q)-1][2]     2.309  0.2163
## Lag[4*(p+q)+(p+q)-1][5]     2.929  0.4202
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                     0.4414  0.5065
## Lag[2*(p+q)+(p+q)-1][5]    0.7236  0.9183
## Lag[4*(p+q)+(p+q)-1][9]    1.3915  0.9640
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.05061 0.500 2.000  0.8220
## ARCH Lag[5]   0.28476 1.440 1.667  0.9443
## ARCH Lag[7]   0.69216 2.315 1.543  0.9578
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.6997
## Individual Statistics:
## mu     0.04737
## omega  0.13041
## alpha1 0.06199
## beta1  0.10421
##
## 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           2.9785 3.012e-03 ***
## Negative Sign Bias  1.2096 2.269e-01
## Positive Sign Bias  0.9676 3.336e-01
## Joint Effect       22.4166 5.342e-05 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     52.12    6.353e-05
## 2    30     51.82    5.704e-03
## 3    40     75.45    4.126e-04
## 4    50     79.99    3.406e-03
##
##
## Elapsed time : 0.03102279
```

```
# Extract estimated parameters
params <- coef(pg_fit)
mu <- params["mu"]
omega <- params["omega"]
alpha <- params["alpha1"]
beta <- params["beta1"]
# Step 4: Write out the fitted GARCH(1,1) model

cat("\nThe fitted GARCH(1,1) model for PG is:\n")
```

```
##
## The fitted GARCH(1,1) model for PG is:
```

```
cat(sprintf("r_t = %.4f + ε_t\n", mu))
```

```
## r_t = 0.8563 + ε_t
```

```
cat("ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)\n")
```

```
## ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)
```

```
cat(sprintf("h_t = %.4f + %.4f * ε_(t-1)^2 + %.4f * h_(t-1)\n", omega, alpha, beta))
```

```
## h_t = 0.8534 + 0.0964 * ε_(t-1)^2 + 0.8624 * h_(t-1)
```

c. Forecast 1 to 5 Steps Ahead

```
# Step 1: Forecast PG log returns 1~5 steps ahead using GARCH(1,1) model

pg_forecast <- ugarchforecast(pg_fit, n.ahead = 5)

# Extract forecasted mean and standard deviation
mean_fc <- fitted(pg_forecast)
std_fc <- sigma(pg_forecast)
var_fc <- std_fc^2

cat("PG log return forecasts (next 1~5 steps):\n")
```

```
## PG log return forecasts (next 1~5 steps):
```

```
for (i in 1:5) {
  cat(sprintf("Step %d: Mean = %.4f, Std = %.4f\n", i, mean_fc[i], std_fc[i]))
}
```

```
## Step 1: Mean = 0.8563, Std = 2.9521
## Step 2: Mean = 0.8563, Std = 3.0347
## Step 3: Mean = 0.8563, Std = 3.1117
## Step 4: Mean = 0.8563, Std = 3.1839
## Step 5: Mean = 0.8563, Std = 3.2515
```

```
# Step 2: Compute 95% confidence interval for 1-step ahead forecast

z_975 <- qnorm(0.975)  # 95% critical z-value

mean_1 <- mean_fc[1]
std_1 <- std_fc[1]

lower <- mean_1 - z_975 * std_1
upper <- mean_1 + z_975 * std_1

cat("\nPG 1-step ahead forecast 95% CI:\n")
```

```
##
## PG 1-step ahead forecast 95% CI:
```

```
cat(sprintf("Mean: %.4f\n", mean_1))
```

```
## Mean: 0.8563
```

```
cat(sprintf("95%% CI: [%.4f, %.4f]\n", lower, upper))
```

```
## 95% CI: [-4.9298, 6.6424]
```

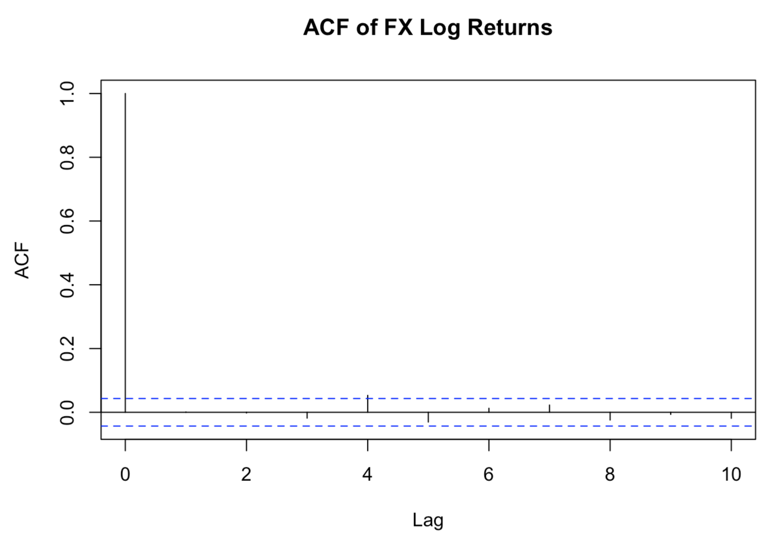
Problem 5: EUR/USD Exchange Rate

Load and Transform Exchange Rate Data

```
# Step 1: Read exchange rate data
# File contains 4 columns: Year, Month, Day, ExchangeRate
fx_df <- read.table("d-exuseu.txt", header = FALSE)
colnames(fx_df) <- c("Year", "Month", "Day", "Rate")

# Combine year/month/day into one Date column
fx_df$Date <- as.Date(paste(fx_df$Year, fx_df$Month, fx_df$Day, sep = "-"))

# Sort by Date
fx_df <- fx_df[order(fx_df$Date), ]

# Step 2: Compute percentage log return
fx_df$LogReturn <- c(NA, 100 * diff(log(fx_df$Rate)))

# Remove NA to get clean log return series
fx_log <- na.omit(fx_df$LogReturn)
```

a. Serial Correlation Test

```
# Step 3a: ACF plot and Ljung-Box test

acf(fx_log, lag.max = 10, main = "ACF of FX Log Returns")
```




```
ljung_fx <- Box.test(fx_log, lag = 10, type = "Ljung-Box")

cat("Ljung-Box Test (lag = 10):\n")
```

```
## Ljung-Box Test (lag = 10):
```

```
print(ljung_fx)
```

```
##
##  Box-Ljung test
##
## data:  fx_log
## X-squared = 11.921, df = 10, p-value = 0.2904
```

```
if (ljung_fx$p.value < 0.05) {
  cat("Significant autocorrelation detected in FX log returns\n")
} else {
  cat("No significant autocorrelation found in FX log returns\n")
}
```

```
## No significant autocorrelation found in FX log returns
```

b. ARCH Effect Test

```
# Step 3b: ARCH-LM test

library(FinTS)
arch_result <- ArchTest(fx_log, lags = 10)

cat("\nARCH-LM Test Result:\n")
```

```
##
## ARCH-LM Test Result:
```

```
cat(sprintf("LM Statistic = %.4f, p-value = %.4f\n", arch_result$statistic, arch_result$p.value))
```

```
## LM Statistic = 26.5889, p-value = 0.0030
```

```
if (arch_result$p.value < 0.05) {
  cat("ARCH effect detected in FX log returns\n")
} else {
  cat("No ARCH effect detected in FX log returns\n")
}
```

```
## ARCH effect detected in FX log returns
```

c. Fit IGARCH(1,1) Model

```
# Step 1: Prepare log return data for exchange rate

fx_log <- na.omit(fx_df$LogReturn)

# Step 2: Fit IGARCH(1,1) model (with normal distribution)

library(rugarch)

# Specify IGARCH(1,1): omega = 0, alpha + beta = 1
igarch_spec <- ugarchspec(
  variance.model = list(model = "iGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(0, 0)),
  distribution.model = "norm"
)

# Fit the model to fx_log
igarch_fit <- ugarchfit(spec = igarch_spec, data = fx_log)

# Step 3: Print estimation results and model expression

show(igarch_fit)
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : iGARCH(1,1)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.011123    0.012801 0.868957 0.384871
## omega   0.000004    0.000108 0.034425 0.972538
## alpha1  0.016037    0.003822 4.196090 0.000027
## beta1   0.983963          NA       NA       NA
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.011123    0.013113 0.848272 0.396286
## omega   0.000004    0.000070 0.052819 0.957876
## alpha1  0.016037    0.004609 3.479639 0.000502
## beta1   0.983963          NA       NA       NA
##
## LogLikelihood : -1867.463
##
## Information Criteria
## ------------------------------------
##
## Akaike       1.8125
## Bayes        1.8207
## Shibata      1.8125
## Hannan-Quinn 1.8155
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.05866  0.8086
## Lag[2*(p+q)+(p+q)-1][2]   0.07752  0.9357
## Lag[4*(p+q)+(p+q)-1][5]   2.88184  0.4291
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      2.339  0.1262
## Lag[2*(p+q)+(p+q)-1][5]     3.983  0.2562
## Lag[4*(p+q)+(p+q)-1][9]     5.800  0.3218
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.7202 0.500 2.000  0.3961
## ARCH Lag[5]    1.2473 1.440 1.667  0.6612
## ARCH Lag[7]    2.4343 2.315 1.543  0.6260
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  0.9087
## Individual Statistics:
## mu     0.3856
## omega  0.3724
## alpha1 0.1687
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          0.846 1.01 1.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias            1.921 0.05484   *
## Negative Sign Bias   2.118 0.03432  **
## Positive Sign Bias   1.154 0.24881
## Joint Effect        11.167 0.01085  **
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     49.02    0.0001822
## 2    30     61.78    0.0003685
## 3    40     59.41    0.0191550
## 4    50     84.01    0.0013687
##
##
## Elapsed time : 0.04500985
```

```
# Step 4: Diagnostic checking of the IGARCH model

# Extract standardized residuals
std_resid <- residuals(igarch_fit, standardize = TRUE)

# Ljung-Box test on residuals (autocorrelation)
ljung_resid <- Box.test(std_resid, lag = 10, type = "Ljung-Box")

cat("\nLjung-Box Test on residuals (lag = 10):\n")
```

```
##
## Ljung-Box Test on residuals (lag = 10):
```

```
print(ljung_resid)
```

```
##
##  Box-Ljung test
##
## data:  std_resid
## X-squared = 9.6109, df = 10, p-value = 0.4753
```

```
# ARCH-LM test on residuals
library(FinTS)
arch_test <- ArchTest(std_resid, lags = 10)

cat("\nARCH-LM Test on standardized residuals:\n")
```

```
##
## ARCH-LM Test on standardized residuals:
```

```
cat(sprintf("LM Statistic = %.4f, p-value = %.4f\n",
            arch_test$statistic, arch_test$p.value))
```

```
## LM Statistic = 11.6152, p-value = 0.3116
```

```
if (arch_test$p.value < 0.05) {
  cat("ARCH effect still present in residuals → model may be inadequate\n")
} else {
  cat("No significant ARCH effect in residuals → model fits well\n")
}
```

```
## No significant ARCH effect in residuals → model fits well
```

```
# Extract parameters
params <- coef(igarch_fit)
mu <- params["mu"]
alpha <- params["alpha1"]
beta <- params["beta1"]
alpha_beta <- alpha + beta

# Print model expression
cat("\nThe fitted IGARCH(1,1) model is:\n")
```

```
##
## The fitted IGARCH(1,1) model is:
```

```
cat(sprintf("r_t = %.4f + ε_t\n", mu))
```

```
## r_t = 0.0111 + ε_t
```

```
cat("ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)\n")
```

```
## ε_t = z_t * sqrt(h_t),   z_t ~ N(0, 1)
```

```
cat(sprintf("h_t = %.4f * ε_(t-1)^2 + %.4f * h_(t-1)\n", alpha, beta))
```

```
## h_t = 0.0160 * ε_(t-1)^2 + 0.9840 * h_(t-1)
```

```
cat(sprintf("(α + β = %.4f)\n", alpha_beta))
```

```
## (α + β = 1.0000)
```

d. Forecast 1 to 4 Steps Ahead

```
# Step 1: Forecast 1~4 steps ahead using the fitted IGARCH(1,1) model

forecast_fx <- ugarchforecast(igarch_fit, n.ahead = 4)
```

```
## Warning in `setfixed<-`(`*tmp*`, value = as.list(pars)): Unrecognized Parameter
## in Fixed Values: beta1...Ignored
```

```
# Extract forecasted mean and standard deviation
mean_fc <- fitted(forecast_fx)
std_fc <- sigma(forecast_fx)
var_fc <- std_fc^2

# Print forecasts
cat("Forecast of FX log returns (USD/EUR):\n")
```

```
## Forecast of FX log returns (USD/EUR):
```

```
for (i in 1:4) {
  cat(sprintf("Step %d: Mean = %.4f, Std = %.4f\n", i, mean_fc[i], std_fc[i]))
}
```

```
## Step 1: Mean = 0.0111, Std = 0.3746
## Step 2: Mean = 0.0111, Std = 0.3746
## Step 3: Mean = 0.0111, Std = 0.3746
## Step 4: Mean = 0.0111, Std = 0.3746
```

```
# Step 2: Compute 95% confidence interval for 1-step ahead forecast

z_975 <- qnorm(0.975)  # 95% critical z value

mean_1 <- mean_fc[1]
std_1 <- std_fc[1]

lower <- mean_1 - z_975 * std_1
upper <- mean_1 + z_975 * std_1

cat("\n1-step ahead forecast 95% CI:\n")
```

```
##
## 1-step ahead forecast 95% CI:
```

```
cat(sprintf("Mean: %.4f\n", mean_1))
```

```
## Mean: 0.0111
```

```
cat(sprintf("95%% CI: [%.4f, %.4f]\n", lower, upper))
```

```
## 95% CI: [-0.7230, 0.7453]
```