ARMA(1,1) Parameter Estimation

Generated Report

2024-10-12

Question 2

Generate 1000 realizations of an ARMA(1,1) process with ϕ = 0.9, θ = 0.5, and σ^2 = 1 for T = 50, 200, 500.

We will estimate the parameters using Maximum Likelihood Estimation (MLE) and compare them to the true values in terms of:

- Mean Squared Error (MSE)
- Mean Absolute Deviation (MAD)
- Coverage of the 95% Confidence Interval

```
# Load necessary libraries
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

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

```
library(MASS)
```

```r
set.seed(123)

# Define parameters
phi <- 0.9
theta <- 0.5
sigma2 <- 1

# Function to generate and fit ARMA(1,1) model
simulate_arma <- function(T) {
  arma_sim <- arima.sim(n = T, list(ar = phi, ma = theta), sd = sqrt(sigma2))
  fit <- tryCatch({
    Arima(arma_sim, order = c(1, 0, 1), method = "ML")
  }, error = function(e) {
    message("Error in fitting ARMA, retrying with differencing...")
    Arima(arma_sim, order = c(1, 1, 1), method = "ML")
  })
  return(fit)
}

# Define sample sizes
T_values <- c(50, 200, 500)
n_sim <- 1000

# Initialize storage for estimates
phi_estimates <- matrix(NA, n_sim, length(T_values))
theta_estimates <- matrix(NA, n_sim, length(T_values))
sigma2_estimates <- matrix(NA, n_sim, length(T_values))

# Run simulations
for (i in 1:length(T_values)) {
  for (j in 1:n_sim) {
    fit <- simulate_arma(T_values[i])
    phi_estimates[j, i] <- coef(fit)["ar1"]
    theta_estimates[j, i] <- coef(fit)["ma1"]
    sigma2_estimates[j, i] <- fit$sigma2
  }
}

# Compute statistics
true_params <- c(phi, theta, sigma2)

mse <- function(estimates, true_value) {
  return(mean((estimates - true_value)^2))
}

mad <- function(estimates, true_value) {
  return(mean(abs(estimates - true_value)))
}

coverage_95_ci <- function(estimates, true_value, se_estimates) {
  lower <- estimates - 1.96 * se_estimates
  upper <- estimates + 1.96 * se_estimates
  return(mean(true_value >= lower & true_value <= upper))
}
```

```
# Initialize matrices for results
mse_results <- matrix(NA, 3, length(T_values))
mad_results <- matrix(NA, 3, length(T_values))
coverage_results <- matrix(NA, 3, length(T_values))

# Calculate MSE, MAD, and 95% coverage
for (i in 1:length(T_values)) {
  mse_results[1, i] <- mse(phi_estimates[, i], phi)
  mse_results[2, i] <- mse(theta_estimates[, i], theta)
  mse_results[3, i] <- mse(sigma2_estimates[, i], sigma2)

  mad_results[1, i] <- mad(phi_estimates[, i], phi)
  mad_results[2, i] <- mad(theta_estimates[, i], theta)
  mad_results[3, i] <- mad(sigma2_estimates[, i], sigma2)

  coverage_results[1, i] <- coverage_95_ci(phi_estimates[, i], phi, sd(phi_estimates[, i]))
  coverage_results[2, i] <- coverage_95_ci(theta_estimates[, i], theta, sd(theta_estimates[,
i]))
  coverage_results[3, i] <- coverage_95_ci(sigma2_estimates[, i], sigma2, sd(sigma2_estimates
[, i]))
}

# Combine results into matrices
mse_matrix <- data.frame(T_50 = mse_results[,1], T_200 = mse_results[,2], T_500 = mse_results
[,3])
mad_matrix <- data.frame(T_50 = mad_results[,1], T_200 = mad_results[,2], T_500 = mad_results
[,3])
coverage_matrix <- data.frame(T_50 = coverage_results[,1], T_200 = coverage_results[,2], T_50
0 = coverage_results[,3])

# Display results
list(MSE = mse_matrix, MAD = mad_matrix, Coverage_95_CI = coverage_matrix)
```

```
## $MSE
##         T_50       T_200        T_500
## 1 0.01725602 0.001626157 0.0005509935
## 2 0.02181333 0.004158268 0.0016060925
## 3 0.04333780 0.010151379 0.0037982942
##
## $MAD
##         T_50      T_200      T_500
## 1 0.09482905 0.03026930 0.01807662
## 2 0.11332543 0.05141670 0.03187021
## 3 0.16343256 0.08065485 0.04970399
##
## $Coverage_95_CI
##    T_50 T_200 T_500
## 1 0.874 0.902 0.929
## 2 0.945 0.943 0.953
## 3 0.950 0.949 0.953
```