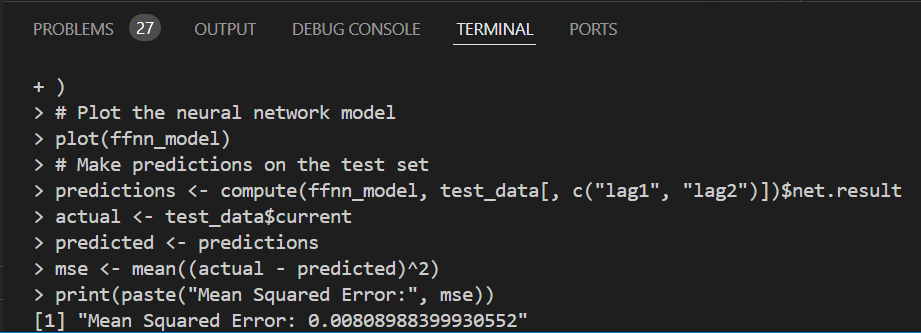
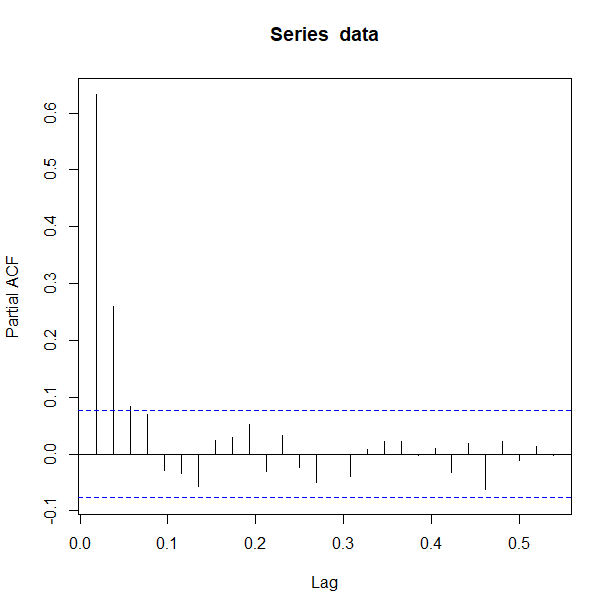
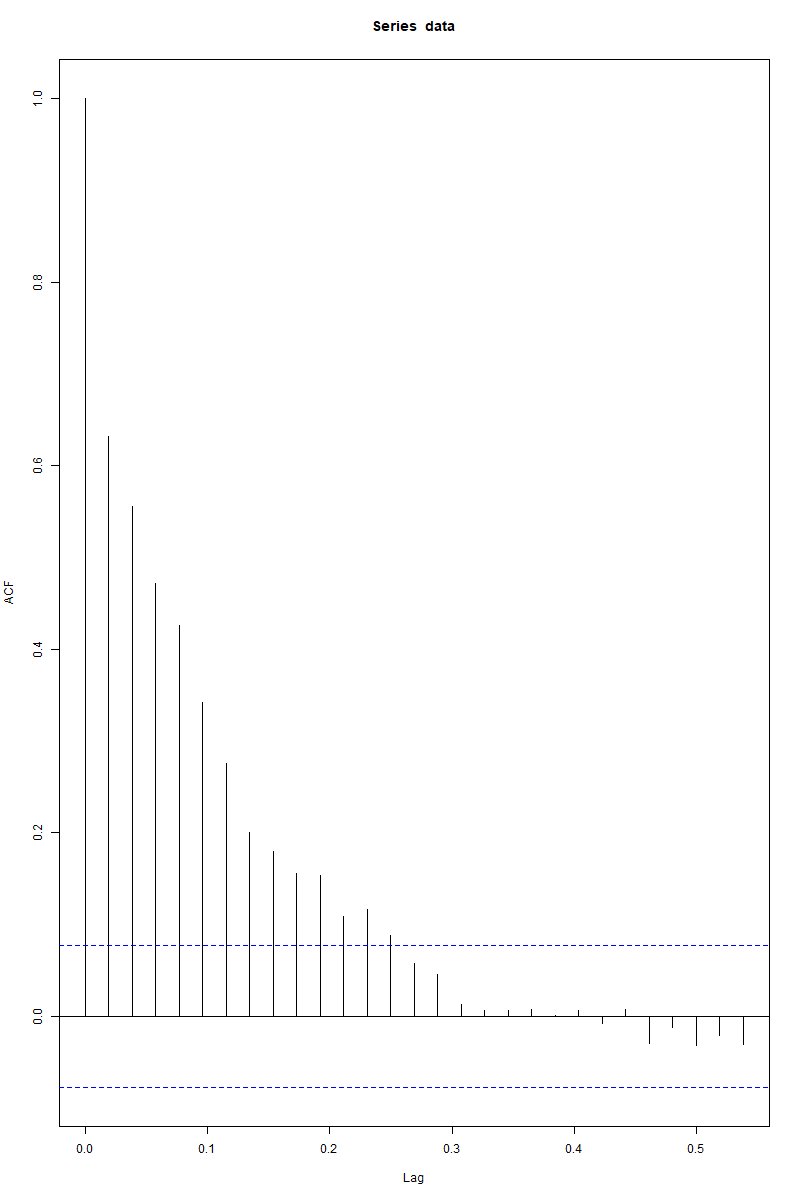
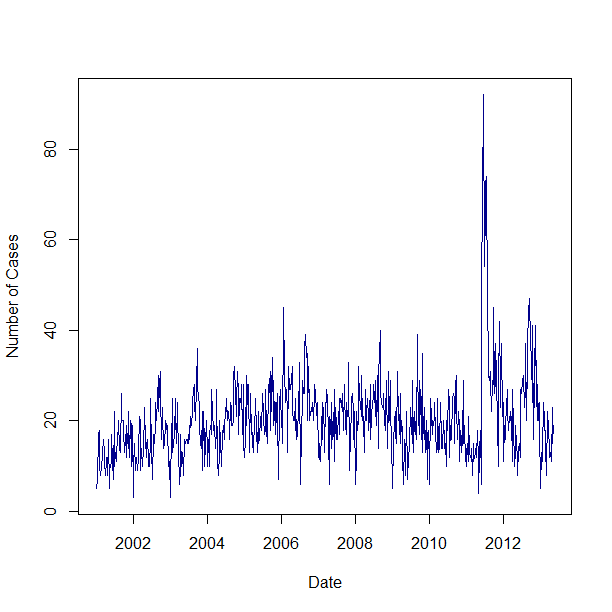
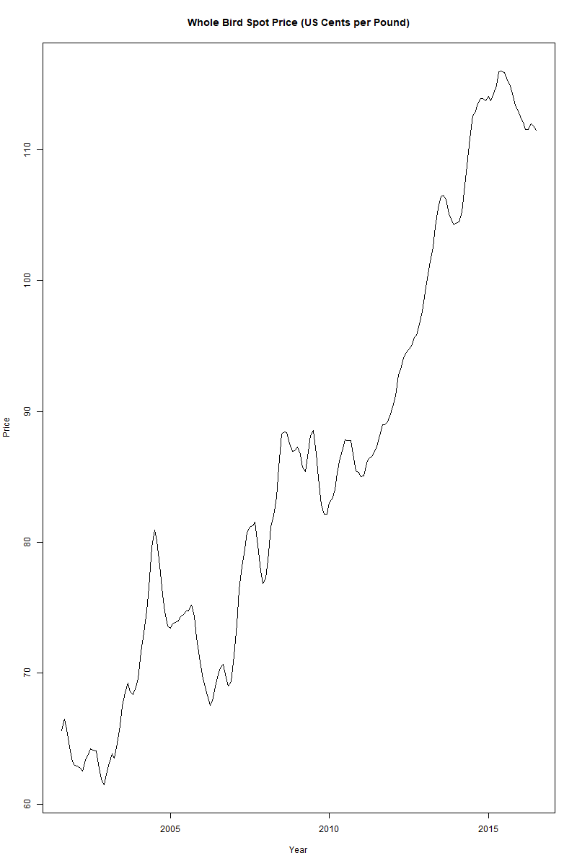
**Neural Network Escherichia Coli**

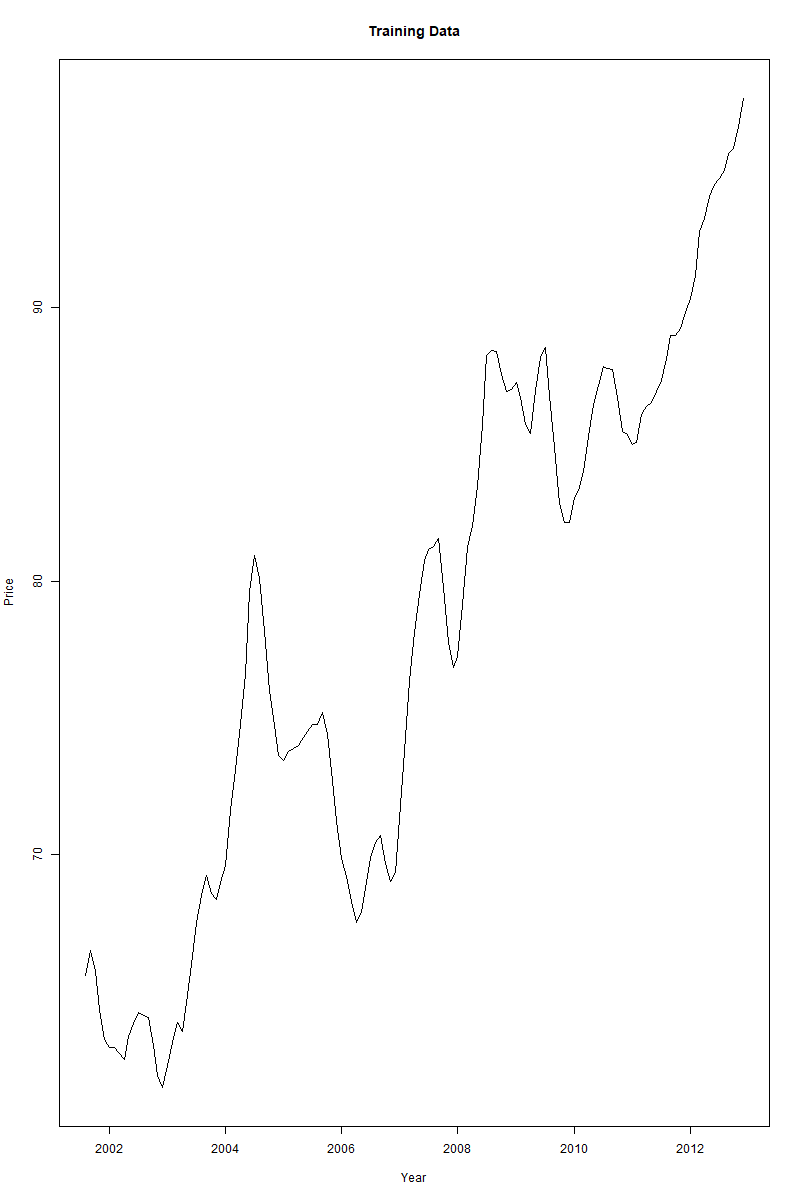
|  |
| --- |
| #install.packages("jsonlite")  #install.packages("tscount")  #install.packages("quantmod")  #install.packages("httpgd")  #install.packages("tscount")  #install.packages("forecast")  #install.packages("dplyr")  #install.packages("neuralnet")  #install.packages("rnn")  # Load necessary libraries  library(tscount)  library(forecast)  library(dplyr)  library(neuralnet)  # Load the E. coli data from tscount package  data("ecoli", package = "tscount")  # Check the structure of the data  head(ecoli)  # Convert the 'cases' column to a time series object  data <- ts(ecoli$cases, start=c(2001,1), frequency=52)  # Plot the time series data  plot(data, xlab="Date", ylab="Number of Cases", col="darkblue")  # ACF and PACF plots  acf(data)  pacf(data)  # Normalization function  normalize <- function(x) {    return ((x - min(x)) / (max(x) - min(x)))  }  # Create lag features and normalize the data  ecoli\_data <- ecoli %>%    mutate(      lag1 = lag(cases, 1),      lag2 = lag(cases, 2),      current = cases  # Keep the original cases as current    ) %>%    na.omit()  # Remove rows with NA values due to lagging  # Normalize the lag features and current values  ecoli\_data <- ecoli\_data %>%    mutate(      lag1 = normalize(lag1),      lag2 = normalize(lag2),      current = normalize(current)    )  # Split the data into training and testing sets  set.seed(123)  train\_indices <- sample(seq\_len(nrow(ecoli\_data)), size = 0.8 \* nrow(ecoli\_data))  train\_data <- ecoli\_data[train\_indices, ]  test\_data <- ecoli\_data[-train\_indices, ]  # Build and train the Feed-Forward Neural Network model  ffnn\_model <- neuralnet(    current ~ lag1 + lag2,  # Predict 'current' using lag features    data = train\_data,    hidden = c(5, 3),  # Two hidden layers with 5 and 3 neurons    linear.output = TRUE  )  # Plot the neural network model  plot(ffnn\_model)  # Make predictions on the test set  predictions <- compute(ffnn\_model, test\_data[, c("lag1", "lag2")])$net.result  # Evaluate model performance  actual <- test\_data$current  predicted <- predictions  # Calculate Mean Squared Error (MSE)  mse <- mean((actual - predicted)^2)  print(paste("Mean Squared Error:", mse)) |



**Recurrent Neural Networks**

|  |
| --- |
| install.packages("keras")  library(keras)  # Install TensorFlow (you only need to run this once)  install\_keras(tensorflow = "2.16.1")  load("C:/Users/amr/Documents/KULIAH/S3/Semester 1/Deret Waktu/R Code/chicken.rda")  #write.csv(chicken, "chicken\_data.csv", row.names = FALSE)  #ls()  # Create a time series object  chicken\_ts <- ts(chicken, start=c(2001, 8), frequency=12)  # Plot the time series  plot(chicken\_ts, main="Whole Bird Spot Price (US Cents per Pound)", ylab="Price", xlab="Year")  # Split the data into training and testing sets  train\_length <- round(length(chicken\_ts) \* 0.8)  # 80% training data  train\_data <- window(chicken\_ts, end=c(2001 + (train\_length-1)/12))  test\_data <- window(chicken\_ts, start=c(2001 + train\_length/12))  plot(train\_data, main="Training Data", ylab="Price", xlab="Year")  # Prepare the data for RNN input (create lagged sequences)  lag\_transform <- function(x, lag=1){    lagged <- c(rep(NA, lag), x[1:(length(x) - lag)])    lagged  }  # Create lagged input features  train\_data\_lagged <- lag\_transform(train\_data, lag=1)  train\_input <- train\_data\_lagged[-1]  # Remove the first NA  train\_output <- train\_data[-1]  # Remove the first value to align  # Reshape the data for RNN input  X\_train <- array(train\_input, dim = c(length(train\_input), 1, 1))  y\_train <- array(train\_output, dim = c(length(train\_output), 1))  # Build the RNN model  model <- keras\_model\_sequential() %>%    layer\_simple\_rnn(units=50, input\_shape=c(1, 1)) %>%    layer\_dense(units=1)  model %>% compile(    loss='mean\_squared\_error',    optimizer='adam'  )  # Train the model  history <- model %>% fit(    X\_train, y\_train,    epochs=100,    batch\_size=1,    verbose=1  )  # Make predictions  X\_test <- array(test\_data[-length(test\_data)], dim=c(length(test\_data)-1, 1, 1))  # Prepare test data for prediction  predictions <- model %>% predict(X\_test)  # Plot actual vs predicted  plot(test\_data, main="Actual vs Predicted", col='blue', type='l')  lines(c(rep(NA, length(train\_data)), predictions), col='red', type='l')  legend("topleft", legend=c("Actual", "Predicted"), col=c("blue", "red"), lty=1) |

****

****