Plot 1:

```{r, echo=FALSE}
library(forecast)
library(tseries)

tsx_data <- read.csv("TSX.csv")


tsx_data$Date <- as.Date(tsx_data$Date, format="%Y-%m-%d")
tsx_series <- ts(tsx_data$Close, frequency=12 )


sarima_model <- auto.arima(tsx_series, seasonal=TRUE, stepwise=FALSE,
approximation=FALSE)

summary(sarima_model)

forecasted_values <- forecast(sarima_model, h=10)  # 'h' is the number of periods you want
to
plot(forecasted_values)
```

Plot 2:

```{r, echo=FALSE}
tsdisplay(residuals(sarima_model))
```

Plot 3:
```{r, echo=FALSE}
if (!require(rjags)) {
  install.packages("rjags")
  library(rjags)
}
if (!require(coda)) {
  install.packages("coda")
  library(coda)
}

tsx_data <- read.csv("TSX.csv")
tsx_data$Date <- as.Date(tsx_data$Date, format="%Y-%m-%d")
tsx_data$TimeIndex <- as.numeric(tsx_data$Date)  # Numeric time index for regression

# Split data into training and testing sets
set.seed(123)  # for reproducibility
train_indices <- sample(1:nrow(tsx_data), 0.8 * nrow(tsx_data))  # 80% for training
```

```
train_data <- tsx_data[train_indices, ]
test_data <- tsx_data[-train_indices, ]

# Prepare data for JAGS
data_jags <- list(
  Close = train_data$Close,
  Open = train_data$Open,
  TimeIndex = train_data$TimeIndex,
  N = nrow(train_data)
)

# JAGS model specification
model_string <- "
model {
  for (i in 1:N) {
    Close[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha + beta1 * Open[i] + beta2 * TimeIndex[i]
  }
  alpha ~ dnorm(0, 0.0001)
  beta1 ~ dnorm(0, 0.0001)
  beta2 ~ dnorm(0, 0.0001)
  tau ~ dgamma(0.01, 0.01)
  sigma <- 1 / sqrt(tau)
}
"

writeLines(model_string, con="tsx_model_time.jags")

inits <- function() {
  list(alpha = rnorm(1, 0, 100), beta1 = rnorm(1, 0, 10), beta2 = rnorm(1, 0, 10), tau =
rgamma(1, 1, 1))
}

params <- c("alpha", "beta1", "beta2", "sigma")

jags_model <- jags.model("tsx_model_time.jags", data = data_jags, inits = inits, n.chains =
3, n.adapt = 500)

# Run the MCMC
mcmc_samples <- coda.samples(jags_model, variable.names = params, n.iter = 5000)

# Convert to matrix for easier handling
sample_matrix <- as.matrix(mcmc_samples)

# Predictive checks on the testing set
test_data$predicted <- apply(sample_matrix, 1, function(params) {
  params["alpha"] + params["beta1"] * test_data$Open + params["beta2"] *
test_data$TimeIndex
```

```
})

# Calculate the mean prediction for each test point
test_data$mean_predicted <- rowMeans(test_data$predicted)

# Plot the actual vs. predicted values
plot(test_data$Date, test_data$Close, type="l", col="blue", xlab="Date", ylab="Close Price",
main="Actual vs Predicted Close Prices")
lines(test_data$Date, test_data$mean_predicted, col="red")
legend("topleft", legend=c("Actual", "Predicted"), col=c("blue", "red"), lty=1)


```