
**Module Code:** AQ050-3-M-TSF

**Module Name:** Time Series Forecasting

# Individual Assignment

## Japan Military Expenditure Analysis & Forecasting

**Student Name:** Muhammad Yousouf Ali Budullah
**TP Number:** TP086704
**Programme:** MSc Data Science & Business Analytics
**Intake Code:** APDMF2501DSBA(BI)(PR)
**Module Lecturer:** Lee Chee Nian

**Date of Submission:** 19 December 2025

# Contents

# 1 Introduction

For this assignment, Japan was selected as the country for this analysis based on the requirement that the chosen nation must fall within the top 30% of global military expenditure in the most recent year available. Using data from the World Banks Military Expenditure (current US$) indicator (World Bank, 2025), countries were filtered using R to ensure that they met both criteria of having at least sixty years of available observations and being within the top 30% of expenditure. Japan satisfied these conditions and was therefore chosen for the forecasting study.

The full R code used for the country filtering and data preparation is provided in Appendix A.

# 2 Time Series Plot and Component Analysis (a)

Figure 1 displays the annual military expenditure of Japan from 1960 to 2023. The series demonstrates a strong and persistent upward trend over time, with a sharp increase from the late 1970s through the early 2000s. In the last few decades, the expenditure has fluctuated around a higher value but remains elevated. Although short-term variations are visible, the most dominant feature of the series is its long-term upward trend. Furthermore no repeating seasonal pattern is present, which is to be expected for annual data.

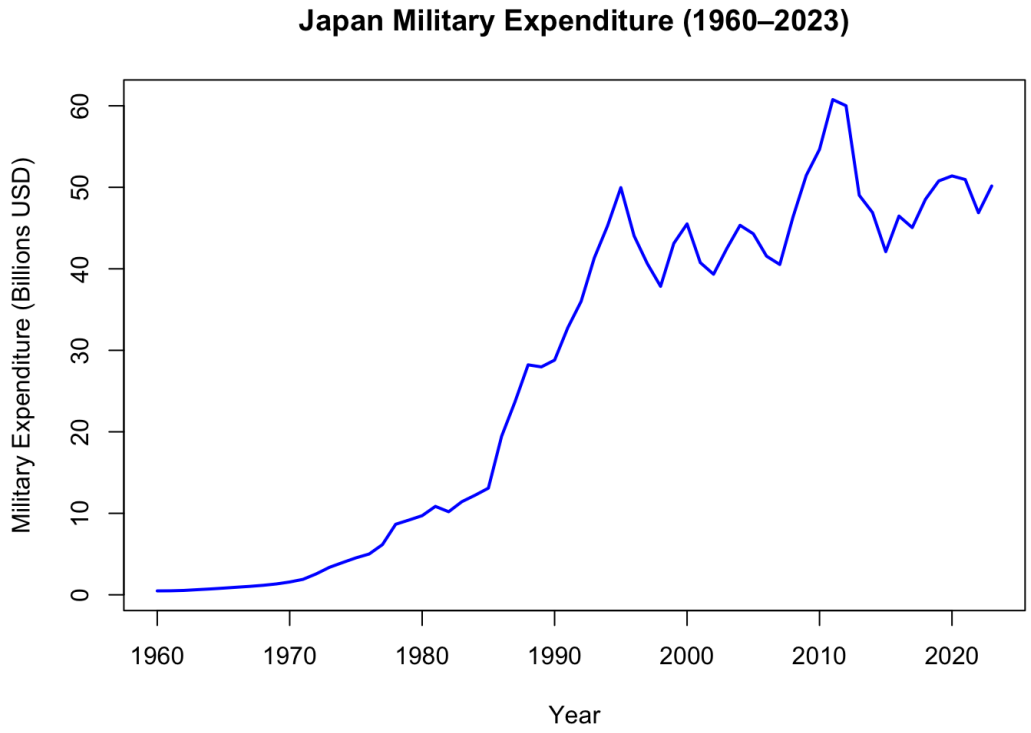


Figure 1: Japan Military Expenditure (1960–2023).

Mann-Kendall test was conducted to confirm the trend observed (see Appendix A for the R code used). The results ($\tau = 0.855$, $p < 2.22 \times 10^{-16}$) provide strong evidence that the upward trend shown in the plot is statistically significant.

Because the data are recorded annually, seasonality cannot occur (frequency = 1). This was further confirmed using the `isSeasonal()` function (see Appendix A for the R code used), which could not detect seasonality due to the insufficient number of observations per cycle. In addition to the trend, the expenditure displays multi-year rises and falls, indicating the presence of cyclical movements, short-term irregular variations can also be seen.

# 3 Selected Forecasting Methods (b)

Three forecasting methods were selected based on the observed behaviour of the military expenditure series, which shows a clear upward trend and no seasonality. The selected methods are Holts Method, a linear trend regression model, and Simple Exponential Smoothing (SES).

## Method 1: Holts Method (Double Exponential Smoothing)

Holts Method extends Simple Exponential Smoothing by explicitly modelling both the level and the trend of a time series. The model is defined as follows.

$$\begin{aligned} L_t &= \alpha A_t + (1-\alpha)(L_{t-1} + T_{t-1}), \\ T_t &= \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}, \\ \hat{y}_{t+k} &= L_t + kT_t, \end{aligned}$$

where $A_t$ is the observed value at time $t$, $L_t$ is the estimated level, $T_t$ is the estimated trend, $\alpha$ and $\beta$ are smoothing parameters ($0 \leq \alpha \leq 1,\ 0 \leq \beta \leq 1$), and $\hat{y}_{t+k}$ is the forecast $k$ periods ahead.

Holts Method is selected because the military expenditure series is a univariate time series with a clear and persistent upward trend. Hyndman and Athanasopoulos (2021) explain that Holts Method is designed for series with trends and provides suitable forecasts by updating both level and slope over time. In addition, evidence from the M4 Forecasting Competition explains that traditional statistical forecasting methods, particularly those from the exponential smoothing family, consistently perform very well across a wide range of real-world datasets and often outperform more complex approaches (Makridakis et al., 2018). Using Holts Method provides a reliable forecast that avoids the risks of overfitting while accurately capturing the underlying trend of the series.

## Method 2: Trend Regression Model

Trend regression models capture the long-term relationship between a series and time. A linear trend model is specified as follows.

$$\hat{y}_t = a + bt$$

where $a$ is the intercept, $b$ represents the average change in expenditure over time, and $\hat{y}_t$ is the fitted value at time $t$.

A linear trend regression model is included to capture the clear trend observed in the military expenditure series. Hyndman and Athanasopoulos (2021) describe trend regression as a simple yet effective method for modeling the relationship between a forecast variable and a single predictor. Research by Green and Armstrong (2015) further highlights that increasing the complexity of a model does not necessarily improve the accuracy of forecasting and can, in some cases, lead to more errors. Trend regression is used to model the clear and persistent upward trend in expenditure without overreacting to temporary fluctuations.

## Method 3: Simple Exponential Smoothing (SES)

Simple Exponential Smoothing produces forecasts by applying exponentially decreasing weights to past observations.

$$F_{t+1} = \alpha A_t + (1 - \alpha) F_t$$

where $F_{t+1}$ is the forecast for the next period, $A_t$ is the observed value at time $t$, and $\alpha$ is the smoothing parameter ($0 \leq \alpha \leq 1$).

Forecasting textbooks describe SES as an appropriate method for time series with no clear trend or seasonality (Hyndman & Athanasopoulos, 2021). Although the military expenditure series displays an upward trend, SES is included as a benchmark forecasting method. Petropoulos et al. (2022) highlights the importance of benchmark models in evaluating forecast performance, with SES considered a standard base method in applied forecasting . Including SES provides a benchmark to assess the performance gains of trend-based models like Holts and trend regression.

# 4 Train/Test Split and Forecasting (c)

The full time series data consists of 64 annual observations (1960–2023). A 70/30 split was applied, producing the following.

- Training set: 1960–2004 (45 observations)
- Testing set: 2005–2023 (19 observations)

Forecasts were derived using the methods justified in the previous section, which include Simple Exponential Smoothing (SES), Holts Method, and the linear trend regression model. Each model was fitted using the training data, and forecasts were produced for the testing period. The R code used to perform the traintest split, model estimation, and forecast generation is provided in Appendix A.

The figures below display the observed values, fitted values for the training period, and forecasts for the testing period.

## Observed, Fitted, and Forecast Values

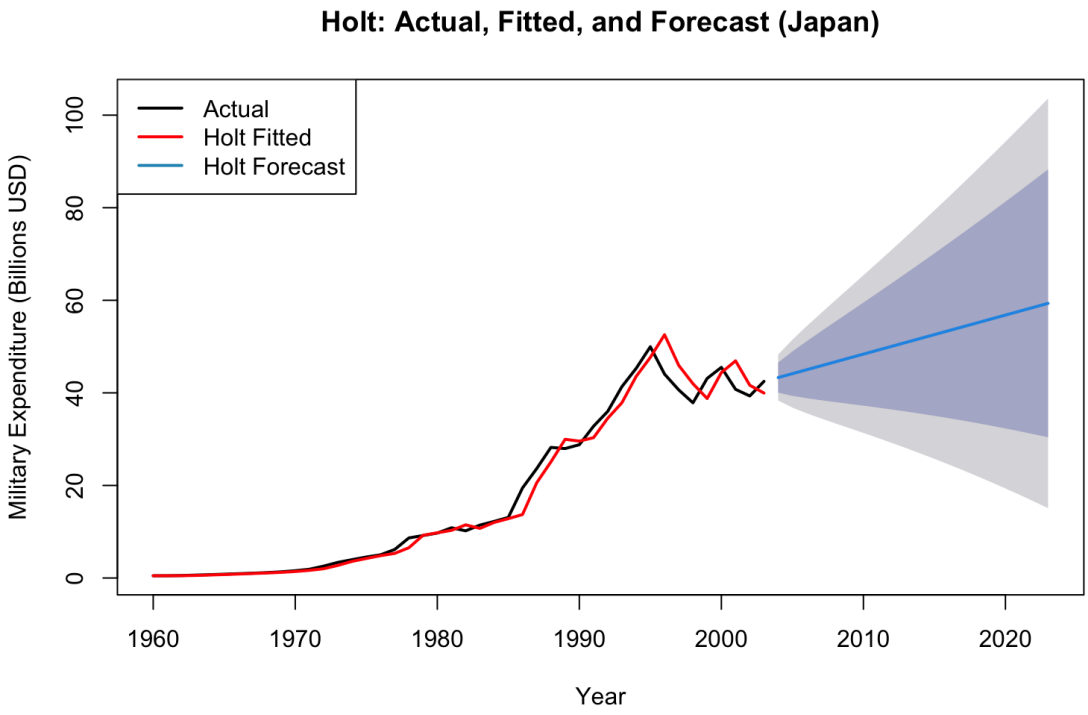


Figure 2: Holts Method: Actual, fitted (training) and forecast (test).

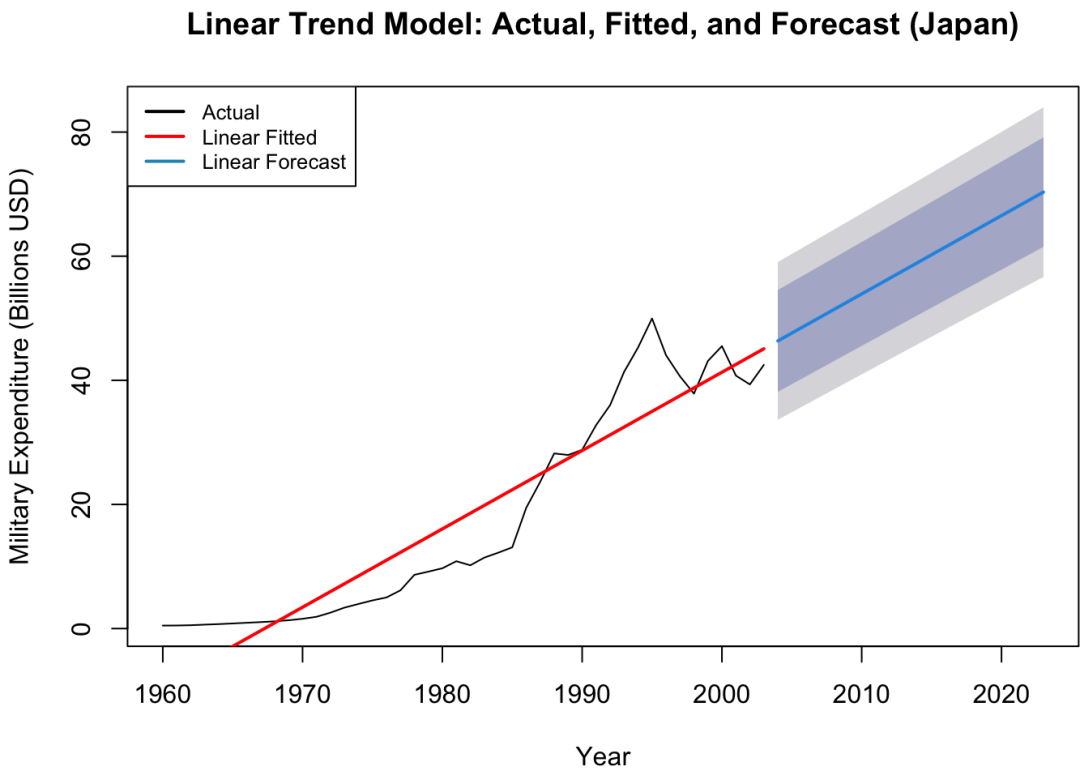


Figure 3: Linear Trend Model: Actual, fitted (training) and forecast (test).

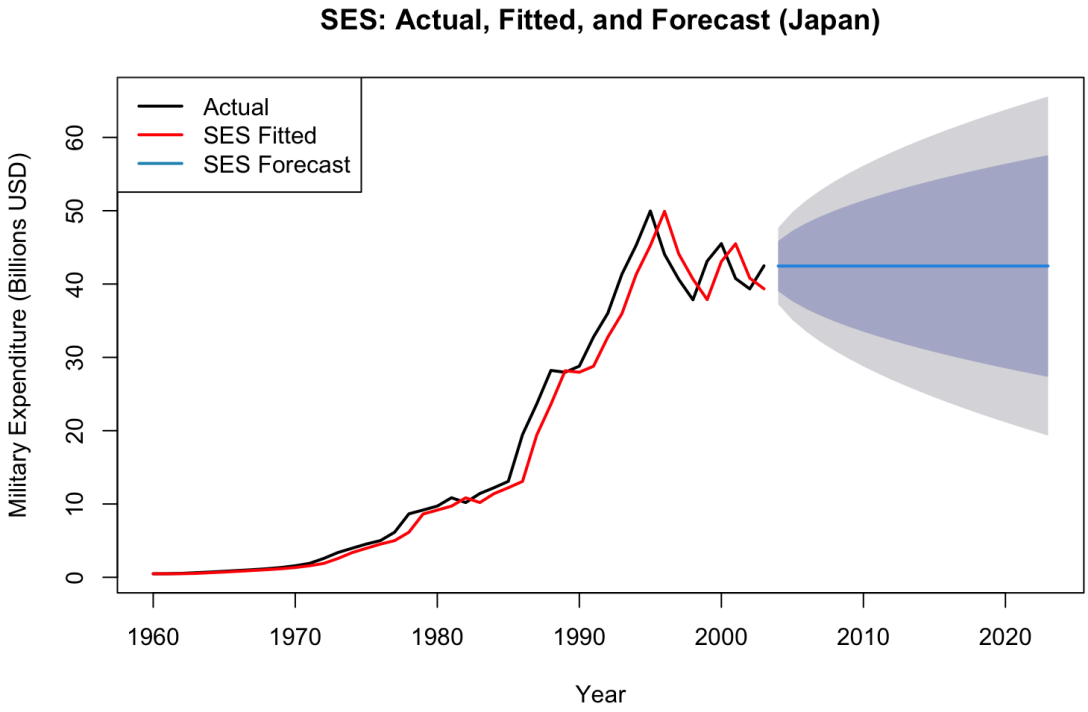


Figure 4: Simple Exponential Smoothing (SES): Actual, fitted (training) and forecast (test).

## Estimated Model Parameters

Table 1, Table 2, and Table 3 display the estimated parameters obtained from the fitted forecasting models using the training data (see Appendix A for the R code & output).

| **Parameter** | **Estimate** |
|---|---|
| Level smoothing ($\alpha$) | 0.9900 |
| Initial level ($l$) | 0.4806 |

Table 1: Estimated parameters for Simple Exponential Smoothing (SES).

| **Parameter** | **Estimate** |
|---|---|
| Level smoothing ($\alpha$) | 0.9900 |
| Trend smoothing ($\beta$) | 0.0973 |
| Initial level ($l$) | 0.4806 |
| Initial trend ($b$) | 0.0118 |

Table 2: Estimated parameters for Holts Method.

| **Coefficient** | **Estimate** | **p-value** |
|---|---|---|
| Intercept | −10.4475 | $< 0.001$ |
| Trend | 1.2621 | $< 0.001$ |

Table 3: Estimated coefficients for the linear trend regression model.

# 5 Forecast Error Evaluation (d)

Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) are chosen because they measure forecast errors in the same units as the original series, making the results easy to understand and compare across models. MAE shows the average size of the forecast errors, while RMSE gives more importance to larger errors. Using both measures provides a clear and balanced view of forecast accuracy for this time series.

Forecast accuracy was evaluated using the testing period from 2005 to 2023. All error values are presented in billions of US dollars, consistent with the rescaled time series used during the forecasting stage. The R code used to compute these forecast errors are provided in Appendix A.

| **Method** | **RMSE (Billion USD)** | **MAE (Billion USD)** |
|---|---|---|
| Simple Exponential Smoothing (SES) | 8.15 | 6.53 |
| Holts Method | 6.95 | 6.05 |
| Linear Trend Regression | 12.61 | 10.66 |

Table 4: Comparison of forecast errors using RMSE and MAE on the test set (2005–2023).

Based on RMSE and MAE, the Holt Method provides the most accurate forecasts, exhibiting the lowest overall error levels. The linear trend regression model performs the worst, suggesting that a simple deterministic trend is ineffective to capture the behaviour of Japans military expenditure. Simple Exponential Smoothing performs better than the linear model, but falls short of Holts Method, indicating the importance of modeling the underlying upward trend. Consequently, Holts Method is identified as the most appropriate forecasting approach for the Japan Military Expenditure series.

# 6 Autocorrelation Analysis and Stationarity Testing (e)

## Autocorrelation Analysis

Figure 5 displays the ACF and PACF of the original Japan military expenditure series. The ACF exhibits very high autocorrelations at low lags that die down slowly while the PACF shows a single significant spike at lag 1, after which it cuts off and remains close to zero. This pattern is consistent with a non-stationary series.

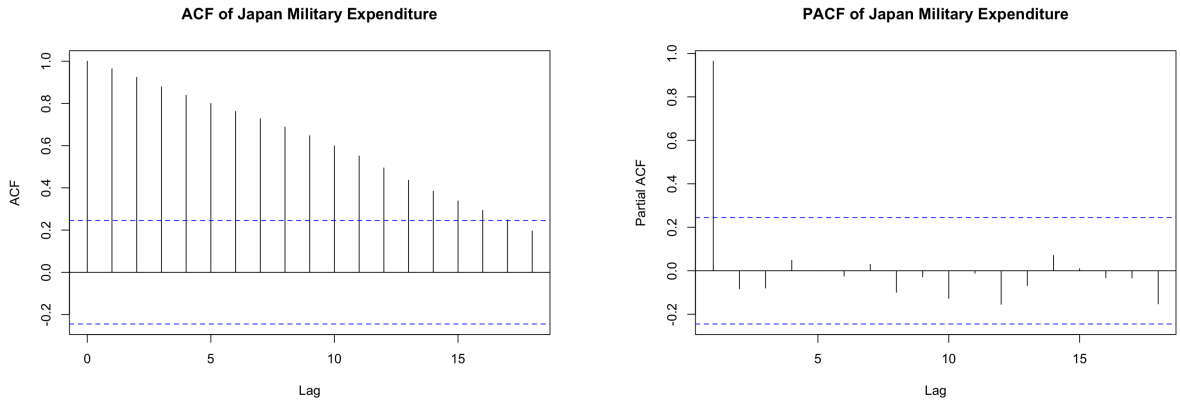


Figure 5: ACF and PACF of Japan Military Expenditure (Original Series).

## Stationarity Testing

Stationarity was determined using the Augmented Dickey–Fuller (ADF) and Kwiatkowski–Phillips–Schmidt–Shin (KPSS) tests at the 5% significance level. The hypotheses the tests for ADF and KPSS are defined as follows.

- **ADF test:**
  - $H_0$: The series contains a unit root (non-stationary).
  - $H_1$: The series does not contain a unit root (stationary).
- **KPSS test:**
  - $H_0$: The series is stationary.
  - $H_1$: The series is non-stationary.

### Original Series

| Test | Statistic | p-value | Decision (5%) |
|---|---|---|---|
| ADF | −1.5326 | 0.7643 | Unit root not rejected |
| KPSS | 1.5868 | 0.01 | Stationarity rejected |

Table 5: Stationarity tests for the original series.

At the 5% significance level, the ADF test does not reject the null hypothesis of a unit root, while the KPSS test rejects the null hypothesis of stationarity. Both tests verify that the series is non-stationary.

### Transformation to Stationarity

To remove non-stationarity, the series was first differenced. The results of the stationarity test for the differenced series are shown in Table 6.

| Test | Statistic | p-value | Decision (5%) |
|---|---|---|---|
| ADF | −4.2326 | 0.01 | Unit root rejected |
| KPSS | 0.0984 | 0.10 | Stationarity not rejected |

Table 6: Stationarity tests for the differenced series.

For the differenced series, the ADF test rejects the null hypothesis of a unit root, and the KPSS test does not reject the null hypothesis of stationarity. Together, these results confirm that the first-differenced series is stationary.

# 7 ARIMA/SARIMA Model Development (f)

## 7.1 Proposed Models (i)

Figure 6 shows the ACF and PACF of the first-differenced Japan military expenditure series.

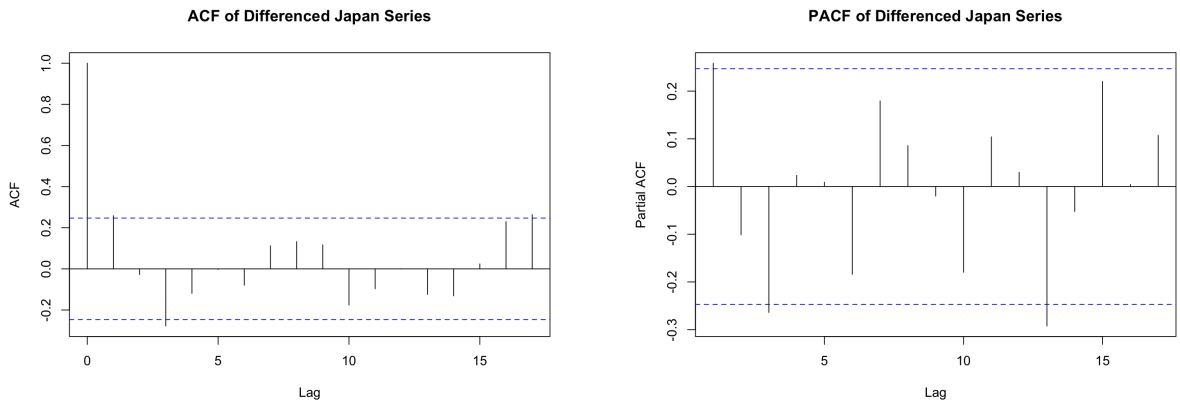


Figure 6: ACF and PACF of the differenced series.

Based on this and the results from `auto.arima`, the following models are suggested.

- **ARIMA(1,1,1):** Based on the ACF and PACF of the differenced series, both show a significant spike at lag 1. The ACF appears to cut off after the first lag, while the PACF does not show a clear cut off or dies down pattern at higher lags. This suggests the presence of both autoregressive and moving average components of low order. As a result, an ARIMA(1,1,1) model is proposed.
- **ARIMA(0,1,1) with drift:** In addition, ARIMA(0,1,1) with drift is also suggested as a model, through automatic model suggestion using `auto.arima()`.

## 7.2 Model Summary Output (ii)

Figure 7 presents the summary outputs for the two suggested ARIMA models fitted to Japans military expenditure series: ARIMA$(1, 1, 1)$ and ARIMA$(0, 1, 1)$ with drift.

```
> summary(mod_arima111)
Series: japan_ts
ARIMA(1,1,1)

Coefficients:
         ar1     ma1
      0.2202  0.0911
s.e.  0.2721  0.2630

sigma^2 = 9.657:  log likelihood = -159.86
AIC=325.71   AICc=326.12   BIC=332.14

Training set error measures:
                    ME     RMSE      MAE      MPE     MAPE      MASE        ACF1
Training set 0.571058 3.033843 2.076269 4.888317 8.601306 0.8906667 -0.02296026
```

ARIMA$(1, 1, 1)$ summary output

```
> summary(mod_arima011)
Series: japan_ts
ARIMA(0,1,1) with drift

Coefficients:
         ma1   drift
      0.2490  0.8004
s.e.  0.1096  0.4714

sigma^2 = 9.326:  log likelihood = -158.74
AIC=323.49   AICc=323.89   BIC=329.92

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set 0.002205951 2.981496 2.058092 -11.14334 17.36564 0.8828691 0.02524989
```

ARIMA$(0, 1, 1)$ with drift summary output

Figure 7: Summary outputs for the fitted ARIMA models.

## 7.3 Model Equations (iii)

**ARIMA(1,1,1):**

$$(1 - \phi_1 B)(1 - B)y_t = (1 + \theta_1 B)\varepsilon_t$$

$$(1 - \phi_1 B)(y_t - y_{t-1}) = \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

$$(y_t - y_{t-1}) - \phi_1(y_{t-1} - y_{t-2}) = \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

$$y_t = (1 + \phi_1)y_{t-1} - \phi_1 y_{t-2} + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

Using the estimated coefficients $\hat{\phi}_1 = 0.2202$ and $\hat{\theta}_1 = 0.0911$, the fitted model is as follows.

$$y_t = 1.2202\, y_{t-1} - 0.2202\, y_{t-2} + \varepsilon_t + 0.0911\, \varepsilon_{t-1}$$

**ARIMA(0,1,1) with Drift:**

$$(1 - B)y_t = (1 + \theta_1 B)\varepsilon_t + c$$

$$y_t - y_{t-1} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + c$$

$$y_t = y_{t-1} + c + \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

Using the estimated parameters $\hat{\theta}_1 = 0.2490$ and $\hat{c} = 0.8004$, the fitted model is as follows.

$$y_t = y_{t-1} + 0.8004 + \varepsilon_t + 0.2490\, \varepsilon_{t-1}$$

## 7.4 Model Adequacy Checks (iv)

Model adequacy was evaluated using residual diagnostics and the Ljung–Box test at the 5% significance level. The corresponding code and outputs are shown in Appendix A.

The hypotheses for Ljung–Box is defined as follows.

- $H_0$: Residuals are not autocorrelated.
- $H_1$: Residuals are autocorrelated.

**ARIMA(1,1,1)**

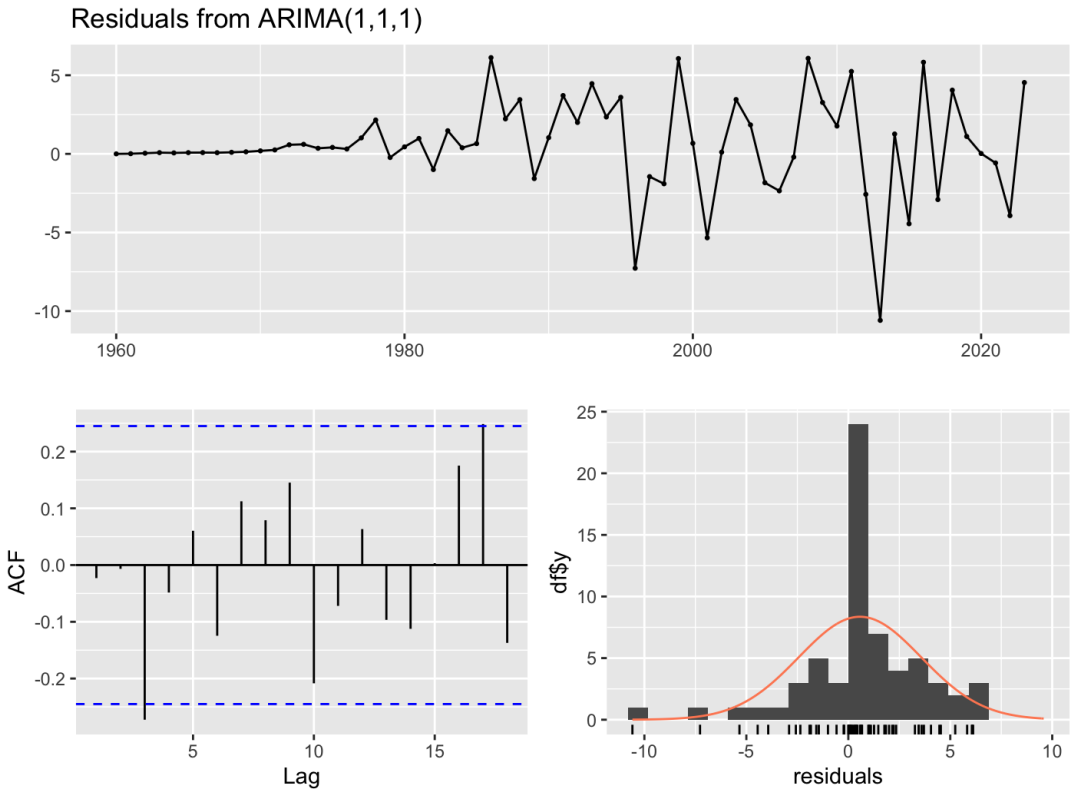


Figure 8: Residual diagnostics for ARIMA(1,1,1).

| Model | Q Statistic | df | p-value | Decision (5%) |
|---|---|---|---|---|
| ARIMA(1,1,1) | 13.168 | 8 | 0.1062 | Adequate |

Table 7: Ljung–Box test results for ARIMA(1,1,1) residuals.

The residuals fluctuate around zero with no clear pattern, and the residual ACF shows no significant autocorrelation. The histogram is approximately normally distributed, indicating near-normal residuals. The Ljung–Box p-value of 0.1062 exceeds 0.05, so the model is adequate.

**ARIMA(0,1,1) with Drift**

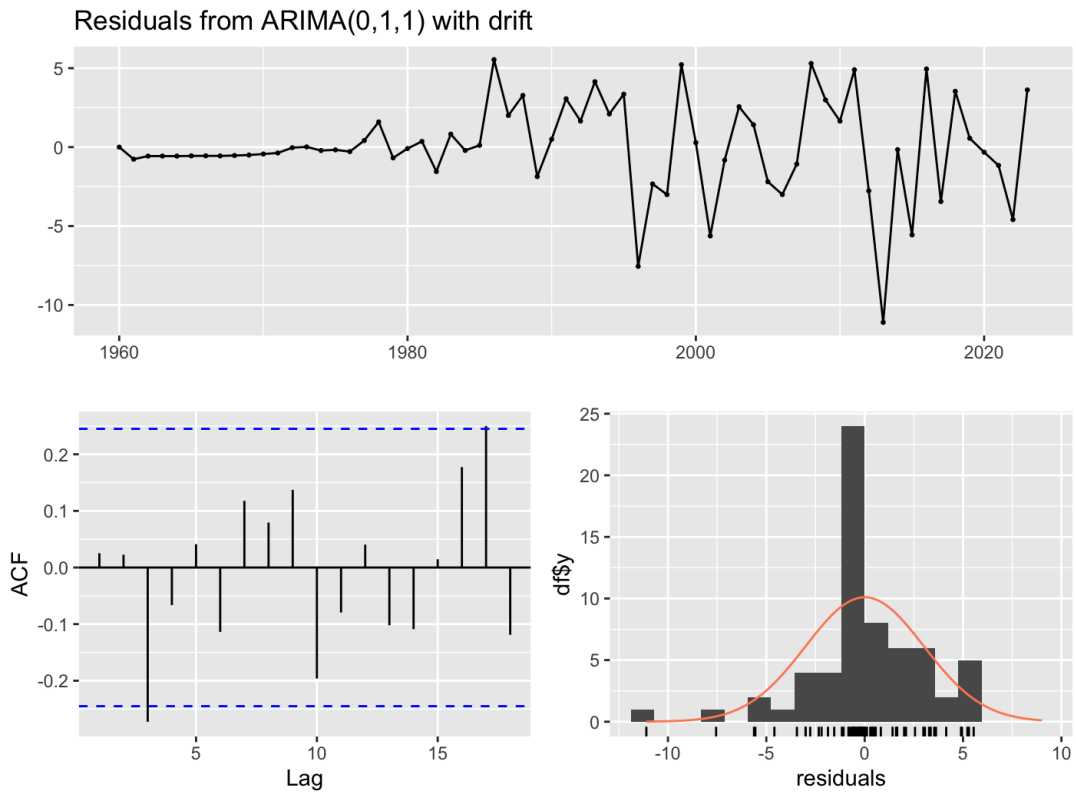


Figure 9: Residual diagnostics for ARIMA(0,1,1) with drift.

| Model | Q Statistic | df | p-value | Decision (5%) |
|---|---|---|---|---|
| ARIMA(0,1,1) with drift | 12.549 | 9 | 0.1841 | Adequate |

Table 8: Ljung–Box test results for ARIMA(0,1,1) with drift residuals.

The residuals show random variation around zero, with no significant autocorrelation in the residual ACF. The histogram is again normally distributed. The Ljung–Box p-value of 0.1841 is greater than 0.05, confirming it is adequate.

Both ARIMA(1,1,1) and ARIMA(0,1,1) with drift pass the residual diagnostics and Ljung–Box test. For both models residuals behave like white noise and satisfy model adequacy conditions.

## 7.5 Parameter Significance (v)

The statistical significance of the ARIMA model parameters was assessed using z-tests at the 5% significance level ($\alpha = 0.05$), with the hypotheses defined as follows.

- $H_0$: Parameter coefficient = 0.
- $H_1$: Parameter coefficient $\neq$ 0.

The corresponding R code and outputs are provided in Appendix A.

**ARIMA(1,1,1)**

For the ARIMA(1,1,1) model, both the AR(1) coefficient ($p = 0.4184$) and the MA(1) coefficient ($p = 0.7289$) are not statistically significant at the 5% level. Therefore, fails to reject the null hypothesis for either parameter, suggesting that both parameters are not significant.

**ARIMA(0,1,1) with Drift**

For the ARIMA(0,1,1) with drift model, the MA(1) coefficient is statistically significant at the 5% level ($p = 0.0231$). Therefore the null hypothesis is rejected for this parameter.

At $\alpha = 0.05$, none of the parameters in the ARIMA(1,1,1) model are statistically significant, whereas the MA(1) term in the ARIMA(0,1,1) with drift model is significant. This provides stronger evidence for using an ARIMA(0,1,1) model with drift.

## 7.6 Forecast Errors (vi)

Table 9 presents the forecast error measures for the ARIMA models using the same criteria as in Part (d), namely the RMSE and MAE. All values are expressed in billions of USD.

| **Model** | **RMSE** | **MAE** |
|---|---|---|
| ARIMA(1,1,1) | 3.03 | 2.08 |
| ARIMA(0,1,1) with Drift | 2.98 | 2.06 |

Table 9: Forecast error measures (RMSE and MAE) for Holts Method and ARIMA models.

Both ARIMA models achieve lower forecast errors however, the ARIMA(0,1,1) model with drift produces the lowest RMSE and MAE overall. Indicating the strongest forecasting performance for Japans military expenditure. It also suggests that including a moving average component and a drift term improves model performance.

# 8 Six-Year Forecast Using Best ARIMA Model (g)

Figure 10 presents the six-year forecast for Japans military expenditure using the selected ARIMA(0,1,1) model with drift. Table 10 displays the corresponding point forecasts and 80% and 95% prediction intervals for the forecast horizon. The corresponding code and outputs are shown in Appendix A.

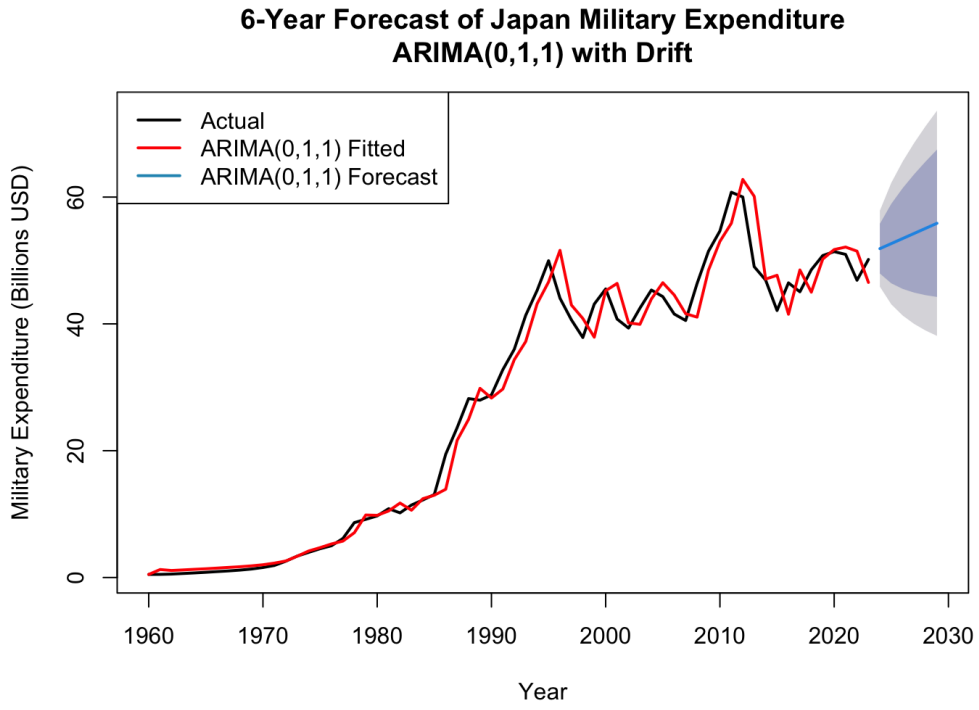


Figure 10: Six-Year Forecast of Japan's Military Expenditure Using the ARIMA(0,1,1) with drift

| Year | Forecast | Lo 80 | Hi 80 | Lo 95 | Hi 95 |
|---|---|---|---|---|---|
| 2024 | 51.86 | 47.95 | 55.78 | 45.88 | 57.85 |
| 2025 | 52.66 | 46.40 | 58.93 | 43.09 | 62.24 |
| 2026 | 53.46 | 45.52 | 61.41 | 41.32 | 65.61 |
| 2027 | 54.26 | 44.94 | 63.59 | 39.99 | 68.53 |
| 2028 | 55.06 | 44.53 | 65.60 | 38.96 | 71.17 |
| 2029 | 55.87 | 44.26 | 67.48 | 38.11 | 73.62 |

Table 10: Six-year ahead forecasts and prediction intervals (Billions USD) generated using ARIMA(0,1,1) with drift.

# A Appendix: R Code and Additional Outputs

The following appendix contains the R code and printed outputs used in the analysis. Only supporting material not shown in the main report is included here.

## Code: Country Filtering and Data Preparation

```r
library(dplyr)

# ------------------------------------------------------------
# Identify countries that meet requirements (Top 30% expenditure,
# with at least 60 years of valid observations)
# ------------------------------------------------------------

final_list <- WB_WDI_MS_MIL_XPND_CD %>%
  rename(
    country = REF_AREA_LABEL,
    year    = TIME_PERIOD,
    mil_exp = OBS_VALUE
  ) %>%
  mutate(year = as.numeric(year)) %>%
  group_by(country) %>%
  filter(sum(!is.na(mil_exp)) >= 60) %>%
  ungroup() %>%
  filter(year == max(year, na.rm = TRUE)) %>%
  filter(mil_exp >= quantile(mil_exp, 0.70, na.rm = TRUE)) %>%
  pull(country)

final_list
```

## Output: Filtered Countries

```
[1] "Japan"
```

*Note.* The filtering procedure returned multiple regions and countries. Only Japan is shown here, as it is the selected country for analysis.

## Code: Extracting and Cleaning Japan's Data

```r
# Keep relevant variables and rename them
df_clean <- WB_WDI_MS_MIL_XPND_CD |>
  dplyr::select(
    country = REF_AREA_LABEL,
    year    = TIME_PERIOD,
    mil_exp = OBS_VALUE
  ) |>
  dplyr::mutate(year = as.numeric(year))

# Extract Japan's data and sort by year
japan_military_spending <- df_clean |>
  dplyr::filter(country == "Japan") |>
  dplyr::arrange(year)

str(japan_military_spending)
```

## Output: Structure

```
'data.frame':   64 obs. of 3 variables:
 $ country: chr  "Japan" "Japan" "Japan" "Japan" ...
 $ year   : num  1960 1961 1962 1963 1964 ...
 $ mil_exp: num  480555556 492361111 538194444 629861111 718055556
    ...
```

## Code: Time Series Construction and Trend Testing

```
# ----------------------------
# PART A: Time Series Analysis
# ----------------------------
library(dplyr)
library(tseries)
library(forecast)
library(Kendall)    # Mann Kendall test

# Convert the dataset into time series (frequency = 1, annual)
japan_ts <- ts(japan_military_spending$mil_exp,
               start = min(japan_military_spending$year),
               frequency = 1)

# Rescale for plotting
japan_ts_bil <- japan_ts / 1e9

# Time Series Plot
plot(japan_ts_bil,
     main = "Japan Military Expenditure (1960-2023)",
     xlab = "Year",
     ylab = "Military Expenditure (Billions USD)",
     col = "blue",
     lwd = 2)

# Mann Kendall Trend Test
mk_test <- MannKendall(japan_ts)
mk_test

# Seasonality Test
isSeasonal(japan_ts)
```

## Output: Mann–Kendall Test

```
tau = 0.855, 2-sided pvalue = < 0.000000000000000222
```

## Output: Seasonality Test

```
Error in combined_test(x, freq = freq) :
  The number of observations per cycle (usually years) is 1 and thus
      too small.
```

## Code: Forecasting Methods and Error Measures

```r
# ----------------------------
# PART C & D: Forecasting Methods & Errors
# All values expressed in BILLIONS of USD
# ----------------------------

library(forecast)
library(dplyr)

# ---------------------------------------
# Prepare time series + train/test split
# ---------------------------------------

# Convert to billions for readability
japan_ts <- ts(japan_military_spending$mil_exp / 1e9,
               start = min(japan_military_spending$year),
               frequency = 1)

n <- length(japan_ts)
train_size <- floor(0.7 * n)

train_end_year  <- time(japan_ts)[train_size]
test_start_year <- time(japan_ts)[train_size + 1]

train_japan <- window(japan_ts, end   = train_end_year)
test_japan  <- window(japan_ts, start = test_start_year)

h <- length(test_japan)

# ---------------------------------------
# Method 1 - Simple Exponential Smoothing (SES)
# ---------------------------------------

ses_model <- ses(train_japan, alpha = 0.99, initial = "simple", h =
   h)

plot(ses_model,
     main = "SES: Actual, Fitted, and Forecast (Japan)",
     ylab = "Military Expenditure (Billions USD)",
     xlab = "Year",
     col = "black",
     lwd = 2)

lines(ses_model$fitted, col = "red", lwd = 2)
```

```r

legend("topleft",
       legend = c("Actual", "SES Fitted", "SES Forecast"),
       col = c("black", "red", "#2596be"),
       lty = 1,
       lwd = 2)

# ----------------------------------------
# Method 2 - Holt's Method
# ----------------------------------------

holt_model <- holt(train_japan, alpha = 0.99, initial = "simple", h
   = h)

plot(holt_model,
     main = "Holt: Actual, Fitted, and Forecast (Japan)",
     ylab = "Military Expenditure (Billions USD)",
     xlab = "Year",
     col = "black",
     lwd = 2)

lines(holt_model$fitted, col = "red", lwd = 2)

legend("topleft",
       legend = c("Actual", "Holt Fitted", "Holt Forecast"),
       col = c("black", "red", "#2596be"),
       lty = 1,
       lwd = 2)

# ----------------------------------------
# Method 3 - Linear Trend Regression
# ----------------------------------------

linear_model <- tslm(train_japan ~ trend)

test_trend <- data.frame(
  trend = seq(length(train_japan) + 1,
              length(train_japan) + h)
)

linear_forecast <- forecast(linear_model,
                            newdata = test_trend,
                            h = h)

plot(linear_forecast,
```

```
     main = "Linear Trend Model: Actual, Fitted, and Forecast (Japan
        )",
     ylab = "Military Expenditure (Billions USD)",
     xlab = "Year")

lines(fitted(linear_model), col = "red", lwd = 2)

legend("topleft",
       legend = c("Actual", "Linear Fitted", "Linear Forecast"),
       col = c("black", "red", "#2596be"),
       lty = 1,
       lwd = 2,
       cex = 0.8)

# ----------------------------------------
# 5. Display Parameters
# ----------------------------------------
ses_model$model$par
holt_model$model$par
summary(linear_model)

# ----------------------------------------
# Forecast Errors (Billions USD)
# ----------------------------------------

ses_acc    <- accuracy(ses_model, test_japan)
holt_acc   <- accuracy(holt_model, test_japan)
linear_acc <- accuracy(linear_forecast$mean, test_japan)

ses_acc
holt_acc
linear_acc
```

## Output: Parameters

```
> ses_model$model$par
    alpha         l
0.9900000 0.4805556
> holt_model$model$par
     alpha       beta          l          b
0.99000000 0.09732261 0.48055556 0.01180556
> summary(linear_model)

Call:
```

```
tslm(formula = train_japan ~ trend)

Residuals:
    Min      1Q  Median      3Q     Max
-9.2789 -4.6099 -0.8813  3.8252 14.9742

Coefficients:
             Estimate Std. Error t value             Pr(>|t|)
(Intercept) -10.44747    1.84600   -5.66           0.00000123 ***
trend         1.26208    0.07145   17.66 < 0.0000000000000002 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .
    0.1         1

Residual standard error: 6.018 on 42 degrees of freedom
Multiple R-squared:  0.8814,  Adjusted R-squared:  0.8785
F-statistic:   312 on 1 and 42 DF,  p-value: < 0.00000000000000022
```

## Output: Forecast Error

```
> ses_acc
                     ME     RMSE       MAE        MPE     MAPE
                        MASE       ACF1 Theil's U
Training set 0.9635961 2.662398 1.836646  9.237544 11.58833
   0.9823245 0.3164454         NA
Test set     6.2077674 8.149869 6.525333 11.776811 12.55164
   3.4900536 0.6903032  1.903656
> holt_acc
                      ME     RMSE       MAE        MPE      MAPE
                         MASE       ACF1 Theil's U
Training set  0.1962618 2.529633 1.585638  4.906766  9.009683
   0.8480735 0.2746422         NA
Test set     -2.6585572 6.947505 6.050857 -6.371179 12.330608
   3.2362816 0.7560415  1.733011
> linear_acc
                 ME     RMSE       MAE       MPE     MAPE       ACF1
                    Theil's U
Test set -9.673382 12.60658 10.66208 -20.8112 22.45925 0.7951786
   3.26078
```

## Code: Autocorrelation and Stationarity Analysis

```r
# ----------------------------
# PART E: Autocorrelation & Stationarity Analysis
# ----------------------------

library(forecast)
library(tseries)

# ---------------------------------------------------
# Time series object (anual data)
# ---------------------------------------------------

japan_ts <- ts(japan_military_spending$mil_exp,
               start = min(japan_military_spending$year),
               frequency = 1)

# ---------------------------------------------------
# Autocorrelation analysis (original series)
# ---------------------------------------------------

acf(japan_ts,
    main = "ACF of Japan Military Expenditure")

pacf(japan_ts,
     main = "PACF of Japan Military Expenditure")

# ---------------------------------------------------
# Stationarity tests (original series)
# ---------------------------------------------------

# Augmented Dickey Fuller Test
adf_test <- adf.test(japan_ts)
adf_test

# KPSS Test
kpss_test <- kpss.test(japan_ts)
kpss_test

# Suggested differencing order
ndiffs(japan_ts)

# ---------------------------------------------------
# First differencing
# ---------------------------------------------------
```

```r

japan_diff <- diff(japan_ts)
#Convert to billions
plot(japan_diff/ 1e9,
     main = "Differenced Japan Military Expenditure",
     ylab = "Differenced Values (Billions USD)",
     xlab = "Year",
     col = "blue")

# ------------------------------------------------------
# Stationarity tests (on differenced series)
# ------------------------------------------------------

adf_diff <- adf.test(japan_diff)
adf_diff

kpss_diff <- kpss.test(japan_diff)
kpss_diff

# ------------------------------------------------------
# Autocorrelation analysis (on differenced series)
# ------------------------------------------------------

acf(japan_diff,
    main = "ACF of Differenced Series")

pacf(japan_diff,
     main = "PACF of Differenced Series")
```

## Output: Stationarity Tests

```
> adf_test

  Augmented Dickey-Fuller Test

data:  japan_ts
Dickey-Fuller = -1.5326, Lag order = 3, p-value = 0.7643
alternative hypothesis: stationary

> kpss_test

  KPSS Test for Level Stationarity

data:  japan_ts
```

```
KPSS Level = 1.5868, Truncation lag parameter = 3, p-value = 0.01

> ndiffs(japan_ts)
[1] 1

> adf_diff

  Augmented Dickey-Fuller Test

data:  japan_diff
Dickey-Fuller = -4.2326, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary

> kpss_diff

  KPSS Test for Level Stationarity

data:  japan_diff
KPSS Level = 0.098352, Truncation lag parameter = 3, p-value = 0.1
```

## Code: ARIMA Modelling

```
# ----------------------------
# PART F: ARIMA Modeling
# ----------------------------
library(forecast)
library(tseries)
library(lmtest)

# ------------------------------------------------------------
# Define time series (scaled to billions USD)
# ------------------------------------------------------------

japan_ts <- ts(japan_military_spending$mil_exp/1e9,
               start = min(japan_military_spending$year),
               frequency = 1)

# ------------------------------------------------------------
# (f)(i)  Select ARIMA models
# ------------------------------------------------------------
auto.arima(japan_ts, trace = TRUE)

# ------------------------------------------------------------
# (f)(ii) Fit ARIMA models
# ------------------------------------------------------------
# Values  in billions as japan_ts was scaled

# ARIMA(1,1,1)
mod_arima111 <- Arima(japan_ts, order = c(1,1,1))

# ARIMA(0,1,1) with drift
mod_arima011 <- Arima(japan_ts, order = c(0,1,1),include.drift =
   TRUE)

summary(mod_arima111)
summary(mod_arima011)

# ------------------------------------------------------------
# (f)(iv) Model adequacy checks
# ------------------------------------------------------------
checkresiduals(mod_arima111)
checkresiduals(mod_arima011)

# ------------------------------------------------------------
# (f)(v) Coefficient significance tests
```

```
# ------------------------------------------------------------
coeftest(mod_arima111)
coeftest(mod_arima011)

# ------------------------------------------------------------
# (f)(vi) Forecast error measures
# ------------------------------------------------------------
accuracy(mod_arima111)
accuracy(mod_arima011)
```

## Output: Auto Arima

```
> auto.arima(japan_ts, trace = TRUE)

 ARIMA(2,1,2) with drift         : Inf
 ARIMA(0,1,0) with drift         : 326.0715
 ARIMA(1,1,0) with drift         : 323.9396
 ARIMA(0,1,1) with drift         : 323.8947
 ARIMA(0,1,0)                    : 327.8553
 ARIMA(1,1,1) with drift         : 325.9804
 ARIMA(0,1,2) with drift         : 325.1067
 ARIMA(1,1,2) with drift         : 325.7833
 ARIMA(0,1,1)                    : 324.4632

 Best model: ARIMA(0,1,1) with drift

Series: japan_ts
ARIMA(0,1,1) with drift

Coefficients:
          ma1   drift
      0.2490  0.8004
s.e.  0.1096  0.4714

sigma^2 = 9.326:  log likelihood = -158.74
AIC=323.49   AICc=323.89   BIC=329.92
```

## Output: Model Summaries

```
> summary(mod_arima111)
Series: japan_ts
ARIMA(1,1,1)

Coefficients:
```

```
         ar1     ma1
      0.2202  0.0911
s.e.  0.2721  0.2630

sigma^2 = 9.657:  log likelihood = -159.86
AIC=325.71   AICc=326.12   BIC=332.14

Training set error measures:
                    ME     RMSE      MAE      MPE     MAPE      MASE
                          ACF1
Training set 0.571058 3.033843 2.076269 4.888317 8.601306 0.8906667
   -0.02296026
> summary(mod_arima011)
Series: japan_ts
ARIMA(0,1,1) with drift

Coefficients:
         ma1   drift
      0.2490  0.8004
s.e.  0.1096  0.4714

sigma^2 = 9.326:  log likelihood = -158.74
AIC=323.49   AICc=323.89   BIC=329.92

Training set error measures:
                      ME     RMSE      MAE       MPE     MAPE
                         MASE       ACF1
Training set 0.002205951 2.981496 2.058092 -11.14334 17.36564
   0.8828691 0.02524989
```

## Output: Residual Diagnostics (Ljung–Box Test)

```
> checkresiduals(mod_arima111)

  Ljung-Box test

data:  Residuals from ARIMA(1,1,1)
Q* = 13.168, df = 8, p-value = 0.1062

Model df: 2.   Total lags used: 10

> checkresiduals(mod_arima011)

  Ljung-Box test
```

```
data:  Residuals from ARIMA(0,1,1) with drift
Q* = 12.549, df = 9, p-value = 0.1841

Model df: 1.   Total lags used: 10
```

## Output: Coefficient Significance Tests

```
> coeftest(mod_arima111)

z test of coefficients:

    Estimate Std. Error z value Pr(>|z|)
ar1 0.220158   0.272093  0.8091   0.4184
ma1 0.091142   0.262952  0.3466   0.7289

> coeftest(mod_arima011)

z test of coefficients:

      Estimate Std. Error z value Pr(>|z|)
ma1    0.24895    0.10958  2.2720  0.02309 *
drift  0.80039    0.47142  1.6978  0.08954 .
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .
    0.1         1
```

## Output: Forecast Accuracy (Training Set)

```
> accuracy(mod_arima111)
                    ME     RMSE      MAE       MPE     MAPE      MASE
                              ACF1
Training set 0.571058 3.033843 2.076269 4.888317 8.601306 0.8906667
   -0.02296026
> accuracy(mod_arima011)
                      ME     RMSE      MAE        MPE     MAPE
                         MASE        ACF1
Training set 0.002205951 2.981496 2.058092 -11.14334 17.36564
   0.8828691 0.02524989
```

## Six-Year Forecast Using ARIMA(0,1,1) with Drift

```
# ----------------------------
# PART G: Forecast 6 years
# ----------------------------

# Forecast the next 6 years
forecast_6yr <- forecast(mod_arima011, h = 6)

# Print forecast values
forecast_6yr

# ------------------------------------------------------------
# Plot forecast and fitted values
# ------------------------------------------------------------

plot(forecast_6yr,
     main = "6-Year Forecast of Japan Military Expenditure\nARIMA
        (0,1,1) with Drift",
     ylab = "Military Expenditure (Billions USD)",
     xlab = "Year",
     col = "black",
     lwd = 2)

# Add fitted values
lines(fitted(mod_arima011), col = "red", lwd = 2)

# Add legend
legend("topleft",
       legend = c("Actual", "ARIMA(0,1,1) Fitted", "ARIMA(0,1,1)
          Forecast"),
       col = c("black", "red", "#2596be"),
       lty = 1,
       lwd = 2)
```

## Output: Forecast of next 6 years

```
> forecast_6yr
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2024       51.86341 47.94964 55.77718 45.87781 57.84900
2025       52.66379 46.40191 58.92567 43.08707 62.24052
2026       53.46418 45.52033 61.40803 41.31511 65.61325
2027       54.26456 44.93727 63.59185 39.99970 68.52942
2028       55.06495 44.53442 65.59547 38.95990 71.16999
2029       55.86533 44.25562 67.47505 38.10980 73.62086
```

# References


Green, K. C., & Armstrong, J. S. (2015). Simple versus complex forecasting: The evidence. *Journal of Business Research*, *68*(8), 1678–1685. https://doi.org/10.1016/j.jbusres.2015.03.026

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.) [Accessed 2025]. OTexts. https://otexts.com/fpp3/

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, *34*(4), 802–808. https://doi.org/10.1016/j.ijforecast.2018.06.001

Petropoulos, F., Makridakis, S., Assimakopoulos, V., & Nikolopoulos, K. (2022). Forecasting: Theory and practice. *International Journal of Forecasting*, *38*(3), 705–871. https://doi.org/10.1016/j.ijforecast.2021.11.001

World Bank. (2025). Military expenditure (current us$) [Retrieved from https://data360.worldbank.org/indicator/WB_WDI_MS_MIL_XPND_CD].