Time series Assignment

Plazzogna Marco, Yan Wenbin

Professor: Ngatchou Wandji Joseph

Contents

May 21, 2024

1 Data

All the code in this section can be found in Appendix A.1 and A.2.

The seasonally and working-day adjusted (SA-WDA) industrial production index for the manufacture of weapons and ammunition (base 100 in 2021) is a crucial measure within the NAF rev. 2 classification, level Class, item 25.40. This index tracks monthly changes in production, serving as an essential indicator for monitoring the industry's business cycle and identifying early turning points. Specifically, it provides insights into the defense sector's production activities and economic impact. Published monthly, it ensures stakeholders have up-to-date and relevant information.

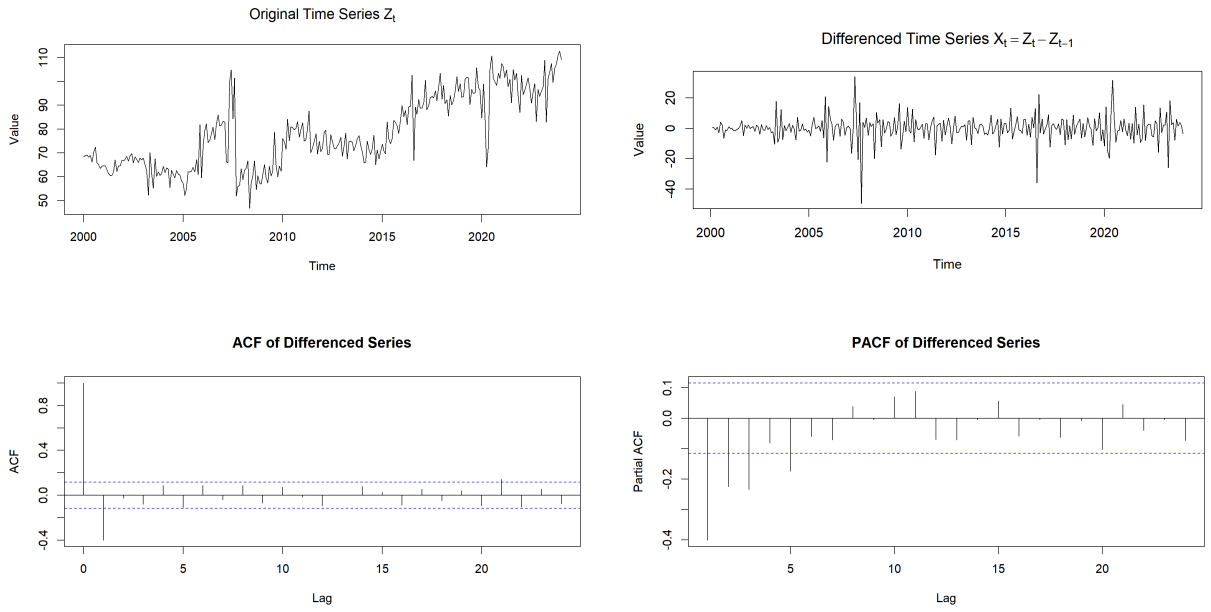

Figure 1: Data Representation

The time series plot (jan/2000-jan/2024) shows an overall increasing, although not linear trend, with values rising from around 50 in 2000 to over 110 by 2023. There are noticeable short-term fluctuations and variability, indicating irregular movements possibly due to seasonal effects or cyclical factors. High volatility is observed particularly between 2008-2010 and from 2020 onwards, likely due to macroeconomics effects, suggesting periods of increased activity. Since 2020, a sharper increase and higher volatility suggest a recent change in the underlying process affecting the variable. We implemented ADF and PP tests to check whether it has a trend. ADF prefers not to reject it while PP prefers to reject it. Since we also agree that there is a trend, we finally decided to difference it.

The differenced time series $X_t = \Delta Z_t := Z_t - Z_{t-1}$ of the original data Z_t appears to be stationary, with a relatively constant mean (most likely equal to 0) and variance over time, suggesting that differencing has successfully removed any trends and made the series more stable. Fluctuations around the zero value are more uniform compared to the original series, further supporting the notion of stationarity. Although there is some volatility, especially notable around 2008 and 2020, it is less pronounced compared to the original series, indicating that differencing has reduced some irregular movements.

The ACF plot of the differenced series X_t shows no significant peaks outside the 95% confidence intervals, indicating the absence of seasonality. If there were seasonality, we would expect to see significant autocorrelations at specific lags (e.g., every 12 months for yearly seasonality). The autocorrelations rapidly decay to near zero after the first lag, which is typical for a stationary series, supporting the conclusion that the differenced series does not have significant autocorrelation at higher lags. The PACF plot shows no significant peaks outside the 95% confidence intervals, suggesting no underlying autoregressive process influencing the series at specific lags, further supporting the absence of seasonality and indicating that the series does not have strong dependencies at particular lags. Minor significant lags within the first five periods are within the confidence intervals, implying weak autocorrelation that does not indicate a clear pattern or seasonal structure. For these reasons, we would prefer to use an ARMA model with parameters $p \leq 5$ and $q \leq 1$.

2 ARMA Models

All the code in this section can be found in Appendix A.1 and A.3.

In this part, in order to select and evaluate ARMA (AutoRegressive Moving Average) models for a given time series, we begin by defining two functions, `Ljung.Box2` and `Ljung.Box`, which perform the Ljung-Box test to check for residual autocorrelation in time series data. The `Ljung.Box2` function calculates the Ljung-Box test statistic, degrees of freedom, and p-value for a given number of lags, while the `Ljung.Box` function extends this test across multiple lag lengths, optionally plotting the p-values to visualize the results.

Next, the `modelchoice` function evaluates ARMA(p, q) models by fitting them to the data and checking the significance of the AR (AutoRegressive) and MA (Moving Average) coefficients. It also ensures the absence of residual autocorrelation using the Ljung-Box test. This function returns a summary of the model parameters, including whether the AR and MA coefficients are significant and whether the residuals show no autocorrelation.

The `armamodelchoice` function generates all possible combinations of AR and MA orders up to specified maximum values (pmax and qmax). It evaluates each combination using the `modelchoice` function, and returns a data frame of the results, including the significance of coefficients and residual diagnostics.

The main script then filters for well-adjusted and valid models from the results, identifying those that meet the criteria of having significant coefficients and no residual autocorrelation. For each selected ARMA model, it fits the model to the data using the `Arima` function and calculates their AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) values. These criteria are used to determine the best-fitting models, as they balance model fit and complexity, with lower values indicating better models.

The script creates a comparison table of AIC and BIC values for each model, allowing for easy identification of the best models based on these criteria. It then identifies the models with the lowest AIC and BIC values as the best models. Summaries of the best models according to AIC and BIC are displayed, providing detailed information about the model parameters and fit.

Residual diagnostics are performed on the best AIC and BIC model to ensure its adequacy (in fact in our case they prefer the same model). This includes checking the residuals for any remaining autocorrelation using the Ljung-Box test. The test results are plotted to visualize the p-values across different lags, with a horizontal line indicating the 0.05 significance level. If the residuals show no significant autocorrelation, it confirms the model's adequacy.

Finally, the best ARMA model for the processed data based on the AIC criterion is selected and returned, which in our case also minimizes the BIC criterion, resulting in an ARIMA(0,0,1) model.

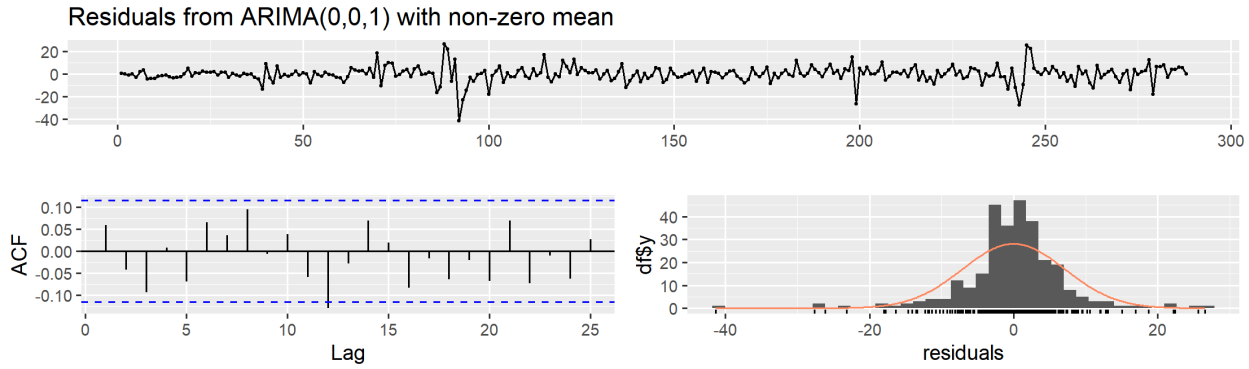

Figure 2: Residuals Analysis for ARIMA(0,0,1) Model

Figure 2 shows the residuals from the ARIMA(0,0,1) model. The top plot displays the residual time series, indicating that the residuals are roughly centered around zero with no apparent trends or patterns. This suggests that the model has adequately captured the underlying process. The bottom-left plot shows the ACF (AutoCorrelation Function) of the residuals. Most of the autocorrelations fall within the 95% confidence intervals (dashed blue lines), indicating that there is no significant autocorrelation left in the residuals, which is a desirable property for a well-fitted model. The bottom-right plot is a histogram of the residuals with a superimposed normal density curve. The residuals appear to follow a roughly normal distribution, with a mean close to zero, suggesting that the assumption of normally distributed errors is reasonable for this model.

However, it is important to note that the results might be influenced by external factors, particularly around the years 2008 and 2020. The global financial crisis in 2008 and the COVID-19 pandemic in 2020 could introduce irregularities and unexpected variations in the data, which the model may not fully account for, potentially affecting its performance.

Table 1 presents the results of the Ljung-Box test for the residuals from the ARIMA(0,0,1) model across lags 1 to 24. The p-values for all lags are above the 0.05 significance level, indicating that we fail to reject the null hypothesis of no autocorrelation in the residuals. This further supports the adequacy of the model.

Lag	Statistic	DF	p-value	Lag	Statistic	DF	p-value	Lag	Statistic	DF	p-value
1	0.00	0	NaN	9	9.90	8	0.2718746	17	20.54	16	0.1968713
2	1.55	1	0.2135640	10	10.36	9	0.3218545	18	21.79	17	0.1930638
3	4.11	2	0.1282974	11	11.41	10	0.3265800	19	21.92	18	0.2355422
4	4.13	3	0.2480566	12	16.49	11	0.1239577	20	23.35	19	0.2223964
5	5.54	4	0.2366010	13	16.73	12	0.1600526	21	24.87	20	0.2063134
6	6.84	5	0.2331901	14	18.20	13	0.1500613	22	26.53	21	0.1868712
7	7.21	6	0.3016077	15	18.31	14	0.1929653	23	26.56	22	0.2283307
8	9.89	7	0.1948043	16	20.45	15	0.1553145	24	27.81	23	0.2229840

Table 1: Ljung-Box Test of Residuals from ARIMA(0,0,1) for Lag Range 1-24

Also note that, since we differenced the original data, this effectively means we implemented the ARIMA(0,1,1) model to the original data in our analysis:

$$X_t = \Delta Z_t = \theta_0 + \epsilon_t + \theta_1 \epsilon_{t-1}, \tag{1}$$

where in our case, $\theta_0 = 0.1422$ and $\theta_1 = -0.6543$.

In summary, the ARIMA(0,0,1) model is selected as the best-fitting model for the corrected series $X_t = \Delta Z_t$ and the ARIMA(0,1,1) model as the best-fitting model for the chosen series Z_t, based on both AIC and BIC criteria. The model chosen effectively captures the underlying structure of the time series data, with well-behaved residuals indicating that the model assumptions are met. The Ljung-Box test further validates the model by showing no significant autocorrelation in the residuals across various lags.

One final note regarding the behaviour of the series: we previously observed two pivotal points likely corresponding to the 2008 financial crisis and the COVID-19 pandemic. During these periods, there were significant changes in the trend of the series. Consequently, it may be beneficial to segment the data into three distinct periods and estimate separate models for each: from 2000 to 2008/2009, from 2008/2009 to 2020, and from 2020 to the end in 2024.

3 Predictions

All the code in this section can be found in Appendix A.1 and A.4.

Denote T as the length of the series. Assume the series' residuals are Gaussian, that is, $\varepsilon_t \sim \mathcal{N}\left(0, \sigma^2\right)$ with $\sigma^2 > 0$. Given $\{X_t, \epsilon_t : 1 \leq t \leq T\}$, our best expectation of X_{T+1} and X_{T+2} would be:

$$\begin{cases} X_{T+1|T} = \theta_0 + \theta_1 \epsilon_T, \\ X_{T+2|T} = \theta_0. \end{cases}$$

Hence, the error would be respectively:

$$\begin{cases} X_{T+1} - X_{T+1|T} = \epsilon_{T+1}, \\ X_{T+2} - X_{T+2|T} = \epsilon_{T+2} + \theta_1 \epsilon_{T+1}. \end{cases}$$

Since the series' residuals are Gaussian, it is then easy to see that:

$$\begin{cases} X_{T+1} - X_{T+1|T} \sim \mathcal{N}\left(0, \sigma^2\right), \\ X_{T+2} - X_{T+2|T} \sim \mathcal{N}\left(0, (1+\theta_1^2)\sigma^2\right). \end{cases}$$

Hence, the α-confidence interval for them would be:

$$\begin{cases} \tilde{X}_{T+1} \in \left[\theta_0 + \theta_1\epsilon_T - \sigma q_{1-\frac{\alpha}{2}}, \theta_0 + \theta_1\epsilon_T + \sigma q_{1-\frac{\alpha}{2}}\right], \\ \tilde{X}_{T+2} \in \left[\theta_0 - \sqrt{1+\theta_1^2}\sigma q_{1-\frac{\alpha}{2}}, \theta_0 + \sqrt{1+\theta_1^2}\sigma q_{1-\frac{\alpha}{2}}\right]. \end{cases}$$

We should also notice the key hypotheses used to derive this confidence region:

1. The differentiated time series follows an ARMA(0, 0, 1) model.
2. The residuals of the series follow a Gaussian distribution centered at zero with a known variance.

By implementing the results above, we can obtain the future projection with 95% confidence bounds as below,

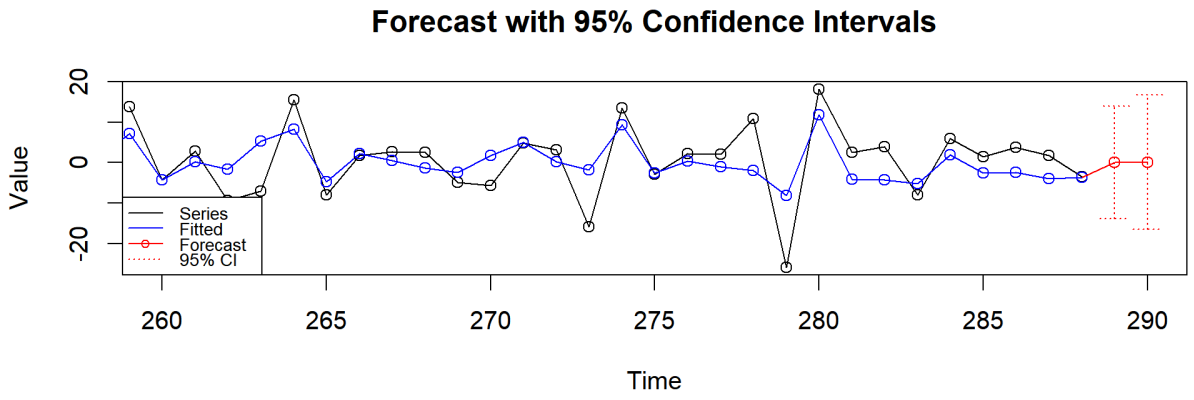

Figure 3: Future Projection with 95% Confidence Bounds Based on Historical Data

Figure 3 shows the last 30 observations of the differenced series along with the corresponding fitted values from the model. This focus on the final portion of the series is due to the need for clear visualization of the two new predicted values and their respective 95 % confidence intervals, which was difficult to achieve when displaying the entire series.

The two prediction are consistent with the type of model that we found for the differenced series, that is as MA(1). In fact, as the first prediction T+1 is still influenced by the observation at time T, then from that point on the predictions are equal to the mean estimated by the model that is 0.1422.

The two predictions align with the MA(1) model identified for the differenced series. Specifically, the first prediction at $T+1$ is influenced by the observation at time T. From that point onward, the predictions converge to the model's estimated mean of 0.1422, which is consistent with the theoretical behavior of MA(1) models.

Open Question:

To determine if the additional information Y_t allows for an improved prediction of X_{T+1}, we need to evaluate the correlation and causality between the two time series. Firstly, there should be a significant correlation between Y_t and X_t, indicating that changes in Y_t are associated with changes in X_t. More importantly, we must establish that Y_t Granger-causes X_t and, as a consequence, verify that there is no spurious correlation between them. Granger causality implies that past values of Y_t contain information that helps predict future values of X_t beyond the information already contained in past values of X_t. To test for Granger causality, we can perform a Granger causality test, which involves estimating a regression model where X_t is regressed on its own past values and past values of Y_t. If the coefficients on the lagged values of Y_t are statistically significant, we conclude that Y_t Granger-causes X_t. This causality test helps identify whether incorporating Y_t as an exogenous variable in the forecasting model for X_t can improve the model's predictive performance.

Finally, having the information provided by Y_{T+1} sooner, can enhance the prediction of X_{T+1}. An illustrative example could be related to news from war zones impacting the French industry's production. For instance, an announcement by President Emmanuel Macron regarding increased defense spending or new defense contracts could provide early

indications of how the production index will evolve. By incorporating such timely information from Y_t, which might capture these announcements and their immediate effects, we can improve the accuracy of our predictions for the industrial production index X_{T+1}.

Appendix

A.1 Custom Functions

Load necessary libraries

```r
library(tseries)   # For time series analysis
library(zoo)       # For working with indexed totally ordered observations
library(forecast)  # For forecasting functions
require(fUnitRoots)  # For unit root tests
require(urca)        # For unit root and cointegration tests
```

pierce.test function

```r
# This function implements the Pierce (or portmanteau) test to check for residual autocorrelation.
# It accepts four arguments:
# - x: The time series data or residuals (as a zoo object) to analyze.
# - m: The number of lags to include in the test.
# - s: The step size (typically 1).
# - para: The number of model parameters to exclude from the degrees of freedom.
#
# The function:
# 1. Computes the autocorrelation function (ACF) up to m*s lags.
# 2. Calculates the test statistic as per the Pierce test formula.
# 3. Returns a list containing:
#    - `statistic`: The calculated test statistic.
#    - `p.value`: The p-value indicating the significance of the test statistic.
pierce.test <- function(x = NULL, m = NULL, s = NULL, para = NULL) {
  if (is.null(x) || is.null(m) || is.null(s) || is.null(para)) {
    stop("All parameters (x, m, s, para) must be provided")
  }
  x <- coredata(x)  # Extract numeric data from zoo object
  kk <- acf(x, m * s, plot = FALSE)  # Calculate autocorrelation function
  n <- length(x)  # Number of observations
  statistics <- 0
  for (i in 1:m) {
    statistics <- statistics + (1 / (n - (i * s))) * kk$acf[(i * s) + 1]^2
  }
  statistics <- statistics * n * (n + 2)
  p.value <- 1 - pchisq(statistics, m - para)
  return(list(statistic = statistics, p.value = p.value))  # Return values as a list
}
```

Ljung.Box2 function

```r
# This function performs the Ljung-Box test to check residual autocorrelation.
# It accepts three parameters:
# - x: The time series data or residuals (as a zoo object) to analyze.
# - lag: The number of lags to include in the test.
# - param: The number of model parameters to adjust for.
#
# The function:
# 1. Computes the ACF up to the specified `lag` (ignoring the first autocorrelation).
# 2. Calculates the test statistic using the Ljung-Box formula.
# 3. Returns a list containing:
#    - `statistic`: The test statistic.
#    - `gdl`: The degrees of freedom (lag minus the number of parameters).
#    - `pvalue`: The p-value indicating the significance of the test.
Ljung.Box2 <- function(x, lag = 15, param = 1) {
  if (NCOL(x) > 1) stop("x is not a vector or univariate time series")
  x <- coredata(x)  # Extract numeric data from zoo object
  cor <- acf(x, lag.max = lag, plot = FALSE)
  n <- length(x)
  PARAMETER <- lag - param
  obs <- cor$acf[2:(lag + 1)]
  STATISTIC <- n * (n + 2) * sum((1 / seq(n - 1, n - lag)) * obs^2)
  PVAL <- 1 - pchisq(STATISTIC, lag - param)
  return(list(statistic = STATISTIC, gdl = PARAMETER, pvalue = PVAL))
}
```

Ljung.Box function

```r
# The Ljung.Box function provides an extended Ljung-Box test across multiple lag lengths.
# It accepts four parameters:
# - x: The time series data or residuals (as a zoo object) to analyze.
# - maxlag: The maximum number of lags to include in the analysis.
# - par: The number of model parameters to adjust for.
# - all: If TRUE, computes the test across all lags up to `maxlag` and plots the p-values.
#
# The function:
# 1. Calls `Ljung.Box2` for each lag up to `maxlag`, excluding the specified `par` parameters.
# 2. Constructs a matrix containing the test statistic, degrees of freedom, and p-value at each lag.
# 3. Optionally plots the p-values as a bar chart if `all` is TRUE.
# 4. Returns a matrix containing the statistics for each lag.
Ljung.Box <- function(x, maxlag = 24, par = 0, all = TRUE) {
  if (all == TRUE) {
    LB <- matrix(0, nrow = maxlag, ncol = 3)
    dimnames(LB) <- list(NULL, c("statistic", "gdl", "pvalue"))
    for (i in (par + 1):maxlag) {
      lbi <- Ljung.Box2(x, lag = i, param = par)
      LB[i, ] <- c(round(lbi$statistic, 2), lbi$gdl, lbi$pvalue)
    }
    plot(LB[, 3], type = "h", lwd = 4, ylim = c(0, 1), ylab = "P-Value", xlab = "Lag")
    abline(h = 0.05, lty = 2, col = "red")
    abline(h = 0)
  }
  if (all == FALSE) LB <- Ljung.Box2(x, lag = maxlag, param = par)

  return(LB)
}
```

stat.mod function

```r
# The stat.mod function provides comprehensive model diagnostics for a fitted time series model.
# It takes three arguments:
# - fit.y: The fitted model object (e.g., ARIMA or other time series model).
# - maxlag: The maximum number of lags for the Ljung-Box test to check residual autocorrelation.
# - ordini: The number of model parameters.
#
# The function calculates and prints:
# 1. Coefficients, standard errors, t-statistics, and p-values of the model parameters.
# 2. Model diagnostics, such as the residual variance (sigma2), Akaike Information Criterion (AIC), and
    log-likelihood.
# 3. Results from the Ljung-Box test to check for autocorrelation in the residuals up to `maxlag` lags.
#
# The results are formatted and displayed for clear interpretation of the model's fit and adequacy.
stat.mod=function(fit.y,maxlag=15,ordini=0)
{
  Coef=fit.y$coef
  Std.Err=sqrt(diag(fit.y$var.coef))
  tstat=fit.y$coef/sqrt(diag(fit.y$var.coef))
  pval=2*pnorm(abs(tstat),lower.tail=F)
  tabella=data.frame(Coef,Std.Err,tstat,pval)
  sigma2=fit.y$sigma2
  names(sigma2)="sigma2"
  AIC=fit.y$aic
  names(AIC)="AIC"
  loglik=fit.y$loglik
  names(loglik)="loglik"
  cc1=Ljung.Box(fit.y$resid,maxlag,ordini)
  riga0="------------ Parameters estimation -----------"
  names(riga0)=" "
  riga="-----------------------------------------------"
  names(riga)=" "
  print(riga0, quote=F)
  print(tabella)
  print(riga, quote=F)
  riga2="------------ Ljung-Box Test --------------"
  names(riga2)=" "
  print(c(sigma2,AIC,loglik))
  print(riga2, quote=F)
  print(t(cc1))
}
```

Qtests function

```r
# This function performs a series of Ljung-Box tests up to lag `k` for the given series.
# It accepts three arguments:
# - series: The time series data to analyze.
# - k: The maximum lag to test.
# - fitdf: The number of model parameters to adjust for.
#
# The function:
# 1. Applies the Ljung-Box test at each lag up to `k`.
# 2. Returns a matrix containing the lag and the corresponding p-value.
Qtests <- function(series, k, fitdf=0) {
  pvals <- apply(matrix(1:k), 1, FUN=function(l) {
    pval <- if (l<=fitdf) NA else Box.test(series, lag=l, type="Ljung-Box", fitdf=fitdf)$p.value
    return(c("lag"=l,"pval"=pval))
  })
  return(t(pvals))
}
```

adfTest_valid function

```r
# This function performs the Augmented Dickey-Fuller (ADF) test for stationarity, increasing lags until no autocorrelation in residuals.
# It accepts three arguments:
# - series: The time series data to analyze.
# - kmax: The maximum number of lags to test.
# - type: The type of test ("c" for constant, "ct" for constant and trend, etc.).
#
# The function:
# 1. Performs the ADF test with increasing lags until residuals show no autocorrelation.
# 2. Prints whether the residuals are autocorrelated or not at each step.
# 3. Returns the final ADF test result.
adfTest_valid <-
  function(series,kmax,type){ #ADF tests until no more autocorrelated residuals
    k <- 0
    noautocorr <- 0
    while (noautocorr==0){
      cat(paste0("ADF with ",k, " lags: residuals OK? "))
      adf <- adfTest(series,lags=k,type=type)
      pvals <- Qtests(adf@test$lm$residuals,24,fitdf=length(adf@test$lm$coefficients))[,2]
      if (sum(pvals<0.05,na.rm=T) == 0) {
        noautocorr <- 1; cat("OK \n")}
      else cat("NO \n")
      k <- k + 1
    }
    return(adf)
  }pvals))
}
```

modelchoice function

```r
# This function fits an ARIMA model and checks the significance of AR and MA terms and the residuals' autocorrelation.
# It accepts four arguments:
# - p: The AR order.
# - q: The MA order.
# - data: The time series data.
# - k: The maximum number of lags for the Ljung-Box test.
#
# The function:
# 1. Tries to fit an ARIMA model with the given orders.
# 2. Checks the significance of AR and MA terms and residuals' autocorrelation.
# 3. Returns a list with the model's coefficients, standard errors, and Ljung-Box test results.
modelchoice <- function(p, q, data, k = 24) {
  estim <- try(arima(data, order = c(p, 0, q), optim.control = list(maxit = 20000)), silent = TRUE)

  if (class(estim) == "try-error") {
    return(c("p" = p, "q" = q, "arsignif" = NA, "masignif" = NA, "resnocorr" = NA, "ok" = NA))
  }
```

modelchoice function

```r
  coefs <- coef(estim)
  se <- sqrt(diag(vcov(estim)))

  arsignif <- if (p == 0) NA else {
    ar_coefs <- coef(estim)[1:p]
    ar_se <- sqrt(diag(vcov(estim)))[1:p]
    sum(2 * (1 - pnorm(abs(ar_coefs / ar_se))) <= 0.05) == p
  }

  masignif <- if (q == 0) NA else {
    ma_coefs <- coef(estim)[(p + 1):(p + q)]
    ma_se <- sqrt(diag(vcov(estim)))[(p + 1):(p + q)]
    sum(2 * (1 - pnorm(abs(ma_coefs / ma_se))) <= 0.05) == q
  }

  resnocorr <- sum(Box.test(estim$residuals, lag = k, type = "Ljung-Box")$p.value <= 0.05, na.rm = TRUE
      ) == 0
  checks <- c(arsignif, masignif, resnocorr)
  ok <- as.numeric(sum(checks, na.rm = TRUE) == (3 - sum(is.na(checks))))

  ljung_box <- Ljung.Box(estim$residuals, maxlag = k)

  # Print coefficients and their standard errors
  cat(paste0("ARMA(", p, ",", q, ") coefficients:\n"))
  print(coefs)
  cat("Standard errors:\n")
  print(se)

  # Print Ljung-Box test result
  cat("Ljung-Box test result:\n")
  print(ljung_box)

  return(list(p = p, q = q, arsignif = arsignif, masignif = masignif, resnocorr = resnocorr, ok = ok,
              coefficients = coefs, se = se, ljung_box = ljung_box))
}
```

armamodelchoice function

```r
# This function finds the best ARMA model within given p and q ranges by checking each model's validity
    .
# It accepts three arguments:
# - pmax: The maximum AR order to test.
# - qmax: The maximum MA order to test.
# - y_df: The differenced time series data.
#
# The function:
# 1. Tests all combinations of AR and MA orders up to pmax and qmax.
# 2. Stores results for each model's coefficients, standard errors, and diagnostic checks.
# 3. Returns a data frame with the comparison of all tested models.
armamodelchoice <- function(pmax, qmax, y_df) {
  pqs <- expand.grid(p = 0:pmax, q = 0:qmax)
  results <- list()

  for (i in 1:nrow(pqs)) {
    p <- pqs$p[i]
    q <- pqs$q[i]
    cat(paste0("Computing ARMA(", p, ",", q, ") \n"))
    result <- modelchoice(p, q, coredata(y_df))
    results[[i]] <- result
  }

  comparison <- do.call(rbind, results)
  comparison <- as.data.frame(comparison, stringsAsFactors = FALSE)
  comparison[, c("p", "q", "arsignif", "masignif", "resnocorr", "ok")] <-
    lapply(comparison[, c("p", "q", "arsignif", "masignif", "resnocorr", "ok")], as.numeric)

  return(comparison)
}
```

A.2 Codes for Part I

Codes for Part I

```r
## Set working directory to the location of the data file
# setwd("C:/Users/Marco Plazzogna/Desktop/MAGISTRALE/PRIMO ANNO/SECONDO SEMESTRE/TIME SERIES/Assignment")

# Read data from a CSV file, skipping the first 4 lines and assuming ";" as the separator
data <- read.csv("valeurs_mensuelles.csv", sep = ";", skip = 4, header = FALSE)

# Extract the second column (assuming it contains the time series data)
serie <- data[, 2]

# Convert the series into a zoo object starting from January 2000 with monthly frequency
y <- zoo(serie, order.by = as.Date(seq(from = as.Date("2000-01-01"), by = "month", length.out = length(serie))))

# Plot the original time series
plot(y, main = "Original Time Series", ylab = "Value", xlab = "Time")

# Perform Augmented Dickey-Fuller test to check for stationarity
adf <- adfTest_valid(y,24,"ct")
print(adf)

adf <- adfTest_valid(y,24,"c")
print(adf)

# Perron-Phillips test to check for stationarity
pp.test(y)
y.pp = ur.pp(y, type = "Z-tau", model = "trend")
summary(y.pp)

# Optionally, add ACF and PACF plots for the series to inspect autocorrelations
acf(coredata(y), main = "ACF of Series")
pacf(coredata(y), main = "PACF of Series")

# Difference the series to achieve stationarity
y_df = diff(y)

# Plot the differenced series
plot(y_df, main = "Differenced Time Series", ylab = "Value", xlab = "Time")

# Plot ACF and PACF for the differenced series
acf(coredata(y_df), main = "ACF of Differenced Series")
pacf(coredata(y_df), main = "PACF of Differenced Series")

# Set maximum AR and MA orders for model selection
pmax = 5
qmax = 1
```

A.3 Codes for Part II

Codes for Part II

```r
# Find the best ARMA models within the specified p and q ranges
armamodels <- armamodelchoice(pmax,qmax, y_df)

# Select models that are well-adjusted and valid
selec <- armamodels[armamodels[,"ok"]==1&!is.na(armamodels[,"ok"]),] #models well adjusted and valid
selec[, c("p", "q", "arsignif", "masignif", "resnocorr", "ok")]

# Fit several ARMA models
models <- list()
aic_values <- numeric()
bic_values <- numeric()

# Find the best model with AIC and BIC
for (i in 1:nrow(selec)) {
  p <- selec$p[i]
  q <- selec$q[i]
  fit <- tryCatch({
    Arima(coredata(y_df), order = c(p, 0, q))
  }, error = function(e) NULL)
```

Codes for Part II

```
  if (!is.null(fit)) {
    models[[paste0("ARMA(", p, ",", q, ")")]] <- fit
    aic_values <- c(aic_values, fit$aic)
    bic_values <- c(bic_values, BIC(fit))
  }
}

# Create a data frame to compare models
comparison <- data.frame(
  Model = names(models),
  AIC = aic_values,
  BIC = bic_values
)

# Print the model comparison table
print(comparison)

# Find the best models based on AIC and BIC
best_aic_model_index <- which.min(comparison$AIC)
best_bic_model_index <- which.min(comparison$BIC)

best_aic_model_name <- names(models)[best_aic_model_index]
best_bic_model_name <- names(models)[best_bic_model_index]

best_aic_model <- models[[best_aic_model_name]]
best_bic_model <- models[[best_bic_model_name]]

# Display summaries of both models
cat("Model with the best AIC:\n")
summary(best_aic_model)

cat("\nModel with the best BIC:\n")
summary(best_bic_model)

best_model = best_aic_model

# Check residuals of the best AIC model
cat("\nResidual diagnostics for the best AIC model:\n")
checkresiduals(best_model)

# Perform Ljung-Box test on residuals of the best AIC model
cat("\nLjung-Box test for residuals (best AIC model):\n")
Ljung.Box(best_model$residuals, maxlag = 24, par=1)
```

A.4 Codes for Part III

Codes for Part III

```
h <- 2

# Generate forecasts
forecasts <- forecast(best_model, h = h, level = 0.95)

# Extract the last 30 observations from the original series
last_30_index <- tail(index(forecasts$x), 30)
last_30_data <- window(forecasts$x, start = last_30_index[1])

# Extract forecast values and confidence intervals
forecast_values <- forecasts$mean
lower_bound <- forecasts$lower  # 95% lower bound
upper_bound <- forecasts$upper  # 95% upper bound

# Assuming you have the index of the last fitted value and the first forecast value
last_fitted_time <- tail(time(forecasts$fitted), 1)
first_forecast_time <- head(time(forecasts$mean), 1)

last_fitted_value <- tail(forecasts$fitted, 1)
first_forecast_value <- head(forecasts$mean, 1)

# Plot the last 30 observations with forecasts and confidence intervals
plot(last_30_data, main = "Forecast with 95% Confidence Intervals", xlab = "Time", ylab = "Value", type
     = "l", xlim=c(260,290))
lines(forecasts$fitted, col = "blue")
points(last_30_data, col="black")
```

Codes for Part III

```
points(forecasts$fitted, col="blue")
points(time(forecasts$mean), forecasts$mean, col = "red")
arrows(time(forecasts$mean), lower_bound, time(forecasts$mean), upper_bound, code = 3, angle = 90,
    length = 0.1, col = "red", lty=3)
lines(forecasts$mean, col="red")

# Draw a link between the last blue point and the first red point
segments(last_fitted_time, last_fitted_value, first_forecast_time, first_forecast_value, col="red")

# Add legend
legend("bottomleft", legend = c("Series", "Fitted", "Forecast", "95% CI"), col = c("black", "blue", "
    red", "red"), lty = c(1, 1, 1, 3), pch = c(NA, NA, 1, NA), bty = "o", cex = 0.7)
```