Advanced Macroeconometrics – Assignment 1

Siegfried Hammer (h12229325@wu.ac.at) Max Heinze (h11742049@wu.ac.at)
Tim Koenders (h12215486@wu.ac.at)

April 19, 2023

The executable code that was used in compiling the assignment is available on GitHub at https://github.com/maxmheinze/macrometrics.

Exercise 1

First, we read in the data set, removing the first column indicating the recommended transformation.

```
# Header ------------------------------------------------------------------

rm(list = ls())
gc()

pacman::p_load(tidyverse, urca)


# Read in Data ------------------------------------------------------------

fred <- read.csv("./assignment1/data/fred.csv")[-1, ]
```

Next, we create the desired function `ts_explode()`. Along with the vector to be transformed, it asks for a specification of whether the data is ordered with the latest or earliest value first. It takes the earliest value first as default.

```
# Create the function -----------------------------------------------------

ts_explode <- function(input_vector, start_with_latest = FALSE) {
    # The function is called ts_explode because a single vector explodes into
    # an entire data frame. Boom!

    # Package dplyr required for lag() function
    require(dplyr)

    # Reverse input vector if user specifies it is sorted latest to earliest
    input_vector <- if (start_with_latest == FALSE) {
        input_vector
    } else {
        rev(input_vector)
    }

    # Do the transformations, assign transformed vectors
    original <- input_vector
    log_transformed <- log(input_vector)
    mom_growth <- input_vector/dplyr::lag(input_vector, 1) - 1
    yoy_growth <- input_vector/dplyr::lag(input_vector, 12) - 1
    yoy_growth_lagged <- dplyr::lag(input_vector, 12)/dplyr::lag(input_vector, 24) -
        1
```

```
    # Create a data frame to export, reverse ordering back to original in case
    # start_with_latest = TRUE was specified
    export_df <- if (start_with_latest == FALSE) {
        data.frame(original, log_transformed, mom_growth, yoy_growth, yoy_growth_lagged)
    } else {
        data.frame(original = rev(original), log_transformed = rev(log_transformed),
            mom_growth = rev(mom_growth), yoy_growth = rev(yoy_growth), yoy_growth_lagged =
            rev(yoy_growth_lagged))
    }

    # Display warnings regarding ordering and units of growth rates
    warning("By default, ts_explode() assumes that values are ordered from earliest to
    latest. If your vector is ordered from latest to earliest, specify `start_with_latest =
    TRUE`!")
    warning("Growth rates are given in decimals, not in percent!")

    # Return the data frame
    return(export_df)
}
```

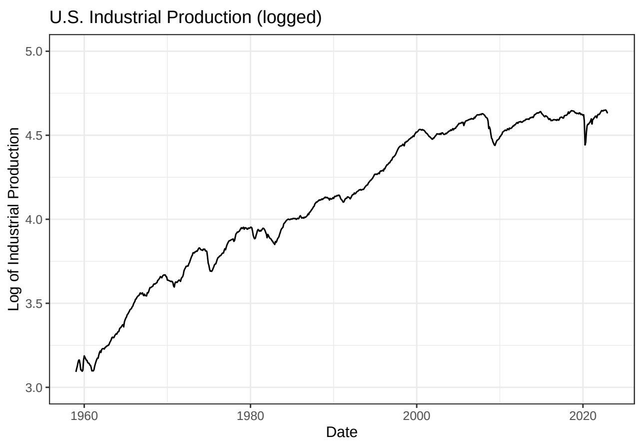
Using `ts_explode()`, we create the data frame `ind_prod` including all transformations of the `INDPRO` variable. We bind the data frame together with the date column, which we transform from character to date. All other changes in the resulting data frame are of cosmetic nature.

```
# Prepare Industrial Production Data Frame --------------------------------

ind_prod <- fred$sasdate %>%
    cbind(ts_explode(fred$INDPRO)) %>%
    as_tibble() %>%
    mutate(date = lubridate::mdy(.)) %>%
    select(-.) %>%
    relocate(date, .before = original)
```

Next, we plot both the logged variable and the year-on-year growth rate.

```
# Create Log Plot ---------------------------------------------------------

ind_prod %>%
    ggplot() + geom_line(aes(x = date, y = log_transformed)) + labs(title = "U.S. Industrial
    Production (logged)",
    x = "Date", y = "Log of Industrial Production") + ylim(3, 5) + theme_bw()
```

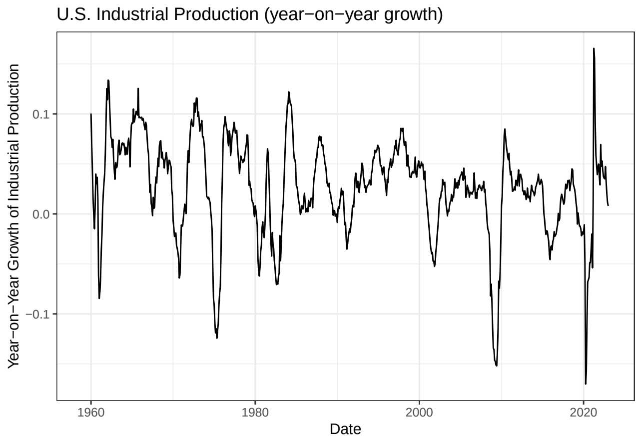
```
## Warning: Removed 1 row containing missing values (`geom_line()`).
```




```
# Create Growth Plot ------------------------------------------------------

ind_prod %>%
```

```
ggplot() + geom_line(aes(x = date, y = yoy_growth)) + labs(title = "U.S. Industrial
    Production (year-on-year growth)",
    x = "Date", y = "Year-on-Year Growth of Industrial Production") + theme_bw()
```

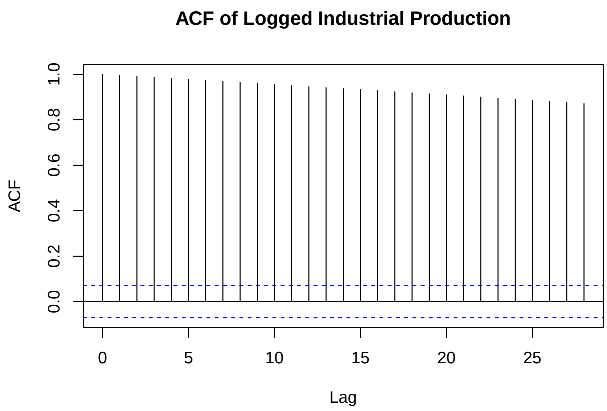
```
## Warning: Removed 13 rows containing missing values (`geom_line()`).
```



The plots above depict the logged time series of U.S. industrial production and its year-on-year growth rates. The logged time series of U.S. industrial production appears to exhibit a generally upward trend over time. The logged time series of U.S. industrial production also shows evidence of cyclical patterns, with periods of expansion and contraction occurring in a cyclical manner. There are also periodic fluctuations in the year-on-year growth rates, which suggests the presence of seasonality in U.S. industrial production.

Next we assess the properties of both logged industrial production and its yearly growth rate.

```
# Assessing the time series properties ----------------------------------------

# ACF plot and ADF test(s) of Logged Industrial Production

stats::acf(ind_prod$log_transformed[!is.na(ind_prod$log_transformed)], main = "ACF of Logged
Industrial Production")
```




```
urtest1a = ur.df(ind_prod$log_transformed[!is.na(ind_prod$log_transformed)], type = "trend",
    selectlags = "AIC")
summary(urtest1a)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
```

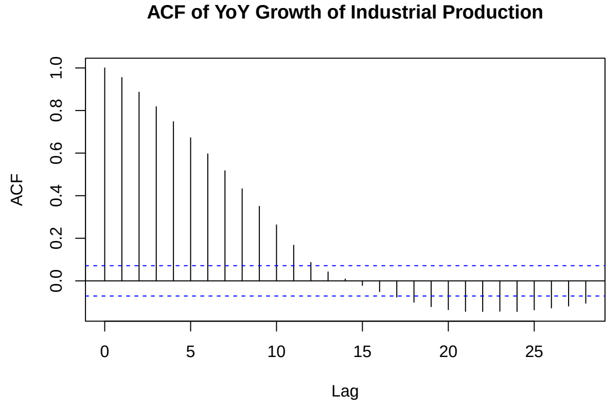
```
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.131711 -0.003784  0.000414  0.004158  0.054957
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.923e-02  1.006e-02   1.911   0.0564 .
## z.lag.1     -4.932e-03  2.998e-03  -1.645   0.1004
## tt           6.338e-06  6.121e-06   1.036   0.3007
## z.diff.lag   2.889e-01  3.458e-02   8.353 3.12e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.009501 on 763 degrees of freedom
## Multiple R-squared:  0.0963, Adjusted R-squared:  0.09275
## F-statistic:  27.1 on 3 and 763 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -1.6449 7.7617 3.7276
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

```
urtest2a = ur.df(ind_prod$log_transformed[!is.na(ind_prod$log_transformed)], type = "drift",
    selectlags = "AIC")
summary(urtest2a)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.131010 -0.003791  0.000361  0.004097  0.055859
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0093366  0.0031640   2.951  0.00327 **
## z.lag.1     -0.0019295  0.0007638  -2.526  0.01173 *
## z.diff.lag   0.2866319  0.0345154   8.304 4.55e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.009501 on 764 degrees of freedom
## Multiple R-squared:  0.09503,    Adjusted R-squared:  0.09266
## F-statistic: 40.11 on 2 and 764 DF,  p-value: < 2.2e-16
##
```

```
##
## Value of test-statistic is: -2.5263 11.1054
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

```
# ACF plot and ADF test(s) of Year-on-Year Growth Rate

stats::acf(ind_prod$yoy_growth[!is.na(ind_prod$yoy_growth)], main = "ACF of YoY Growth of
Industrial Production")
```




```
urtest1b = ur.df(ind_prod$yoy_growth[!is.na(ind_prod$yoy_growth)], type = "trend",
    selectlags = "AIC")
summary(urtest1b)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.112205 -0.005982  0.000166  0.005728  0.134883
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.846e-03  1.088e-03   2.615   0.0091 **
## z.lag.1     -6.130e-02  1.047e-02  -5.852 7.24e-09 ***
## tt          -3.507e-06  2.295e-06  -1.528   0.1268
## z.diff.lag   3.272e-01  3.435e-02   9.527  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01324 on 751 degrees of freedom
## Multiple R-squared:  0.1286, Adjusted R-squared:  0.1251
## F-statistic: 36.94 on 3 and 751 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -5.8523 11.4203 17.1258
```

```
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

```
urtest2b = ur.df(ind_prod$yoy_growth[!is.na(ind_prod$yoy_growth)], type = "drift",
    selectlags = "AIC")
summary(urtest2b)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.113192 -0.006224  0.000028  0.005802  0.133866
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0014075  0.0005468   2.574   0.0102 *
## z.lag.1     -0.0570068  0.0100998  -5.644 2.35e-08 ***
## z.diff.lag   0.3246308  0.0343372   9.454  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01325 on 752 degrees of freedom
## Multiple R-squared:  0.1259, Adjusted R-squared:  0.1235
## F-statistic: 54.14 on 2 and 752 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -5.6443 15.934
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

The ACF of Logged Industrial Production reveals a highly persistent nature of the time series with significant autocorrelation coefficients, particularly between Logged Industrial Production and its first five lags, which are extremely close to 1. The autocorrelation coefficients remain highly significant in the displayed laggs. This highly persistent nature suggests potential non-stationarity in the series. For a more appropriate procedure, we apply the Augmented Dickey-Fuller (ADF) unit root tests.

Type: "trend" (urtest1a) H0: The time series is random walk around a trend H1: The time series is trend-stationary

As the coefficient of trend (tt) is insignificant, we never apply the ADF test with drift and trend. Instead, we move on to the second equation including only the drift component.

Type: "drift" (urtest2a) H0: The time series is random walk around a drift H1: The time series is (log-level) stationary around a drift

As the drift component is significant, and the absolute value of the test-statistic (2.5263) is below all the critical values for conventional significance levels, we conclude that we fail to reject the null hypothesis that the time series is random walk around a drift.

Thus, we conclude that Logged Industrial Production is I(1).

The ACF of YoY Growth rate of Industrial Production shows positive persistence, but to a lesser degree than the former series. The correlation coefficient decreases relatively quickly and even turns significantly negative after the 17th lag, indicating the presence of a pattern or cycle in the data at specific time lags. Now we apply the Augmented Dickey-Fuller (ADF) unit root tests.

Type: "trend" (urtest1b) H0: The time series is random walk around a trend H1: The time series is trend-stationary

As the coefficient of trend (tt) is insignificant, we never apply the ADF test with drift and trend. Instead, we move on to the second equation including only the drift component.

Type: "drift" (urtest2b) H0: The time series is random walk around a drift H1: The time series is stationary around a drift

As the drift component is significant, and the absolute value of the test-statistic (5.6443) is above all the critical values for conventional significance levels, we conclude that we reject the null hypothesis that the time series is random walk around a drift.

Thus, we conclude that the YoY Growth of Industrial Production is I(0).

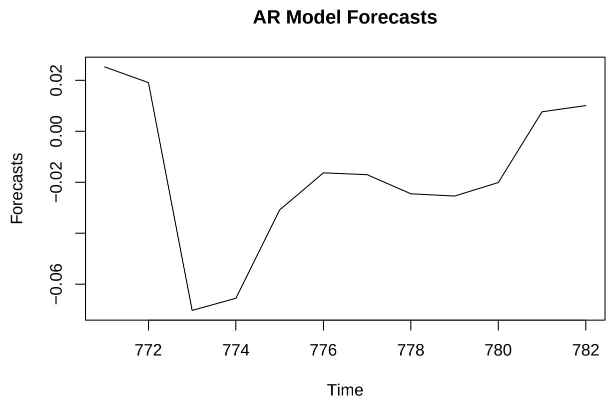
```
# Estimate AR model ----------------------------------------------------------

ar_model_yoygrowth = ar.ols(ind_prod$yoy_growth[!is.na(ind_prod$yoy_growth)])
summary(ar_model_yoygrowth)
```

```
##               Length Class  Mode
## order           1    -none- numeric
## ar             26    -none- numeric
## var.pred        1    -none- numeric
## x.mean          1    -none- numeric
## x.intercept     1    -none- numeric
## aic            29    -none- numeric
## n.used          1    -none- numeric
## n.obs           1    -none- numeric
## order.max       1    -none- numeric
## partialacf      0    -none- NULL
## resid         757    -none- numeric
## method          1    -none- character
## series          1    -none- character
## frequency       1    -none- numeric
## call            2    -none- call
## asy.se.coef     2    -none- list
```

```
# Generate forecasts for the next year
forecasts <- predict(ar_model_yoygrowth, n.ahead = 12)
print(forecasts)
```

```
## $pred
## Time Series:
## Start = 758
## End = 769
## Frequency = 1
##  [1]  0.025269455  0.019107977 -0.070284530 -0.065514117 -0.030876699
##  [6] -0.016306685 -0.017024809 -0.024510326 -0.025405957 -0.020109857
## [11]  0.007656505  0.010083272
##
## $se
## Time Series:
## Start = 758
## End = 769
## Frequency = 1
##  [1] 0.01016206 0.01654370 0.02111975 0.02485038 0.02826507 0.03122660
##  [7] 0.03401918 0.03654871 0.03859429 0.04052347 0.04271988 0.04482780
```

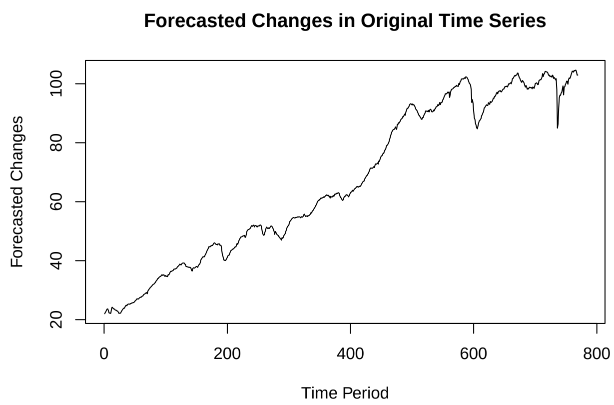
```r
# Extract the forecasted values and the corresponding time period
forecasted_values <- as.vector(forecasts$pred)
time_period <- seq_along(forecasted_values) + length(ind_prod$yoy_growth)

# Plot the forecasts
plot(time_period, forecasted_values, type = "l", main = "AR Model Forecasts", xlab = "Time",
    ylab = "Forecasts")
```




```r
# Average YoY growth rate
mean(ind_prod$yoy_growth[!is.na(ind_prod$yoy_growth)])
```

```
## [1] 0.02552905
```

```r
# Forecast change in original time series based on forecasted year-on-year
# growth rate
last_observed_original <- tail(ind_prod$original, n = 1)
forecasted_change <- last_observed_original * forecasted_values

# Combine original time series with forecasted change
forecasted_original <- c(ind_prod$original, forecasted_change)

# Define time periods for forecasted changes
time_period_forecasted_change <- seq_along(forecasted_original)

# Plot forecasted changes
plot(time_period_forecasted_change, forecasted_original, type = "l", xlab = "Time Period",
    ylab = "Forecasted Changes", main = "Forecasted Changes in Original Time Series")
```



The lag order is determined by default in the ar.ols() function using the Akaike Information Criterion (AIC) as the method for order selection. AIC is a model selection criterion that balances model goodness of fit and complexity. Lower AIC values are preferred. The ar.ols() function computes the AIC value for different lag orders and selects the lag order that minimizes the AIC as the optimal lag order for the AR model.

If the AR model is capturing the underlying patterns wel, then as the YoY growth rate of industrial production is stationary, the forecasts are likely to converge towards their long-run average of around 0.026 (2.6%).

```r
# Bonus (AR Model of (Logged) Industrial
# Production-----------------------------

# Define function to calculate RMSE for AR model with given lag order
calculate_rmse <- function(lag_order, holdout_period, ind_prod) {

    # Exclude the holdout period from the end of the sample
    train_data <- ind_prod$log_transformed[1:(length(ind_prod$log_transformed) -
        holdout_period)]

    # Estimate AR model
    ar_model <- ar.ols(train_data, aic = FALSE, order.max = lag_order)

    # Produce forecasts for the holdout period
    forecasts <- predict(ar_model, n.ahead = holdout_period)

    # Extract predicted values for the holdout period
    predicted_values <- forecasts$pred

    # Extract realized values for the holdout period
    realized_values <- ind_prod$log_transformed[(length(ind_prod$log_transformed) -
        holdout_period + 1):length(ind_prod$log_transformed)]

    # Remove missing values from predicted_values and realized_values
    predicted_values <- na.omit(predicted_values)
    realized_values <- na.omit(realized_values)

    # Check and adjust length of predicted_values and realized_values
    if (length(predicted_values) > length(realized_values)) {
        predicted_values <- predicted_values[1:length(realized_values)]
    } else if (length(realized_values) > length(predicted_values)) {
        realized_values <- realized_values[1:length(predicted_values)]
    }

    # Compute RMSE
    rmse <- sqrt(mean((predicted_values - realized_values)^2))
    return(rmse)
}

# Specify lag orders to compare
lag_orders <- c(1, 2, 3, 4, 5, 6, 7, 8)

# Specify holdout periods to compare
holdout_periods <- c(6, 12, 24)

# Initialize matrix to store RMSE results
results <- matrix(NA, nrow = length(lag_orders), ncol = length(holdout_periods))

# Iterate over lag orders and holdout periods to calculate RMSE for each model
for (i in 1:length(lag_orders)) {
    for (j in 1:length(holdout_periods)) {
        rmse <- calculate_rmse(lag_orders[i], holdout_periods[j], ind_prod)
        results[i, j] <- rmse
    }
}

# Print RMSE results
print(results)
```

```
##             [,1]        [,2]       [,3]
## [1,] 0.01083095 0.009360575 0.05575435
## [2,] 0.01020996 0.007806726 0.07005326
## [3,] 0.01017002 0.008160002 0.06738367
## [4,] 0.01018277 0.008012830 0.06806190
## [5,] 0.01018192 0.007866010 0.06822266
## [6,] 0.01007486 0.008113162 0.06834463
## [7,] 0.01016670 0.008140459 0.07030717
## [8,] 0.01027124 0.008264455 0.07092213
```

The AR Model with a holdout period of 12 months and a lag order of 2 performs the based among the alternatives as it minimizes the RMSE. When specifying a holdout period of 6 (24) months, we would, based on minimizing the RMSE, select an AR model with 6 (1) lags. Thus, the preferred specification (i.e. optimal lag order) depends on the selected holdout period.