Time Series Econometrics: Homework assignment 2

Joaquín Barrutia Álvarez

1 Plot the time series with proper labeling of axes and dates. Also plot the autocorrelation function (ACF) and the partial autocorrelation function (PACF) of the series. Based on these plots, make a guess of an ARMA(p, q) model that you believe could be appropriate.

Before we start applying trying to find an ARMA(p,q) model, we should make sure that the process Y_t is stationary, where Y is the Change in Volume Seasonally Adjusted GDP and $t = 1981Q2, ..., 2023Q3$.

We know that if neither the expectation μ_t nor the autocovariances γ_{jt} depend on the time t, the the process for Y_t is said to be covariance-stationary or weakly stationary.

$$E(Y_t) = \mu \text{ for all } t$$
$$E(Y_t - \mu)(Y_{t-j} - \mu) = \gamma_{jt} < \infty \text{ for all } t \text{ and any } j$$

This is brought to attention because AR or MA are not applicable on non-stationary series. In situations where the stationary criteria are not met, the primary step is to make the time series stationary, and afterward, attempt stochastic models for forecasting this time series.

First we can start looking at the trend by looking at the blue line in Figure 1. It is evident that Y_t doesn't display a deterministic trend.

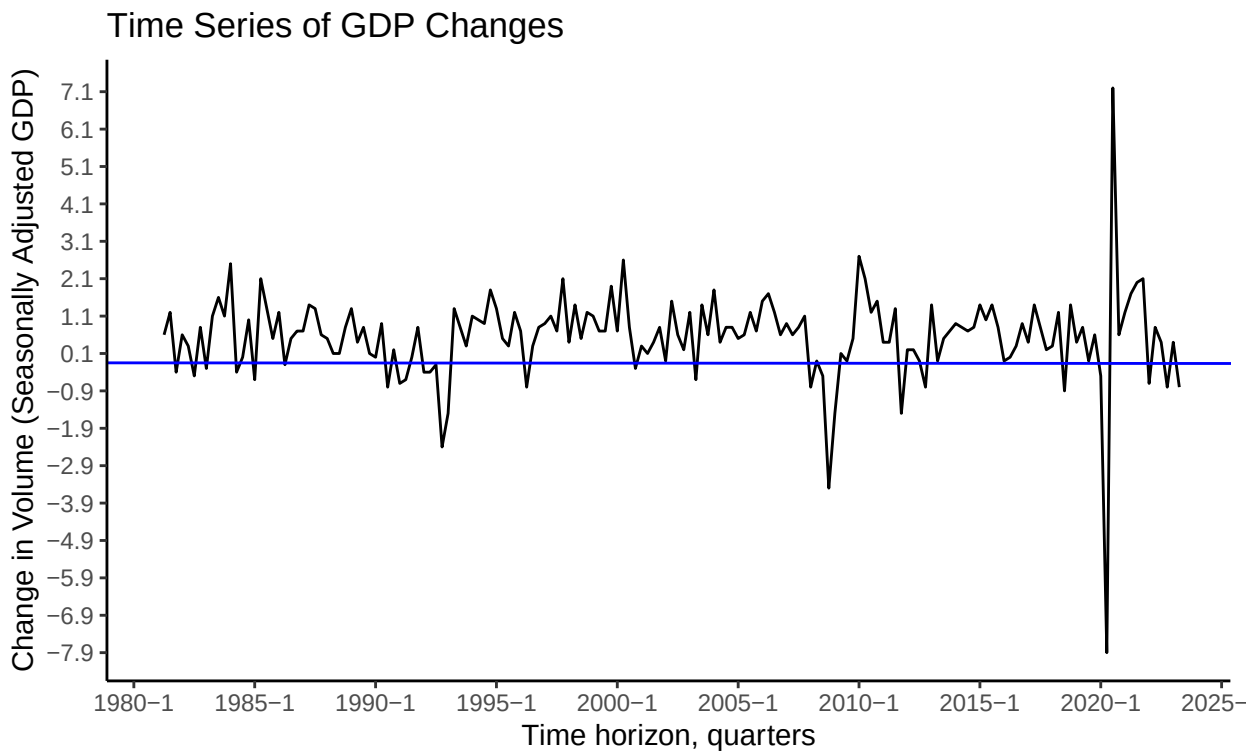

Figure 1: Time series of GDP change in volume seasonally adjusted with trend

We can further check if our series is stationary by using the Augmented Dickey–Fuller:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \cdots + \delta_{p-1} \Delta y_{t-p+1} + \varepsilon_t,$$

where β is the coefficient on a time trend and p the lag order of the autoregressive process. The inclusion of lag terms of order p in the ADF formulation permits the consideration of higher-order autoregressive processes. The unit root test is conducted with the null hypothesis that $\gamma = 0$, while the alternative hypothesis suggests $\gamma < 0$. After calculating the test statistic, it can be compared to the appropriate critical value for the Dickey–Fuller test.

$$\mathrm{DF}_\tau = \frac{\hat{\gamma}}{\mathrm{SE}(\hat{\gamma})}$$

The result of the test is computed here using a lag order = 5.

```
adf.test(df$change_in_vol)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  df$change_in_vol
## Dickey-Fuller = -6.2445, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

This evidence suggests that our data is stationary.

We can now take a look at our ACF and PACF. The autocorrelation function (ACF) evaluates the correlation between observations in a time series at various lags. The ACF for a time series Y is defined as: $Corr(y_t, y_{t-k})$

for k = 1, 2, ... While the PACF can be calculated by using the Durbin–Levinson Algorithm. By looking at the ACF and PACF our series seems to follow a white noise process. This wouldn't be a good model to explain a macroeconomic variable such as GDP so we should try to look what's causing our data to look like this.

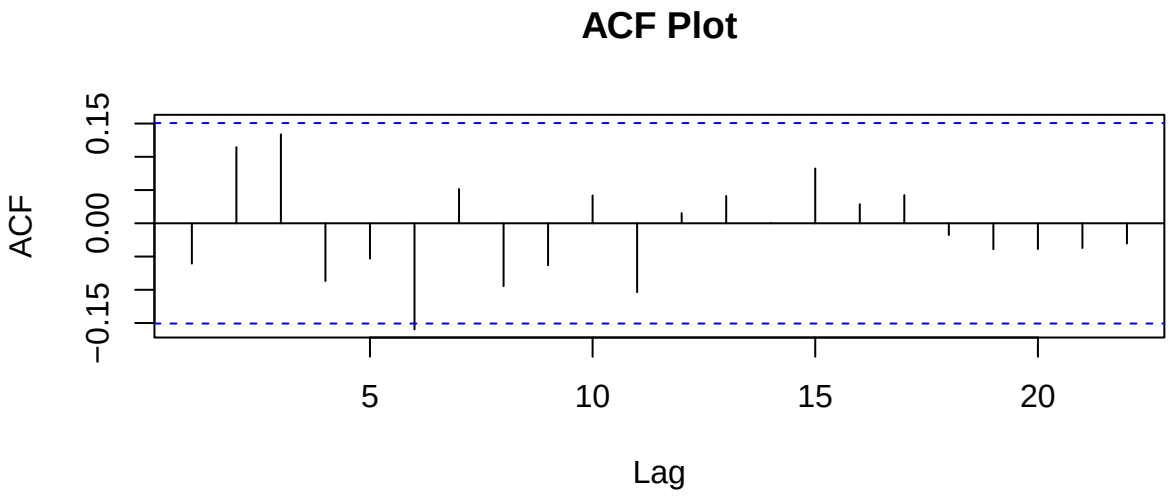

Figure 2: ACF full sample

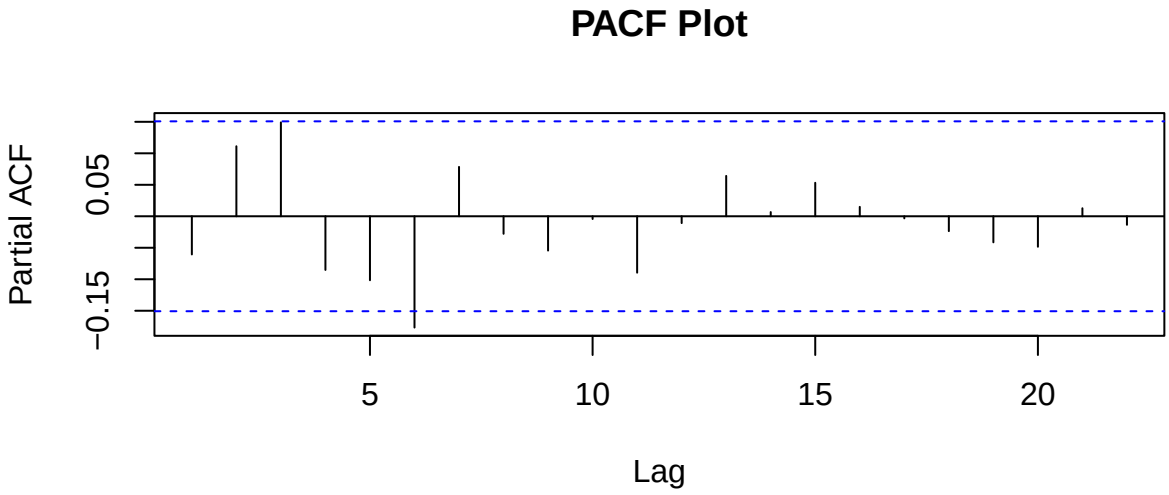

Figure 3: PACF full sample

It is evident by looking at Figure 4 that there is a significant shock in 2020 related to the COVID pandemic. These observations are outliers in our data. Since this event was an almost isolated, exogenous factor affecting macroeconomic variables, we should not incorporate it into our analysis. Our primary focus is on understanding the underlying trends and patterns in GDP volume changes. Including such extreme outliers could distort our analysis and misrepresent the typical behavior of the economy over time. Therefore, removing these outliers is essential to obtain a more accurate and meaningful representation of the data,

allowing us to make informed economic assessments and predictions. For these reasons we are going to filter our data and only work with observations from before 2020.

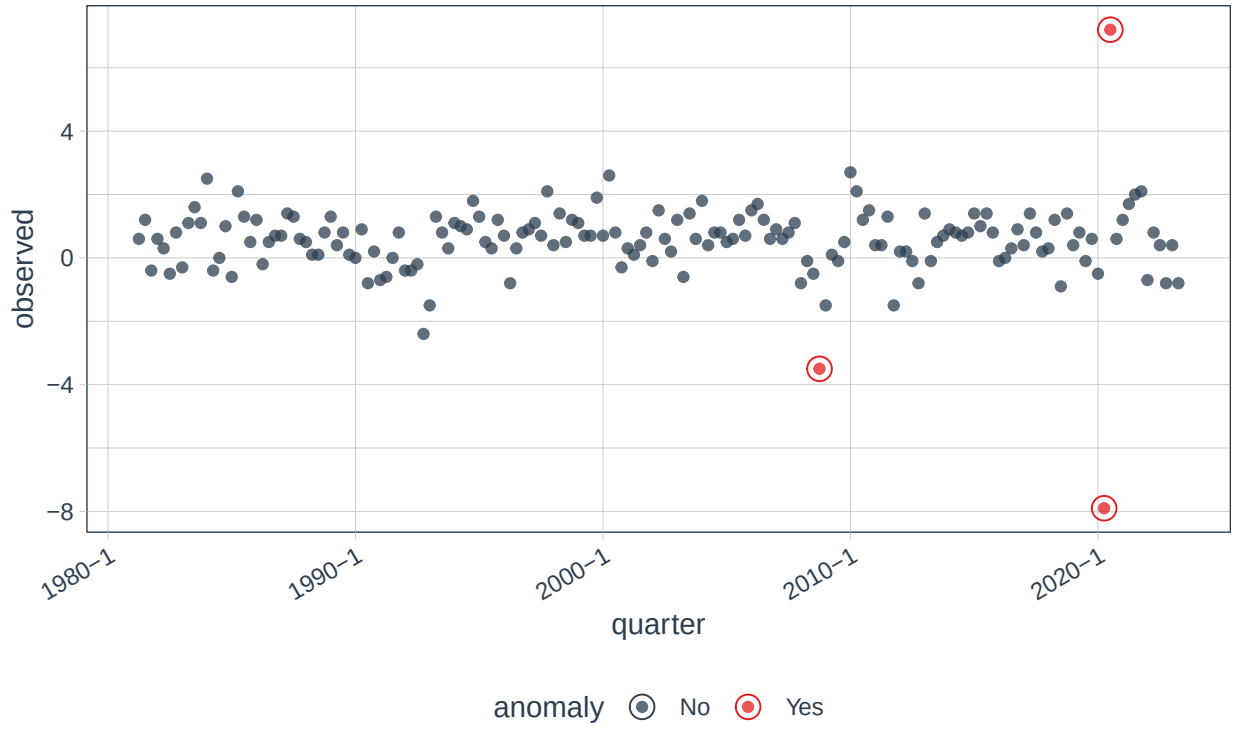

Figure 4: Time series of GDP change in volume seasonally adjusted showing outliers

By looking at the new ACF and PACF plots in Figure 6 and 8 we can think that they follow an ARMA(3,3) process. And this is the model I will use in the next questions.

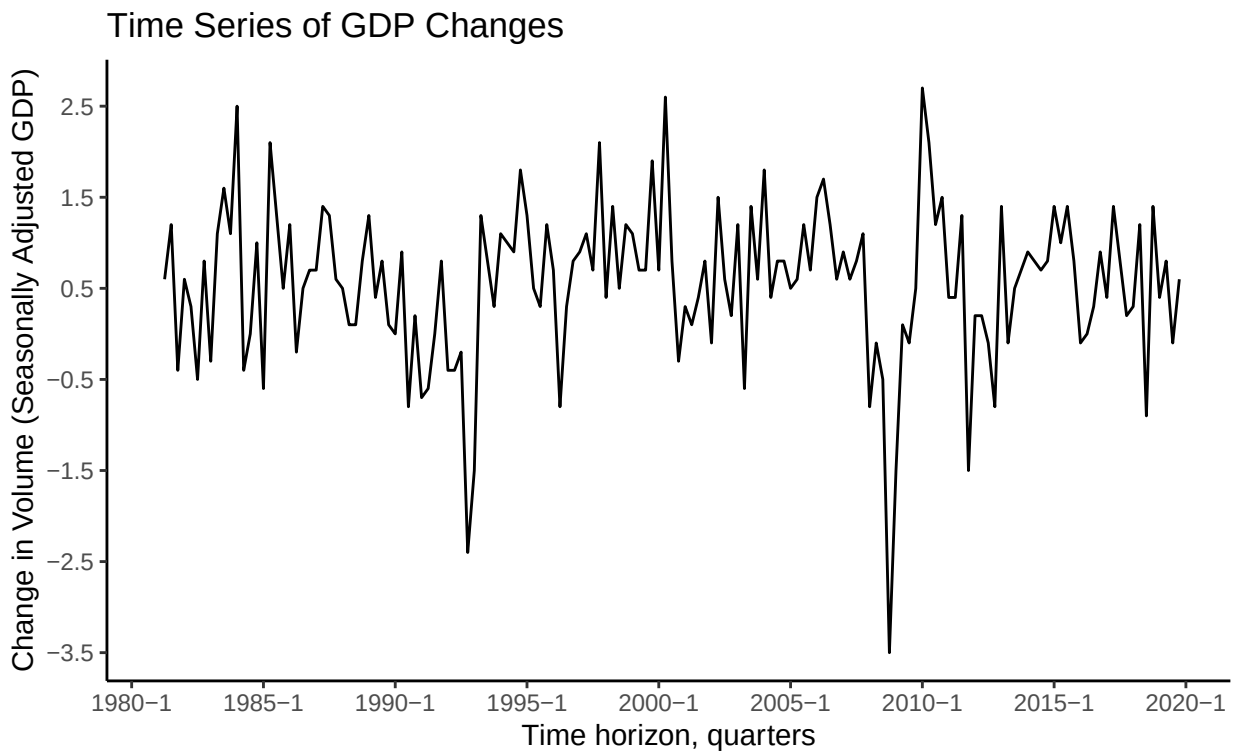

Figure 5: Time series of GDP change in volume seasonally reduced sample.

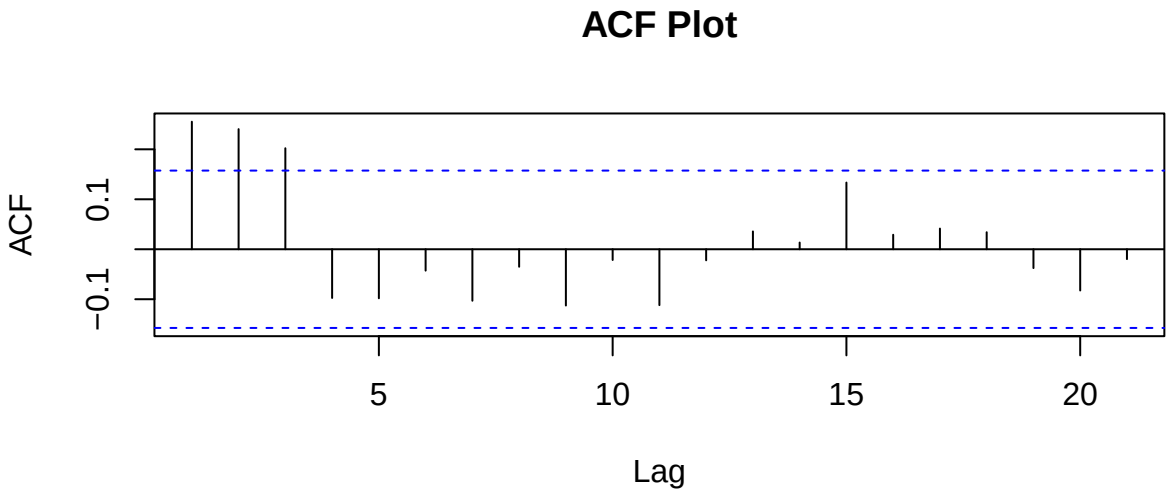

Figure 6: ACF reduced sample

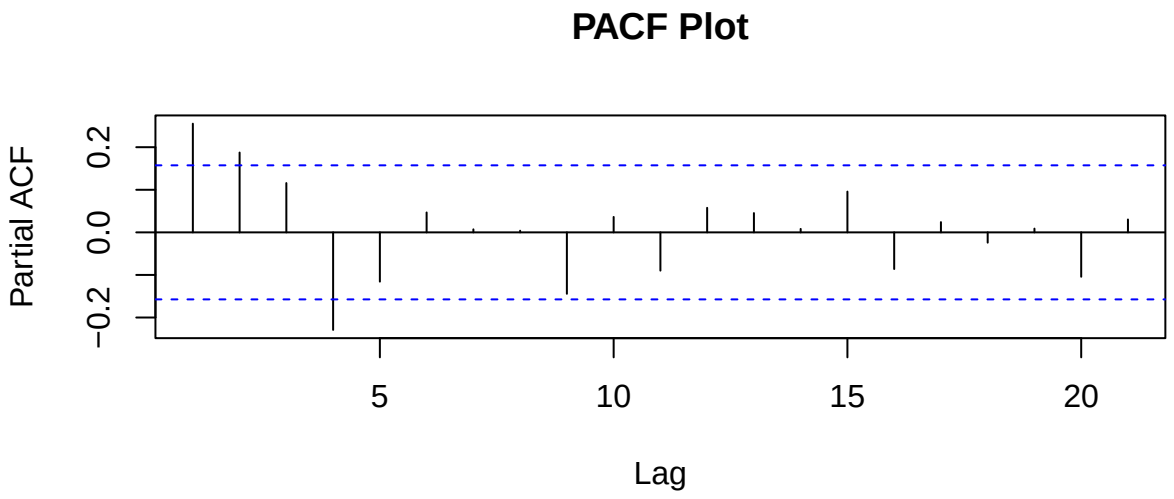

Figure 7: PACF reduced sample

2 Perform a forecast evaluation of one- and two-step predictions, where the models to be evaluated are:

- (a) ARMA(1, 1), ARMA(2, 1), ARMA(1, 2) and ARMA(2, 2)
- (b) Your choice of ARMA(p, q) in the previous part
- (c) The "best" ARMA according to auto.arima() in the forecast package1, with max.p and max.q set to 4

The model suggested by auto.arima() was an ARMA(4,0). We will use the following criteria to evaluate our models:

$$RMSE_h^{(j)} = \sqrt{\sum_{t=2011Q4}^{2015Q2-h} \frac{(y_{t+h} - \hat{y}_{t+h|t}^{(j)})^2}{\#forecasts}},$$

$$MAD_h^{(j)} = \sum_{t=2011Q4}^{2015Q2-h} \frac{|y_{t+h} - \hat{y}_{t+h|t}^{(j)}|}{\#forecasts},$$

$$Bias_h^{(j)} = \sum_{t=2011Q4}^{2015Q2-h} \frac{y_{t+h} - \hat{y}_{t+h|t}^{(j)}}{\#forecasts}$$

The results in Tables 1 display evaluation metrics for the different ARMA models used for making one step predictions. Here are some observations based on the results:

In terms of **RMSE (Root Mean Square Error)**, the models for 1-step predictions exhibit values ranging from 0.5388 to 0.6628. The lowest RMSE is associated with the ARMA(2,2) model, indicating it has the smallest average squared prediction error among these models for 1-step predictions.

For **MAD (Mean Absolute Deviation)**, the MAD values vary between 0.3899 and 0.4338. The ARMA(2,1) model has the lowest MAD, suggesting it has the smallest absolute prediction error on average for 1-step predictions.

In the context of **Bias**, representing the difference between predictions and actual values, bias values range from 0.0497 to 0.0779. The ARMA(4,0) model has the smallest bias, implying it tends to be closer to the actual observations on average for 1-step predictions.

Overall, the 1-step prediction analysis suggests that the ARMA(2,2) model performs well with low RMSE and MAD values, while the ARMA(4,0) model exhibits the least bias.

The results in Tables 2 display evaluation metrics for the different ARMA models used for making two step predictions. Here are some observations based on the results:

1. **RMSE (Root Mean Square Error):**
   - RMSE values range from 0.5315 to 0.6106.
   - The ARMA(2,2) model has the lowest RMSE value (0.5315), indicating that, on average, its predictions have the lowest mean squared error compared to the other models.
   - The ARMA(4,0) model has the highest RMSE value (0.6106), suggesting it has the highest mean squared error in its predictions.
2. **MAD (Mean Absolute Deviation):**
   - MAD values vary between 0.3959 and 0.4272.
   - Similar to RMSE, the ARMA(2,2) model has the lowest MAD value (0.3959), implying it has the lowest mean absolute error in its predictions on average.
   - The ARMA(1,1) model has the highest MAD value (0.4272), indicating a higher mean absolute error in its predictions compared to the other models.
3. **Bias (Difference between Predictions and Actual Observations):**
   - Bias values range from 0.05795 to 0.10784.
   - The ARMA(4,0) model has the lowest bias (0.05795), suggesting that, on average, it slightly underestimates actual observations.
   - The ARMA(1,2) model has the highest bias (0.10784), indicating a tendency to overestimate actual observations on average.

In summary:

- The ARMA(2,2) model appears to perform the best in terms of RMSE and MAD, as it has the lowest values for both metrics, suggesting better accuracy and fit to the data.
- The ARMA(4,0) model has the lowest bias, indicating that, despite having a higher RMSE, it tends to make predictions that are closer to actual observations on average.
- The ARMA(1,1) and ARMA(1,2) models have relatively high values in RMSE, MAD, and bias, suggesting they may not be the most suitable models for these predictions.

Comparing the 1-step and 2-step predictions, it's evident that in most cases, the RMSE and MAD values tend to increase for 2-step predictions compared to 1-step predictions. This is expected because making predictions further into the future is generally more challenging and prone to errors. The ARMA(2,2) model continues to perform well with relatively low RMSE and MAD values in both 1-step and 2-step predictions, suggesting it maintains good predictive accuracy for short-term and slightly longer-term forecasts. The ARMA(4,0) model still exhibits low bias, indicating that it tends to make predictions that are close to actual observations on average for both 1-step and 2-step predictions. The ARMA(1,1) and ARMA(1,2) models also show consistent performance characteristics in both 1-step and 2-step predictions, but their RMSE and MAD values are higher compared to the ARMA(2,2) model. Overall, the choice of the appropriate model should consider the specific forecasting horizon we are interested in. If short-term predictions (1-step) are crucial, then models like ARMA(2,2) or ARMA(4,0) may be preferred. However, if we need slightly longer-term predictions (2-step), it's important to be aware of the increased uncertainty and potentially higher error rates.

In conclusion, there is no model that performs better in the three criteria used. However, the choice of the model should not rely solely on these metrics. It's important to consider the context and theory behind the models, as well as perform additional cross-validation tests or further comparisons to make an informed decision on which model is best suited for our specific research questions.

Table 1: model performance one-step predictions

Model	RMSE	MAD	Bias
ARMA11	0.5706326	0.4144694	0.0762017
ARMA12	0.5618414	0.4193861	0.0746327
ARMA21	0.5601498	0.3899594	0.0779191
ARMA22	0.5388195	0.3965804	0.0503330
ARMA33	0.6213255	0.4147595	0.0711462
ARMA40	0.6627717	0.4338094	0.0496943

Table 2: model performance one-step predictions

Model	RMSE	MAD	Bias
ARMA11	0.5553659	0.4272136	0.0887858
ARMA12	0.5535479	0.4145349	0.1078370
ARMA21	0.5515731	0.4202408	0.0978405
ARMA22	0.5315436	0.3959362	0.0763662
ARMA33	0.5814621	0.4022592	0.0928301
ARMA40	0.6105751	0.4207046	0.0579522

Finally, we can take a look at Figures 8 and 9 to visually see the performance of our models. We can see that our predictions are sometimes really far away from the actual data, this models may be useful in some specific settings, but for really precise and accurate predictions we should incorporate more information to our models, which will be discussed further in future lectures.

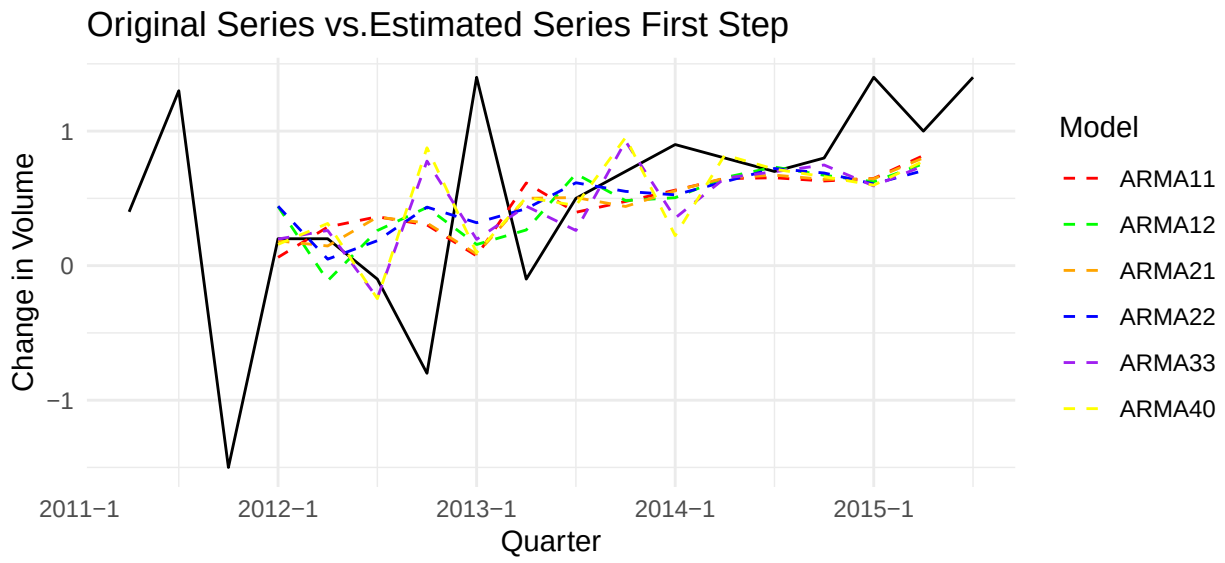

Figure 8: PACF reduced sample

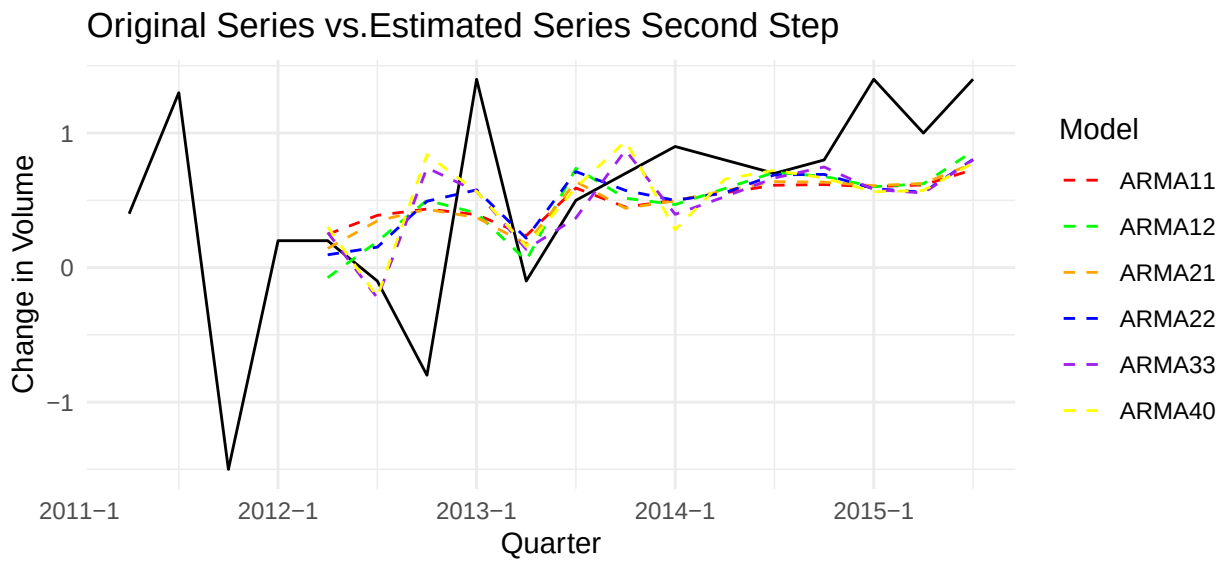

Figure 9: PACF reduced sample

3 Appendix: Code

```
# This chunk sets echo = TRUE as default, that is, print all code.
# knitr::opts_chunk$set can be used to set other notebook generation options, too.
# include=FALSE inside curly brackets makes this block not be included in the pdf.
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(fig.pos = "!H", out.extra = "")
library(pxweb)
library(zoo)
library(ggplot2)
library(tidyverse)
library(forecast)
library(tibbletime)
library(anomalize)
library(timetk)
library(tseries)
library(kableExtra)
#Install packages

# Define the API endpoint URL
url <- "https://api.scb.se/OV0104/v1/doris/en/ssd/START/NR/NR0103/NR0103B/NR0103ENS2010T10SKv"

d <- pxweb_interactive("https://api.scb.se/OV0104/v1/doris/en/ssd/START/NR/NR0103/NR0103B/NR0103ENS2010

df <- tibble(d["data"]$data) %>%
  rename(change_in_vol =
           `Seasonally adjusted, change in volume, previous period, percent`
           ) %>% mutate(quarter = as.yearqtr(quarter, format = "%YK%q")) %>%
  drop_na(change_in_vol) %>% select(quarter, change_in_vol)

df <- read_csv("time_series.csv") %>%
  rename(change_in_vol =
           `Seasonally adjusted, change in volume, previous period, percent`
           ) %>% mutate(quarter = as.yearqtr(quarter, format = "%YK%q")) %>%
  drop_na(change_in_vol) %>% select(quarter, change_in_vol)
#plot the complete series

# Fit a linear regression model to  data
model <- lm(change_in_vol ~ time(quarter), data = df)

ggplot(data = df, aes(x = quarter, y = change_in_vol)) +
  geom_line() +
  labs(x = "Time horizon, quarters", y = "Change in Volume (Seasonally Adjusted GDP)") +
  ggtitle("Time Series of GDP Changes") +
  scale_x_yearqtr(n = 10, lim = c(as.yearqtr("1981-1"), max(df$quarter))) +
  scale_y_continuous(breaks = seq(min(df$change_in_vol), max(df$change_in_vol), by = 1)) +
  theme_classic() +
  geom_abline(intercept = coef(model)[1], slope = coef(model)[2], color = "blue")
adf.test(df$change_in_vol)
plot(acf(df$change_in_vol, plot=F)[1:30], main = "ACF Plot")
plot(pacf(df$change_in_vol, plot=F)[1:30], main = "PACF Plot")
df_anomalized <- df %>%
  time_decompose(change_in_vol, merge = TRUE) %>%
  anomalize(remainder) %>%
```

```r
  time_recompose()
df_anomalized %>% glimpse()

df_anomalized %>% plot_anomalies(ncol = 3, alpha_dots = 0.75)
df_filter <- df %>% filter(quarter<"2020Q1")
ggplot(data = df_filter, aes(x = quarter, y = change_in_vol)) +
  geom_line() +
  labs(x = "Time horizon, quarters", y = "Change in Volume (Seasonally Adjusted GDP)") +
  ggtitle("Time Series of GDP Changes") +
  scale_x_yearqtr(n = 10, lim = c(as.yearqtr("1981-1"), max(df_filter$quarter))) +
  scale_y_continuous(breaks = seq(min(df_filter$change_in_vol), max(df_filter$change_in_vol), by = 1)) +
  theme_classic()
plot(acf(df_filter$change_in_vol, plot=F)[1:30], main = "ACF Plot")
plot(pacf(df_filter$change_in_vol, plot=F)[1:30], main = "PACF Plot")
# select auto.arima
auto.arima(df_filter$change_in_vol, max.p = 4, max.q = 4)


# Specify the evaluation period
start_date <- as.yearqtr("2011Q4")
end_date <- as.yearqtr("2015Q2")

# Initialize storage for evaluation metrics
evaluation_results_1 <- data.frame(
  Model = character(),
  quarter = numeric(),
  estimate = numeric(),
  Step = numeric()
)

evaluation_results_2 <- data.frame(
  Model = character(),
  quarter = numeric(),
  estimate = numeric(),
  Step = numeric()
)

# Define the ARMA models
models <- list(
  ARMA11 = c(1, 0, 1),  # ARMA(1, 1)
  ARMA21 = c(2, 0, 1),  # ARMA(2, 1)
  ARMA12 = c(1, 0, 2),  # ARMA(1, 2)
  ARMA22 = c(2, 0, 2),  # ARMA(2, 2)
  ARMA40 = c(4, 0, 0),  # ARMA(4, 0),
  ARMA33 = c(3, 0, 3)   # ARMA(3, 3)
)


# Loop through the evaluation period
current_date <- start_date
while (current_date < end_date) {
  for (model_name in names(models)) {
    # Subset data up to the current date
```

```
    subset_data <- df_filter %>%
      filter(quarter <= current_date)


    # Estimate the ARMA model
    model <- Arima(subset_data$change_in_vol, order = models[[model_name]])

    # Make one-step and two-step predictions
    one_step_forecast <- forecast(model, h = 1)
    two_step_forecast <- forecast(model, h = 2)

    # Extract the forecasts
    one_step_forecast_value <- one_step_forecast$mean[1]
    two_step_forecast_value <- two_step_forecast$mean[2]

    # Create two dfs to store forecasts
    add_date <- as.yearqtr(current_date+1/4)
    add_date_2 <- as.yearqtr(current_date+2/4)

    evaluation_results_1 <- rbind(evaluation_results_1,
                                  data.frame(
                                    Model = model_name,
                                    quarter = add_date,
                                    estimate = one_step_forecast_value,
                                    Step = 1
                                  ))

    evaluation_results_2 <- rbind(evaluation_results_2,
                                  data.frame(
                                    Model = model_name,
                                    quarter = add_date_2,
                                    estimate = two_step_forecast_value,
                                    Step = 2
                                  ))
  }
  # Move to the next quarter
  current_date <- current_date + 1/4
}

evaluation_results_1 %>% group_by(Model) %>% count()
evaluation_results_2 %>% group_by(Model) %>% count()


results_1 <- left_join(evaluation_results_1, df, by = "quarter") %>%
  mutate(
    rmse = (((change_in_vol - estimate)^2)/14),
    mae = (abs(change_in_vol - estimate))/14,
    bias = (change_in_vol - estimate)/14
  ) %>%
  group_by(Model) %>%
  summarize(
    RMSE = sqrt(sum(rmse)),
    MAD = sum(mae),
```

```
    Bias = sum(bias)
  )

results_2 <- left_join(evaluation_results_2, df, by = "quarter") %>%
  filter(quarter<'2015 Q3') %>%
  mutate(
    rmse = (((change_in_vol - estimate)^2)/13),
    mae = (abs(change_in_vol - estimate))/13,
    bias = (change_in_vol - estimate)/13
  ) %>% group_by(Model) %>%
  summarize(
    RMSE = sqrt(sum(rmse)),
    MAD = sum(mae),
    Bias = sum(bias)
  )
knitr::kable(results_1, format = "latex", booktabs = TRUE, align = "c",
             linesep = c("", "", "\\addlinespace"),
             caption = "model performance one-step predictions") %>%
  kable_styling(latex_options = "HOLD_position")
knitr::kable(results_2, format = "latex", booktabs = TRUE, align = "c",
             linesep = c("", "", "\\addlinespace"),
             caption = "model performance one-step predictions") %>%
  kable_styling(latex_options = "HOLD_position")
df_filter_2 <- df_filter %>% filter(quarter>"2011Q1" & quarter<"2015Q4")


ggplot(df_filter_2, aes(x = quarter, y = change_in_vol)) +
  geom_line(color = "black") +
  labs(x = "Quarter", y = "Change in Volume",
       title = "Original Series vs.Estimated Series First Step") +


  geom_line(data = evaluation_results_1,
            aes(x = quarter, y = estimate, color = Model, linetype = Model)) +


  scale_color_manual(values = c("ARMA11" = "red",
                                "ARMA12" = "green",
                                "ARMA21" = "orange",
                                "ARMA22"= "blue",
                                "ARMA40" = "yellow",
                                "ARMA33" = "purple")) +
  scale_linetype_manual(values = c("ARMA11" = "dashed",
                                   "ARMA12" = "dashed",
                                   "ARMA21" = "dashed",
                                   "ARMA22" = "dashed",
                                   "ARMA40" = "dashed",
                                   "ARMA33" = "dashed")) +


  labs(color = "Model") +
```

```
  theme_minimal()
df_filter_2 <- df_filter %>% filter(quarter>"2011Q1" & quarter<"2015Q4")


ggplot(df_filter_2, aes(x = quarter, y = change_in_vol)) +
  geom_line(color = "black") +
  labs(x = "Quarter", y = "Change in Volume",
       title = "Original Series vs.Estimated Series Second Step") +


  geom_line(data = evaluation_results_2, aes(x = quarter, y = estimate,
                                             color = Model,
                                             linetype = Model)) +


  scale_color_manual(values = c("ARMA11" = "red",
                                "ARMA12" = "green",
                                "ARMA21" = "orange",
                                "ARMA22"= "blue",
                                "ARMA40" = "yellow",
                                "ARMA33" = "purple")) +
  scale_linetype_manual(values = c("ARMA11" = "dashed",
                                   "ARMA12" = "dashed",
                                   "ARMA21" = "dashed",
                                   "ARMA22" = "dashed",
                                   "ARMA40" = "dashed",
                                   "ARMA33" = "dashed")) +


  labs(color = "Model") +


  theme_minimal()
```