W271 Assignment 8

```
library(tidyverse)
library(magrittr)
library(patchwork)

library(lubridate)

library(tsibble)
library(feasts)
library(forecast)

library(sandwich)
library(lmtest)

library(nycflights13)
library(blsR)

library(fable)
library(fabletools)
library(slider)
```

```
theme_set(theme_minimal())
```

(14 points total) Question-1: Is Unemployment an Autoregressive or a Moving Average Process?

You did work in a previous homework to produce a data pipeline that pulled the unemployment rate from official BLS sources. Reuse that pipeline to answer this final question in the homework:

> "Are unemployment claims in the US an autoregressive or a moving average process?

(1 point) Part-1: Why is the distinction important?

Why is it important to know whether a process is a AR or an MA (or a combination of the two) process? What changes in the ways that you would talk about the process, what changes in the ways that you would fit a model to the process, and what changes with how you would produce a forecast for this process?

Q1P1 Answer

Knowing whether a process is AR, MA, or a mix matters because it defines how persistence and shocks drive the series and how we model and forecast it.

Interpretation:

- AR: current values depend on past values (persistence).
- MA: current values depend on past shocks (transitory effects).

Forecasting:AR forecasts rely on past predicted values; MA forecasts decay faster as past shocks fade. Correctly identifying the process ensures accurate model specification and reliable forecasts.

(1 point) Part-2: Pull in (and clean up) your data pipeline.

In the previous homework, you built a data pipeline to draw data from the BLS. We are asking you to re-use, and if you think it is possible, to improve the code that you wrote for this pipeline in the previous homework.

- Are there places where you took "shortcuts" that could be more fully developed?
- Are the processes that could be made more modular, or better documented so that they are easier for you to understand what they are doing? You've been away from the code that you wrote for a few weeks, and so it might feel like "discovering" the code of a *mad-person* (Who even wrote this???)

Answer

For data pipeline (not in HW 6 pipeline):

- I added logic (if else statement) to check if API key is provided.
- After examining the time trend of male and female time trend, I kept the overall trend only for later tasks.
- Last time I ran into "reaching API request limit" problem. This time I saved the raw query on local drive then read from local drive when I re-knit (for many times).

```
huibin_bls_key <- "e943ccb8989f4599bd7964149a31457d"
bls_set_key(huibin_bls_key)

if (bls_has_key()) {
  print("Key provided. OK.")
} else {
  print("BLS API key not provided")
}
```

```
## [1] "Key provided. OK."
```

```
# Specify series IDs
series_id_list <- c(
  overall = "LNS14000000",
  male    = "LNS14000001",
  female  = "LNS14000002"
)

current_year <- as.numeric(format(Sys.Date(), "%Y"))
current_year
```

```
## [1] 2025
```

```
# The block below queries the data, and was commented out after the results
# saved locally.

#raw_bls_data <- get_n_series_table(
#  series_ids = series_id_list,
#  api_key = bls_get_key(),
#  start_year = current_year - 20,
#  end_year = current_year
#)

# Saved the query results so that I don't use up my api quota
#saveRDS(raw_bls_data, file = "data/my_api_results.Rds")

raw_bls_data <- readRDS("data/my_api_results.Rds")
```

```r
# bls_unset_key()

head(raw_bls_data)
```

```
## # A tibble: 6 x 5
##    year period LNS14000000 LNS14000001 LNS14000002
##   <int> <chr>        <dbl>       <dbl>       <dbl>
## 1  2005 M01            5.3         5.4         5.1
## 2  2005 M02            5.4         5.5         5.3
## 3  2005 M03            5.2         5.3         5.1
## 4  2005 M04            5.2         5.1         5.2
## 5  2005 M05            5.1         5           5.2
## 6  2005 M06            5           5           5.1
```

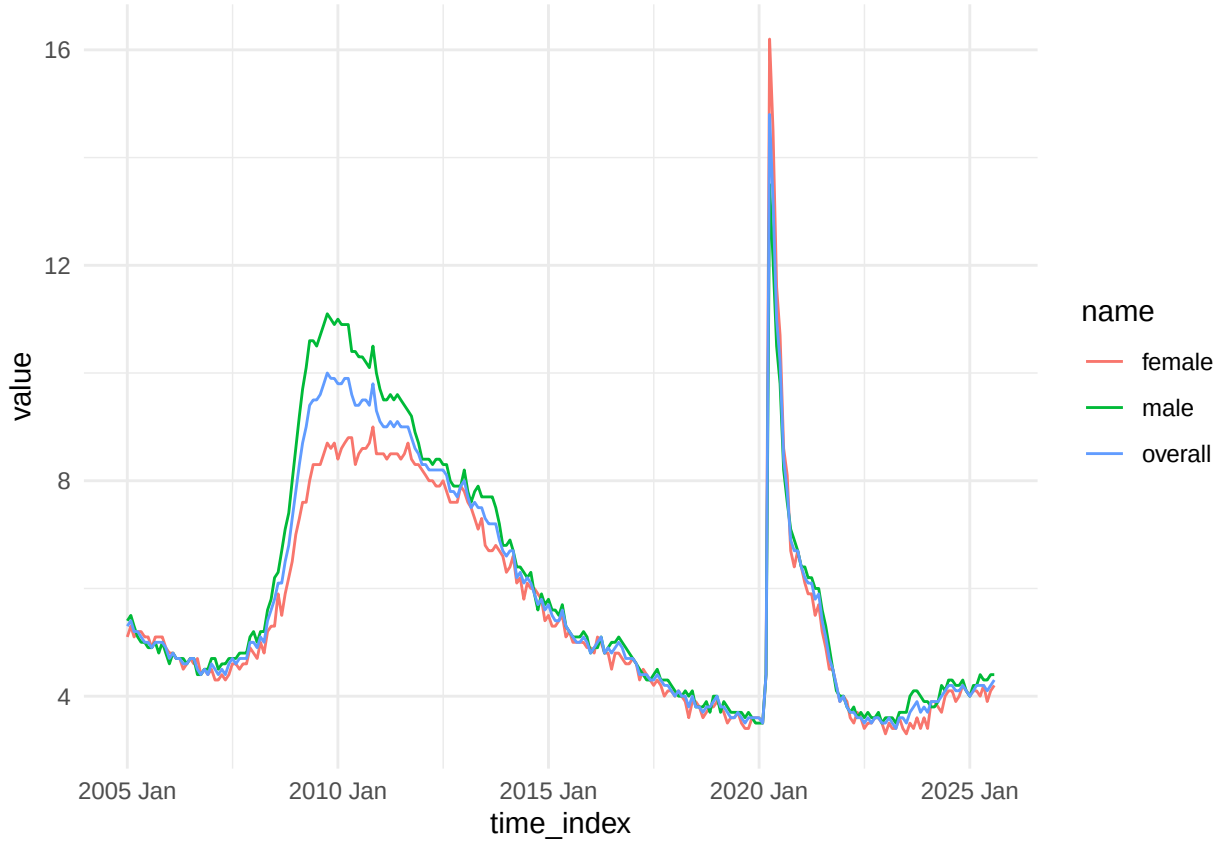
```r
unemployment <-
  raw_bls_data %>%
  pivot_longer(
    cols = starts_with("LNS"),
    names_to = "series_id",
    values_to = "value"
  ) %>%
  mutate(
    date = lubridate::ymd(paste0(year,
                                 "-",
                                 gsub(pattern = "M",
                                      replacement = "",
                                      x = period),
                                 "-01"
                                 )),
    name = case_when(
      series_id == "LNS14000000" ~ "overall",
      series_id == "LNS14000001" ~ "male",
      series_id == "LNS14000002" ~ "female",
      TRUE ~ series_id
                     )
  ) %>%
  mutate(
    year = year(date),
    month = month(date),
    time_index = yearmonth(date)
  ) %>%
  as_tsibble(
    key = name,
    index = time_index
  )

unemployment
```

```
## # A tsibble: 744 x 8 [1M]
## # Key:       name [3]
##     year period series_id   value date       name   month time_index
##    <dbl> <chr>  <chr>       <dbl> <date>     <chr>  <dbl>      <mth>
##  1  2005 M01    LNS14000002   5.1 2005-01-01 female     1   2005 Jan
##  2  2005 M02    LNS14000002   5.3 2005-02-01 female     2   2005 Feb
```

```
##  3  2005 M03    LNS14000002   5.1 2005-03-01 female     3   2005 Mar
##  4  2005 M04    LNS14000002   5.2 2005-04-01 female     4   2005 Apr
##  5  2005 M05    LNS14000002   5.2 2005-05-01 female     5   2005 May
##  6  2005 M06    LNS14000002   5.1 2005-06-01 female     6   2005 Jun
##  7  2005 M07    LNS14000002   5.1 2005-07-01 female     7   2005 Jul
##  8  2005 M08    LNS14000002   4.9 2005-08-01 female     8   2005 Aug
##  9  2005 M09    LNS14000002   5.1 2005-09-01 female     9   2005 Sep
## 10  2005 M10    LNS14000002   5.1 2005-10-01 female    10   2005 Oct
## # i 734 more rows
```

```
unemployment %>%
  ggplot(aes(x = time_index, y = value, color = name)) +
  geom_line(line_width = 1)
```

```
## Warning in geom_line(line_width = 1): Ignoring unknown parameters: `line_width`
```




```
# After examining all three, which are similar, I keep the overall unemployment rate
unemployment <-
  unemployment %>%
  filter(name == "overall")
```

(5 points) Part-3: Conduct an EDA of the data and comment on what you see.

We have presented four **core** plots that are a part of the EDA for time-serires data. Produce each of these plots, and comment on what you see.

Answer

1. Time plot

Unemployment shows some cyclical movements tied to business cycles, with spikes during recessions (e.g., 2008, 2020). Unemployment rises sharply in downturns but gradually returns toward a long-run average.

2. ACF plot

The autocorrelation decays slowly over many lags, suggesting strong persistence — each month's unemployment rate is correlated with many months prior. This indicates an autoregressive structure.

3. Distribution plot

The histogram is not bell-shaped (right-skewed), reflecting periods of high unemployment during recessions. Outliers are present (subprime, Covid-19).

4. PACF plot

The PACF cuts off after lag 1, while later lags are small and insignificant. This is a sign of an AR(1) process.

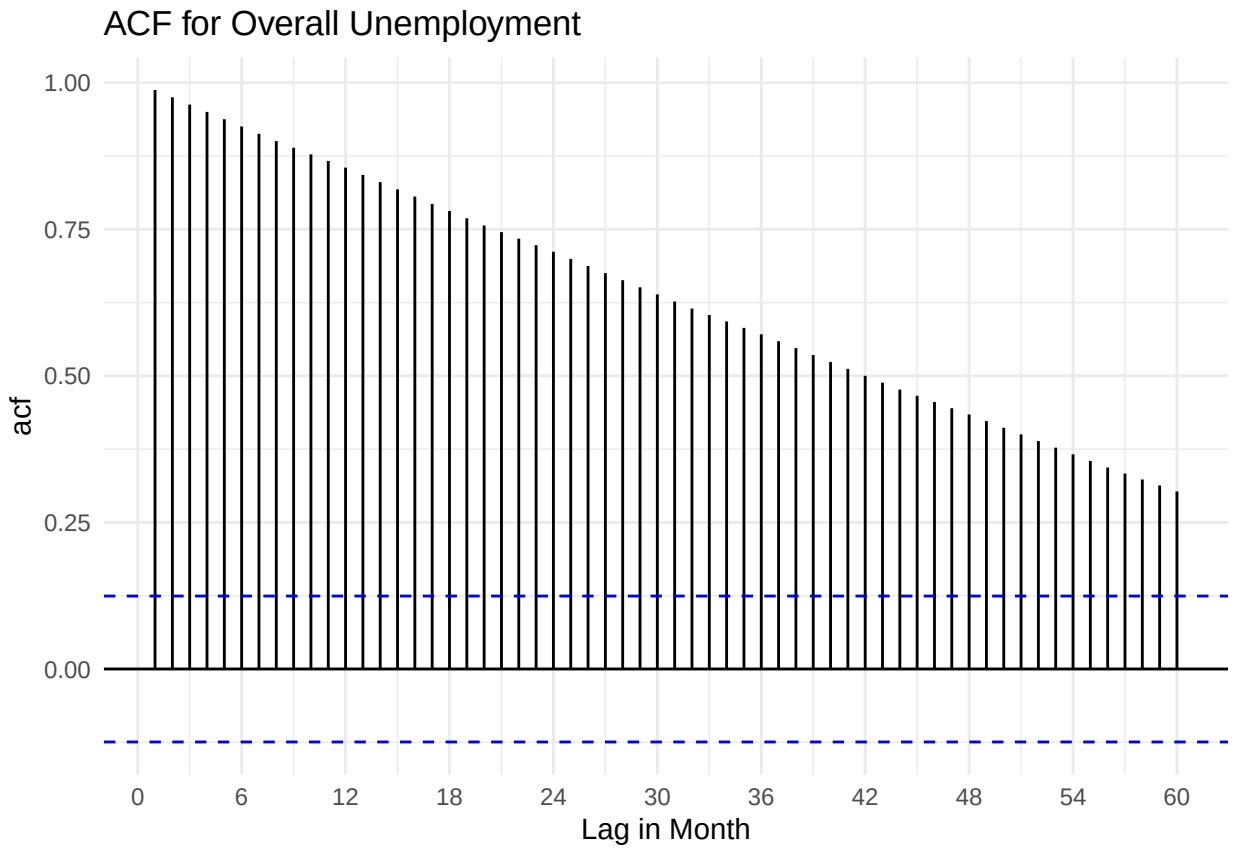
Summary:

The EDA supports modeling unemployment as a stationary AR(1) process — strongly persistent but mean-reverting, with no major structural breaks or variance shifts.

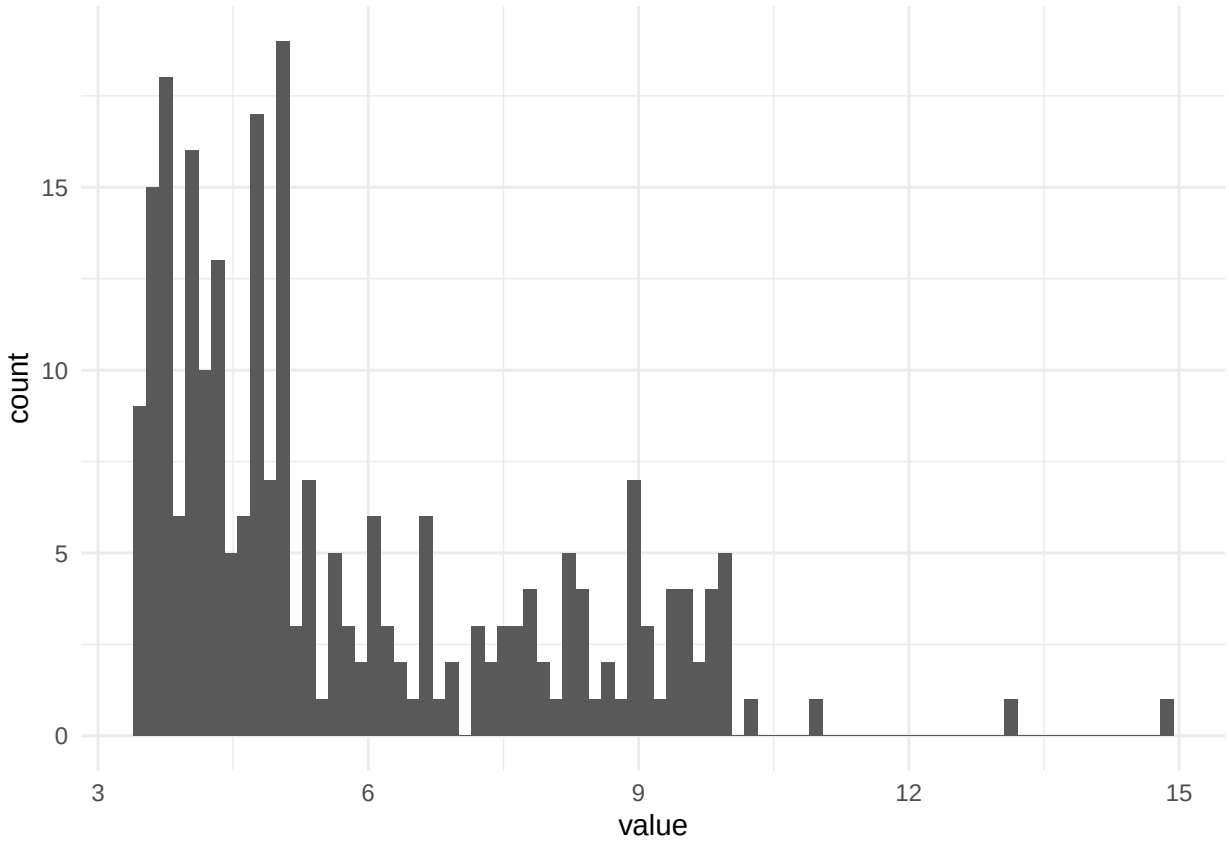
```
#Plot the time trend
unemployment %>%
  ggplot(aes(x = time_index, y = value, color = name)) +
  geom_line(linewidth = 1) +
  labs(
    title = "2005 - 2025 Unemployment by Gender",
    x = "Time",
    color = "Series"
  )
```


```
# Commented out another way of quick time trend
#unemployment %>%
#  autoplot(value)

# Second: ACF Plot
unemployment %>%
#  filter(name == "overall") %>%
  ACF(
    lag_max= 60
  ) %>%
  autoplot() +
  labs(
    title = "ACF for Overall Unemployment",
    x = "Lag in Month"
  )
```

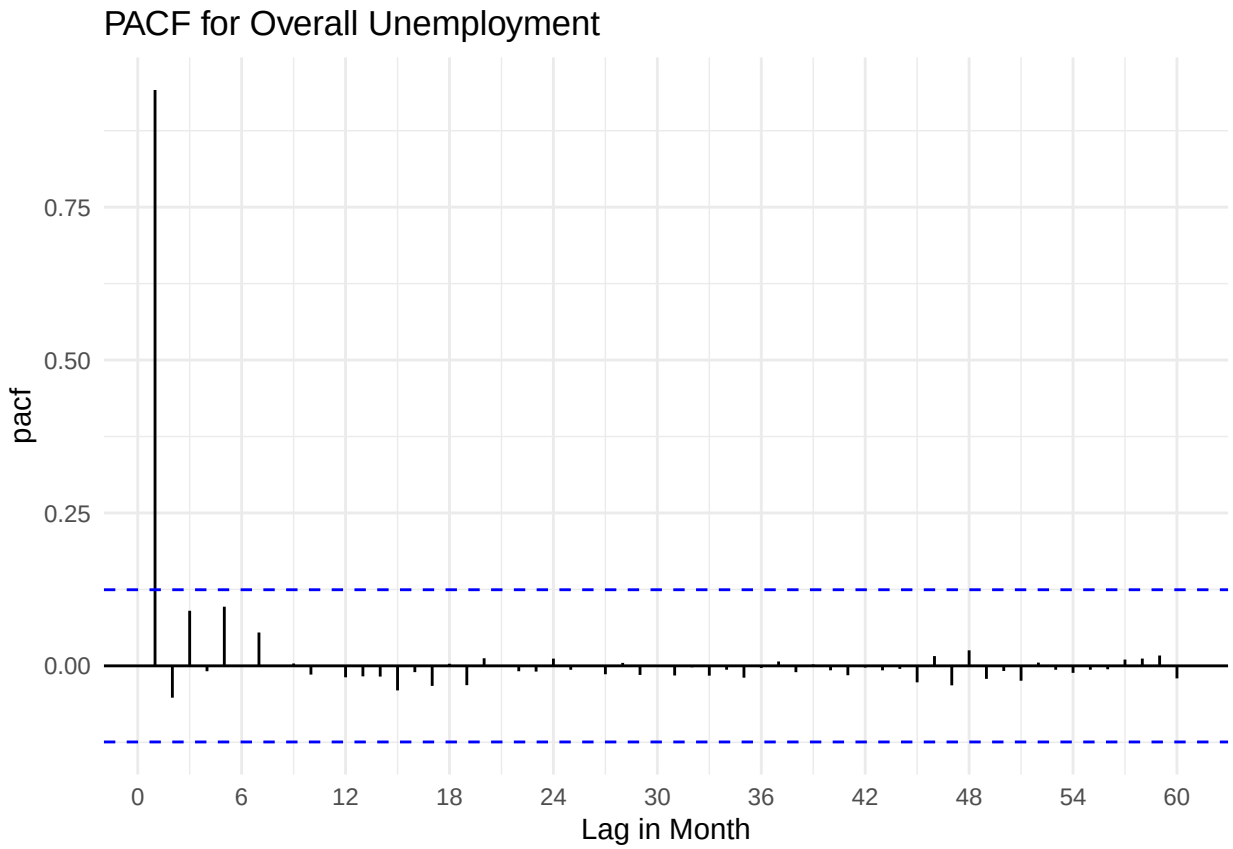
```
## Response variable not specified, automatically selected `var = year`
```




```
# Third: Distribution
unemployment %>%
#  filter(name == "overall") %>%
  ggplot(aes(x = value)) +
  geom_histogram(bins = 80)
```




```
# Forth: PACF
unemployment %>%
  filter(name == "overall") %>%
  PACF(value, lag_max = 60) %>%
  autoplot() +
  labs(title = "PACF for Overall Unemployment",
       x = "Lag in Month")
```


PACF for Overall Unemployment
pacf
0.75
0.50
0.25
0.00
0
6
12
18
24
30
36
42
48
54
60
Lag in Month

(1 point) Part-4: Make a Call

Based on what you have plotted and written down in the previous section, would you say that the unemployment rate is an AR, MA or a mix of the two?

Answer

The unemployment rate time series, according to the EDA, particularly the ACF and the PACF, resembles as an AR(1) process. Specifically, autocorrelation decays gradually, PACF cuts off sharply after lag 1 — meaning each month's unemployment rate is strongly predicted by the previous month's rate.

Also I am using my knowledge from macroeconomics that long-run unemployment should be, in theory, a constant, or the natural rate unemployment rate under potential output. This favors the assumption that the time series is a stationary process.

(6 points total) Part-5: Estimate a model

Report the best-fitting parameters from the best-fitting model, and then describe what your model is telling you. In this description, you should:

- (1 point) State, and justify your model selection criteria.

Answer

The AR() model class is selected based on the EDA in the previous part. Then the order is selected to have a low aic and at the same time avoiding overfitting (e.g., order 25).

As shown in the async and the text book, and using the ar() function similarly to the async, the AR(1) model was selected as the best-fitting model because it had the lowest AIC (0 in the normalized output) among all tested orders (p=0 to p=12).

It is worth-noting that the ar() method with the default optimization yields the lowest AIC at lag 25, but the function selects AR(1). AIC, which doesn't penalize overfit as severely as BIC, is the lowest at lag 25 may lead to overfitting.

```
unemployment
```

```
## # A tsibble: 248 x 8 [1M]
## # Key:       name [1]
##     year period series_id   value date       name    month time_index
##    <dbl> <chr>  <chr>       <dbl> <date>     <chr>   <dbl>      <mth>
##  1  2005 M01    LNS14000000   5.3 2005-01-01 overall     1   2005 Jan
##  2  2005 M02    LNS14000000   5.4 2005-02-01 overall     2   2005 Feb
##  3  2005 M03    LNS14000000   5.2 2005-03-01 overall     3   2005 Mar
##  4  2005 M04    LNS14000000   5.2 2005-04-01 overall     4   2005 Apr
##  5  2005 M05    LNS14000000   5.1 2005-05-01 overall     5   2005 May
##  6  2005 M06    LNS14000000   5   2005-06-01 overall     6   2005 Jun
##  7  2005 M07    LNS14000000   5   2005-07-01 overall     7   2005 Jul
##  8  2005 M08    LNS14000000   4.9 2005-08-01 overall     8   2005 Aug
##  9  2005 M09    LNS14000000   5   2005-09-01 overall     9   2005 Sep
## 10  2005 M10    LNS14000000   5   2005-10-01 overall    10   2005 Oct
## # i 238 more rows
```

```
model <-
  unemployment %>%
  pull(value) %>%
  ar(aic = TRUE, bic = TRUE, order.max = 12, method = "yule-walker")

#model$order
#model$aic
#model$ar
#sqrt(model$asy.var.coef)
#model$asy.var
summary(model)
```

```
##              Length Class  Mode
## order          1    -none- numeric
## ar             1    -none- numeric
## var.pred       1    -none- numeric
## x.mean         1    -none- numeric
## aic           13    -none- numeric
## n.used         1    -none- numeric
## n.obs          1    -none- numeric
```

```
## order.max      1    -none- numeric
## partialacf    12    -none- numeric
## resid        248    -none- numeric
## method         1    -none- character
## series         1    -none- character
## frequency      1    -none- numeric
## call           6    -none- call
## asy.var.coef   1    -none- numeric
```

```
#model$bic
#model$resid


# Extract and compare AICc values
#model_comparison <- glance(fit_ar_models) %>%
#  select(.model, AICc, BIC) %>%
#  arrange(AICc)

#model_comparison
```

- (1 point) Interpret the model selection criteria in context of the other models that you also fitted.

Answer

Interpretation: The AR(1) model is the simplest adequate model (compared to higher orders with low AIC values) for the unemployment rate under the assumption of stationarity. It suggests that, according to the ar() method, the predictive power of the unemployment time series is captured by its immediate prior value y(t−1), and adding more lagged terms y(t−2), y(t−3) and so on does not improve the fit enough to justify the increased complexity (parameter count). This result is consistent with the sharp cutoff seen at Lag 1 in the PACF plot.

- (2 points) Interpret the coefficients of the model that you have estimated.

Answer

Interpretation:

AR(1) Coefficient (ϕ_1=0.9416): This coefficient is highly significant (since 0.9416 is many standard errors away from 0).

It measures persistence. A value of 0.9416 means that 94.16% of last month's unemployment rate persists into the current month. The current unemployment rate is strongly driven by the rate in the immediately preceding period. This high value confirms the slow, smooth movement observed in the time plot and ACF.

- (2 points) Produce and interpret the model diagnostic plots to evaluate how well your best-fitting model is performing.

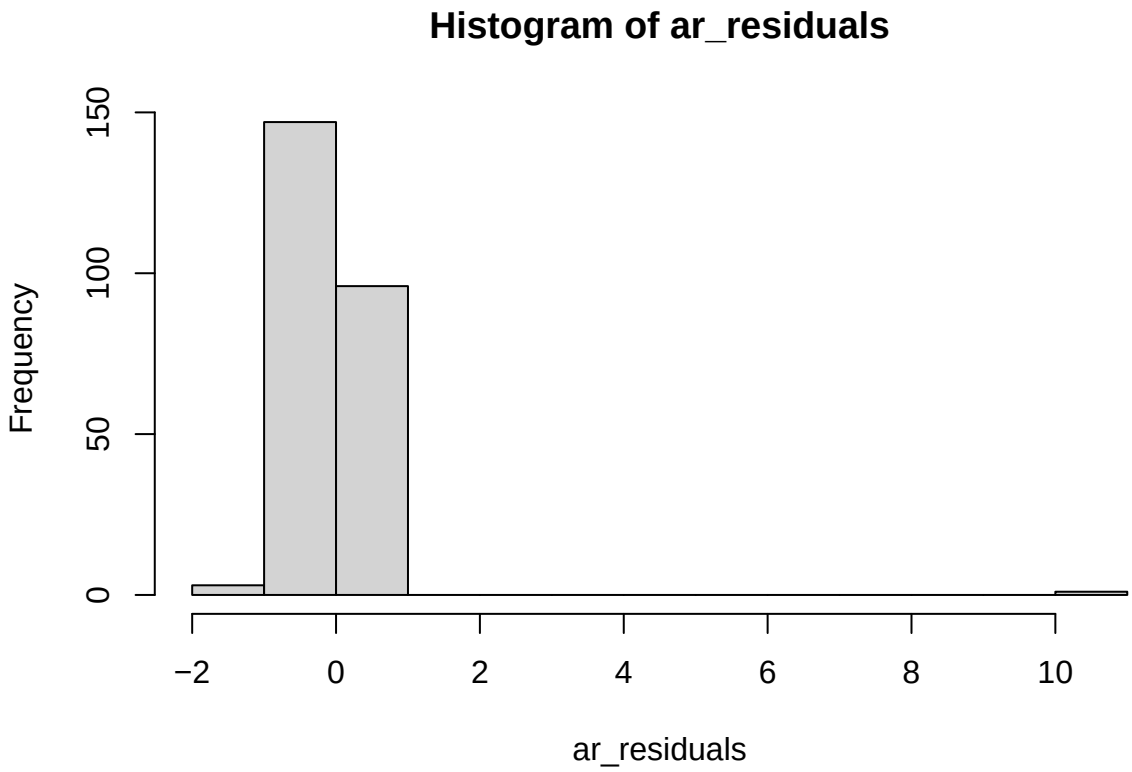
Answer

Ljung-Box test results and the ACF graph of the residuals are shown below in the code chunk.

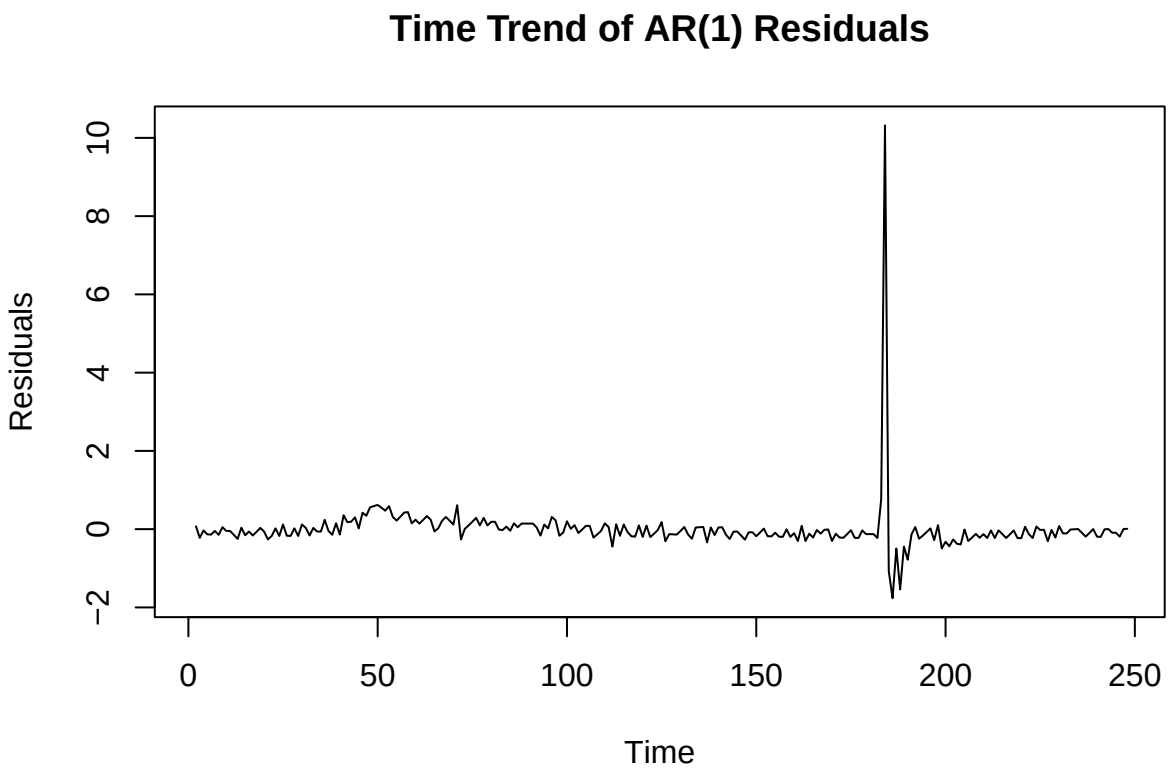
With Ljung-Box test, the p-value is 0.95 and we fail to reject the null hypothesis that the residuals are white noise. This confirms the visual evidence from the ACF plot that the AR(1) model is statistically adequate and fits the data well under the assumption of stationarity.

The ACF plot shows no significant spikes at any lag (Lags 1 through 20 are all within the blue dashed confidence intervals). The AR(1) model has successfully captured the linear autocorrelation in the original unemployment series. The residuals appear to be a sequence of white noise.

```
ar_residuals <- model$resid

hist(ar_residuals)
```



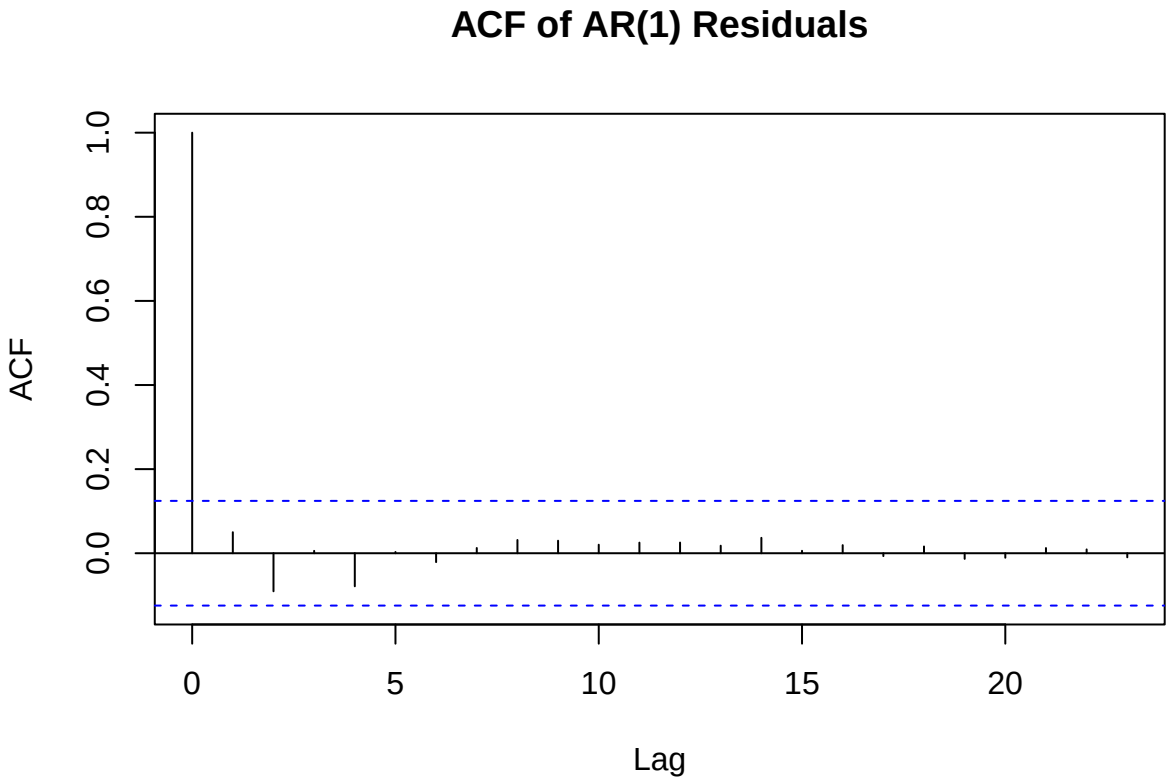

```
plot.ts(ar_residuals,
        main = "Time Trend of AR(1) Residuals",
        ylab = "Residuals",
        xlab = "Time")
```




```
Box.test(ar_residuals, lag = 12, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ar_residuals
## X-squared = 5.3367, df = 12, p-value = 0.9458
```

```
acf(ar_residuals,
    na.action = na.pass,
    main = "ACF of AR(1) Residuals")
```


ACF of AR(1) Residuals
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
Lag

- (1 (optional) point) If, after fitting the models, and interpreting their diagnostics plots, you determine that the model is doing poorly – for example, you notice that the residuals are not following a white-noise process – then, make a note of the initial model that you fitted then propose a change to the data or the model in order to make the model fit better. If you take this action, you should focus your interpretation of the model's coefficients on the model that you think does the best job, which might be the model after some form of variable transformation.

Answer

Diagnostics from the previous part shows that the model is fitting well.

(14 Points Total) Question-2: Forecasting Inflation

The Federal Reserve tracks inflation data across countries on a monthly level.

(1 point) Part-1: Load and Clean Data

Load the CSV provided and store the data in a useful dataframe. Check for missing values and outliers in the data. Perform any cleaning that is necessary.

Also create lagged columns for GBR and CAN and training and test datasets based on pre and post Jan 1, 2022.

Answer

The data is loaded and "sum(is.na(raw_inflation))" returns 0, which indicates no missing values. Initially the tsibble is created in long form (USA, CAN, GBR), then changed to wide form for later tasks.

```
raw_inflation <- read_csv("data/inflation_country.csv")
```

```
## Rows: 632 Columns: 26
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (25): AUT, BEL, CAN, CHE, CHL, DEU, DNK, ESP, FIN, FRA, GBR, GRC, ISL, ...
## date  (1): date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dim(raw_inflation)
```

```
## [1] 632  26
```

```
sum(is.na(raw_inflation))
```

```
## [1] 0
```

```
#raw_inflation

# raw_inflation %>%
#   select(date, USA, GBR, CAN) %>%
#   mutate(
#     date = as_date(date),
#     GBR_lag1 = lag(GBR, 1),
#     CAN_lag1 = lag(CAN, 1)
#   ) %>%
#   pivot_longer(
#     cols = -date,
#     names_to = "country",
#     values_to = "value"
#   ) %>%
#   as_tsibble(
#     key = country,
#     index = date
#   )




# 1. Load the original wide data and select the required columns
inflation_arimax_data <-
  raw_inflation %>%
  select(date, USA, GBR, CAN) %>%
```

```
  mutate(date = yearmonth(date)) %>%
  # Convert to tsibble early for easy lag calculation
  as_tsibble(index = date) %>%

  # 2. Create the lagged variables DIRECTLY on the wide data
  mutate(
    GBR_lag1 = lag(GBR, 1),
    CAN_lag1 = lag(CAN, 1)
  )

# 3. Create Train and Test Datasets (Split at Jan 1, 2022)
split_date <- ymd("2022-01-01")

# Training data
train_data <-
  inflation_arimax_data %>%
  filter(date < split_date)
```

```
## Warning: There was 1 warning in `filter()`.
## i In argument: `date < split_date`.
## Caused by warning:
## ! Incompatible methods ("<.vctrs_vctr", "<.Date") for "<"
```

```
# Test data
test_data <-
  inflation_arimax_data %>%
  filter(date >= split_date)
```

```
## Warning: There was 1 warning in `filter()`.
## i In argument: `date >= split_date`.
## Caused by warning:
## ! Incompatible methods (">=.vctrs_vctr", ">=.Date") for ">="
```

```
train_data
```

```
## # A tsibble: 612 x 6 [1M]
##        date    USA    GBR     CAN GBR_lag1 CAN_lag1
##       <mth>  <dbl>  <dbl>   <dbl>    <dbl>    <dbl>
##  1 1971 Jan 0.0529 0.0849 0.00990  NA       NA
##  2 1971 Feb 0.0500 0.0852 0.0149    0.0849  0.00990
##  3 1971 Mar 0.0471 0.0876 0.0149    0.0852  0.0149
##  4 1971 Apr 0.0416 0.0942 0.0197    0.0876  0.0149
##  5 1971 May 0.0440 0.0982 0.0197    0.0942  0.0197
##  6 1971 Jun 0.0464 0.103  0.0246    0.0982  0.0197
##  7 1971 Jul 0.0436 0.101  0.0244    0.103   0.0246
##  8 1971 Aug 0.0462 0.103  0.0341    0.101   0.0244
##  9 1971 Sep 0.0408 0.0989 0.0343    0.103   0.0341
## 10 1971 Oct 0.0381 0.0937 0.0392    0.0989  0.0343
## # i 602 more rows
```

```
test_data
```

```
## # A tsibble: 20 x 6 [1M]
##        date    USA    GBR    CAN GBR_lag1 CAN_lag1
##       <mth>  <dbl>  <dbl>  <dbl>    <dbl>    <dbl>
##  1 2022 Jan 0.0748 0.0485 0.0514   0.0484   0.0480
```

```
##  2 2022 Feb 0.0787 0.0548 0.0569   0.0485   0.0514
##  3 2022 Mar 0.0854 0.0620 0.0666   0.0548   0.0569
##  4 2022 Apr 0.0826 0.0779 0.0677   0.0620   0.0666
##  5 2022 May 0.0858 0.0784 0.0773   0.0779   0.0677
##  6 2022 Jun 0.0906 0.0817 0.0813   0.0784   0.0773
##  7 2022 Jul 0.0852 0.0880 0.0759   0.0817   0.0813
##  8 2022 Aug 0.0826 0.0865 0.0701   0.0880   0.0759
##  9 2022 Sep 0.0820 0.0881 0.0686   0.0865   0.0701
## 10 2022 Oct 0.0775 0.0961 0.0688   0.0881   0.0686
## 11 2022 Nov 0.0711 0.0938 0.0680   0.0961   0.0688
## 12 2022 Dec 0.0645 0.0924 0.0632   0.0938   0.0680
## 13 2023 Jan 0.0641 0.0890 0.0592   0.0924   0.0632
## 14 2023 Feb 0.0604 0.0919 0.0525   0.0890   0.0592
## 15 2023 Mar 0.0498 0.0884 0.0430   0.0919   0.0525
## 16 2023 Apr 0.0493 0.0782 0.0441   0.0884   0.0430
## 17 2023 May 0.0405 0.0785 0.0336   0.0782   0.0441
## 18 2023 Jun 0.0297 0.0739 0.0281   0.0785   0.0336
## 19 2023 Jul 0.0318 0.0644 0.0327   0.0739   0.0281
## 20 2023 Aug 0.0318 0.0644 0.0327   0.0644   0.0327
```

```
# Display the dimensions to verify the split
print(paste("Train Data Rows:", nrow(train_data)))
```

```
## [1] "Train Data Rows: 612"
```

```
print(paste("Test Data Rows:", nrow(test_data)))
```

```
## [1] "Test Data Rows: 20"
```

```
# 4. Save the new train/test data (overwriting previous files)
#write_csv(train_data, "inflation_train_data.csv")
#write_csv(test_data, "inflation_test_data.csv")
```

(5 points) Part-2: Produce a Model on the Training Data

1. Select inflation data for the US.
2. Produce a 7 observation lag, backward smoother of inflation and plot the original time series with its smoothed trend.
3. Produce a time series model of inflation. This should include:

- Conducting a full EDA and description of the data that you observe.
- Estimating a model that you believe is appropriate after conducting your EDA.
- Evaluating the model performance through diagnostic plots and making any necessary adjustments to satisfy key assumptions.

Q2 Part2 step 1: data wrangling (creating smoother) and EDA.

As shown in the output of the code chunk below:

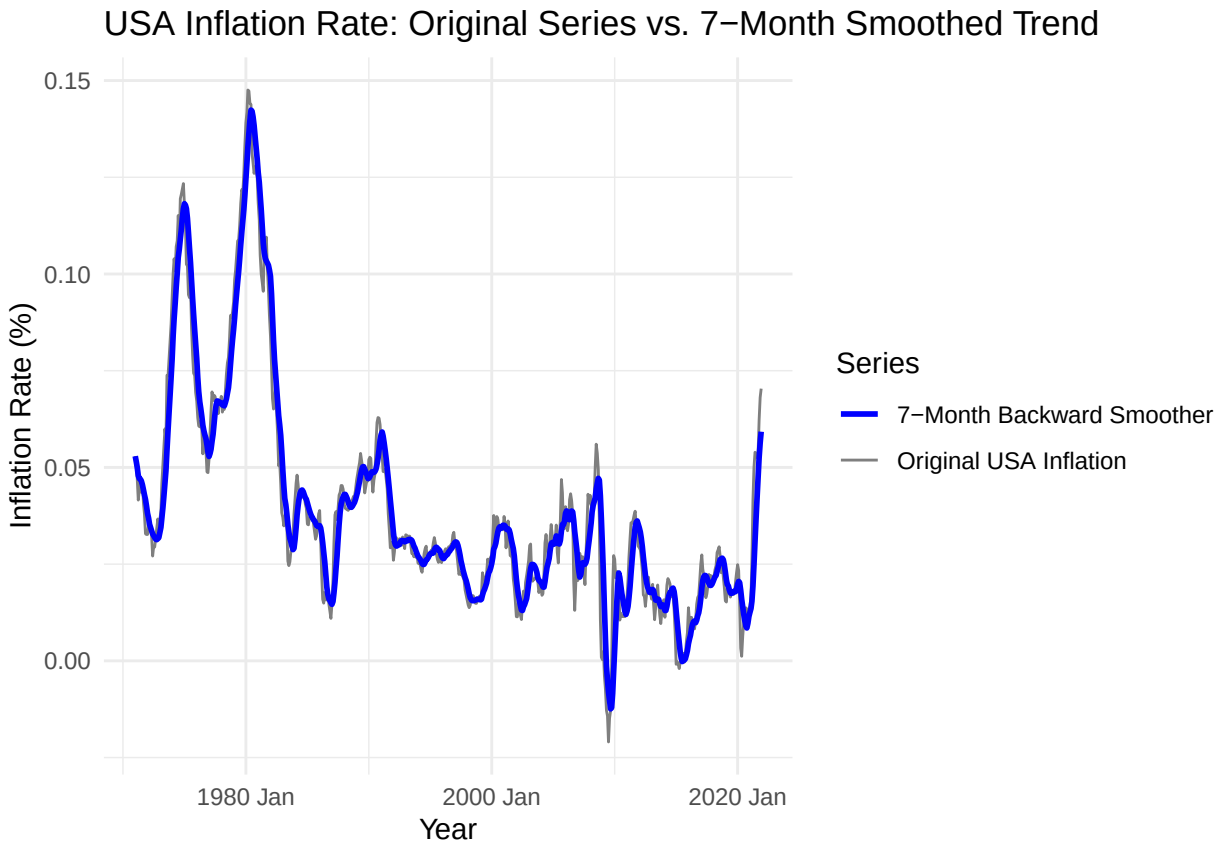
The ACF declines slowly and smoothly from 1 toward 0, taking many lags to taper off. This gradual decay indicates persistence: each month's inflation rate is highly correlated with many months prior. Such slow decay suggests an autoregressive (AR) component rather than an MA one.

PACF has a large spike at lag 1, followed by much smaller and statistically insignificant spikes. This pattern is the signof an AR(1) process.

Visual check (smoothed plot): Inflation has long cycles (multi-year swings) and persistent trends rather than sharp, short-run oscillations. This visually reinforces the conclusion that inflation is highly persistent and well captured by an AR model.

Given these diagnostics, I will start by estimating an AR(1) model for U.S. inflation using the training data, and then I'll test whether a higher-order AR (e.g., AR(2) or AR(3)) or an ARIMA with differencing gives any improvement.

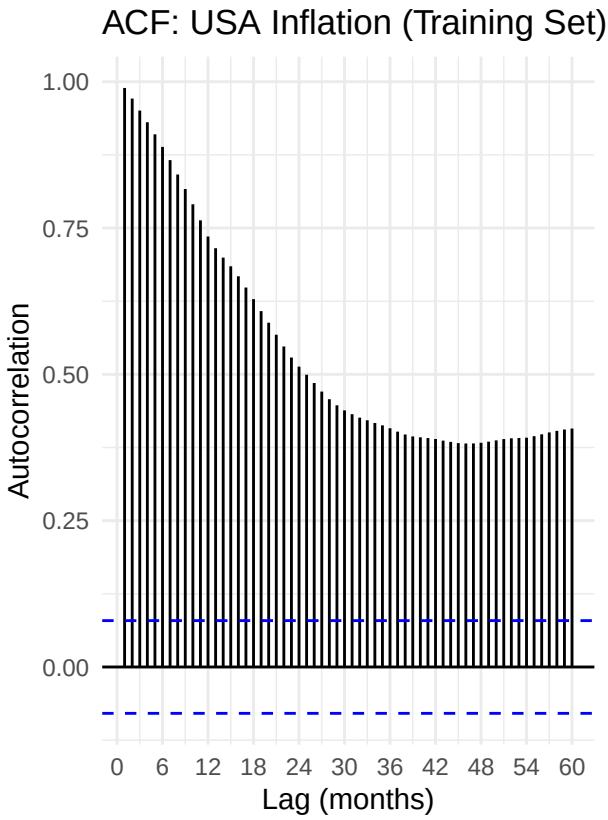
```
train_data_smoothed <-
  train_data %>%
  select(USA) %>%
  mutate(
    US_smoother = slide_dbl(
      .x = USA,
      .f = mean,
      .before = 6,
      .after = 0,
      .complete = FALSE
    )
  )

#class(train_data_smoothed)

smoother_plot <- train_data_smoothed %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = USA, color = "Original USA Inflation")) +
  geom_line(aes(y = US_smoother, color = "7-Month Backward Smoother"), linewidth = 1) +
  labs(
    title = "USA Inflation Rate: Original Series vs. 7-Month Smoothed Trend",
    y = "Inflation Rate (%)",
    x = "Year",
    color = "Series"
  ) +
  scale_color_manual(values = c("Original USA Inflation" = "gray50", "7-Month Backward Smoother" = "blu
  theme_minimal()

smoother_plot
```

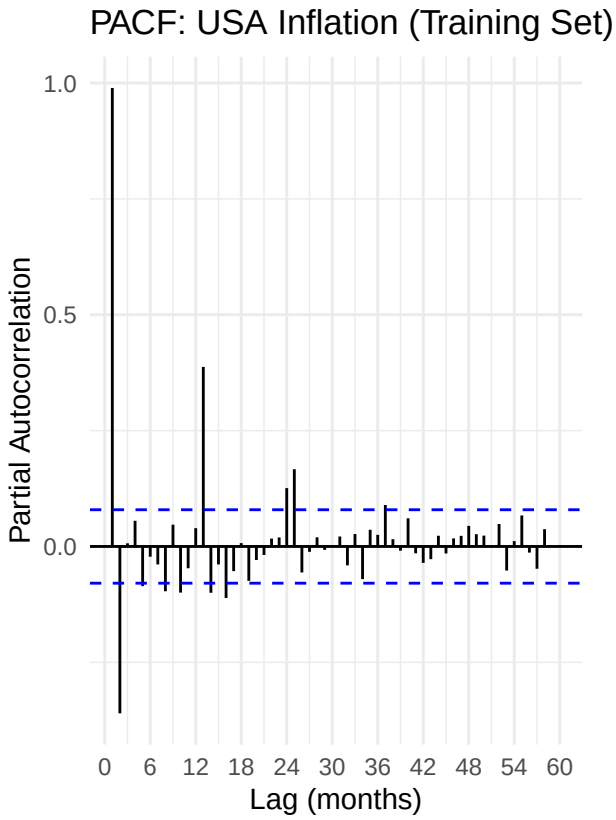



```
us_train <- train_data %>%
select(date, USA) %>%
as_tsibble(index = date)

sum(is.na(train_data$USA))
```

```
## [1] 0
```

```
acf_us <- us_train %>% ACF(USA, lag_max = 60)
pacf_us <- us_train %>% PACF(USA, lag_max = 60)

p_acf <- autoplot(acf_us) +
labs(title = "ACF: USA Inflation (Training Set)",
x = "Lag (months)", y = "Autocorrelation")

p_pacf <- autoplot(pacf_us) +
labs(title = "PACF: USA Inflation (Training Set)",
x = "Lag (months)", y = "Partial Autocorrelation")

p_acf + p_pacf
```


ACF: USA Inflation (Training Set)
Autocorrelation
1.00
0.75
0.50
0.25
0.00
0
6
12
18
24
30
36
42
48
54
60
Lag (months)


PACF: USA Inflation (Training Set)
Partial Autocorrelation
1.0
0.5
0.0
0
6
12
18
24
30
36
42
48
54
60
Lag (months)

Q2 Part 2 step 2

Estimating a model

The code block below shows the estimated model. The ar() selects AR(1). "order" in summary of the fitted model is 1, but "ar" in summary is 25. This is because order 25 gives the smallest aic (normalized to 0, same as the async).

With some googling and ChatGPTing around, the most plausible explanation is that sometimes with highly persistent data (like inflation), the yule-walker algorithm overfits at higher orders — it keeps adding lags until it "almost perfectly fits" the data, driving the AIC down.

```
usa_infl_ar1 <- train_data %>%
select(date, USA) %>%
  ar()

summary(usa_infl_ar1)
```

```
##              Length Class  Mode
## order          1    -none- numeric
## ar            25    -none- numeric
## var.pred       1    -none- numeric
## x.mean         1    -none- numeric
## aic           28    -none- numeric
## n.used         1    -none- numeric
## n.obs          1    -none- numeric
## order.max      1    -none- numeric
## partialacf    27    -none- numeric
## resid        612    -none- numeric
## method         1    -none- character
## series         1    -none- character
## frequency      1    -none- numeric
## call           2    -none- call
## asy.var.coef 625    -none- numeric
```

```
#usa_infl_ar1$aic
```

Q2 Part 2 step 3

Evaluating the model performance through diagnostic plots and making any necessary adjustments to satisfy key assumptions.

As the code block below shows:

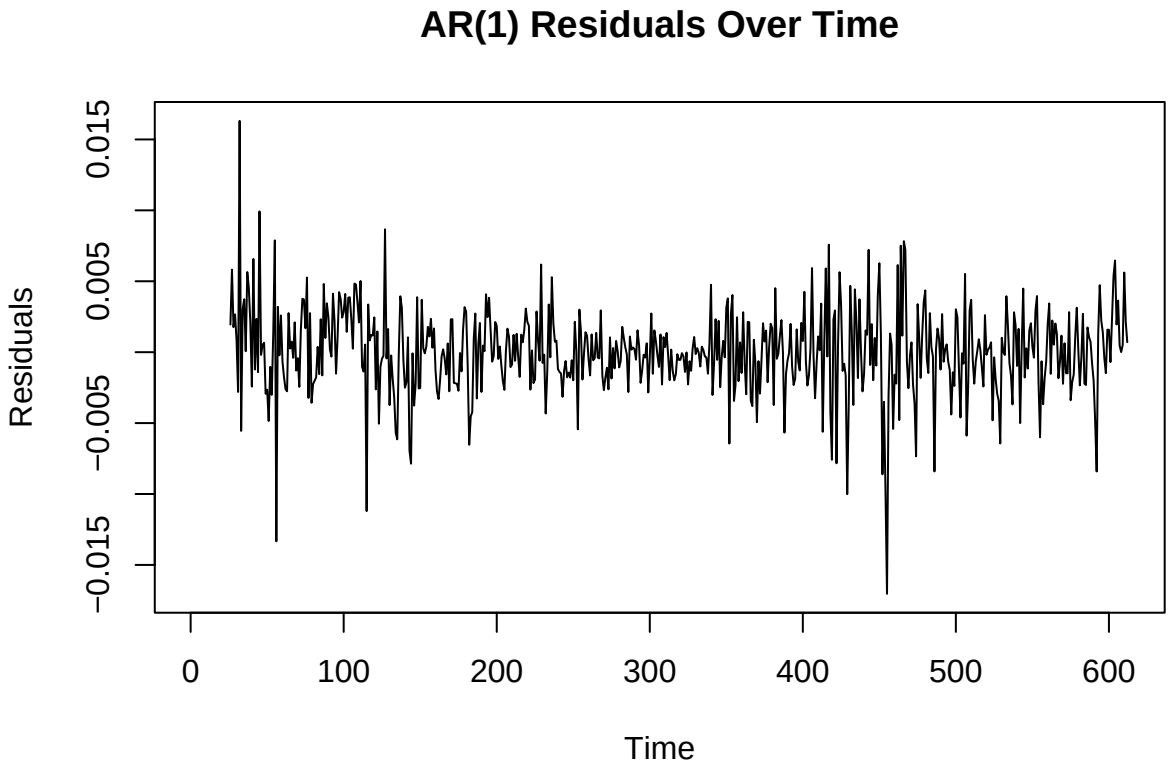
1. Residual time plot The residuals fluctuate randomly around zero without any visible trends or structural breaks. This indicates the mean equation is probably well specified.
2. Histogram Residuals are approximately bell-shaped and symmetric around zero. This suggests the normality assumption for AR(1) errors is likely reasonable.
3. ACF and PACF of residuals No spikes exceed the blue 95 % confidence bands beyond lag 0. This means no significant serial correlation remains visually.
4. Ljung–Box test p-value is $0.0004 < 0.05$, and we reject the null. Statistically, there is autocorrelation left in the residuals even though it's small visually.

Overall, this AR(1) model captures some of the serial dependence, but not all.

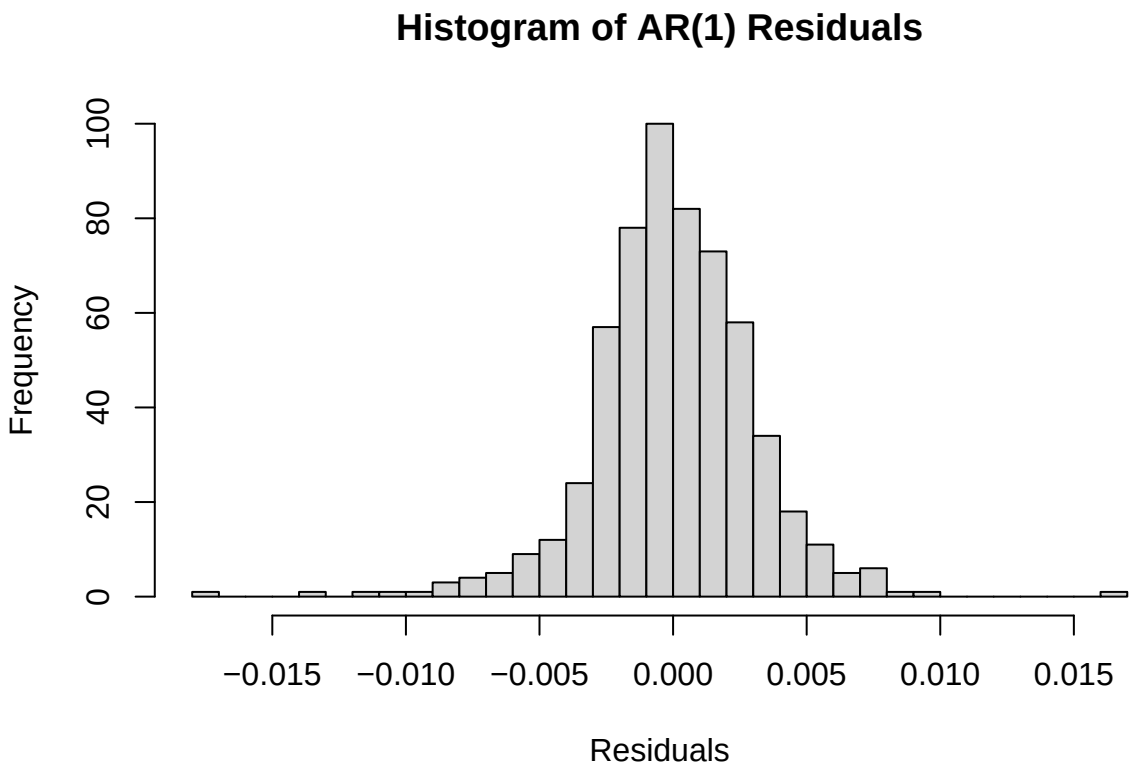
To address this, a natural adjustment is to allow for additional autoregressive lags (as shown in the second code chunk below) but the Ljung-Box test still comes back rejecting the null.

So, perhaps the stationary assumption or the white-noise assumption needs to be adjusted. I stopped here for Q2 Part 2.

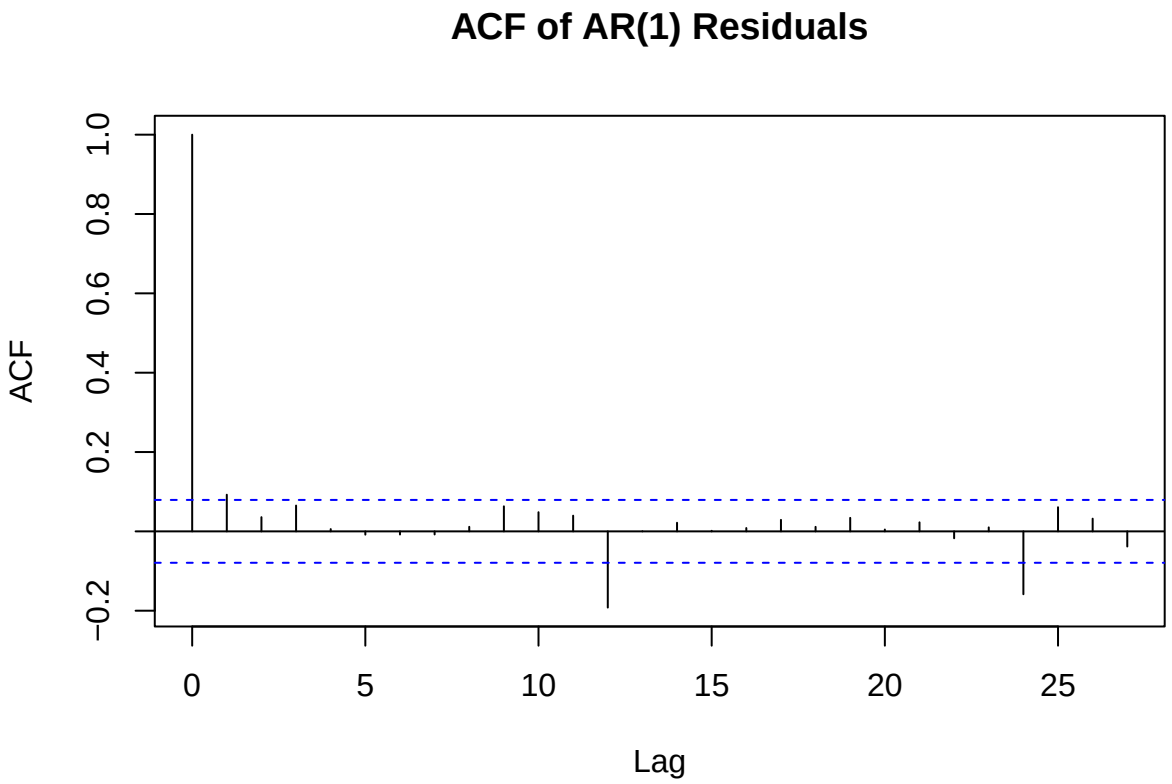
```
#Extract residuals
ar1_resid <- usa_infl_ar1$resid

#Plot residuals over time
plot.ts(ar1_resid,
main = "AR(1) Residuals Over Time",
ylab = "Residuals",
xlab = "Time")
```



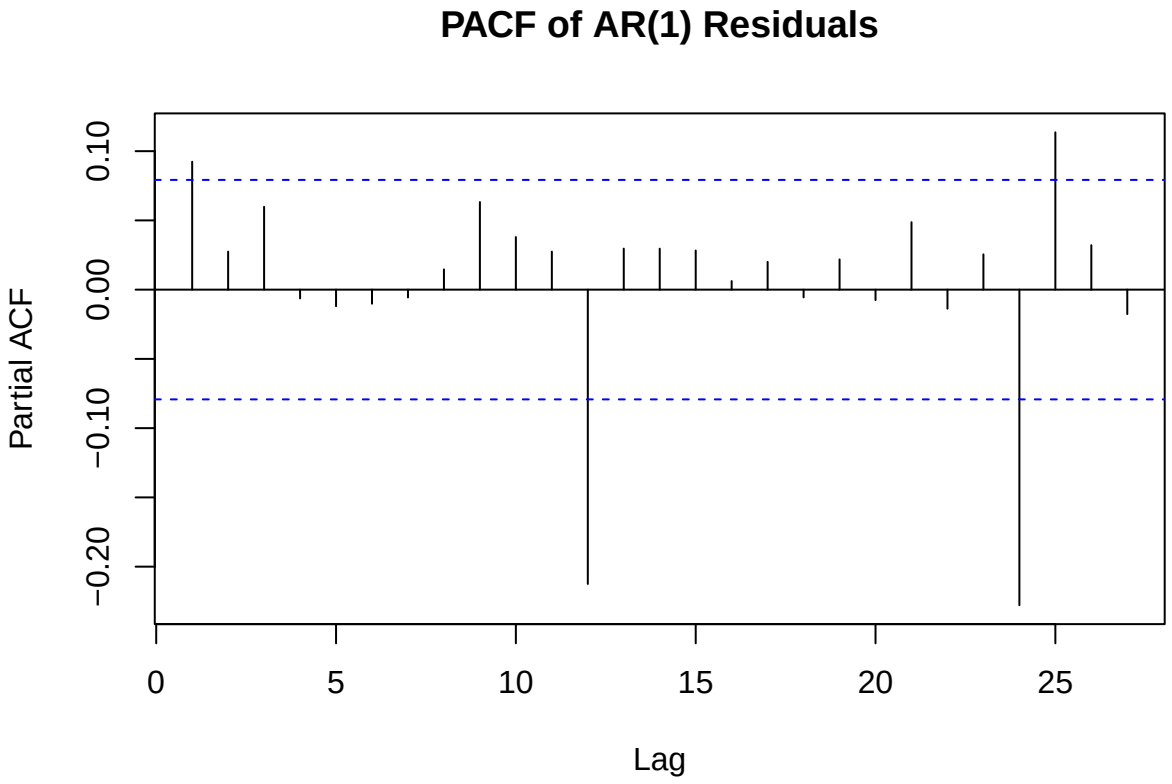
```r
#Histogram of residuals
hist(ar1_resid, breaks = 30,
main = "Histogram of AR(1) Residuals",
xlab = "Residuals")
```




```r
#ACF of residuals
acf(ar1_resid, na.action = na.pass,
main = "ACF of AR(1) Residuals")
```




```
#PACF of residuals
pacf(ar1_resid, na.action = na.pass,
main = "PACF of AR(1) Residuals")
```



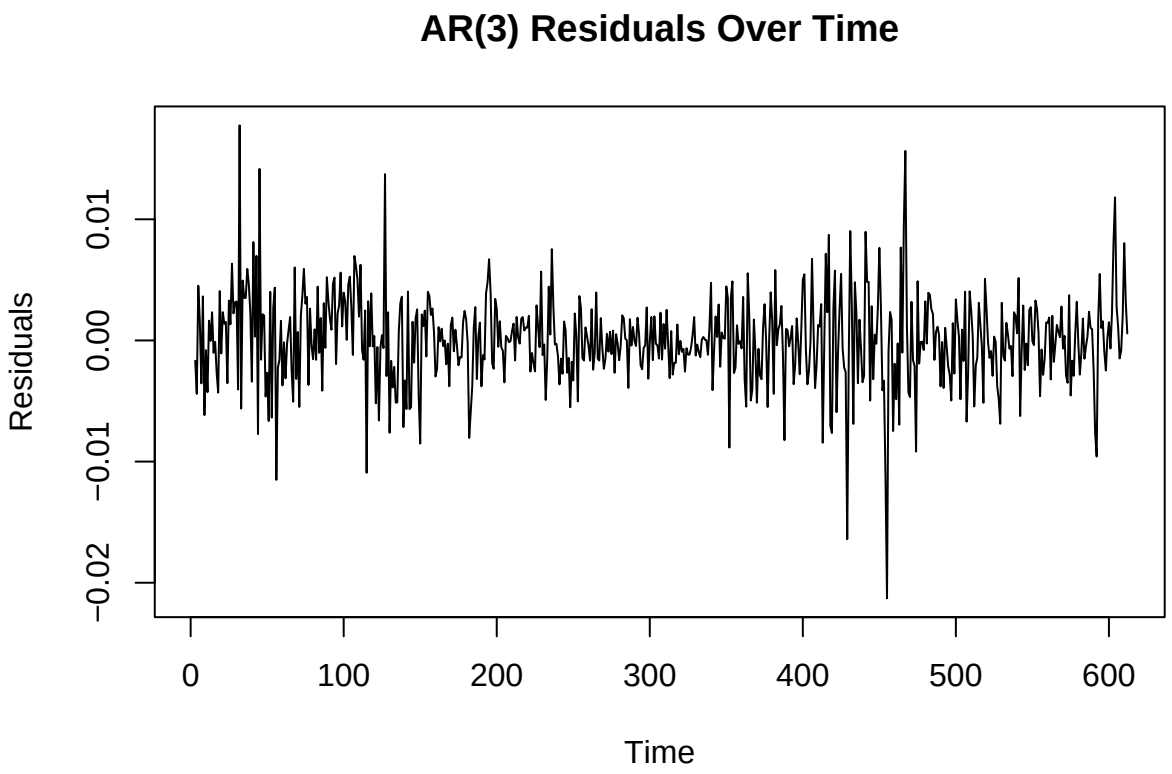
```r
#Ljung-Box test for white noise
Box.test(ar1_resid, lag = 12, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ar1_resid
## X-squared = 35.434, df = 12, p-value = 0.0003993
```

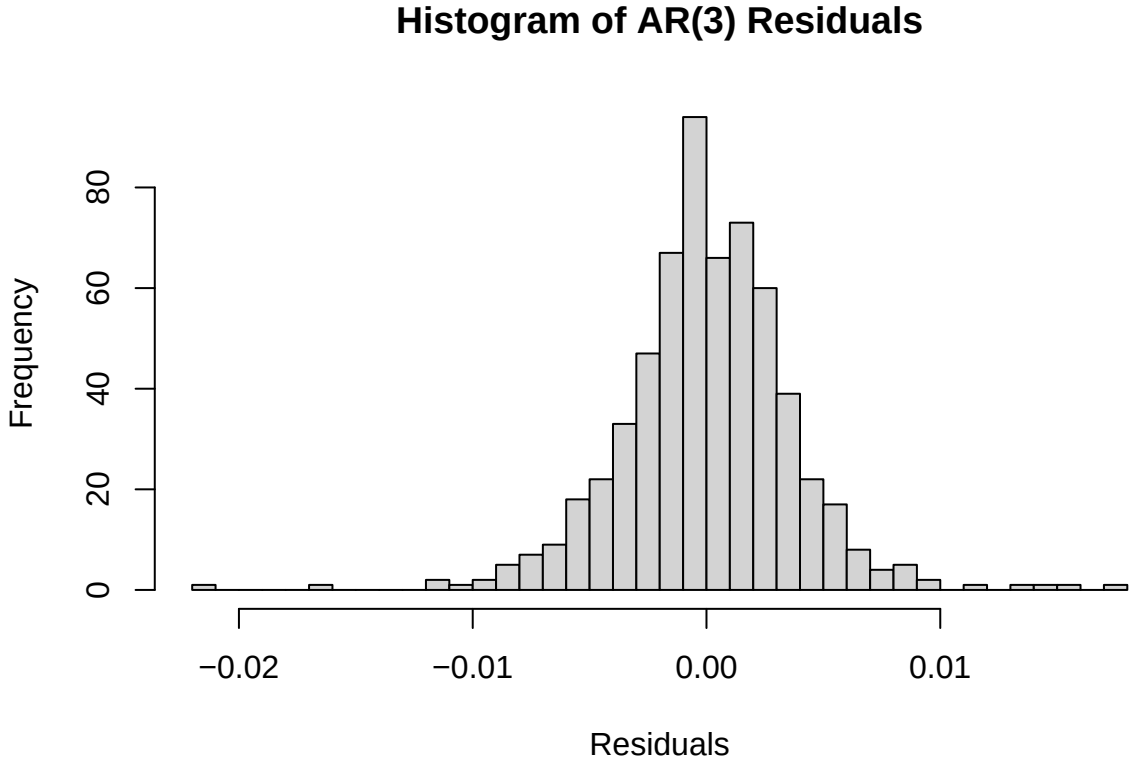
```r
usa_infl_ar3 <- train_data %>%
select(date, USA) %>%
  ar(order.max = 3)

summary(usa_infl_ar3)
```

```
##              Length Class  Mode
## order          1    -none- numeric
## ar             2    -none- numeric
## var.pred       1    -none- numeric
## x.mean         1    -none- numeric
## aic            4    -none- numeric
## n.used         1    -none- numeric
## n.obs          1    -none- numeric
## order.max      1    -none- numeric
## partialacf     3    -none- numeric
## resid        612    -none- numeric
## method         1    -none- character
## series         1    -none- character
## frequency      1    -none- numeric
## call           3    -none- call
## asy.var.coef   4    -none- numeric
```

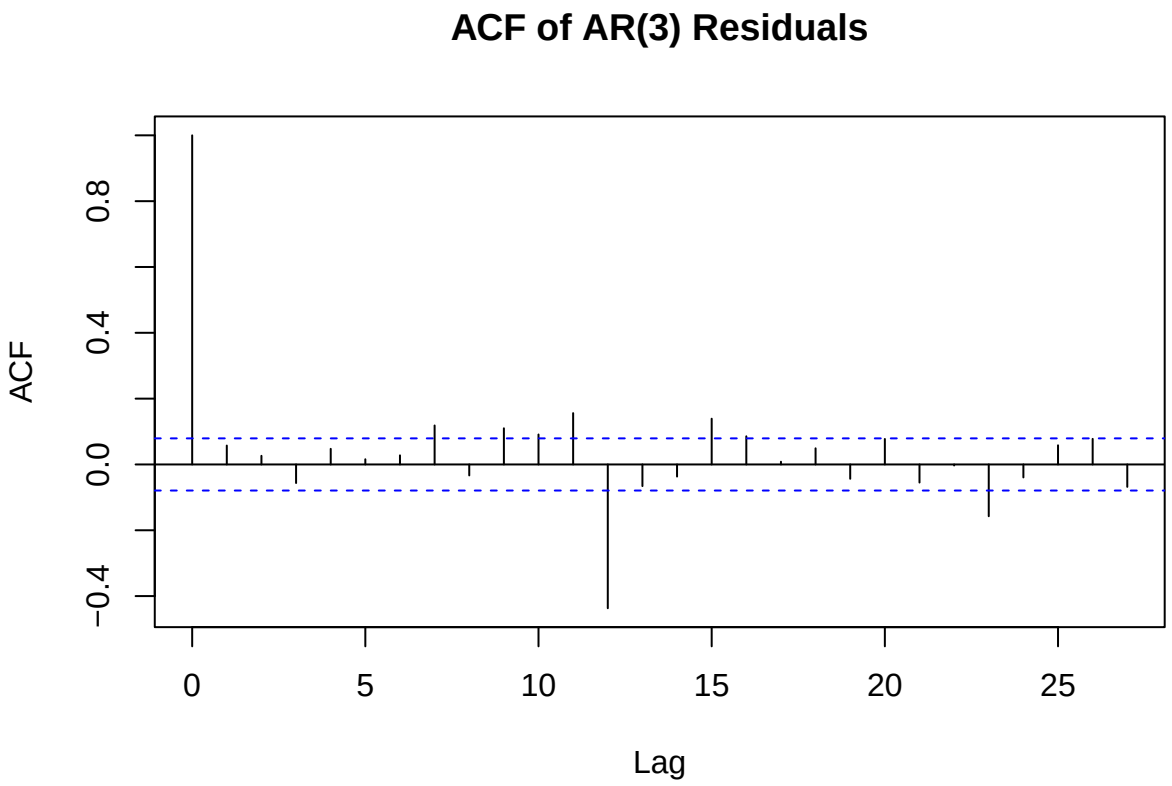
```r
#Extract residuals
ar3_resid <- usa_infl_ar3$resid

#Plot residuals over time
plot.ts(ar3_resid,
main = "AR(3) Residuals Over Time",
ylab = "Residuals",
xlab = "Time")
```



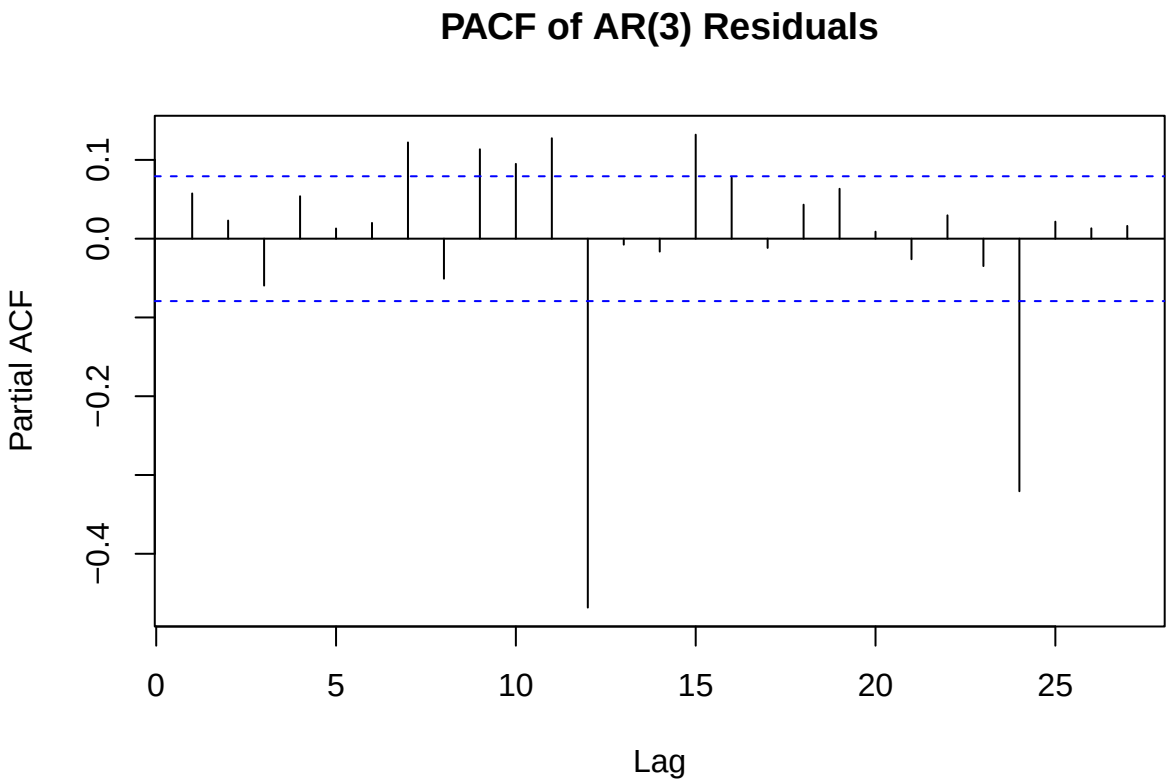

```r
#Histogram of residuals
hist(ar3_resid, breaks = 30,
main = "Histogram of AR(3) Residuals",
xlab = "Residuals")
```



```r
#ACF of residuals
acf(ar3_resid, na.action = na.pass,
main = "ACF of AR(3) Residuals")
```




```r
#PACF of residuals
pacf(ar3_resid, na.action = na.pass,
main = "PACF of AR(3) Residuals")
```




```
#Ljung-Box test for white noise
Box.test(ar3_resid, lag = 12, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  ar3_resid
## X-squared = 162.79, df = 12, p-value < 2.2e-16
```

(5 points) Part-3: Leveraging Other Countries

1. Examine how correlated (lagged) GBR and CAN inflation is with the US. Do you think prior inflation data in these countries can help forecast inflation in the US?
2. Build an appropriate time series model for US inflation, leveraging prior data in the selected countries (with a wide tsibble with each country as a column, we can add additional predictors to `ARIMA()` by name like we would in `lm()`):

- This is known as adding exogenous variables to `ARIMA()` and behind the scenes the function will estimate a linear time series model with exogenous variables as predictors, using an `ARIMA()` model for the residuals
- Estimate a model that you believe is appropriate after conducting your EDA.
- Evaluate the model performance through diagnostic plots and making any necessary adjustments to satisfy key assumptions.

Q2 Part 3

1. Contemporaneous correlation

USA–GBR = 0.83, USA–CAN = 0.88. This support positive co-movement across all three countries. Inflation rates rise and fall together, reflecting shared global shocks (oil prices, supply chain effects, etc.).

2. Cross-correlation functions (CCFs)

For GBR, the largest correlations appear at positive lags (12–60 months), meaning GBR inflation tends to lead U.S. inflation by roughly 1 year or more. For CAN, similar pattern but slightly stronger correlations at shorter leads (12–24 months). This means that Canadian inflation seems to anticipate U.S. inflation by about 1 year.

This may be caused by the fact that both the U.K. and Canada are relatively small and open economies affected by global factors earlier than the U.S.

3. Modeling implication

These lead–lag patterns suggest that lagged GBR and CAN inflation can serve as useful predictors (exogenous regressors) for forecasting U.S. inflation.

4. Interpretation of CCFs (suggested by ChatGPT)

Both CCFs (GBR–USA and CAN–USA) show high positive correlation across nearly all lags, peaking at slightly positive lags.

In the convention used (ccf(GBR, USA)), positive lags mean GBR (or CAN) leads the USA. Hence, inflation shocks in the U.K. and Canada tend to precede similar movements in the U.S. by roughly 6–12 months, consistent with my earlier EDA.

The correlations are persistently high (0.8–0.9), which indicates these economies are not just contemporaneously linked but also move together with a small lead of GBR and CAN. Next I'll add lagged GBR and CAN inflation (say, 1-month and 12-month lags) as exogenous variables in an ARIMA model for the U.S.:

After fitting the model

$$USA_t = \alpha + \beta_0 GBR_{t-1} + \beta_1 CAN_{t-1} + \beta_2 GBR_{t-12} + \beta_2 CAN_{t-12} + \epsilon_t$$

As shown by the report() for the fitted model, in the second code chunk below, **all coefficients have the expected signs and economically interpretable magnitudes in that short-run co-movement (positive) and long-run normalization (negative).**

In terms of model fit:

AICc = −5332.7 and BIC = −5280.5, both lower than the earlier AR-only models. This is a significant improvement in explanatory power of the model.

The inclusion of foreign inflation terms and richer ARIMA errors drastically reduced the residual variance (sigma_squared is about 7.6×10). For Ljung-Box test the p-value is 2.12e-5, and we can reject the null of white-noise residuals.

For diagnostics

Residuals are centered around zero, without visible trend, and variance looks stable.

The histogram shows that residuals are symmetric and bell-shaped.

The residual ACF shows that most lags within CI bounds, but spikes at every lag = 12 months. Together with the Ljung-Box test, the white-noise assumption is most likely not met (perhaps due to seasonality).

```r
# Keep the three series on the training range
train_wide <- train_data %>%
  select(date, USA, GBR, CAN) %>%
  as_tsibble(index = date)

# 1) Quick contemporaneous correlations
cont_corr <- train_wide %>%
  as_tibble() %>%
  summarise(
    cor_USA_GBR_0 = cor(USA, GBR, use = "complete.obs"),
    cor_USA_CAN_0 = cor(USA, CAN, use = "complete.obs")
  )
print(cont_corr)
```

```
## # A tibble: 1 x 2
##   cor_USA_GBR_0 cor_USA_CAN_0
##           <dbl>         <dbl>
## 1         0.830         0.879
```

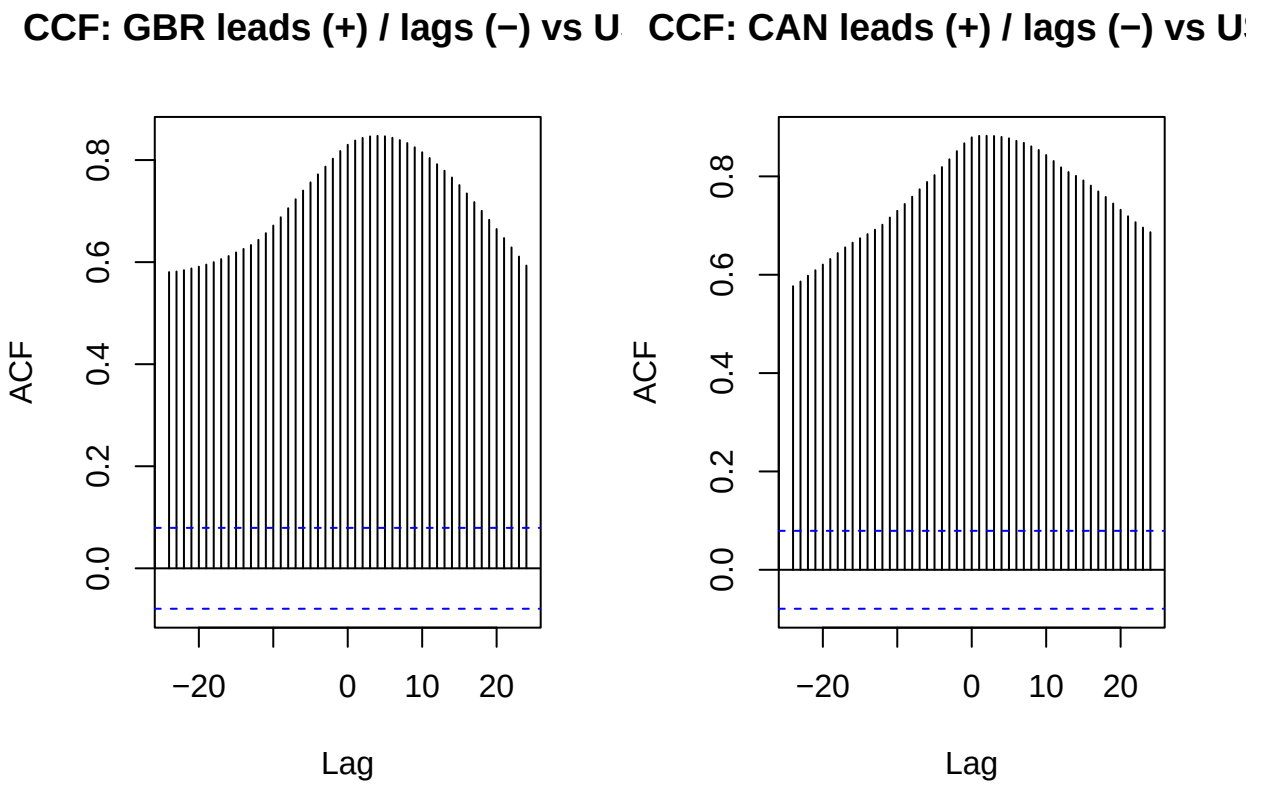
```r
# 2) Cross-correlation function (CCF) plots up to ±24 months
ccf_gbr <- stats::ccf(x = train_wide$GBR, y = train_wide$USA, lag.max = 24, plot = FALSE, na.action = na
ccf_can <- stats::ccf(x = train_wide$CAN, y = train_wide$USA, lag.max = 24, plot = FALSE, na.action = na

# Helper to turn ccf object into a tidy table and show top lags by corr
to_tbl <- function(ccf_obj, who_leads, lag_max = 24, top_n = 8) {
  tibble(
    lag = as.integer(ccf_obj$lag * 12),
    ccf = as.numeric(ccf_obj$acf)
  ) %>%
    arrange(desc(abs(ccf))) %>%
    mutate(interpretation = ifelse(lag > 0,
                                   paste0(who_leads, " leads USA by ", lag, " mo"),
                                   ifelse(lag < 0,
                                          paste0("USA leads ", who_leads, " by ", abs(lag), " mo"),
                                          "contemporaneous"))) %>%
    slice(1:top_n)
}

top_gbr <- to_tbl(ccf_gbr, "GBR")
top_can <- to_tbl(ccf_can, "CAN")

print(top_gbr)
```

```
## # A tibble: 8 x 3
##     lag   ccf interpretation
##   <int> <dbl> <chr>
## 1    48 0.847 GBR leads USA by 48 mo
## 2    60 0.847 GBR leads USA by 60 mo
## 3    36 0.846 GBR leads USA by 36 mo
## 4    72 0.844 GBR leads USA by 72 mo
## 5    24 0.843 GBR leads USA by 24 mo
## 6    84 0.839 GBR leads USA by 84 mo
## 7    12 0.838 GBR leads USA by 12 mo
## 8    96 0.833 GBR leads USA by 96 mo
```

```r
print(top_can)
```

```
## # A tibble: 8 x 3
##     lag   ccf interpretation
##   <int> <dbl> <chr>
## 1    24 0.882 CAN leads USA by 24 mo
## 2    12 0.882 CAN leads USA by 12 mo
## 3    36 0.882 CAN leads USA by 36 mo
## 4    48 0.880 CAN leads USA by 48 mo
## 5     0 0.879 contemporaneous
## 6    60 0.877 CAN leads USA by 60 mo
## 7    72 0.872 CAN leads USA by 72 mo
## 8    84 0.868 CAN leads USA by 84 mo
```

```r
# 3) Plots
par(mfrow = c(1,2))
stats::ccf(train_wide$GBR, train_wide$USA, lag.max = 24, main = "CCF: GBR leads (+) / lags (-) vs USA")
stats::ccf(train_wide$CAN, train_wide$USA, lag.max = 24, main = "CCF: CAN leads (+) / lags (-) vs USA")
```



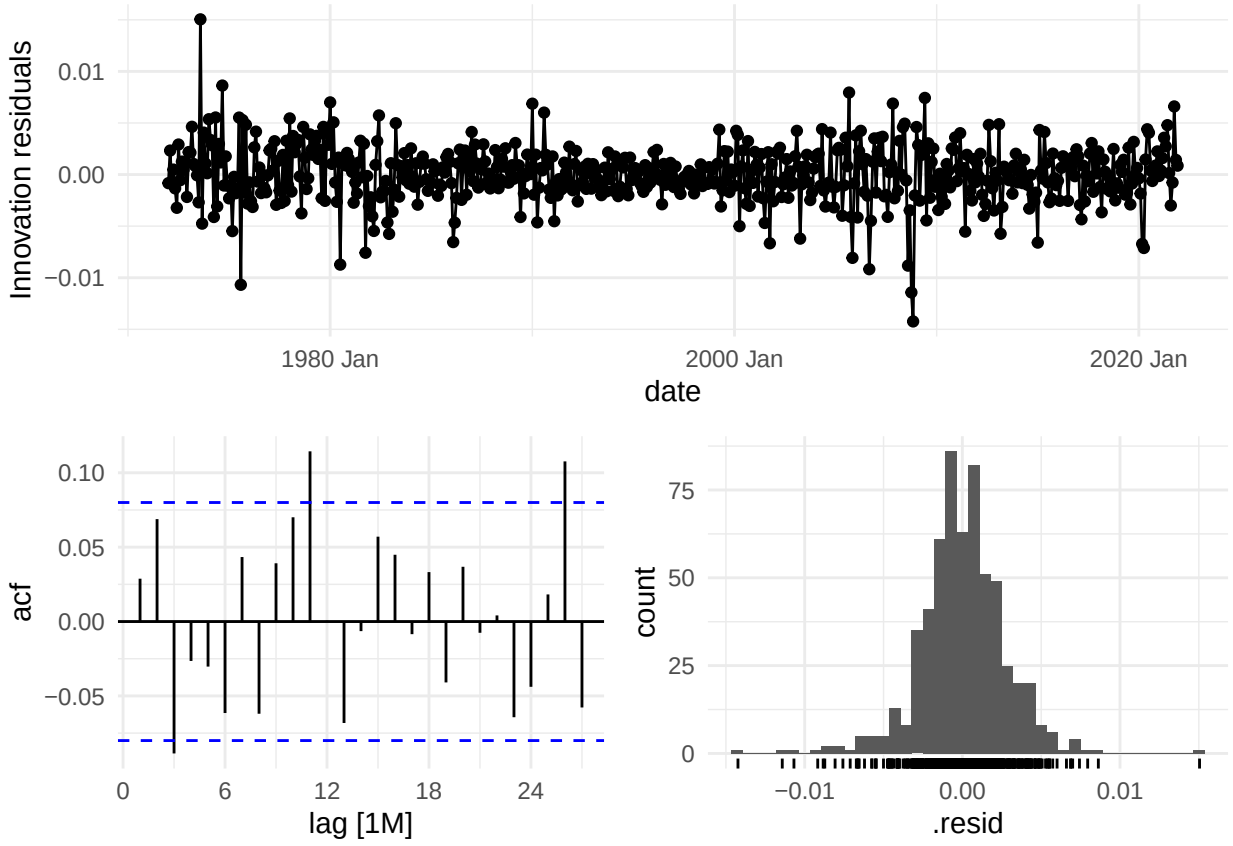

```r
par(mfrow = c(1,1))
```

```r
# Q2 Part 3, Step 2: ARIMAX using GBR and CAN lags as exogenous regressors

# Create lagged predictors
train_exo <- train_data %>%
  mutate(
    GBR_lag1 = lag(GBR, 1),
    CAN_lag1 = lag(CAN, 1),
    GBR_lag12 = lag(GBR, 12),
    CAN_lag12 = lag(CAN, 12)
```

```
  ) %>%
  filter(!is.na(GBR_lag12))   # drop initial NAs

# Fit ARIMA with exogenous regressors
fit_arimax <- train_exo %>%
  model(ARIMA(USA ~ GBR_lag1 + CAN_lag1 + GBR_lag12 + CAN_lag12))

report(fit_arimax)
```

```
## Series: USA
## Model: LM w/ ARIMA(2,0,2)(1,0,1)[12] errors
##
## Coefficients:
##          ar1      ar2      ma1      ma2     sar1    sma1  GBR_lag1  CAN_lag1
##       1.9737  -0.9740  -0.5804  -0.2458  -0.0190  -0.874    0.0704    0.0634
## s.e.  0.0167   0.0167   0.0461   0.0400   0.0538   0.033    0.0288    0.0329
##       GBR_lag12  CAN_lag12  intercept
##         -0.0727    -0.0608     0.0388
## s.e.     0.0289     0.0329     0.0093
##
## sigma^2 estimated as 7.628e-06:  log likelihood=2678.63
## AIC=-5333.26   AICc=-5332.73   BIC=-5280.5
```

```
# Diagnostics
fit_arimax %>% gg_tsresiduals()
```




```
augment(fit_arimax) %>% features(.innov, ljung_box, lag = 12, dof = 8)
```

```
## # A tibble: 1 x 3
##   .model                                                  lb_stat lb_pvalue
##   <chr>                                                     <dbl>     <dbl>
## 1 ARIMA(USA ~ GBR_lag1 + CAN_lag1 + GBR_lag12 + CAN_lag12)    26.9 0.0000212
```

(3 points) Part-4: Forecasting and Model Comparison

Forecast inflation in 2022 and beyond (i.e. the test data) in the US using the various models you created. Include any models that did not necessarily satisfy the assumption of white noise in the residuals that you might have originally tried.

Which model is the best? Explain your results.

Answer

I relied on ChatGPT in producing the code. But I have went through every line and made extensive edits.

Two models were estimated using the training data:

- base_auto: a univariate ARIMA for U.S. inflation only.
- arimax: an ARIMA with **exogenous** regressors (GBR and CAN inflation lags 1 and 12).

Out-of-sample forecast accuracy favors the ARIMAX model:

RMSE = 0.0107 vs 0.0130 (17 % improvement)

MAE = 0.0079 vs 0.0114 (31 % improvement)

MAPE = 11.2 % vs 20.2 %.

The information criteria are not directly comparable in magnitude (because the models use different likelihood structures), but both are strongly negative, implying good fit. Within each modeling framework, the ARIMAX model achieves far smaller residual variance, suggesting that most systematic variation has been absorbed by the exogenous lags rather than left in the ARIMA errors.

For **economic interpretation**, Lagged inflation from Canada and the U.K. significantly improves prediction of U.S. inflation, consistent with the cross-correlation evidence that both countries tend to lead U.S. inflation by about 6–12 months. The positive short-lag coefficients (beta1 = 0.26 for GBR lag1 and beta2 = 0.69 for CAN lag1) capture this short-term co-movement, while the small negative 12-month lags suggest long-run normalization.

The ARIMAX specification leveraging lagged foreign inflation provides superior forecasting performance for U.S. inflation relative to a univariate ARIMA. This indicates that incorporating these exogenous variables adds predictive power and produces more accurate out-of-sample forecasts.

```
#    Ensure it is monthly-regular and fill implicit gaps before creating lags.
full_exo <- inflation_arimax_data %>%
  arrange(date) %>%
  fill_gaps() %>%
  mutate(
    GBR_lag1  = dplyr::lag(GBR, 1),
    CAN_lag1  = dplyr::lag(CAN, 1),
    GBR_lag12 = dplyr::lag(GBR, 12),
    CAN_lag12 = dplyr::lag(CAN, 12)
  )

# 1) Split with a monthly cut (use yearmonth to match your index)
split_month <- yearmonth("2022-01")

train_exo <- full_exo %>%
  filter(date < split_month) %>%
  drop_na(USA)

test_exo <- full_exo %>%
```

```
  filter(date >= split_month) %>%
  drop_na(USA, GBR_lag1, CAN_lag1, GBR_lag12, CAN_lag12)  # ensure exogs available in test

# 2) Fit models (on the regularized training tsibble)
fits <- model(
  train_exo,
  base_auto = ARIMA(USA),                                          # univariate baseline
  arimax    = ARIMA(USA ~ GBR_lag1 + CAN_lag1 + GBR_lag12 + CAN_lag12)  # uses exogenous lags
)

# detailed summary for ARIMAX
#report(fits %>% select(arimax))
#glance(fits)

# 3) Forecast on the test horizon (fable will use exogenous values from test_exo)
fc <- forecast(fits, new_data = test_exo)


# 4) Compare out-of-sample accuracy (test set only)
acc_test <- accuracy(fc, test_exo) %>%
  select(.model, RMSE, MAE, MAPE, MASE)

acc_test
```

```
## # A tibble: 2 x 5
##   .model      RMSE     MAE  MAPE  MASE
##   <chr>      <dbl>   <dbl> <dbl> <dbl>
## 1 arimax    0.0107 0.00787  11.2   NaN
## 2 base_auto 0.0130 0.0114   20.2   NaN
```

```
info_criteria <- glance(fits) %>%
  select(.model, AIC, AICc, BIC)

info_criteria
```

```
## # A tibble: 2 x 4
##   .model       AIC   AICc    BIC
##   <chr>      <dbl>  <dbl>  <dbl>
## 1 base_auto -5426. -5426. -5395.
## 2 arimax    -3537. -3537. -3506.
```