ECB Project - Coding Appendix

Contents

Preliminaries

Setup

```
# --------- 1. Clear memory
rm(list=ls())

# --------- 2. Load here package to enable finding script loading other packages
require(here)

# --------- 3. Set directory
getwd()
setwd("...")

# --------- 4. Load packages & helper functions for plots and tables
source(here("scripts/packages.R"))
source(here("helpers/raw_plotter.R"))
source(here("helpers/stationarity_tables.R"))
source(here("helpers/taylor_tables.R"))
source(here("helpers/struct_breaks_tables.R"))
source(here("helpers/roll_TR_plotter.R"))
source(here("helpers/pseudo_outofsample_tables.R"))
source(here("helpers/pseudo_outofsample_plots.R"))
source(here("helpers/actual_forecast_displays.R"))

# Api key for data
fredr_set_key("e0169694a62c1337f1969e3872605eca")

# --------- 5. Dates (to automatically get the latest data from API calls
start_date <- "1999-01-01"
end_date <- Sys.Date()

# For replication
set.seed(2025)
```

Interactive Option Selection

- Use Hamilton Filter:
  - TRUE: Selects Hamilton method for output gap estimation
  - FALSE: Selects Hodrick-Prescott method for output gap estimation
- Use Inflation Expectations:
  - TRUE: The models used for forecasting will use 12-month ahead inflation expectations from the ECB survey of professional forecasts (average).
  - FALSE: The models used for forecasting will use realised inflation
- Use Formula
  - Formula 1: Actual interest rate regressed on inflation and output gaps
  - Formula 2: Shadow interest rate regressed on inflation and output gaps

  - Formula 3: Actual interest rate regressed on the one-quarter lag of the interest rate and on inflation and output gaps
  - Formula 4: Shadow interest rate regressed on the one-quarter lag of the shadow interest rate and on inflation and output gaps

- Format:
  - html: For outputting in console or knitting to html
  - latex: For knitting to pdf

```
source(here("scripts/options_config.R"))

#temp remove soon
format <- "html"
format <- "latex"
```

```
source(here("scripts/taylor_rule_formulas.R"))
```

Data

Sources & Explanations

- FRED Data Source: European Central Bank, ECB Deposit Facility Rate for Euro Area [ECBDFR], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/ECBDFR. The data is the Deposit Facility and is directly reprinted from the ECB. It's in Percent and not seasonally adjusted. The ECB Monetary Policy is steered through this rate
- Shadow Interest Rate: The Shadow rate was developed by **wuTimeVaryingLowerBound2017** and does quantify a hypothetical removal of the ZLB. The rate does not track 1:1 on the deposit facility in the data before the ZLB, instead being closer to the refinancing rate. (maybe need to adjust, oui je crois faux, surtout la partie sur refinancing rate)
- GDP: The GDP Data is quarterly real GDP in 2010 Euros in million retrieved from FRED via Eurostat. The data is seasonally adjusted.
- Potential GDP: Either estimated with the Hodrick-Prescott filter or the Hamilton filter, based on the formula below:

$$\text{HP Filter:} \quad \min_{\tau} \left(\sum_{t=1}^{T} (y_t - \tau_t)^2 + \lambda \sum_{t=2}^{T-1} \left[(\tau_{t+1} - \tau_t) - (\tau_t - \tau_{t-1}) \right]^2 \right)$$

$$\text{Hamilton:} \quad y_t = \beta_0 + \sum_{j=1}^{p} \beta_j y_{t-h-j+1} + v_t \quad \text{(where } v_t \text{ is the cycle)}$$

- Inflation:
  - Realised: The Inflation data is from Eurostat and measures HICP monthly data (annual rate of change). It's the index that the ECB uses for Inflation and is not seasonally adjusted, but the fact that it represents the year-on-year change in prices implies there is no seasonality.
  - Expectations: Expected Inflation is the ECB's survey of professional forecasters. It tracks professional forecasts of HICP inflation 12 months in advance.

Loading & Preparation Data

```
#copypaste data script and explain
source(here("scripts/data.R"))
```

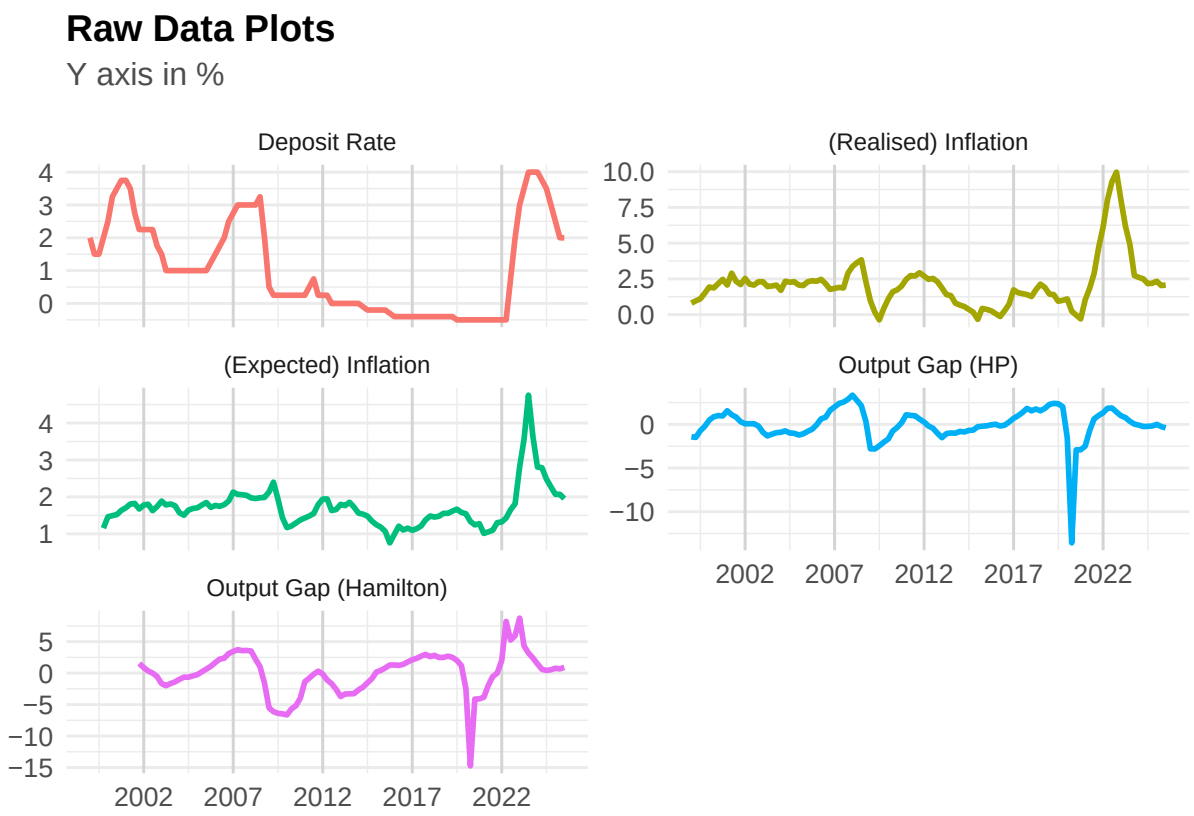
Options Configuration

```
source(here("scripts/options_implement.R"))
```

- CONFIGURATION: Using Hamilton Filter for output gap estimation.
- CONFIGURATION: Using realised inflation in Taylor Rule forecasting.

Raw Data Plots

```r
# Who cares about the method here, only result matters
raw_data_plotter(data = data)
```



Data Properties

```r
# copy paste both stat and coint funcs and explain
source(here("helpers/stationarity_tests.R"))
```

```r
# -----  Run Tests -----

# Use helper to run stationarity checks for our main variables
test_rate <- check_stationarity(data$rate, "Interest Rate")
test_inflation <- check_stationarity(data$inflation, "Inflation")
test_output_gap <- check_stationarity(na.omit(data$output_gap), "Output Gap")

# Given results for rate and inflation, run tests on 1st diffs
test_rate_diff <- check_stationarity(diff(data$rate), "Interest Rate (1st Diff)")
test_inflation_diff <- check_stationarity(diff(data$inflation), "Inflation (1st Diff)")

# Since rate and inflation are I(1), run a cointegration test
coint_test = check_coint(data$rate, data$inflation,
            var_name1 = "Interest Rate", var_name2 = "Inflation")
```

Table 1: Summary of Stationarity Tests (ADF & KPSS)

Variable	ADF p-value	KPSS p-value	Result
Interest Rate	0.4350	0.0359	Non-Stationary I(1): ADF confirms unit root, KPSS rejects stationarity.
Inflation	0.2257	0.1000	Conflicting Results: ADF confirms unit root, while KPSS suggests stationarity.
Output Gap	0.0131	0.1000	Stationary I(0): ADF rejects unit root, KPSS confirms stationarity.
Interest Rate (1st Diff)	0.0100	0.1000	Stationary I(0): ADF rejects unit root, KPSS confirms stationarity.
Inflation (1st Diff)	0.0100	0.1000	Stationary I(0): ADF rejects unit root, KPSS confirms stationarity.

Table 2: Cointegration Test Results

Variables	p-value	Result
Interest Rate & Inflation	0.0585	Not Cointegrated

```
# ----- Print Results -----

# Combine stationarity results
all_stationarity_results <- rbind(test_rate,
                                  test_inflation,
                                  test_output_gap,
                                  test_rate_diff,
                                  test_inflation_diff)

# Tables
generate_stat_tests_table(stat_tests = all_stationarity_results)
```

```
generate_coint_tests_table(coint_test = coint_test)
```

```
# Cleanup
rm(all_stationarity_results, coint_test, test_inflation,
   test_inflation_diff, test_output_gap, test_rate, test_rate_diff)
```

Taylor Rule Estimation

Without Lags

The "No Lag" Taylor Rule specifications follow the regression expressed in equation (1). We run 6 variations total of this, starting with the base case using the actual deposit rate, realised inflation (gap) and the output gap. We then mix cases where we use the shadow rate, expected inflation (gap) and finally including both expected and realised inflation (gap).

$$i_t = \pi^* + \beta(\pi_t - \pi^*) + \gamma(y_t - \bar{y_t}) \tag{1}$$

```
TR <- lm(rate ~ realised_inflation_gap + output_gap, data = data)
TRsr <- lm(shadowrate ~ realised_inflation_gap + output_gap, data = data)

models <- list(TR, TRsr)
taylor_regression_to_table(models, caption = "No Lag, No Expectations")
```

Table 3: No Lag, No Expectations

	TR	TR w/ SR
(Intercept)	0.8460	-0.8296
	(0.8943)	(2.6530)
realised_inflation_gap	0.1946	0.6812
	(0.4318)	(0.6058)
output_gap	0.0839	-0.0528
	(0.1632)	(0.2763)
N	96	96
R2	0.1722	0.1146

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
TR_e <- lm(rate ~ exp_inflation_gap + output_gap, data = data)
TRsr_e <- lm(shadowrate ~ exp_inflation_gap + output_gap, data = data)
TR_ie <- lm(rate ~ realised_inflation_gap + exp_inflation_gap +
                   output_gap, data = data)
TRsr_ie <- lm(shadowrate ~ realised_inflation_gap + exp_inflation_gap +
                           output_gap, data = data)

models <- list(TR_e, TRsr_e, TR_ie, TRsr_ie)
taylor_regression_to_table(models, caption = "No Lag, with
                                              Inflation Expectations")
```

Table 4: No Lag, with Inflation Expectations

	TR	TR w/ SR	TR	TR w/ SR
(Intercept)	1.3522 ***	0.3098	1.3470 ***	0.1984
	(0.3169)	(1.5706)	(0.3707)	(1.6114)
exp_inflation_gap	1.7282 ***	3.7705 **	1.7165 ***	3.5220 **
	(0.3556)	(1.3951)	(0.3483)	(1.1795)
output_gap	0.0711	-0.0015	0.0671	-0.0872
	(0.0469)	(0.1872)	(0.0811)	(0.2131)
realised_inflation_gap			0.0146	0.3120
			(0.1278)	(0.3423)
N	96	96	96	96
R2	0.6180	0.3914	0.6182	0.4089

*** p < 0.001; ** p < 0.01; * p < 0.05.

Lagged Models

The lagged Taylor Rule specifications follow the regression expressed in equation (2) which is the same as (1) but also including a one-quarter lag for the interest rate. We then run the same 6 variations as for the models without the lag.

$$i_t = \pi^* + \phi i_{t-1} + \beta(\pi_t - \pi^*) + \gamma(y_t - \bar{y_t}) \tag{2}$$

```
lTR <- lm(rate ~ rate_lag + realised_inflation_gap + output_gap, data = data)
lTRsr <- lm(shadowrate ~ shadowrate_lag + realised_inflation_gap +
                         output_gap, data = data)

models <- list(lTR, lTRsr)
taylor_regression_to_table(models, caption = "Interest Rate Lag,
                                              No Expectations")
```

```
lTR_e <- lm(rate ~ rate_lag + exp_inflation_gap + output_gap, data = data)
lTRsr_e <- lm(shadowrate ~ shadowrate_lag + exp_inflation_gap +
                           output_gap, data = data)
lTR_ie <- lm(rate ~ rate_lag + realised_inflation_gap + exp_inflation_gap +
                    output_gap, data = data)
lTRsr_ie <- lm(shadowrate ~ shadowrate_lag + realised_inflation_gap +
                            exp_inflation_gap + output_gap, data = data)

models <- list(lTR_e, lTRsr_e, lTR_ie, lTRsr_ie)
taylor_regression_to_table(models, caption = "Interest Rate Lag, with
                                              Inflation Expectations")
```

Table 5: Interest Rate Lag, No Expectations

	TR	TR w/ SR
(Intercept)	0.0512	-0.0756
	(0.0406)	(0.0592)
rate_lag	0.9195 ***	
	(0.0381)	
realised_inflation_gap	0.0877 *	0.2493 ***
	(0.0348)	(0.0343)
output_gap	0.0219	-0.0113
	(0.0163)	(0.0184)
shadowrate_lag		0.9535 ***
		(0.0173)
N	96	96
R2	0.9597	0.9828

*** p < 0.001; ** p < 0.01; * p < 0.05.

Checking for structural breaks

```
#only paste chow one
source(here("helpers/struct_breaks_tests.R"))

# 1st Suspected break: Start of ZLB in 2012 Q3
#  -> R = 55 in evaluation chunk

# Suspected break: Covid
#  -> R = 85 in evaluation chunk

# Run test using helper function
chow_tests(data, break1 = 55, break2 = 85,
           events_name <- c("ZLB Start", "COVID-19 Start"))
```

```
#only paste bp one
source(here("helpers/struct_breaks_tests.R"))

# Run test using helper function
bp_tests(data)
```

Table 6: Interest Rate Lag, with Inflation Expectations

	TR	TR w/ SR	TR	TR w/ SR
(Intercept)	0.1805 *	0.0012	0.1343 ***	-0.0876
	(0.0791)	(0.0852)	(0.0376)	(0.0594)
rate_lag	0.8612 ***		0.8750 ***	
	(0.0436)		(0.0381)	
exp_inflation_gap	0.2381 **	0.1295	0.1530 ***	-0.0548
	(0.0734)	(0.1038)	(0.0434)	(0.0618)
output_gap	0.0449 *	0.0592	0.0234	-0.0106
	(0.0218)	(0.0474)	(0.0165)	(0.0181)
shadowrate_lag		0.9631 ***		0.9586 ***
		(0.0287)		(0.0187)
realised_inflation_gap			0.0768 *	0.2528 ***
			(0.0378)	(0.0346)
N	96	96	96	96
R2	0.9547	0.9714	0.9614	0.9829

*** p < 0.001; ** p < 0.01; * p < 0.05.

Rolling Estimation (for structural breaks)

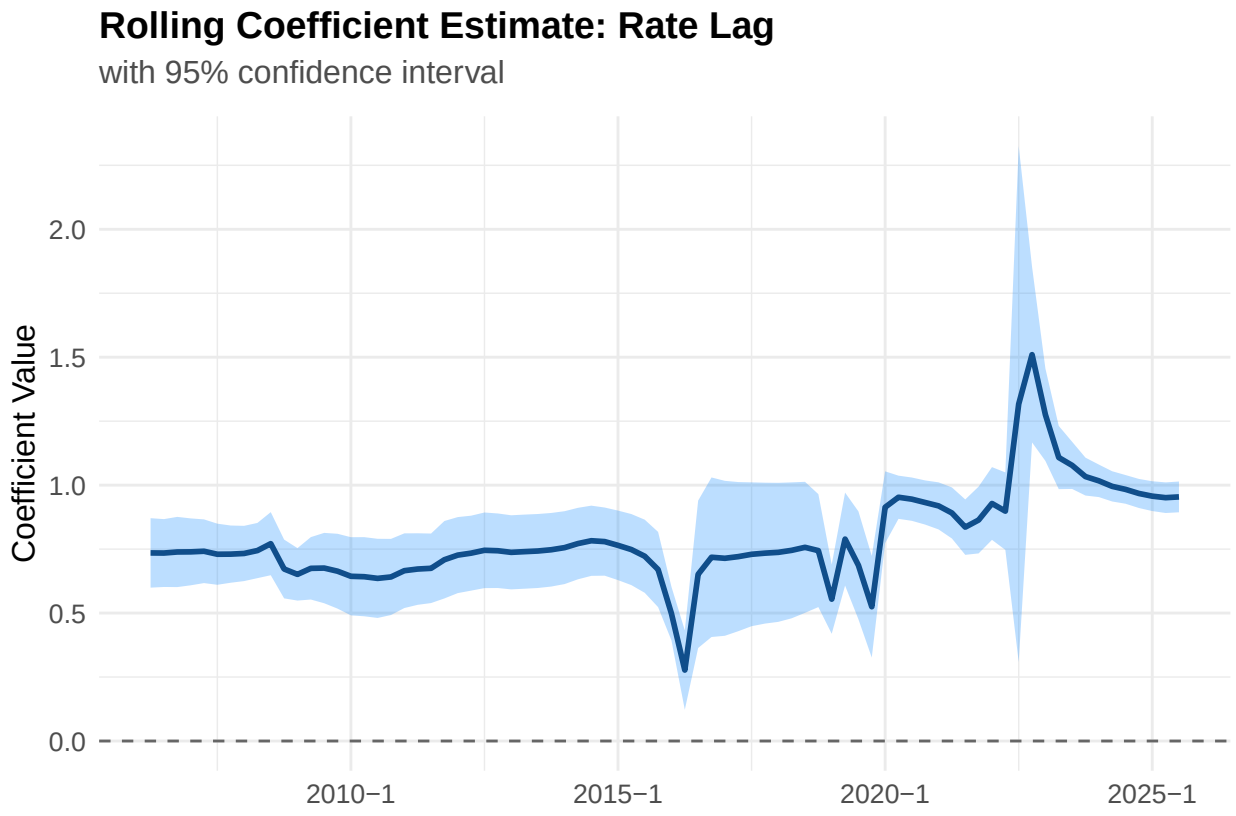
```
source(here("helpers/roll_TR_estimator.R"))

# Estimate a rolling-window (W in quarters) Taylor Rule specification
TR_roll <- estimate_rolling_TR(data, W = 30)

# Plotting (note: this part is not modular, obviously)
plot_rolling_coefs(TR_roll, "rate_lag", var_name_title="Rate Lag")
```

Table 7: Chow tests for suspected structural breaks

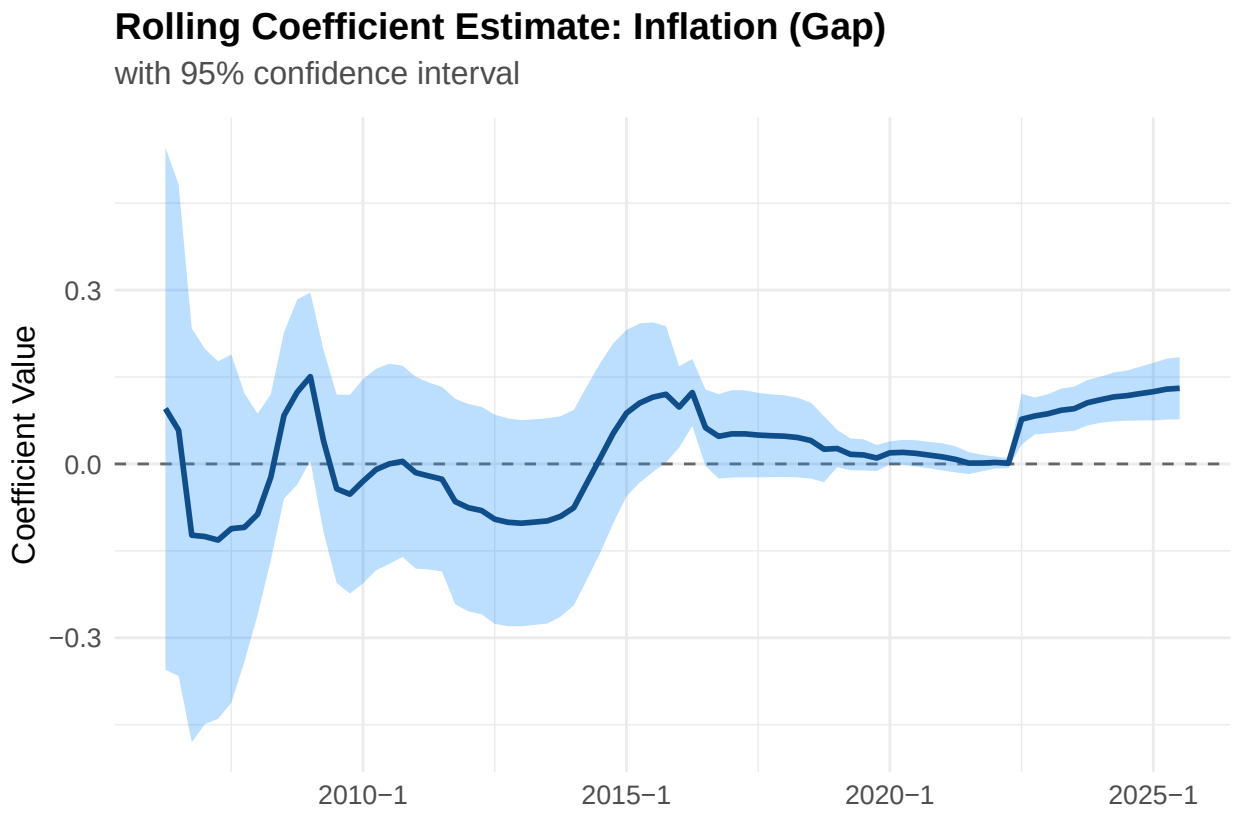
Event	Date	p-value
ZLB Start	2012 Q3	0.0018
COVID-19 Start	2020 Q1	0.1230

Table 8: Bai-Perron test for multiple breaks

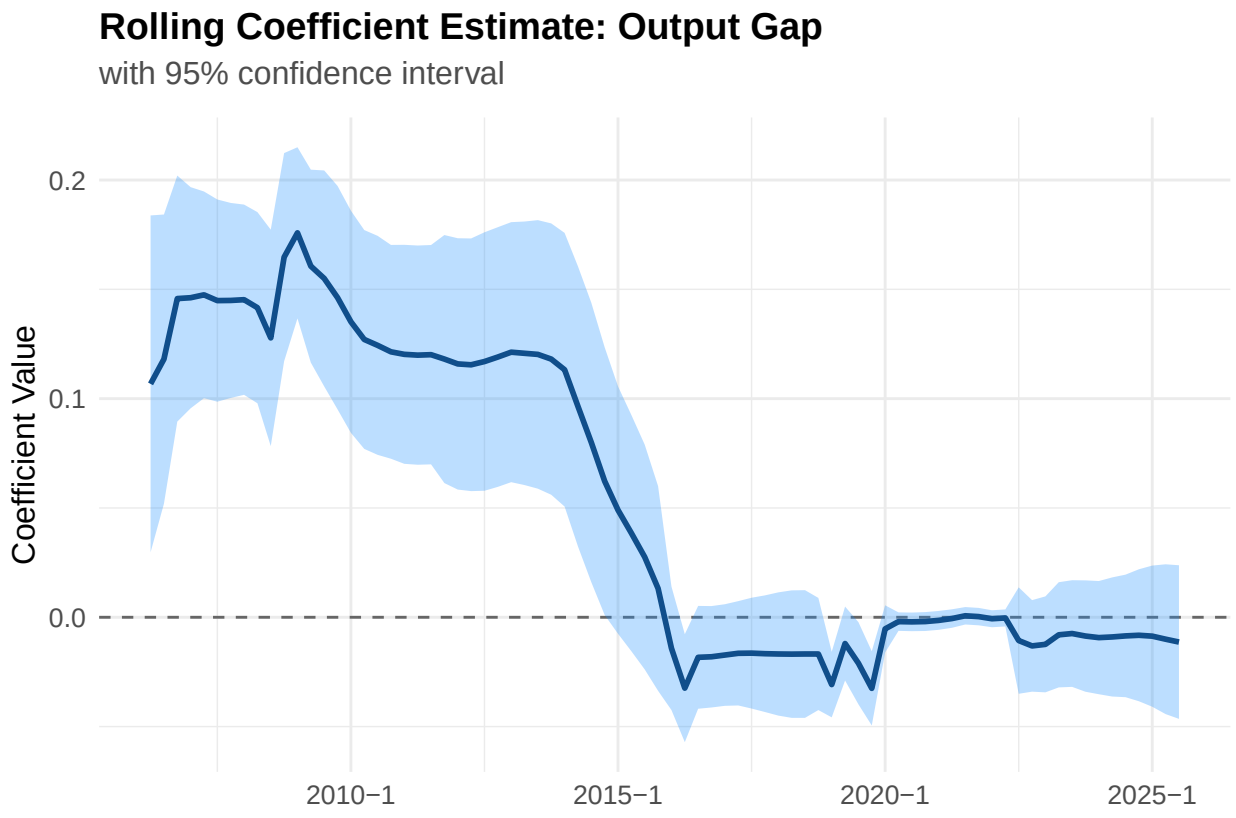
Detected Breaks
2006 Q2
2019 Q2




```
plot_rolling_coefs(TR_roll, "inflation_gap", var_name_title="Inflation (Gap)")
```




```
plot_rolling_coefs(TR_roll, "output_gap", var_name_title="Output Gap")
```


Rolling Coefficient Estimate: Output Gap
with 95% confidence interval
Coefficient Value
0.2
0.1
0.0
2010−1
2015−1
2020−1
2025−1

Forecasting Model Evaluation

Methods

Describe method with auto arima and so on.

```
source(here("helpers/auto_ARIMA_replic.R"))
```

Pseudo Out of Sample Estimation

Pseudo-out of sample rolling estimation scheme of direct forecasts for all Taylor Rule formulas and a benchmark ARIMA specification.

```
#add explain parameters
R = 85 # Chow: Structural breaks at R=55 and R=85
cat("Evaluation sample starts after ",as.character(data$quarter[R]),".",sep="")
```

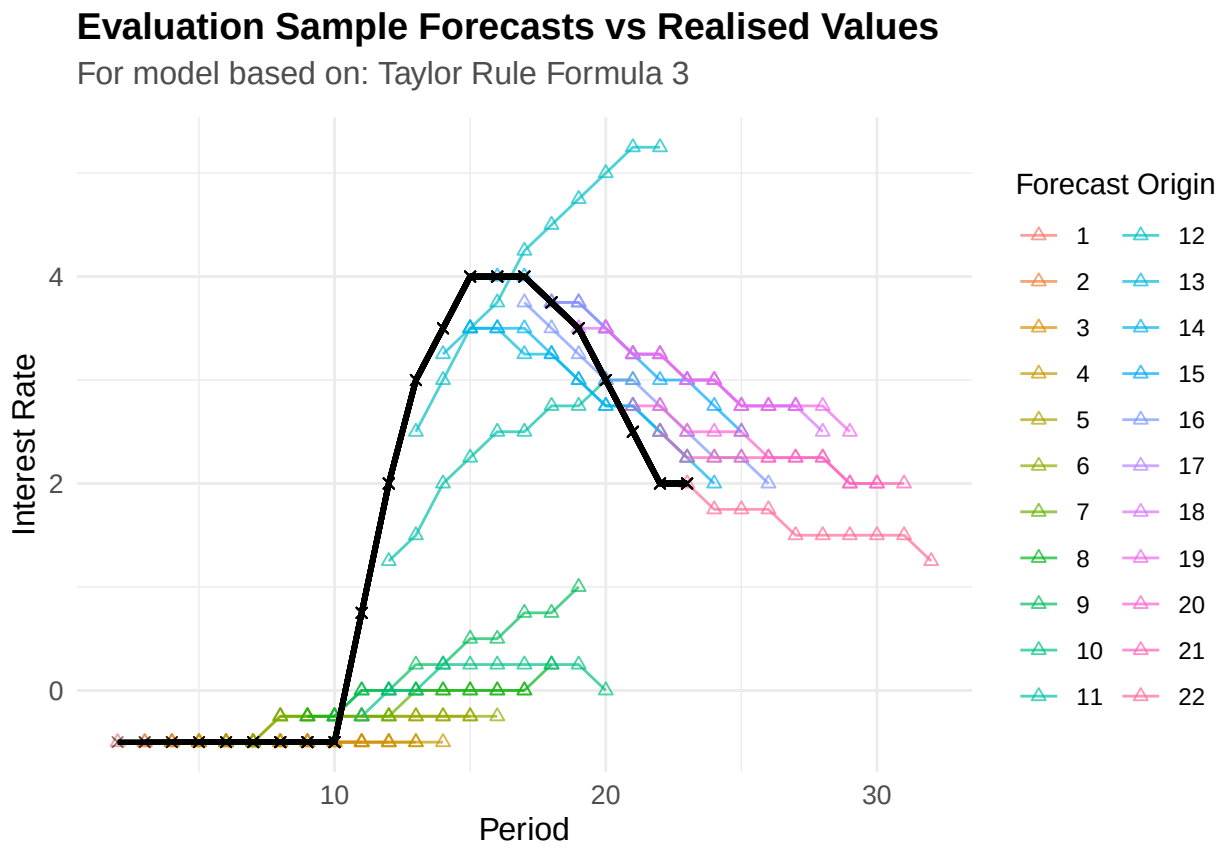
```
## Evaluation sample starts after 2020 Q1.
```

```
P = nrow(data) - R #but will effectively be: P = T-h-R
H = 10 # Number of different horizons (takes 10 to go until 2027 Q4)
```

```
source(here("scripts/pseudo_out_of_sample_estimation.R"))
```

Spaghetti Plots

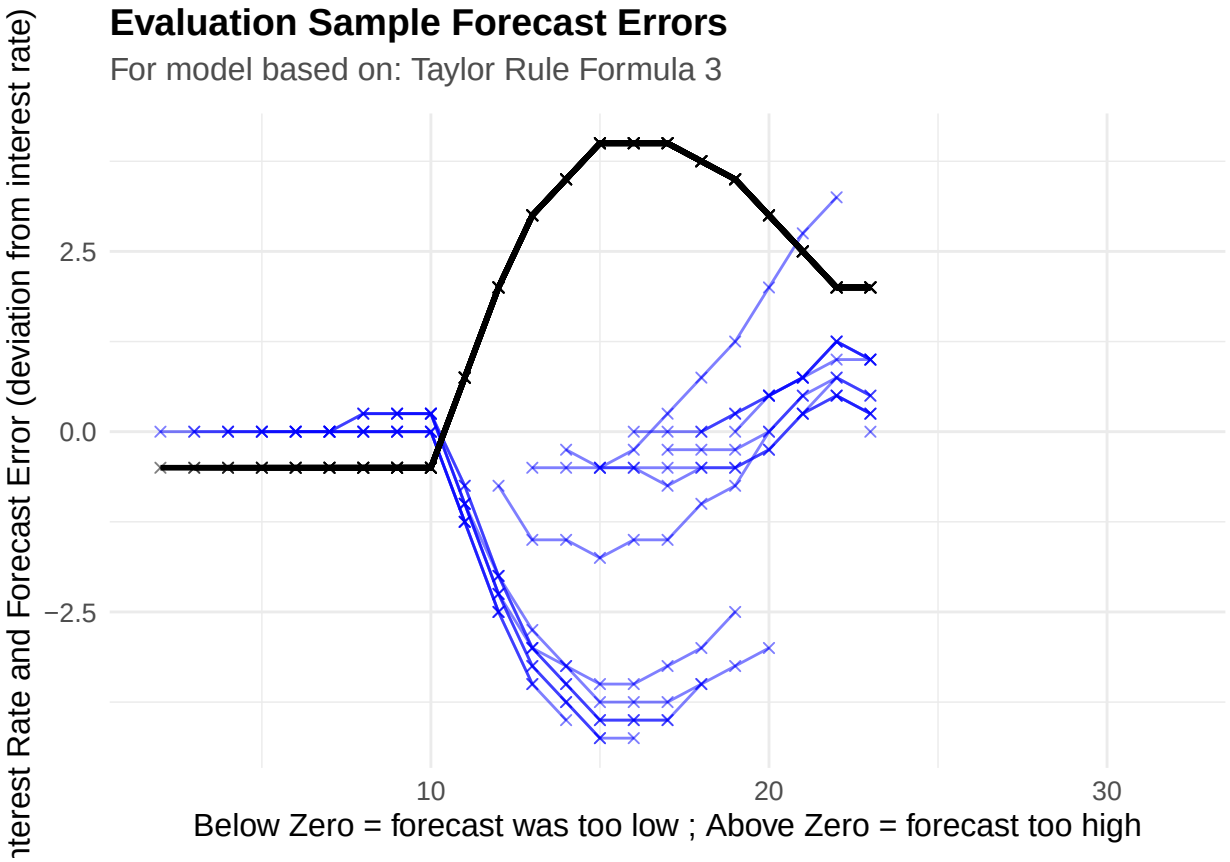
```
spaghetti_plotter(evals = eval_all_models)
```



Forecast Errors

Plots of FE

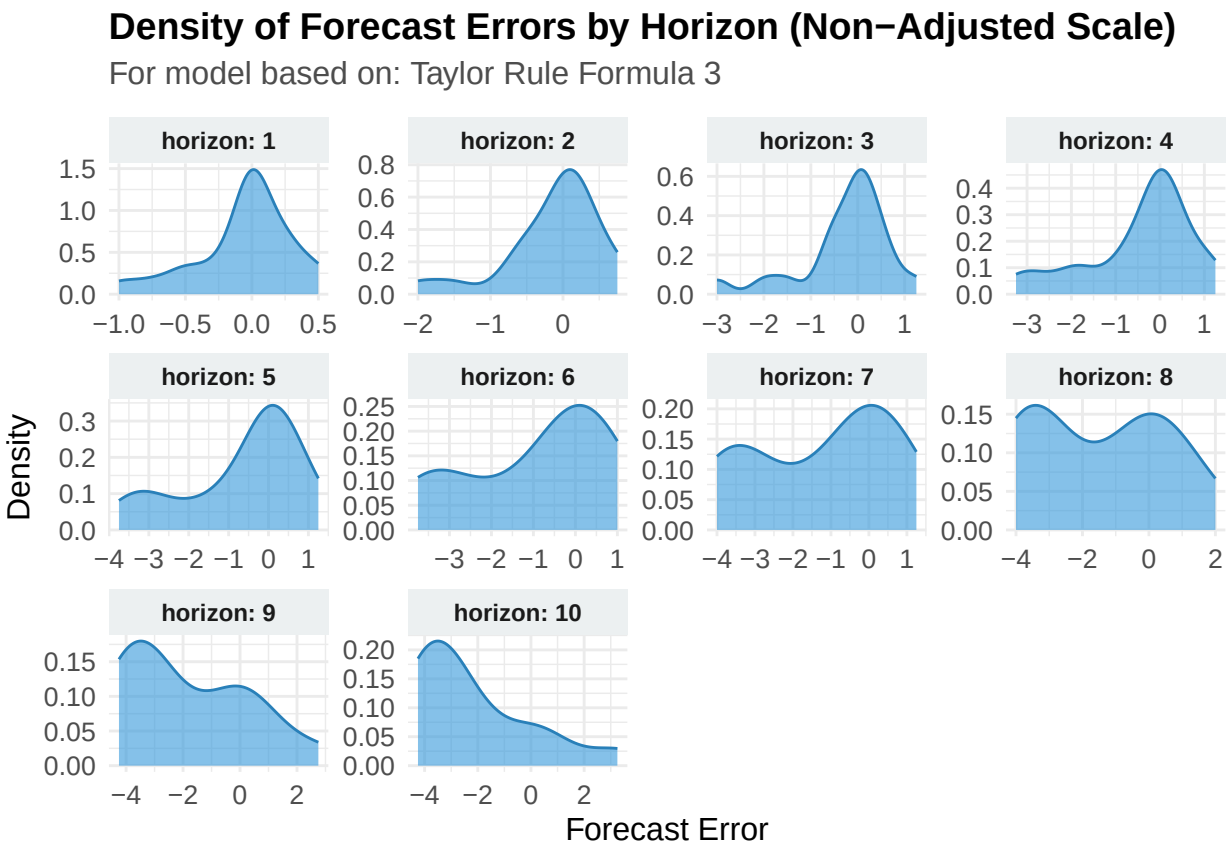
```
FE_spaghetti_plotter(evals = eval_all_models)
```



Density of FE

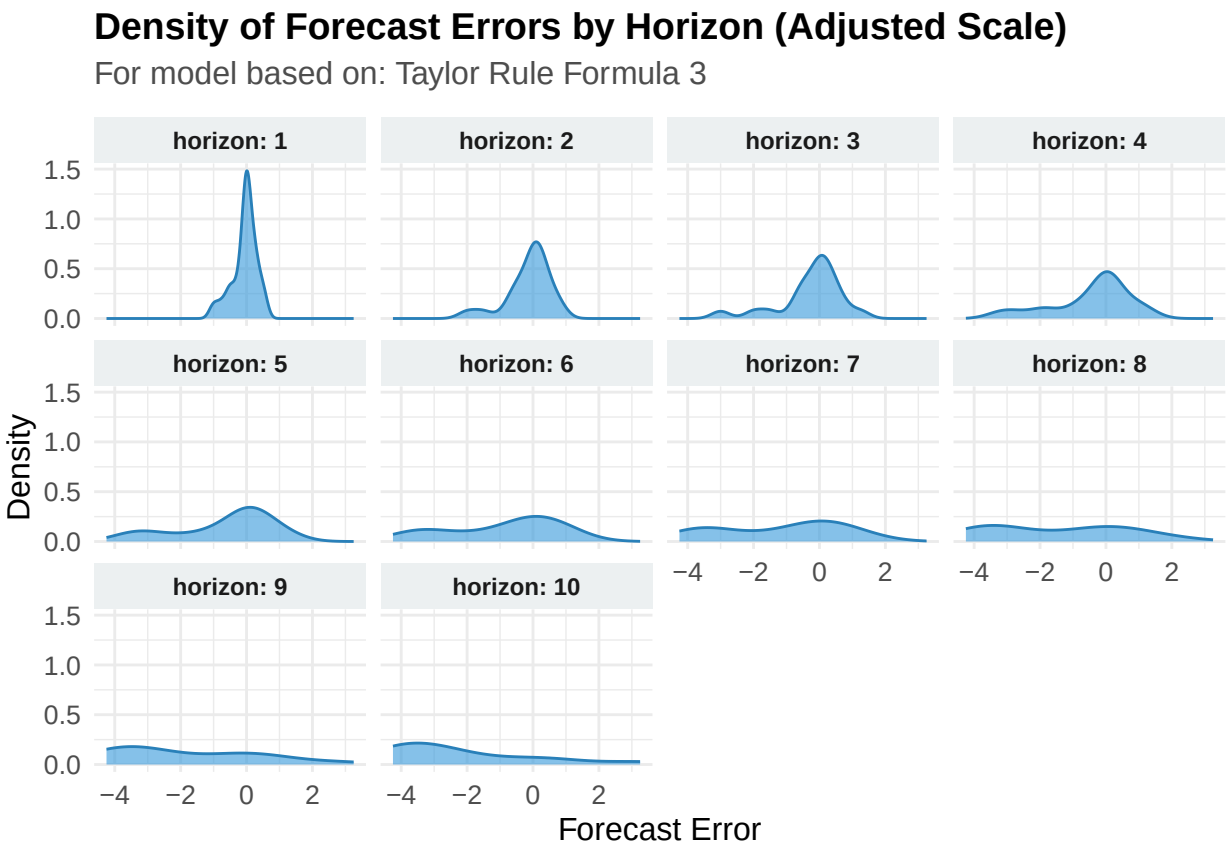
```
FE_density_plotter_unscaled(evals = eval_all_models)
```

```
## Warning: Removed 45 rows containing non-finite outside the scale range
## (`stat_density()`).
```



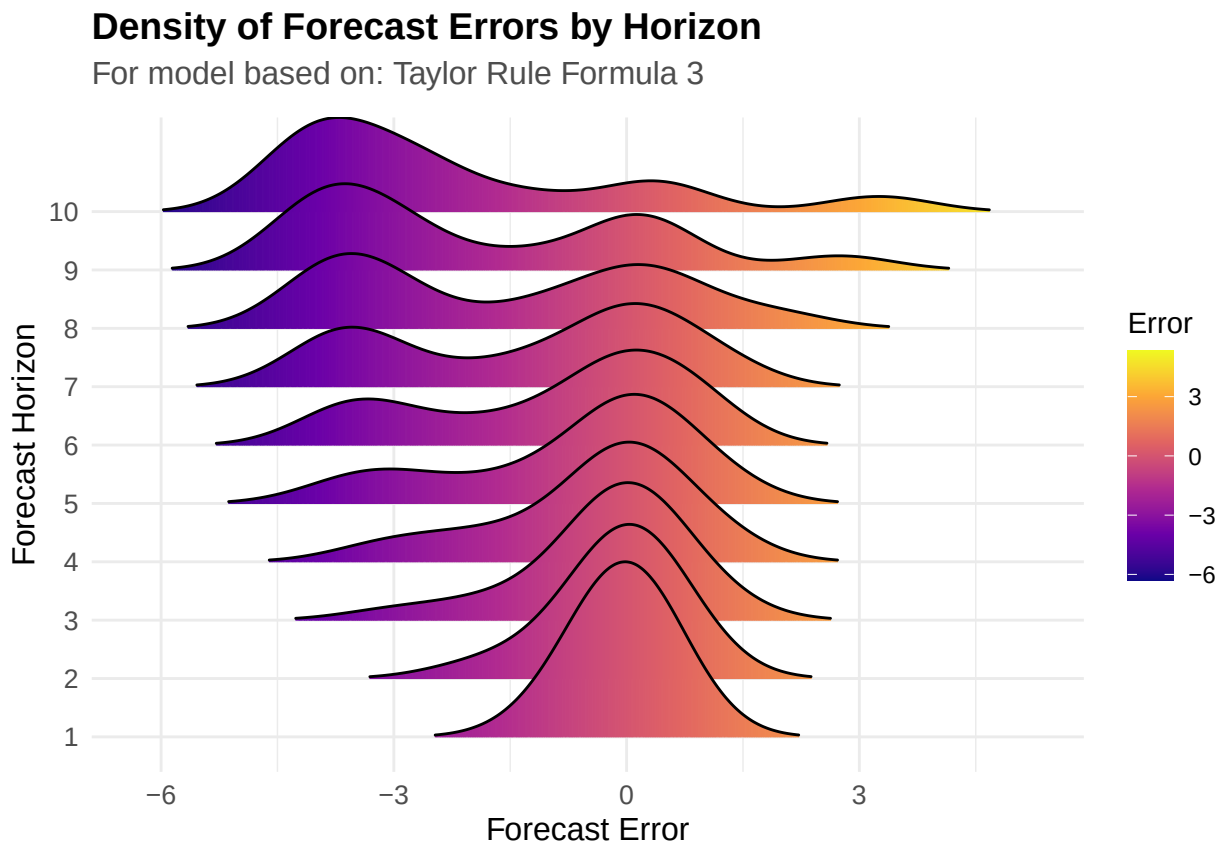

```
FE_density_plotter_scaled(evals = eval_all_models)
```

```
## Warning: Removed 45 rows containing non-finite outside the scale range
## (`stat_density()`).
```



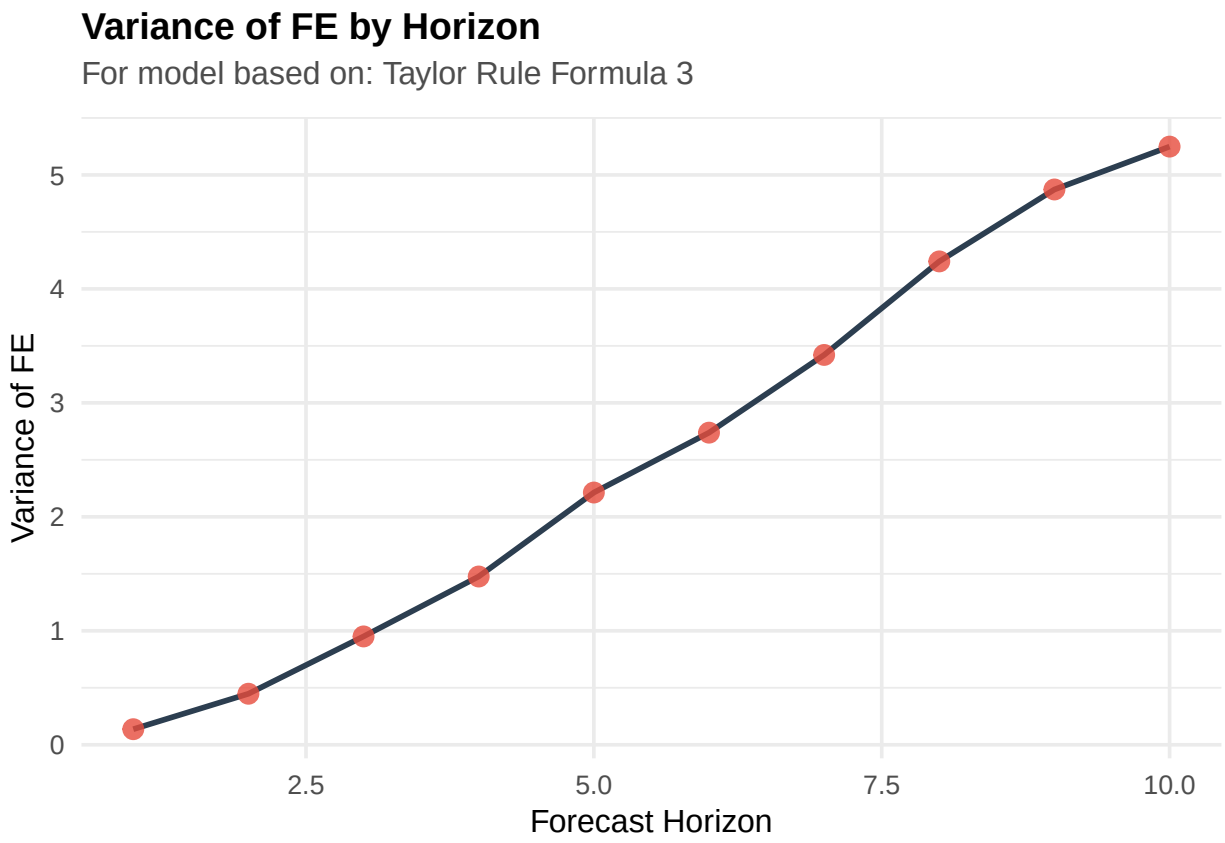

```
FE_density_plotter_ridges(evals = eval_all_models)
```

```
## Warning: `stat(x)` was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(x)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Picking joint bandwidth of 0.688
```



Variance of FE

```
FE_variance_plotter(var_by_horizon)
```



Horizon 1 Autocorrelation of FE

Mmodel 1, 2, 3 are autocorrelated at h=1.

```
#Durbin Watson tests for first autocorrelation at h=1

# Select Forecast Columns
formula_cols <- grep("^F_TR_FORMULA_", names(eval_all_models), value = TRUE)

# Automatically set max horizon for loop
max_h <- max(eval_all_models$horizon)

# Loop for each model
for (e_model_name in formula_cols) {

  # Extract and regress errors
  all_forecast_errors <- eval_all_models[["actuals"]] - eval_all_models[[e_model_name]]
  h1_errors_vector <- all_forecast_errors[eval_all_models$horizon == 1]
  temp_model <- lm(h1_errors_vector ~ 1)

  # Durbin-Watson Test
  dw_test_result <- durbinWatsonTest(temp_model)

  # Print output
  cat(sprintf("\n--- Durbin-Watson Test for Model: %s (h=1) ---\n", e_model_name))
```

```
  print(dw_test_result)
  cat(sprintf("DW Statistic: %.4f\n", dw_test_result$statistic))}
```

— Durbin-Watson Test for Model: F_TR_FORMULA_1 (h=1) — lag Autocorrelation D-W Statistic p-value 1 0.9175895 0.1455016 0 Alternative hypothesis: rho != 0

— Durbin-Watson Test for Model: F_TR_FORMULA_2 (h=1) — lag Autocorrelation D-W Statistic p-value 1 0.8986753 0.1798937 0 Alternative hypothesis: rho != 0

— Durbin-Watson Test for Model: F_TR_FORMULA_3 (h=1) — lag Autocorrelation D-W Statistic p-value 1 0.4973872 1.002973 0.022 Alternative hypothesis: rho != 0

— Durbin-Watson Test for Model: F_TR_FORMULA_4 (h=1) — lag Autocorrelation D-W Statistic p-value 1 -0.07327455 2.115385 0.804 Alternative hypothesis: rho != 0

Overall Autocorrelation of FE

We don't reject H0 -> h+1 are uncorrelated 1,2, are autocorrelated, 3 and 4 arent

```
#Ljung Box Test, with Columns & max lags set in Durbin Watson code chunk

# Loop through each TR
for (e_model_name in formula_cols) {

    cat(sprintf("--- Checking Errors for Model: %s ---\n", e_model_name))

    # Calculate the errors of the model
    all_errors <- eval_all_models[["actuals"]] - eval_all_models[[e_model_name]]

    # Loop over all forecast horizons to test for autocorrelation
    for (h in 1:max_h){

        # select errors for each horizon
        h_errors <- all_errors[eval_all_models$horizon == h]

        # max lag according to slides around sqrt T
        T_errors <- length(h_errors)
        max_lag <- 4 #round(sqrt(T_errors)) # use 4 or 5 chose 4 because 1 year has 4 quarters

        # Ljung-Box Test
        lb_test_result <- Box.test(h_errors,
                                   lag = max_lag,
                                   type = "Ljung-Box")

        #Print Results
        cat(sprintf("Horizon h=%d (N=%d, Lags=%d): Q=%.2f, p-value=%.3f\n",
                    h, T_errors, max_lag, lb_test_result$statistic, lb_test_result$p.value))}
        cat("\n")}
```

— Checking Errors for Model: F_TR_FORMULA_1 — Horizon h=1 (N=22, Lags=4): Q=47.79, p-value=0.000 Horizon h=2 (N=22, Lags=4): Q=45.55, p-value=0.000 Horizon h=3 (N=22, Lags=4): Q=44.69, p-value=0.000 Horizon h=4 (N=22, Lags=4): Q=40.67, p-value=0.000 Horizon h=5 (N=22, Lags=4): Q=27.72, p-value=0.000 Horizon h=6 (N=22, Lags=4): Q=16.42, p-value=0.003 Horizon h=7

(N=22, Lags=4): Q=10.25, p-value=0.036 Horizon h=8 (N=22, Lags=4): Q=9.62, p-value=0.047 Horizon h=9 (N=22, Lags=4): Q=7.04, p-value=0.134 Horizon h=10 (N=22, Lags=4): Q=8.80, p-value=0.066

— Checking Errors for Model: F_TR_FORMULA_2 — Horizon h=1 (N=22, Lags=4): Q=43.55, p-value=0.000 Horizon h=2 (N=22, Lags=4): Q=31.45, p-value=0.000 Horizon h=3 (N=22, Lags=4): Q=18.25, p-value=0.001 Horizon h=4 (N=22, Lags=4): Q=5.79, p-value=0.215 Horizon h=5 (N=22, Lags=4): Q=2.22, p-value=0.695 Horizon h=6 (N=22, Lags=4): Q=1.52, p-value=0.824 Horizon h=7 (N=22, Lags=4): Q=1.55, p-value=0.819 Horizon h=8 (N=22, Lags=4): Q=1.09, p-value=0.896 Horizon h=9 (N=22, Lags=4): Q=0.95, p-value=0.918 Horizon h=10 (N=22, Lags=4): Q=0.74, p-value=0.947

— Checking Errors for Model: F_TR_FORMULA_3 — Horizon h=1 (N=22, Lags=4): Q=8.50, p-value=0.075 Horizon h=2 (N=22, Lags=4): Q=15.23, p-value=0.004 Horizon h=3 (N=22, Lags=4): Q=15.87, p-value=0.003 Horizon h=4 (N=22, Lags=4): Q=18.15, p-value=0.001 Horizon h=5 (N=22, Lags=4): Q=17.36, p-value=0.002 Horizon h=6 (N=22, Lags=4): Q=16.86, p-value=0.002 Horizon h=7 (N=22, Lags=4): Q=16.02, p-value=0.003 Horizon h=8 (N=22, Lags=4): Q=14.72, p-value=0.005 Horizon h=9 (N=22, Lags=4): Q=12.30, p-value=0.015 Horizon h=10 (N=22, Lags=4): Q=8.83, p-value=0.065

— Checking Errors for Model: F_TR_FORMULA_4 — Horizon h=1 (N=22, Lags=4): Q=1.82, p-value=0.770 Horizon h=2 (N=22, Lags=4): Q=9.02, p-value=0.061 Horizon h=3 (N=22, Lags=4): Q=11.89, p-value=0.018 Horizon h=4 (N=22, Lags=4): Q=12.75, p-value=0.013 Horizon h=5 (N=22, Lags=4): Q=11.97, p-value=0.018 Horizon h=6 (N=22, Lags=4): Q=11.28, p-value=0.024 Horizon h=7 (N=22, Lags=4): Q=11.98, p-value=0.017 Horizon h=8 (N=22, Lags=4): Q=12.30, p-value=0.015 Horizon h=9 (N=22, Lags=4): Q=12.41, p-value=0.015 Horizon h=10 (N=22, Lags=4): Q=11.81, p-value=0.019

Errors Normally Distributed

Run the Jarque Bera Test with helpers

```
# in final markdown, input just the relevant code, i.e. the tests made
# who cares about code to create tables, that stays in helpers
source(here("helpers/pseudo_outofsample_tests.R"))
```

```
# Run with helper functions
# missing benchmark
generate_all_jb_reports(
  formula_cols =  formula_cols,
  eval_all_models = eval_all_models,
  max_h = max_h)
```

Table 9: Jarque–Bera Test Results

horizon	F_TR_FORMULA_1	F_TR_FORMULA_2	F_TR_FORMULA_3	F_TR_FORMULA_4
1	1.3147 (0.518)	1.4536 (0.483)	3.0503 (0.218)	17.8432 (0.000 ***)
2	1.5416 (0.463)	1.3227 (0.516)	8 (0.018 **)	2.9683 (0.227)
3	1.7299 (0.421)	1.6514 (0.438)	7.5554 (0.023 **)	2.5318 (0.282)
4	1.4723 (0.479)	30.1444 (0.000 ***)	2.75 (0.253)	2.2488 (0.325)
5	1.4886 (0.475)	58.129 (0.000 ***)	2.3399 (0.310)	2.194 (0.334)
6	1.5736 (0.455)	67.3941 (0.000 ***)	1.8789 (0.391)	2.4866 (0.288)
7	0.9188 (0.632)	64.089 (0.000 ***)	1.6653 (0.435)	3.159 (0.206)
8	0.3858 (0.825)	56.1628 (0.000 ***)	1.3626 (0.506)	2.9565 (0.228)
9	0.3242 (0.850)	42.1115 (0.000 ***)	1.3005 (0.522)	3.2125 (0.201)
10	1.5089 (0.470)	30.9638 (0.000 ***)	3.1801 (0.204)	3.0005 (0.223)

Absolute Performance: Efficiency & Bias

```
# in final markdown, input just the relevant code, i.e. the tests made
# who cares about code to create tables, that stays in helpers
source(here("helpers/pseudo_outofsample_tests.R"))
```

```
# Call MZ-test helper function 4 times.
#  Note: These reports is wrapped in trycatch as it sometimes fails
#         If it does fail, simply decrease R in order to have more
#          observations, removing potential multicolinearity.

# MZ Report 1: Actual Rate, No Lag
mz_report_1 <- tryCatch({
  generate_absolute_performance_report(
    F_model = F_TR_1,
    Actual_values = Actuals,
    H = H,
    model_caption = "Mincer-Zarnowitz Test: Actual Rate, No Lag",
    format = format)}, error = function(e) {
  message("Error generating MZ Report (Actual Rate, No Lag): ", e$message)
  message("Skipping this report and continuing...")
  return(NULL)})

# MZ Report 2: Shadow Rate, No Lag (
mz_report_2 <- tryCatch({
  generate_absolute_performance_report(
    F_model = F_TR_2,
    Actual_values = Actuals,
    H = H,
    model_caption = "Mincer-Zarnowitz Test: Shadow Rate, No Lag",
    format = format)}, error = function(e) {
  message("Error generating MZ Report (Shadow Rate, No Lag): ", e$message)
  message("Skipping this report and continuing...")
  return(NULL)})

# MZ Report 3: Actual Rate, with Lag
mz_report_3 <- tryCatch({
  generate_absolute_performance_report(
    F_model = F_TR_3,
    Actual_values = Actuals,
    H = H,
    model_caption = "Mincer-Zarnowitz Test: Actual Rate, with Lag",
    format = format)}, error = function(e) {
  message("Error generating MZ Report (Actual Rate, with Lag): ", e$message)
  message("Skipping this report and continuing...")
  return(NULL)})

# MZ Report 4: Shadow Rate, with Lag
mz_report_4 <- tryCatch({
  generate_absolute_performance_report(
    F_model = F_TR_4,
    Actual_values = Actuals,
    H = H,
```

Table 10: Mincer-Zarnowitz Test: Actual Rate, No Lag

h	Alpha	Beta	pv(Joint)
1	1.2076	0.3116	0.184
2	1.2531	0.3795	0.756
3	0.9920	0.8033	0.857
4	0.8109	1.2440	0.218
5	0.8868	1.4210	0.007 ***
6	1.0360	1.4710	0.000 ***
7	1.2366	1.3372	0.000 ***
8	1.5904	1.0953	0.000 ***
9	2.1531	0.6397	0.000 ***
10	2.8582	0.0750	0.004 ***

Note: pv(Joint) is the p-value for the joint hypothesis H_0: (Alpha, Beta) = (0, 1). A high p-value means we fail to reject the null hypothesis of an unbiased, efficient forecast. * Signif. codes: '***' 0.01, '**' 0.05, '*' 0.1

```
    model_caption = "Mincer-Zarnowitz Test: Shadow Rate, with Lag",
    format = format)}, error = function(e) {
  message("Error generating MZ Report (Shadow Rate, with Lag): ", e$message)
  message("Skipping this report and continuing...")
  return(NULL)})

# MZ Report 5: Benchmark
mz_report_BM <- tryCatch({
  generate_absolute_performance_report(
    F_model = F_BM,
    Actual_values = Actuals,
    H = H,
    model_caption = "Mincer-Zarnowitz Test: Benchmark ARIMA",
    format = format)}, error = function(e) {
  message("Error generating MZ Report (Benchmark ARIMA): ", e$message)
  message("Skipping this report and continuing...")
  return(NULL)})

list(mz_report_1, mz_report_2, mz_report_3, mz_report_4, mz_report_BM)
```

[[1]]

[[2]]

[[3]]

[[4]]

[[5]]

Table 11: Mincer-Zarnowitz Test: Shadow Rate, No Lag

h	Alpha	Beta	pv(Joint)
1	1.8200	-0.3633	0.000 ***
2	1.8598	-0.2299	0.000 ***
3	1.7793	-0.0457	0.000 ***
4	1.8299	0.0095	0.000 ***
5	1.9916	-0.0130	0.000 ***
6	2.1898	-0.0409	0.000 ***
7	2.4185	-0.0665	0.000 ***
8	2.6798	-0.0894	0.000 ***
9	2.8483	-0.0617	0.000 ***
10	3.0489	-0.0389	0.000 ***

Note:
pv(Joint) is the p-value for the joint hypothesis H_0: (Alpha, Beta) = (0, 1). A high p-value means we fail to reject the null hypothesis of an unbiased, efficient forecast.
* Signif. codes: '***' 0.01, '**' 0.05, '*' 0.1

Table 12: Mincer-Zarnowitz Test: Actual Rate, with Lag

h	Alpha	Beta	pv(Joint)
1	0.0547	1.0015	0.793
2	0.2077	0.9484	0.793
3	0.4454	0.8750	0.731
4	0.7365	0.7927	0.655
5	1.1190	0.6535	0.492
6	1.4765	0.5449	0.372
7	1.8810	0.3823	0.108
8	2.2834	0.1895	0.023 **
9	2.6784	0.0002	0.001 ***
10	3.0516	-0.1807	0.000 ***

Note:
pv(Joint) is the p-value for the joint hypothesis H_0: (Alpha, Beta) = (0, 1). A high p-value means we fail to reject the null hypothesis of an unbiased, efficient forecast.
* Signif. codes: '***' 0.01, '**' 0.05, '*' 0.1

Table 13: Mincer-Zarnowitz Test: Shadow Rate, with Lag

h	Alpha	Beta	pv(Joint)
1	0.0681	0.9143	0.087 *
2	0.1930	0.7986	0.186
3	0.4392	0.6389	0.007 ***
4	0.7626	0.4854	0.000 ***
5	1.1300	0.3526	0.000 ***
6	1.4534	0.2523	0.000 ***
7	1.8018	0.1659	0.000 ***
8	2.1810	0.0875	0.000 ***
9	2.6095	0.0196	0.000 ***
10	3.0870	-0.0420	0.000 ***

Note:
pv(Joint) is the p-value for the joint hypothesis H_0: (Alpha, Beta) = (0, 1). A high p-value means we fail to reject the null hypothesis of an unbiased, efficient forecast.
* Signif. codes: '***' 0.01, '**' 0.05, '*' 0.1

Table 14: Mincer-Zarnowitz Test: Benchmark ARIMA

h	Alpha	Beta	pv(Joint)
1	0.1487	0.9447	0.382
2	0.3943	0.8503	0.193
3	0.7096	0.7125	0.197
4	1.0656	0.5675	0.290
5	1.4463	0.4072	0.146
6	1.8131	0.2622	0.144
7	2.1639	0.1192	0.006 ***
8	2.4781	-0.0143	0.001 ***
9	2.7540	-0.1508	0.000 ***
10	2.9915	-0.2735	0.000 ***

Note:
pv(Joint) is the p-value for the joint hypothesis H_0: (Alpha, Beta) = (0, 1). A high p-value means we fail to reject the null hypothesis of an unbiased, efficient forecast.
* Signif. codes: '***' 0.01, '**' 0.05, '*' 0.1

Relative Performance (against benchmark)

```
# in final markdown, input just the relevant code, i.e. the tests made
# who cares about code to create tables, that stays in helpers
source(here("helpers/pseudo_outofsample_tests.R"))
```

```
# Call DM-test helper function 4 times.

# Report 1: Actual Rate, No Lag
report_1 <- generate_relative_performance_report(
  FE_TR_model = FE_TR_1,
  FE_BM_model = FE_BM,
  H = H,
  model_caption = "MSFE Comparison, Trained on Actual Rate, No Lag",
  format = format)

# Report 2: Shadow Rate, No Lag
report_2 <- generate_relative_performance_report(
  FE_TR_model = FE_TR_2,
  FE_BM_model = FE_BM,
  H = H,
  model_caption = "MSFE Comparison, Trained on Shadow Rate, No Lag",
  format = format)

# Report 3: Actual Rate, with Lag
report_3 <- generate_relative_performance_report(
  FE_TR_model = FE_TR_3,
  FE_BM_model = FE_BM,
  H = H,
  model_caption = "MSFE Comparison, Trained on Actual Rate, with Lag",
  format = format)

# Report 4: Shadow Rate, with Lag
report_4 <- generate_relative_performance_report(
  FE_TR_model = FE_TR_4,
  FE_BM_model = FE_BM,
  H = H,
  model_caption = "MSFE Comparison, Trained on Shadow Rate, with Lag",
  format = format)

list(report_1, report_2, report_3, report_4)
```

[[1]]

[[2]]

[[3]]

[[4]]

Table 15: MSFE Comparison, Trained on Actual Rate, No Lag

h	MSFE TR	MSFE BM	Ratio	DM Two-Sided	DM Greater	DM Lesser
1	3.9517	0.1591	24.8393	0.000 ***	1.000	0.000 ***
2	3.9732	0.6458	6.1521	0.032 **	0.984	0.016 **
3	3.4469	1.6062	2.1459	0.386	0.807	0.193
4	3.1086	2.8224	1.1014	0.915	0.542	0.458
5	3.0938	4.3368	0.7134	0.603	0.302	0.698
6	3.2794	5.8787	0.5578	0.124	0.062 *	0.938
7	3.5234	7.4570	0.4725	0.000 ***	0.000 ***	1.000
8	4.0667	9.0333	0.4502	0.001 ***	0.000 ***	1.000
9	4.8839	10.1786	0.4798	0.000 ***	0.000 ***	1.000
10	5.8798	11.5721	0.5081	0.000 ***	0.000 ***	1.000

Note: TR refers to the forecast made with an estimated Taylor Rule. BM refers to a benchmark of the interest rate using an ARIMA model. Ratio < 1 indicates that the TR model has lower MSFE.
DM Test Alternative Hypotheses (H_A): * 'DM Greater' tests if the TR model is significantly more accurate than the BM model. † 'DM Lesser' tests if the TR model is significantly less accurate than the BM model.

Table 16: MSFE Comparison, Trained on Shadow Rate, No Lag

h	MSFE TR	MSFE BM	Ratio	DM Two-Sided	DM Greater	DM Lesser
1	8.7557	0.1591	55.0357	0.000 ***	1.000	0.000 ***
2	9.4762	0.6458	14.6728	0.002 ***	0.999	0.001 ***
3	9.4969	1.6062	5.9125	0.030 **	0.985	0.015 **
4	12.0493	2.8224	4.2692	0.209	0.895	0.105
5	16.4410	4.3368	3.7910	0.313	0.844	0.156
6	22.9412	5.8787	3.9024	0.363	0.819	0.182
7	30.3125	7.4570	4.0650	0.384	0.808	0.192
8	40.6792	9.0333	4.5032	0.336	0.832	0.168
9	48.6339	10.1786	4.7781	0.256	0.872	0.128
10	59.1154	11.5721	5.1084	0.144	0.928	0.072 *

Note: TR refers to the forecast made with an estimated Taylor Rule. BM refers to a benchmark of the interest rate using an ARIMA model. Ratio < 1 indicates that the TR model has lower MSFE.
DM Test Alternative Hypotheses (H_A): * 'DM Greater' tests if the TR model is significantly more accurate than the BM model. † 'DM Lesser' tests if the TR model is significantly less accurate than the BM model.

Table 17: MSFE Comparison, Trained on Actual Rate, with Lag

h	MSFE TR	MSFE BM	Ratio	DM Two-Sided	DM Greater	DM Lesser
1	0.1335	0.1591	0.8393	0.491	0.246	0.754
2	0.4435	0.6458	0.6866	0.196	0.098 *	0.902
3	0.9719	1.6062	0.6051	0.136	0.068 *	0.932
4	1.5987	2.8224	0.5664	0.104	0.052 *	0.948
5	2.5347	4.3368	0.5845	0.037 **	0.018 **	0.982
6	3.4632	5.8787	0.5891	0.008 ***	0.004 ***	0.996
7	4.7305	7.4570	0.6344	0.009 ***	0.004 ***	0.996
8	6.2083	9.0333	0.6873	0.002 ***	0.001 ***	0.999
9	7.9732	10.1786	0.7833	0.013 **	0.006 ***	0.994
10	9.7356	11.5721	0.8413	0.080 *	0.040 **	0.960

Note: TR refers to the forecast made with an estimated Taylor Rule. BM refers to a benchmark of the interest rate using an ARIMA model. Ratio < 1 indicates that the TR model has lower MSFE.
DM Test Alternative Hypotheses (H_A): [*] 'DM Greater' tests if the TR model is significantly more accurate than the BM model.
[†] 'DM Lesser' tests if the TR model is significantly less accurate than the BM model.

Table 18: MSFE Comparison, Trained on Shadow Rate, with Lag

h	MSFE TR	MSFE BM	Ratio	DM Two-Sided	DM Greater	DM Lesser
1	0.1591	0.1591	1.0000	1.000	0.500	0.500
2	0.5536	0.6458	0.8571	0.715	0.357	0.643
3	1.5969	1.6062	0.9942	0.946	0.473	0.527
4	3.5822	2.8224	1.2692	0.371	0.815	0.185
5	6.7917	4.3368	1.5661	0.230	0.885	0.115
6	11.0846	5.8787	1.8856	0.076 *	0.962	0.038 **
7	17.4023	7.4570	2.3337	0.079 *	0.960	0.040 **
8	26.3667	9.0333	2.9188	0.100	0.950	0.050 *
9	37.3973	10.1786	3.6741	0.096 *	0.952	0.048 **
10	51.5048	11.5721	4.4508	0.056 *	0.972	0.028 **

Note: TR refers to the forecast made with an estimated Taylor Rule. BM refers to a benchmark of the interest rate using an ARIMA model. Ratio < 1 indicates that the TR model has lower MSFE.
DM Test Alternative Hypotheses (H_A): [*] 'DM Greater' tests if the TR model is significantly more accurate than the BM model.
[†] 'DM Lesser' tests if the TR model is significantly less accurate than the BM model.

Table 19: For model based on: Taylor Rule Formula 3

Horizon: Quarter	Taylor Rule Forecast	Benchmark Forecast	Inflation Forecast	Output Gap Forecast
1: 2025 Q4	2.00	2.00	2.17	1.24
2: 2026 Q1	1.75	2.25	2.03	1.38
3: 2026 Q2	1.75	2.25	2.10	1.33
4: 2026 Q3	1.75	2.25	2.09	1.11
5: 2026 Q4	1.75	2.25	2.09	0.79
6: 2027 Q1	1.50	2.25	2.08	0.42
7: 2027 Q2	1.50	2.25	2.08	0.07
8: 2027 Q3	1.50	2.25	2.07	-0.22
9: 2027 Q4	1.50	2.25	2.07	-0.42
10: 2028 Q1	1.25	2.25	2.07	-0.52

Actual Forecast Model

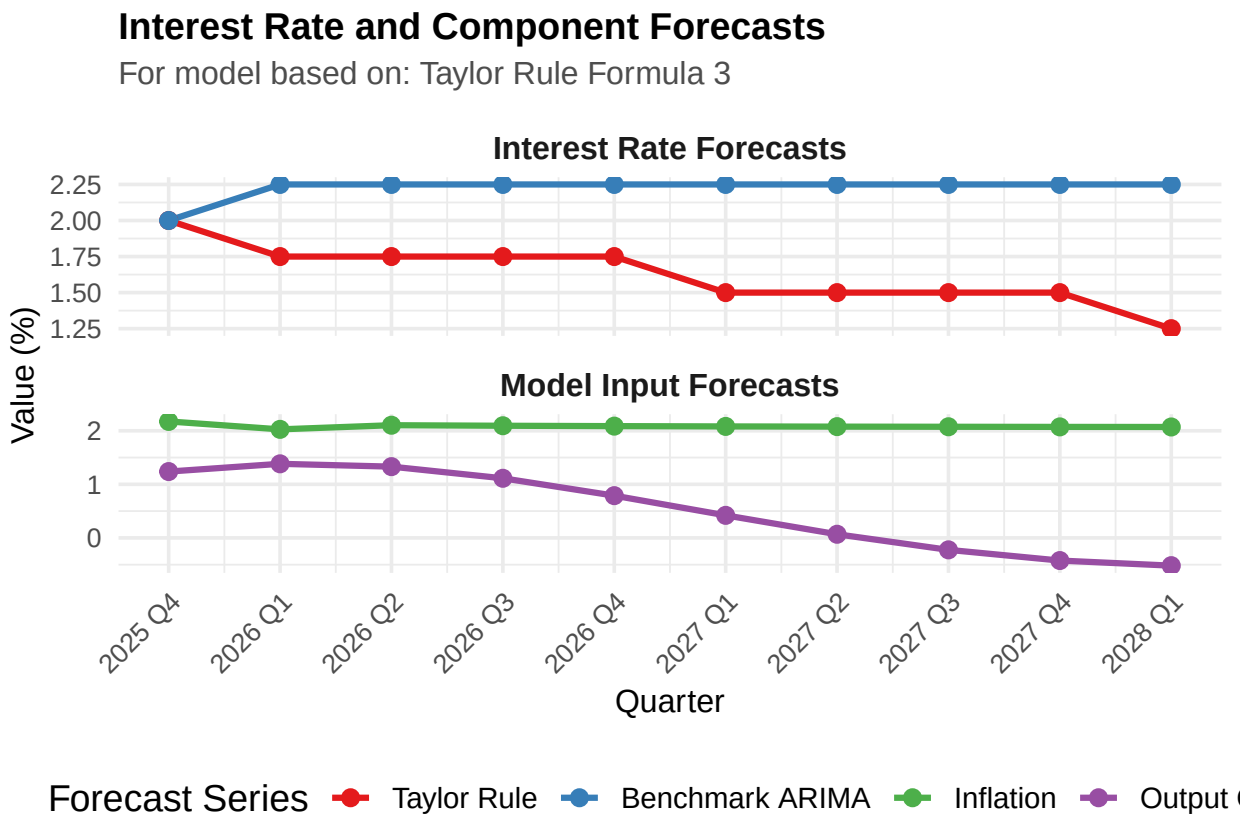
Forecasting

```
source(here("helpers/actual_forecast_estimator.R"))

final_forecasts <- our_predict(data = data, formula = model_formula, H = H)
```

[1] "This model is fitted to the variable 'Inflation Gap' specified as ARIMA(1, 1, 4)." [1] "This model is fitted to the variable 'Output Gap' specified as ARIMA(2, 0, 3)." [1] "This model is fitted to the variable 'Interest Rate' specified as ARIMA(2, 1, 0)."

```
display_forecasts(final_forecasts,
                  caption = paste("For model based on:", model_name),
                  format = format)
```

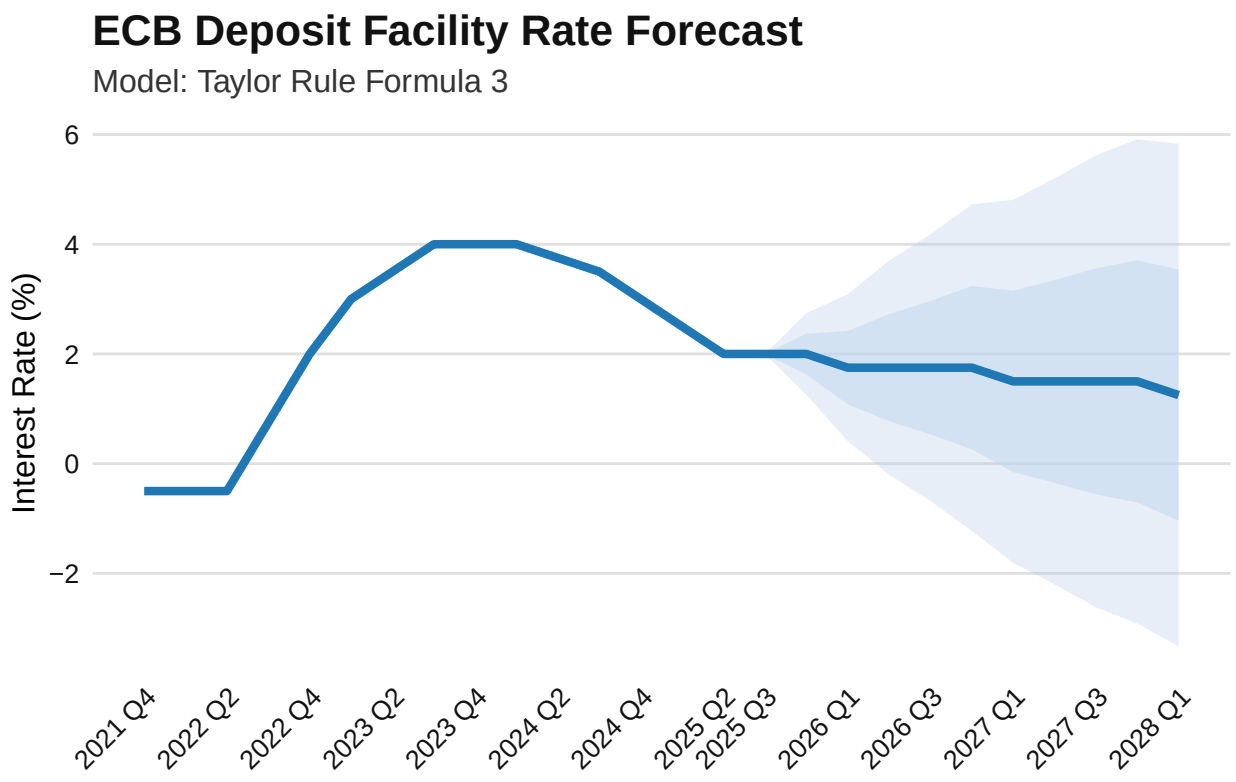
```
plot_forecasts(final_forecasts)
```



Prediction Intervals

Prepare Data

```
source(here("helpers/actual_forecast_estimator.R"))

# Computes prediction intervals
final_interval <- our_predict_intervals(estimated_variance = var_by_horizon,
                                        forecast = final_forecasts)

# Plots prediction intervals
plot_forecasts_pred_int(data, intervals = final_interval)
```


ECB Deposit Facility Rate Forecast
Model: Taylor Rule Formula 3
Interest Rate (%)
6
4
2
0
−2
2021 Q4
2022 Q2
2022 Q4
2023 Q2
2023 Q4
2024 Q2
2024 Q4
2025 Q2
2025 Q3
2026 Q1
2026 Q3
2027 Q1
2027 Q3
2028 Q1
Shaded areas: ±1 S.D. (dark) / ±2 S.D. (light)