# Forecast Comparisons

In this example, we want to see if the ARMA model would have produced better forecasts than the mean. To do that, we'll need to construct *pseduo-OOS forecasts*. The definition of a pseudo-OOS forecast that we'll use in this class is as follows.

**Pseudo-OOS forecast:** A forecast of an observation in our dataset that's made using only data that would have been available at an earlier date.

Suppose you have a monthly dataset that runs through December 2025. If you use data through to December 2019 to estimate a forecasting model, and then calculate a forecast of the January 2020 observation, you've made a pseudo-OOS forecast. The January 2020 observation is already in your dataset.

Why would you make a forecast of the January 2020 observation if that value is already in your dataset? You would do that because you could then compare the forecast against the actual value and compute a forecast error.

It would be unusual to compute only one pseudo-OOS forecast. In order to properly evaluate a forecasting model's performance, you need a reasonable sample size. In practice, it would be unusual to make fewer than 30 pseudo-OOS forecasts, because even a sample of size 30 is very small to be doing inference.

An *out-of-sample forecast comparison* is done to compare the accuracy of two or more forecasting models. Out-of-sample forecast comparisons are done by computing a series of pseudo-OOS forecasts for each model. A statistic such as mean squared error (MSE) is computed for each model and the models can then be ranked in order from best to worst.

A critical point is that this is a measure of *past* forecast performance over a specific time period. Model performance can change in the future. Nonetheless, when you actually want to put a forecasting model in production, you'll want to know how well it would have performed in the past, and how its performance would compare with other models you could have used.

# Comparisons With Simple Forecasting Models

There are costs associated with building a sophisticated forecasting model. Someone needs to do the initial work to come up with the optimal model specification, data may need to be purchased, the model needs to be updated every time new data is released, and the updated forecasts need to be communicated with others. During periods like the pandemic of 2020 or the government shutdown of 2025, new data might not be released on schedule, or the data is unreliable. The person or team in charge of producing the forecasts may have to spend hours deciding on the best way to proceed.

It's reasonable to ask if the time and money invested in maintaining a sophisticated forecasting model is worth it. Simple models, such as the random walk or the sample average, are alternatives that come without a meaningful cost. The random walk forecast is equal to the most recent value in your dataset. The mean forecast is nothing but the sample mean. A

minimal standard for a sophisticated forecasting model to meet is that it performs better than a simple forecasting model. If they perform about the same, or even worse, if the simple forecasting model performs better, there is no reason to invest time building and maintaining the sophisticated model.

Let's see how these comparisons are done using an example from gasoline markets. We will compare the forecast performance of the mean forecast against the forecast performance of an ARMA model. Here are the relevant parameters of the comparison:

- We will produce pseudo-OOS forecasts of the change in the price of gasoline for each month from January 2000 to December 2025. This will give us 312 forecasts and 312 forecast errors to work with. This *sample split* is roughly 50/50 (50% used for the initial estimation and predictions being calculated for the last 50% of the sample).
- We will produce forecasts for $h = 1$, i.e., we will only consider one-step ahead forecasts. We could alternatively use a larger value of $h$, such as $h = 6$, which would correspond to evaluating the performance for forecasts of the price of gasoline six months ahead of time. Or we could look at multiple time horizons. For this example, we will only consider $h = 1$.
- The metric for comparison is the *mean squared prediction error*. This is the most common metric for comparison of forecasts.
- We will limit ourselves to the ARMA(1,1) model to keep the example simple. In practice, you'd want to select the lag lengths of the ARMA model before estimation. That's an unnecessary complication for the purposes of this example.

# Example: Gasoline price forecasting

Load the data and compute the percentage change in gas price:

```
library(tstools)
gas <- import.fred("https://raw.githubusercontent.com/bachmeil/econ890spr26/refs/heads/m
dgas <- pctChange(gas)
start(dgas)
end(dgas)
```

That gives you the start and end dates of the gasoline series. Making the first forecast in January 2000 approximately gives us a 50/50 sample split (at least 50% of the sample for model estimation and making forecasts for the last 50% of observations).

Now make pseduo-OOS forecasts for the mean model. To forecast January 2000, we would have estimated the mean on data through December 1999:

```
mean(window(dgas, end=c(1999,12)))
```

The forecast for January 2000 would have been a 0.313% increase. Now make a forecast of February 2000, but since we'd have had the January 2000 data when we made that forecast, we have to do a new estimation:

```
mean(window(dgas, end=c(2000,1)))
```

Repeat the process to make a forecast for March 2000 using data through February 2000:

```
mean(window(dgas, end=c(2000,2)))
```

There's nothing *complicated* about this, but it's *ridiculously repetitive*. There's no way you can manually calculate all of the pseudo-OOS forecasts in any reasonable forecast comparison. We have to find a way to automate the process. A convenient way to automate this type of process in R is to use the `lapply` function, so let's see how to do that.

# Automating the Process

First, let's define the end dates for the samples we'll be using. Since we want to forecast January 2000 through December 2025, and our forecast horizon is $h = 1$, we will do the estimation for the first forecast using data through December 1999, and for the last forecast using data through November 2025.

```
estimation.dates <- dates(c(1999,12), c(2025,11), 12)
estimation.dates
```

The other thing we need is a function that estimates the mean if you give it the end date of the sample:

```
mean.forecast.calc <- function(e) {
  mu <- mean(window(dgas, end=e))
  return(mu)
}
```

For instance, if we want to make a forecast using data through January 2010, we'd do this:

```
mean.forecast.calc(c(2020,1))
```

How do we make forecasts for *all* of the sample end dates in `estimation.dates`? The easiest way is to use the `lapply` function. The first argument will be the estimation end dates and the second argument will be the function. `lapply` goes through the estimation end dates one by one, evaluating `mean.forecast.calc` repeatedly and capturing the output:

```
mean.forecasts.raw <- lapply(estimation.dates, mean.forecast.calc)
mean.forecasts.raw
```

The only downside to `lapply` is that it returns the output as a list, which would be hard to work with further. We'll use the tstools function `collect` to convert the list to a vector:

```
mean.forecasts.vector <- collect(mean.forecasts.raw, output="numeric")
```

The first argument is the list we want to convert. The `output` parameter says to save the values as a numerical vector. Finally, let's convert the forecast vector to a time series object. (In the future, `collect` will be modified to allow a direct conversion to a time series, but that functionality has not yet been implemented, so we need to do it manually.)

```
mean.forecasts <- ts(mean.forecasts.vector, start=c(2000,1),
  frequency=12)
```

Now you can plot the forecasts:

```
plot(mean.forecasts)
```

We care about the mean squared forecast error of the models, so calculate the forecast errors we would have seen in the past:

```
mean.errors <- window(dgas,start=c(2000,1)) - mean.forecasts
```

And then calculate the MSE for the mean forecasting model:

```
mean(mean.errors^2)
```

Now we need to repeat the process for the ARMA(1,1) model. I will provide the code without explanation, since it's a modification of the earlier code:

```
arma.forecast.calc <- function(e) {
  dgas.sample <- window(dgas, end=e)
  fit <- armafit(dgas.sample, 1, 1)
  return(prediction(fit, 1))
}

arma.forecasts.raw <- lapply(estimation.dates,
                             arma.forecast.calc)

arma.forecasts.vector <- collect(arma.forecasts.raw, output="numeric")
arma.forecasts <- ts(arma.forecasts.vector, start=c(2000,1),
  frequency=12)
arma.errors <- window(dgas,start=c(2000,1)) - arma.forecasts
mean(arma.errors^2)
```

And we see that the MSE of the ARMA(1,1) model is lower than the mean forecasting model. Hence, it is reasonable to continue putting in the extra effort to maintain the ARMA(1,1) forecasts.

## Generalization

There is nothing in this example that's specific to the mean forecasting model or the ARMA(1,1) model. Every pseudo-OOS forecast comparison follows the same steps to compute the forecasts. You might compute other statistics for the forecast errors, you might change the model, etc., but this same template can always be used.