Homework 4: [YOUR NAME HERE]

Introduction to Time Series, Fall 2024

Due Friday November 8

The total number of points possible for this homework is 34. The number of points for each question is written below, and questions marked as "bonus" are optional (points awarded for bonus problems can be used to earn back points that you may have lost on other parts of this homework but will not put you above full credit). Submit the **knitted pdf file** from this Rmd to Gradescope.

If you collaborated with anybody for this homework, put their names here:

Backshift commuting

1. (1 pt) Let B denote the backshift operator. Given any integers $k, \ell \geq 0$, explain why $B^k B^\ell = B^\ell B^k$.
2. (2 pts) Using Q1, if $\phi_1, \dots, \phi_k$ and $\varphi_1, \dots, \varphi_\ell$ are any coefficients, show that

$$(1 + \phi_1 B + \cdots + \phi_k B^k)(1 + \varphi_1 B + \cdots + \varphi_\ell B^\ell) = (1 + \varphi_1 B + \cdots + \varphi_\ell B^\ell)(1 + \phi_1 B + \cdots + \phi_k B^k)$$

3. (2 pts) Verify the result in Q2 with a small code example.
4. (3 pts) Using Q2, show that we can write a SARIMA model equivalently as

$$\phi(B)\Phi(B^s)\nabla^d \nabla_s^D x_t = \theta(B)\Theta(B^s) w_t$$

and

$$\Phi(B^s)\phi(B)\nabla_s^D \nabla^d x_t = \Theta(B^s)\theta(B) w_t.$$

Long-range ARIMA

5. (1 pt) Let $\nabla = 1 - B$ denote the difference operator. Suppose that $\nabla x_t = 0$ for all t. Prove that x_t must be a constant sequence.
6. (2 pts) Suppose that $\nabla x_t = u$ for all t, where u is an arbitrary constant. Prove that x_t must be a linear function of t, of the form $x_t = a + bt$.
7. (3 pts) Suppose that $\nabla x_t = u + vt$ for all t, where u, v are again arbitrary constants. Prove that x_t must be a quadratic function of t, of the form $x_t = a + bt + ct^2$.
8. (1 pt) Using Q6 and Q7, prove that if $\nabla^2 x_t = u$ for all t, where u is a constant, then x_t must be a quadratic function of t.
9. (4 pts) Consider an ARMA(1,1) model:

$$(1 - \phi B)x_t = (1 + \theta B)w_t.$$

Suppose that our estimates pass the "unit root test", $|\hat{\phi}|, |\hat{\theta}| < 1$, which we will assume implicitly henceforth. Unravel the forecast iteration described in lecture to show that the forecast $\hat{x}_{t+h|t}$ from this ARMA model approaches zero as $h \to \infty$.

10. (2 pts) Consider an ARMA(1,1) model, with intercept:

$$(1-\phi B)x_t = c + (1+\theta B)w_t.$$

Unravel the forecast iteration to show that $\hat{x}_{t+h|t}$ approaches a nonzero constant as $h \to \infty$.

11. (Bonus) Now consider the extension to ARIMA$(1, d, 1)$:

$$(1-\phi B)\nabla^d x_t = c + (1+\theta B)w_t.$$

Use Q5–Q10 to argue the following:

- If $c = 0$ and $d = 1$, then $\hat{x}_{t+h|t}$ approaches a constant as $h \to \infty$.
- If $c = 0$ and $d = 2$, then $\hat{x}_{t+h|t}$ approaches a linear trend as $h \to \infty$.
- If $c \neq 0$ and $d = 1$, then $\hat{x}_{t+h|t}$ approaches a linear trend as $h \to \infty$.
- If $c \neq 0$ and $d = 2$, then $\hat{x}_{t+h|t}$ approaches a quadratic trend as $h \to \infty$.

Time series CV

When we learned time series cross-validation in lecture (weeks 3-4, "Linear regression and prediction"), we implemented it "manually", by writing a loop in R to iterate over time, rebuild models, and so on. The `fable` package in R does it differently. It relies on data being stored in a class that is known as a `tsibble`, which is like a special data frame for time series. You can then use a function called `stretch_tsibble()` in order to "prepare it" for time series cross-validation. Take a look at what it does with this example:

```r
library(tidyverse)
library(fpp3)

dat = tsibble(date = as.Date("2023-10-01") + 0:9,
              value = 1:10 + rnorm(10, sd = 0.25),
              index = date)
dat
```

```
## # A tsibble: 10 x 2 [1D]
##    date       value
##    <date>     <dbl>
##  1 2023-10-01 0.847
##  2 2023-10-02 2.10
##  3 2023-10-03 3.09
##  4 2023-10-04 4.00
##  5 2023-10-05 5.18
##  6 2023-10-06 6.19
##  7 2023-10-07 7.10
##  8 2023-10-08 8.10
##  9 2023-10-09 8.71
## 10 2023-10-10 9.73
```

```r
dat_stretched = dat |> stretch_tsibble(.init = 3)
dat_stretched
```

```
## # A tsibble: 52 x 3 [1D]
## # Key:       .id [8]
##    date       value   .id
##    <date>     <dbl> <int>
##  1 2023-10-01 0.847     1
##  2 2023-10-02 2.10      1
##  3 2023-10-03 3.09      1
##  4 2023-10-01 0.847     2
##  5 2023-10-02 2.10      2
##  6 2023-10-03 3.09      2
##  7 2023-10-04 4.00      2
##  8 2023-10-01 0.847     3
##  9 2023-10-02 2.10      3
## 10 2023-10-03 3.09      3
## # i 42 more rows
```

What this does is it takes the first 3 entries of the time series and assigns them `.id = 1`. Then it appends the first 4 entries of the time series and assigns them `.id = 2`. Then it appends the first 5 entries of the time series and assigns them `.id = 3`, and so on. Downstream, when we go to fit a forecast model with fable, it (by default) will fit a separate model to the data in each level of the `.id` column. And by making forecasts at a (say) horizon `h = 1`, these are actually precisely the 1-step ahead forecasts that we would generate in time series CV:

```r
dat_fc = dat_stretched |>
  model(RW = RW(value ~ drift())) |>
  forecast(h = 1)
dat_fc
```

```
## # A fable: 8 x 5 [1D]
## # Key:     .id, .model [8]
##      .id .model date                value .mean
##    <int> <chr>  <date>             <dist> <dbl>
## 1      1 RW     2023-10-04 N(3.5, 0.073)  3.53
## 2      2 RW     2023-10-05 N(4.6, 0.065)  4.60
## 3      3 RW     2023-10-06 N(5.8, 0.068)  5.78
## 4      4 RW     2023-10-07 N(6.8, 0.052)  6.78
## 5      5 RW     2023-10-08 N(8.1, 0.069)  8.13
## 6      6 RW     2023-10-09  N(9.3, 0.06)  9.26
## 7      7 RW     2023-10-10  N(9.5, 0.11)  9.51
## 8      8 RW     2023-10-11   N(11, 0.16) 11.2
```

The `.mean` column give us the point forecast. To evaluate these, we could join the original data `dat` to the point forecasts in `dat_fc`, and then align by the `date` column, and compute whatever metrics we wanted. However, there is also a handy function to do all of this for us, called `accuracy()`. This computes a bunch of common metrics, and here we just pull out the `MAE` column:

```r
accuracy(dat_fc, dat) |> select(.model, MAE)
```

```
## # A tibble: 1 x 2
##   .model   MAE
##   <chr>  <dbl>
## 1 RW     0.365
```

Now for the questions.

12. (3 pts) A clear advantage to the above workflow is convenience: we have to write less code. A disadvantage is that it can be inefficient, and in particular, memory inefficient. To see this, consider using this to do time series CV on a sequence with n observations and burn-in time t_0. We store this as a `tsibble`, call it `x`, with `n` rows, and then we run `stretch_tsibble(x, .init = t0)`. How many rows does the output have? Derive the answer mathematically (as an explicit formula involving n, t_0), and then verify it with a couple of code examples.

13. (4 pts) Show that the MAE result for the random walk forecasts produced above, on the data in `dat`, matches the MAE from a manual implementation of time series CV with the same forecaster. (Your manual implementation can build off the code from the regression lecture, and/or from previous

homeworks.)

14. (6 pts) Consider the `leisure` data set from the HA book, which the code excerpt below (taken from the ARIMA lecture) prepares for us. Use time series CV, implemented using `stretch_tsibble()`, `model()`, and `forecast()`, as described above, to evaluate the MAE of the following four models:

- $\text{ARIMA}(2,1,0)$
- $\text{ARIMA}(0,1,2)$
- $\text{ARIMA}(2,1,0)(1,1,0)_{12}$
- $\text{ARIMA}(0,1,2)(0,1,1)_{12}$

The last models are motivated by the exploratory analysis done in lecture. The first two remove the seasonal component, which should not be very good (since there is clear seasonality in the data). For each model you should use a burn-in period of length 50 (i.e., set `.init = 50` in the call to `stretch_tsibble()`). A key difference in how you implement time series CV to the above examples: you should consider 1-step, 2-step, all the way through 12-step ahead forecasts. But do not worry! This can be handled with an appropriate call to `forecast()`. For each model, calculate the MAE by averaging over all forecast horizons (1 through 12). Report the results and rank the models by their MAE.

```r
leisure = us_employment |>
  filter(Title == "Leisure and Hospitality", year(Month) > 2000) |>
  mutate(Employed = Employed/1000) |>
  select(Month, Employed)
```

15. (Bonus) Break down the MAE for the forecasts made in Q14 by forecast horizon. That is, for each $h = 1, \dots, 12$, calculate the MAE of the h-step ahead forecasts made by each model. Make a plot with the horizon h on the x-axis and MAE on the y-axis, and compare in particular the models $\text{ARIMA}(2,1,0)(1,1,0)_{12}$ and $\text{ARIMA}(0,1,2)(0,1,1)_{12}$. Do you see anything interesting happening here in the comparison between their MAE as we vary the horizon h?

16. (Bonus^2) Evaluate the forecasts made by auto-ARIMA in this time series CV pipeline. Remember, this means that auto-ARIMA will be rerun (yikes!) at each iteration in time series CV. This may take a very long time to run (which is why this is a Bonus^2). If it finishes for you, how does its MAE compare?