Homework 5: [YOUR NAME HERE]

Introduction to Time Series, Fall 2024

Due Friday November 22

The total number of points possible for this homework is 48. 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:

Notes about this homework

This homework is an exercise in Covid-19 forecasting. You will build some basic forecasters of Covid-19 deaths at the national (U.S.) level, based on frameworks you've learned in the last few lectures: ARIMA and ETS. You should know that, in true (prospective) Covid-19 forecasting, the situation is much harder than the one you are facing in this homework. This is because of **data revisions**: the forecasters in true (prospective) Covid-19 forecasting did not have access to the same data in real-time that you have access to now, in retrospect. Instead, they had access to preliminary data that was subject to revisions, sometimes very large and irregular ones, making forecasting much harder. See, e.g., McDonald et al. (2021), for a discussion of the impact of revisions on forecasting.

Also, in most operational forecasting enterprises, epidemic/pandemic forecasting included, we would typically be trying to leverage exogenous signals and sources of information to guide to our forecasts (beyond statistical models like ARIMA or ETS which only rely on historical information about the target of interest). Here we will pursue this only in limited fashion, at the end of this homework.

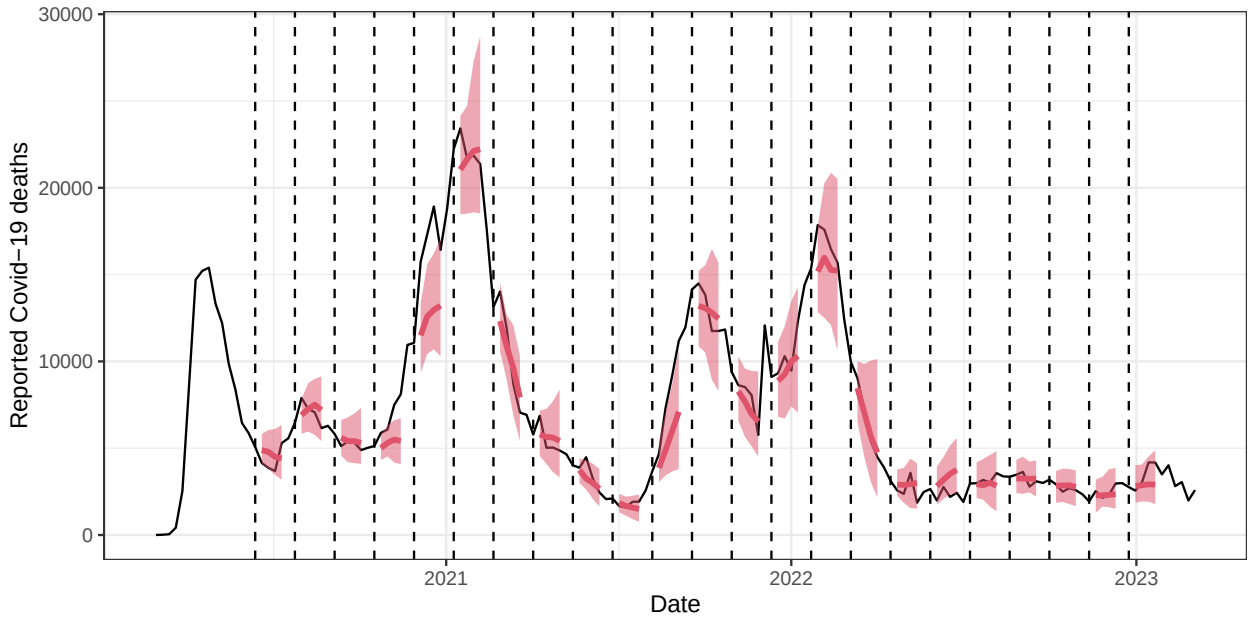
Covid-19 cases and deaths

First, download the data files `cases_deaths.csv` and `covidhub_fc.csv` from the course GitHub repo. The former contains weekly reported Covid-19 cases and death counts in the U.S., from February 29, 2020 to March 4, 2023. The latter contains 1-4 week ahead forecasts of weekly reported Covid-19 death counts, from June 13, 2020 to March 4, 2023. These forecasts were generated by the CovidHub ensemble model, which is an ensemble of all qualifying forecast submissions to the U.S. Covid-19 Forecast Hub, and was the basis of official CDC communications during the pandemic. See, e.g., Ray et al. (2022) for discussion of the ensemble model.

The code below loads in these data frames (which you can run once the downloaded files are in your working directory) and plots death curves along a set of 1-4 week ahead forecasts, across the pandemic.

```
library(tidyverse)

cases_deaths = read.csv("cases_deaths.csv")
covidhub_fc = read.csv("covidhub_fc.csv")

cases_deaths = cases_deaths |>
  mutate(date = as.Date(date))
```

```
covidhub_fc = covidhub_fc |>
  mutate(target_date = as.Date(target_date))
fc_dates = seq.Date(
  as.Date("2020-06-13"),
  as.Date("2023-01-01"),
  by = 6 * 7)

g = ggplot() +
  geom_line(data = cases_deaths, aes(x = date, y = deaths)) +
  labs(x = "Date", y = "Reported Covid-19 deaths") +
  theme_bw() + theme(legend.position = "none")

layers = list()
for (fc_date in fc_dates) {
  layers = c(
    layers,
    geom_vline(xintercept = fc_date, linetype = 2),
    geom_ribbon(data = covidhub_fc |>
                  mutate(forecast_date = target_date - h * 7) |>
                  filter(forecast_date == fc_date),
                aes(x = target_date, ymin = forecast_0.1, ymax = forecast_0.9),
                fill = 2, alpha = 0.5),
    geom_line(data = covidhub_fc |>
                mutate(forecast_date = target_date - h * 7) |>
                filter(forecast_date == fc_date),
              aes(x = target_date, y = forecast_0.5),
              color = 2, linewidth = 1.25))
}

g + layers
```



As you can see from the plot, the ensemble model produces both point forecasts and prediction intervals. In fact, the prediction intervals are derived from predicted quantiles. The point forecast is stored in the `forecast_0.5` column, whereas the 80% prediction interval is formed by taking the difference of the

`forecast_0.9` and `forecast_0.1` columns (and similarly for other levels). Lastly, the `h` column indicates horizon of the forecast (in weeks), and the `target_date` column denotes the target date (aligned by the end of the week), whose death count the ensemble model is predicting. The forecast date can hence be derived from `target_date - h * 7` (which, since `target_date` is stored as a `Date` variable, will be understood correctly to mean subtract off `h * 7` days from the given date).

Evaluating the ensemble model's MAE

1. (4 pts) Before building any forecasters of our own, we'll evaluate the ensemble model's mean absolute error (MAE). We won't be able to beat this with our models, but it'll give us a sense of how gold-standard forecasts perform, for this problem. First, you need join the death data to the forecast data. Using `left_join()`, join `covidhub_fc` and `cases_deaths` by date: specifically, the `target_date` variable in the first data set should be matched to the `date` variable in the second. As a check, you should see for that the 4 week ahead prediction for the target date July 4, 2020, the point forecast is 5169.654 and the death count is

2. After you've joined the data sets, compute the MAE of the point forecasts per horizon, report the results.

3. (2 pts) Using the same joined data set from Q1, compute the coverage of 80% prediction intervals, per horizon, and report the results.

Evaluating the baseline model's MAE

3. (5 pts) As one more thing to do before building our own models, we'll develop a simple baseline model and compute its MAE. Make a copy of the joined data from Q1, and overwrite the point forecasts with the true value of deaths from `h` weeks ago. In other words, this baseline simply predicts (at all horizons) that we will see the same number of new reported Covid-19 deaths as what we observed in the last week. (Its forecast trajectories are thus flat lines, extrapolating forward from the latest observation.) After you've done this, compute the MAE of the baseline forecasts, per horizon, and report the results.

4. (Bonus) Develop a method to form an 80% prediction interval around the baseline model's point forecasts. This should still be an ex-ante prediction in the sense that it should not be using data after the forecast date. Once you've done this, compute the coverage of these prediction intervals, per horizon, and report the results.

ARIMA models

5. (2 pts) Now we'll start building our own models, starting with ARIMA. Because the data is highly nonstationary (recall the plot above which showed death counts along with the ensemble forecasts), we'll consider differencing. Plot the first and second differences of the death data, and comment on what you find.

6. (5 pts) Fit two models: ARIMA(1,1,0) and ARIMA(2,1,0), using `ARIMA()` from the `fable` package. Each model should be fit only using the data up through June 6, 2020. (This is also therefore the forecast date.) These models should be fit *with* the inclusion of a constant but *without* any seasonality terms (specify the formula carefully to ensure this). After fitting these models, use these to make 1-4 week ahead forecasts and plot them (along with the true death counts through June 6, 2020) with `autoplot()`. Hint: before calling `fable::model()` you'll have to convert `cases_deaths`to a `tsibble` object, with `index = date`.

7. (4 pts) For the two models from Q6, check that their 1 week ahead point forecasts match the formulas you know underlie them. To do so, extract the coefficients from the fitted models, form the 1 week ahead predictions manually, and demonstrate that these match the given forecasts.

8. (6 pts) Implement time series cross-validation (CV) for these ARIMA models, ARIMA(1,1,0) and ARIMA(2,1,0), each with constants (and no seasonality). The simplest way to do this will be write a loop (to refresh yourself, look back at the lecture code from weeks 3-4, "Linear regression and prediction", or from the homeworks after that). As you saw in the last homework, the `fable` package provides its own way to run time series CV, but it will use up too much memory (and be too cumbersome for the computations that involve exogenous features, later).

   The code below provides a scaffold that you should build on, where we run time series CV over June 6, 2020 to February 4, 2023. Make sure to explicitly define the horizon `h` in forecasts stored in the `fc` object, which will be useful for subsequent computations.The end result of this question will be a `tibble`, called `fable_fc`, which contains all of the 1-4 week ahead forecasts made in time series CV by your two ARIMA models. We will continue to append forecasts to this `tibble` in subsequent questions, as we build more models using `fable`.

```
t0 = as.Date("2020-06-06")
t1 = as.Date("2023-02-04")
fc_dates = cases_deaths |> filter(between(date, t0, t1)) |> pull(date)
fable_fc = tibble()

for (i in 1:length(fc_dates)) {
  fc_date = fc_dates[i]
  # cat(as.character(fc_date), "... ")

  # dat = ... construct data set using data up through the forecast date

  # fit = ... fit the ARIMA(1,1,0) and ARIMA(2,1,0) models

  # fc  = ... make forecasts, up to horizon 4; important: it will help to
  #           create a column in fc that indicates the forecast horizon!

  # fable_fc = bind_rows(fable_fc, as_tibble(fc))
}
```

9. (4 pts) Compute the MAE of the time series CV forecasts made by each ARIMA model, per horizon. (In order to do this, you'll need to join `fable_fc` to the death data from `cases_deaths`.) Compare this to the previously-computed MAEs of the ensemble and baseline models by displaying their MAE curves, as a function of horizon, all on the same plot. According to MAE, your ARIMA models should be better than the baseline, but notably worse than the ensemble.

10. (Bonus) Do the same as in Q9, but substituting MAE with coverage of the 80% prediction intervals.

ETS models

11. (8 pts) Fit two ETS models, using `ETS()` from the `fable` package: Holt's linear trend and damped linear trend. Each of these models should have additive errors, and no seasonality. Implement time series CV to evaluate them, just as you did in Q8; you should be appending the forecast from these ETS models to `fable_fc`. Then, as in Q9, plot their MAE curves as a function of horizon, and overlay the curves from all models considered thus far. Discuss what you find.

12. (Bonus) Compute the coverage of 80% prediction intervals from each of the ETS models, per horizon, and plot alongside the coverage curves from all models considered thus far.

Exogenous features

13. (8 pts) As mentioned in the intro to this homework, exogenous features can play a huge role developing useful forecasts. Fortunately, here you already have a fairly obvious candidate for such an exogenous feature: reported Covid-19 case counts. Both intuitively and quantitatively (recall the cross-correlation plots that we computed near the start of the course), we know that this is a leading indicator of Covid-19 deaths. In order to be able to make up to 4 week ahead forecasts, we'll use 4-week-lagged Covid-19 cases as exogenous feature, to add to our ARIMA model. Technically, when you add an exogenous feature in a formula in the call to `ARIMA()`, this fits a regression model with ARIMA errors.

    Consider a regression model of `log(deaths)` on `log(lag(cases, 4))`, with ARIMA(2,1,0) errors. The log transform is to stabilize the variance; we could have done this earlier, in our ARIMA models in the above questions, but it would have made the forecasts from our pure ARIMA models (without exogenous features) too volatile. Include an intercept (constant) term in the model. Implement time series CV for this model, using the code provided below as a scaffold. Compute the MAE per horizon, and compare this to the MAEs from the models considered thus far, on one plot. Discuss what you find—you should have taken a big leap toward the performance of the ensemble! Hint: read the comments below carefully. In order to make forecasts you will need to construct a new data set to pass to `forecast()` (as opposed to simply specifying the horizon) because `forecast()` needs to know where to find the exogenous feature(s) to construct forecasts.

```
cases_deaths = cases_deaths |> mutate(x = lag(cases, 4))

for (i in 1:length(fc_dates)) {
  fc_date = fc_dates[i]
  # cat(as.character(fc_date), "... ")

  # dat = ... construct data set using data up through the forecast date

  # fit = ... fit the regression model using ARIMA(2,1,0) errors

  # new_dat = new_data(dat, 4) |>
  #   left_join(cases_deaths, by = "date")

  # fc = ... make forecasts up to horizon 4, but now by specifying new_dat

  # fable_fc = bind_rows(fable_fc, as_tibble(fc))
}
```

14. (Bonus) Compute the coverage of 80% prediction intervals from the ARIMA model with the exogenous feature (cases), and plot alongside the coverage curves from models considered thus far.

15. (Bonus) For the regression model from Q13 (pick a single regression model that was fit at a single iteration of time series CV), check that its 1 week ahead point forecast matches the formulas you know underlie it. As before, you'll have to extract the coefficients from the fitted model, and then form the 1 week ahead predictions manually, verifying that you get the same result. Hint: here you should take the forecast to be the median of the forecast distribution and not the mean. Because of the data transformation (we're modeling log deaths), the `fable` package will do something to adjust the mean after back-transforming which is nonobvious. This page gives more details.

16. (Bonus) Repurposing the code used to plot the ensemble forecasts given at the start of the homework, plot forecasts from the 5 `fable` models that you've developed, at the same set of forecast dates. (You should have 5 separate plots.)