

# Post-processing forecasts

Forecasts of COVID-19, especially forecasts made by humans, often are not perfectly calibrated and tend to be systematically over-confident. Post-processing may be a way to alleviate this problem and improve overall forecast quality.

## (Possible) Objectives

- Identify different means of post-processing forecasts in a quantile-based format
- Implement the procedures in R or python
- Evaluate and compare performance of the different procedures
- Identify differences when post-processing data on a logarithmic scale
- Find out whether we can adapt a forecast model using information learned from other forecast models
- Investigate how stable forecast performance is over time. How much training data should be used for re-calibration and is it possible to infer future errors from past errors?

## Data

All forecasts are made in a quantile-based format, meaning that forecasters provide a predictive distribution in the form of 23 quantiles (11 prediction intervals plus the median prediction), specifying how likely they think the true observed value will fall in a given range.

### Human forecasts of COVID-19 made in the UK COVID-19 Forecasting Challenge

This data includes all forecasts as well as the true observations.

```
uk_data <- fread("data/full-data-uk-challenge.csv")
```

The data has the following columns:

location	Country code
location_name	Name of the country
target_end_date	Date for which a forecast was made. This is always a Saturday
target_type	The target variable to be predicted, cases or deaths
true_value	The corresponding true observed value
population	population of the target country
forecast_date	Date on which a forecast was made. This is always a Monday
quantile	quantile-level of the predictive distribution
prediction	Predicted value corresponding to the quantile-level specified in 'quantile'
model	Name of the forecaster
target	Summary of the prediction target variable (redundant information)
horizon	Forecast horizon
expert	Whether or not a forecaster self-identified as an expert

Potential difficulties are:

- there only is one location
- there are not a lot of observations
- many forecasters only submitted a few times and drop in and out

However, it is of particular interest to know

- whether proposed procedures work for small data sets with not a lot of training data
- whether we can e.g. learn anything from other forecasters that we can then apply to a forecaster who submitted for the first time.

### Forecasts submitted to the European Forecast Hub

These forecasts were submitted to the European Forecast Hub by different research institutions. The file contains forecasts as well as true observations.

```
hub_data <- rbindlist(  
  list(  
    fread("data/full-data-european-forecast-hub-1.csv"),  
    fread("data/full-data-european-forecast-hub-2.csv")  
  )  
)
```

**Updating the European Forecast Hub data (probably not necessary)** To update the data, clone the whole repository or use subversion (svn) to only download the relevant folder. Run

```
git clone https://github.com/epiforecasts/covid19-forecast-hub-europe/
```

or

```
svn checkout https://github.com/epiforecasts/covid19-forecast-hub-europe/trunk/data-processed
```

To load the forecasts and truth data and to update the csv files, run

```
# load truth data using the covidHubUtils package -----  
library(covidHubUtils) # devtools::install_github("reichlab/covidHubUtils")  
  
truth <- covidHubUtils::load_truth(hub = "ECDC") |>  
  filter(target_variable %in% c("inc case", "inc death")) |>  
  mutate(target_variable = ifelse(target_variable == "inc case",  
                                   "Cases", "Deaths")) |>  
  rename(target_type = target_variable,  
         true_value = value) |>  
  select(-model)  
  
fwrite(truth, "data/weekly-truth-Europe.csv")  
  
# get the correct file paths to all forecasts -----  
folders <- here("data-processed", list.files("data-processed"))  
folders <- folders[  
  !(grepl("\\.R", folders) | grepl(".sh", folders) | grepl(".csv", folders))  
>]  
  
file_paths <- purrr::map(folders,  
  .f = function(folder) {  
    files <- list.files(folder)  
    out <- here::here(folder, files)  
    return(out)} %>%  
  unlist()  
file_paths <- file_paths[grepl(".csv", file_paths)]  
  
# load all past forecasts -----
```

```

# ceate a helper function to get model name from a file path
get_model_name <- function(file_path) {
  split <- str_split(file_path, pattern = "/")[[1]]
  model <- split[length(split) - 1]
  return(model)
}

# load forecasts
prediction_data <- map_dfr(file_paths,
  .f = function(file_path) {
    data <- fread(file_path)
    data[, `:=`(
      target_end_date = as.Date(target_end_date),
      quantile = as.numeric(quantile),
      forecast_date = as.Date(forecast_date),
      model = get_model_name(file_path)
    )]
    return(data)
  }) %>%
  filter(grepl("case", target) | grepl("death", target)) %>%
  mutate(target_type = ifelse(grepl("death", target),
    "Deaths", "Cases"),
    horizon = as.numeric(substr(target, 1, 1))) %>%
  rename(prediction = value) %>%
  filter(type == "quantile",
    grepl("inc", target)) %>%
  select(location, forecast_date, quantile, prediction,
    model, target_end_date, target, target_type, horizon)

# merge forecast data and truth data and save
hub_data <- merge_pred_and_obs(prediction_data, truth,
  by = c("location", "target_end_date",
    "target_type")) |>
  filter(target_end_date >= "2021-01-01") |>
  select(-location_name, -population, -target)

# split forecast data into two to reduce file size
split <- floor(nrow(hub_data) / 2)

fwrite(hub_data[1:split, ],
  file = "data/full-data-european-forecast-hub-1.csv")
fwrite(hub_data[(split + 1):nrow(hub_data), ],
  file = "data/full-data-european-forecast-hub-2.csv")

```

## (Possible) Methods

### Post-processing

**Conformalized Quantile Regression (CQR)** CQR is a two-step process where in step 1 you obtain quantiles and in step 2 you post-process these quantiles to achieve better calibration. What is neat about this workflow is that we already have quantiles and therefore don't need to worry about step 1.

Re-calibration is achieved by computing so-called conformity scores on a training data set and using these conformity scores to adapt future prediction intervals.

The procedure is nicely explained in this poster and in this paper.

**Quantile Regression Averaging (QRA)** Quantile regression averaging is a procedure commonly used to create ensembles from different quantile-based forecasts. The idea is to represent every quantile of the ensemble forecast as a linear combination of quantiles from the individual member distributions. Weights for different quantiles in the linear combination are chosen to minimise the weighted interval score (see below) on a training data set.

This procedure could be adapted such that every quantile of the resulting distribution is a ‘linear combination’ of the quantiles of the raw predictive distribution. Essentially, this would mean e.g. shifting every quantile of the raw distribution by an additive value or multiplying it with a certain value.

This could presumably be done using the `quantgen` package. However, I have not yet had the time to check this is easily possible.

**Adjusting the quantile spread** **General idea** For example, the 90% quantile of a predictive distribution can be understood as the 80% quantile times some value (let’s call it quantile spread), the 80% quantile can be understood as the 70% quantile times some value etc. If a forecast is over-confident that implies that the quantile-spread is too small on average.

Future forecasts could potentially be improved by multiplying every quantile spread with some constant (or even a varying) factor. This factor could be estimated by writing an `optim` function that finds the value of that factor which optimises the weighted interval score (see below) on a training set. This factor could then be used to expand or contract prediction intervals for future forecasts.

Potentially, something similar could also be implemented using the QRA-framework.

Alternatively, the quantile spread could also be defined as a difference, rather than a ratio.

## Other possible methods and keywords

- quantile mapping (e.g. <https://journals.ametsoc.org/view/journals/clim/30/9/jcli-d-16-0652.1.xml>)
- coherent forecasting
- adjusting nominal quantile levels based on empirical coverage in the past

## Forecast evaluation

**Proper scoring rules** Forecasts in a quantile-based forecast can be evaluated using the weighted interval score (WIS, lower values are better), a proper scoring rule. Proper scoring rules incentivise the forecaster to state their true best belief and cannot be ‘cheated’.

The WIS can be decomposed into three components: a penalty for dispersion (i.e. large forecast uncertainty) as well as penalties for over- and underprediction.

$$WIS = Dispersion + Over-prediction + Under-prediction$$

For a single prediction interval, the interval score is computed as

$$IS_{\alpha}(F, y) = (u - l) + \frac{2}{\alpha} \cdot (l - y) \cdot 1(y \leq l) + \frac{2}{\alpha} \cdot (y - u) \cdot 1(y \geq u),$$

where  $1()$  is the indicator function,  $y$  is the true value, and  $l$  and  $u$  are the  $\frac{\alpha}{2}$  and  $1 - \frac{\alpha}{2}$  quantiles of the predictive distribution  $F$ , i.e. the lower and upper bound of a single prediction interval. For a set of  $K$  prediction intervals and the median  $m$ , the WIS is computed as a weighted sum,

$$WIS = \frac{1}{K + 0.5} \cdot (w_0 \cdot |y - m| + \sum_{k=1}^K w_k \cdot IS_{\alpha}(F, y)),$$

where  $w_k$  is a weight for every interval. Usually,  $w_k = \frac{\alpha_k}{2}$  and  $w_0 = 0.5$

The WIS is closely related to the absolute error. Over- and under-prediction should therefore also be understood as a form of absolute forecasting error.

**Forecast calibration Coverage** In addition to the WIS it makes sense to look at forecast calibration in terms of interval coverage. On average, all 50% or 90% prediction intervals should ideally cover 50% or 90% of the true observations. By comparing nominal and empirical coverage we can quickly determine whether a model is over- or underconfident.

**Bias** The over- and under-prediction penalties of the WIS capture absolute forecasting errors. In addition we can examine whether a forecast has relative tendency to over- or under-predict. This is less susceptible to large outlier predictions

### **PIT histograms**

Probability integral transform (PIT) histograms are another way of examining forecast calibration

**Evaluating forecasts in R** Forecasts can be evaluated in R using the `scoringutils` package.

```
scores <- eval_forecasts(uk_data,  
                          summarise_by = c("model", "target_type"))  
  
scores
```