Post Processing Covid-19 Forecasts

Matthias Herp, Joel Beck

supervised by

Nikos Bosse

Chair of Statistics, University of Göttingen

15.03.2022

Table of Contents

Introduction

1 Analysis Tools

Data Analysis is inherently build upon two foundational components: High Quality Data that allows to gain insight into the underlying data generating process and a structured and reproducible way to extract information out of the collected data.

Thus, Section 1.1 introduces the two datasets we worked with whereas Section 1.2 provides an overview about the `postforecasts` package, a unified framework to apply and analyze various post-processing methods.

1.1 Data & Methodology

This section first introduces the two data sources that all of our analysis is based on. Afterwards, the final paragraphs explain our evaluation procedure from a theoretical view point.

1.1.1 UK Covid-19 Crowd Forecasting Challenge

As part of an ongoing research project by the *epiforecasts* group at the London School of Hygiene & Tropical Medicine the UK Covid-19 Crowd Forecasting Challenge[1] consisted of submitting weekly predictions of Covid-19 Cases and Deaths in the United Kingdom. The challenge was not restricted to experienced researchers in the field but rather intended to collect quantile predictions for the upcomning four weeky by non-expert individuals.

One of the main motivations was to gather evidence for or against the hypotheses that humans are highly capable of submitting precise *point forecasts*, yet, at the same time, they tend to be too confident in their beliefs such that prediction *intervals* are chosen too narrow. In fact, this tendency represents one motivation for post-processing: Extract valuable information from point forecasts and adjust the corresponding prediction intervals with a systematic correction procedure.

In case of individuals that are unfamiliar with statistical methodology specifying forecasts for 23 quantiles ranging from 0.01 to 0.99 separately might lead to inconsistencies. Therefore all participants could determine an uncertainty parameter around their median prediction via an interactive web application such that all quantile predictions could be concluded in an automatic fashion. Note that this procedure leads to *symmetric* forecast intervals.

The results of the 12-week challenge are publicly available[2].

1.1.2 European Covid-19 Forecast Hub

According to their webpage[3] the European Covid-19 Forecast Hub collects *short-term forecasts of Covid-19 cases and deaths across Europe, created by a multitude of infectious disease modelling teams.*

In contrast to the compact UK data described above, the European Forecast Hub data contains almost two million observations for over 20 European countries. Further, the forecasters are knowledgeable research groups that submit their weekly predictions based on statistical models. Although the data collection continues in regular frequency up to this day, our data set is limited to a 32-week span from March 2021 until October 2021.

1.1.3 Weighted Interval Score

In order to quantify if the post-processed prediction intervals improve the original forecasts we chose the *Weighted Interval Score* (WIS) (Bracher et al. 2021) as our evaluation metric. The WIS is a so-called *Proper Scoring Rule* (Gneiting and Raftery 2007): It incentivizes the forecaster to state their true best belief and cannot be manipulated to its own benefit. It combines measures for interval *sharpness* as well as *overprediction* and *underprediction* and can thus be understood as a trade-off between prediction *accuracy* and *precision*.

[1] https://www.crowdforecastr.org/2021/05/11/uk-challenge/
[2] https://epiforecasts.io/uk-challenge/
[3] https://covid19forecasthub.eu/index.html

More specifically, for a given quantile level α, true observed value y as well as lower bound l and upper bound u of the corresponding $(1-\alpha)\cdot 100\%$ prediction interval, the Weighted Interval Score is computed as

$$Score_\alpha(y) = (u-l) + \frac{2}{\alpha}\cdot(l-y)\cdot\mathbf{1}(y\leq l) + \frac{2}{\alpha}\cdot(y-u)\cdot\mathbf{1}(y\geq u).$$

The score of an entire model can then be obtained from a weighted sum over all (included) quantile levels α.

1.1.4 Time Series Cross Validation

Just like any statistical model the post-processing methods must be evaluated on *out-of-sample* data. Rather than starting from the raw data, i.e. the observed Covid-19 Cases and Deaths, our data sets already consist of existing quantile predictions. As a consequence, no part of our data set must be dedicated to fitting the quantile regression models in the first place and our evaluation procedure can be split in two steps:

1. Use a *training set* to learn parameters of the post-processing procedure in consideration.
2. Use a *validation set* to evaluate how the learned parameters generalize to unseen data.

Instead of a hard cut-off between the splits we used *Time Series Cross Validation* to leverage a higher fraction of the data set for training. In contrast to classical cross validation for independent and identically distributed data, time series cross validation iterates through the data set along the time dimension one step at a time.

The process is nicely illustrated in Figure 1[4].

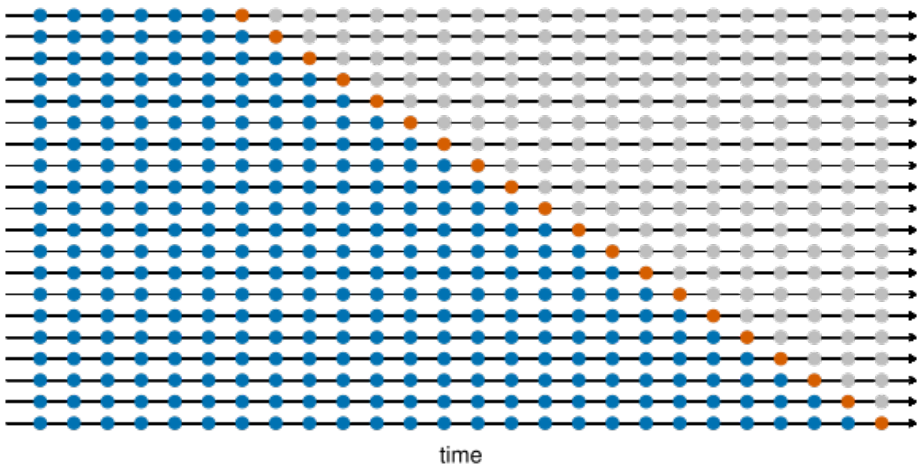

Figure 1: Time Series Cross Validation

At each iteration the validation set is composed of the one step ahead prediction based on all observations prior to and including the current time point. The algorithm typically starts with a minimum number of observations as the initial training set, which can be considered a hyperparameter that has to be specified at the beginning of training.

1.2 The `postforecasts` Package

One core aspect of our project was the development of a fully functional R package that unites a collection of different post-processing algorithms into a well-designed and user friendly interface. This section can be understood as a compact guide how to use our package effectively and explains some of the thought process that went into the implementation. It is worth noting that the `postforecasts` package adheres to all formal requirements for an R package such that `RCMDCHECK` does not produce any warnings or errors.

[4]Image Source: https://otexts.com/fpp3/tscv.html (Hyndman and Athanasopoulos 2021).

1.2.1 Overview

The `postforecasts` functions that are meant to be visible to the end-user can be grouped into three categories:

1. Exploratory

   The `plot_quantiles()`, `plot_intervals()` and `plot_intervals_grid()` functions visualize the development of true Covid19 Cases and Deaths over time as well as corresponding original and post-processed quantile predictions.

2. Model Fitting

   The `update_predictions()` function is the workhorse of the entire `postforecasts` package. It specifies both the raw data and the post-processing method(s) that should be applied to this data set. The function returns a list of $k + 1$ equally shaped data frames for k selected post-processing methods, the first element being the original, possibly filtered, data frame.

   All list elements can be analyzed separately or collectively by stacking them into one large data frame with the `collect_predictions()` function. The combined data frame is designed to work well with analysis functions that are provided by the scoringutils package[5] (Bosse, Sam Abbott, and Gruson 2022). Finally, an ensemble model of all selected methods can be appended which will be explained in Section 4 .

3. Evaluation

   As noted in Section 1.1 the Weighted Interval Score is our primary metric to evaluate the *quality* of prediction intervals. The `score()` function of the `scoringutils` package computes this score for each observation in the data set which can then be aggregated by the related `summarise_scores()` function. Depending on the *granularity* of the aggregation the output might contain many interval scores of vastly different magnitudes. To simplify interpretation the `eval_methods()` function computes *relative* or *percentage* changes in the Weighted Interval Score for each selected method compared to the original quantile predictions. Further, these relative changes can be conveniently visualized by the `plot_eval()` function.

The following section demonstrates the complete workflow described above to give an impression how all these functions interact.

1.2.2 Workflow

We use the Covid-19 data for Germany in 2021 that is provided by the European Forecast Hub.

Figure 2 illustrates the 5%, 20% 80% and 95% quantile predictions of the `EuroCOVIDhub-ensemble` during the summer months of 2021 in Germany.

```
plot_quantiles(
  hub_germany,
  model = "EuroCOVIDhub-ensemble", quantiles = c(0.05, 0.2, 0.8, 0.95)
)
```

The original predictions look quite noisy overall with the clear trend that uncertainty and the interval width increases with growing forecast horizon. Thus, we want to analyze if one particular post-processing method, *Conformalized Quantile Regression* which is explained in much more detail in Section 2, improves the predictive performance for this model on a validation set by computing the Weighted Interval Scores for Covid Cases and Covid Deaths separately.

```
df_updated <- update_predictions(
  hub_germany,
  methods = "cqr", models = "EuroCOVIDhub-ensemble", cv_init_training = 0.5
```

[5] https://epiforecasts.io/scoringutils/

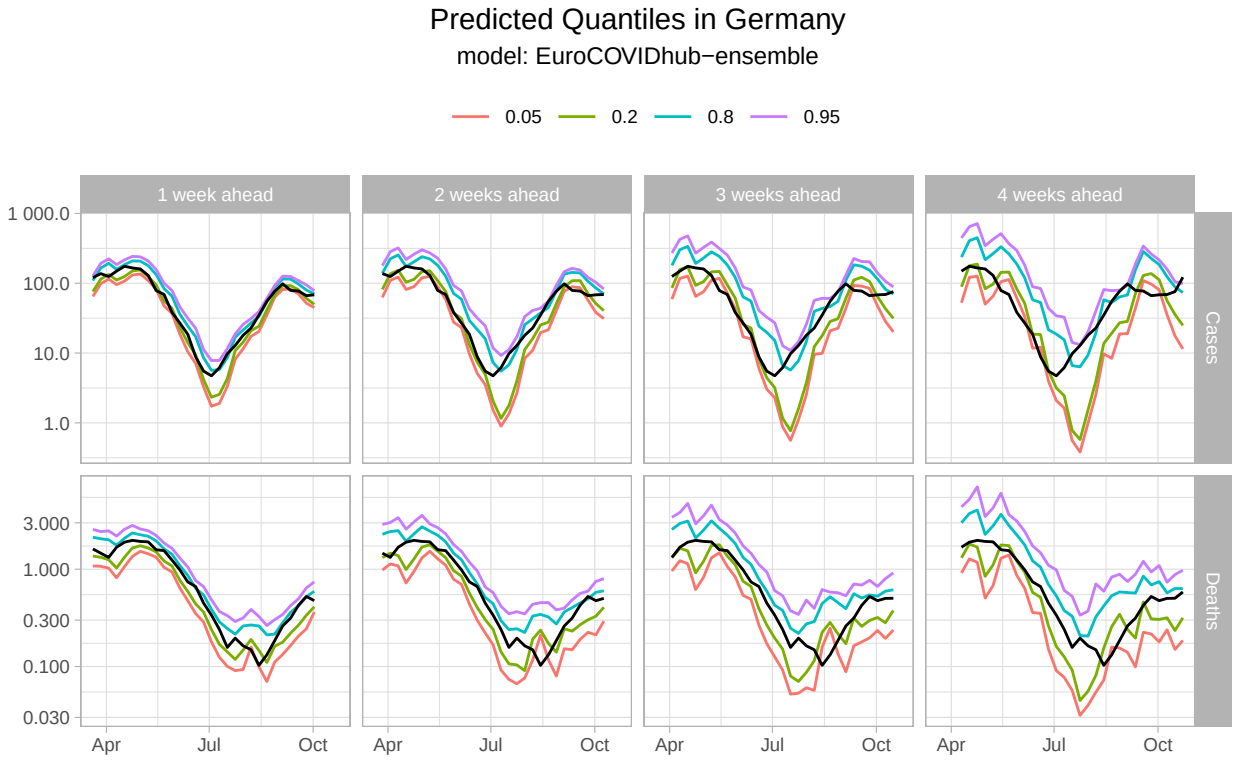

Figure 2: Original Quantile Predictions for Covid-19 Cases and Deaths in Germany 2021

Table 1: Comparison of the Weighted Interval Score after CQR Adjustments

method	target_type	interval_score	dispersion
cqr	Cases	13.37	5.05
original	Cases	13.78	3.81
cqr	Deaths	0.05	0.01
original	Deaths	0.05	0.03

```
)
df_combined <- collect_predictions(df_updated)
```

```
df_combined |>
  extract_validation_set() |>
  scoringutils::score() |>
  scoringutils::summarise_scores(by = c("method", "target_type"))
```

Table 1 shows that CQR improved the Weighted Interval Score for Covid Cases on the validation set, whereas the predictive performance for Covid Deaths dropped slightly.

The `update_predictions()` and `collect_predictions()` combination immediately generalize to multiple post-processing methods. The only syntax change is a vector input of strings for the `methods` argument instead of a single string. Hence, if not desired, the user does not have to worry about which input and output features each method requires in its raw form nor how exactly each method is implemented. This design allows for maximum syntactic consistency through masking internal functionality.

In the output above CQR increased the *dispersion* of the predictions for Cases significantly. These wider

intervals for specific covariate combinations are visualized in Figure 3.

```
plot_intervals(df_combined, target_type = "Cases", horizon = 2, quantile = 0.05)
```

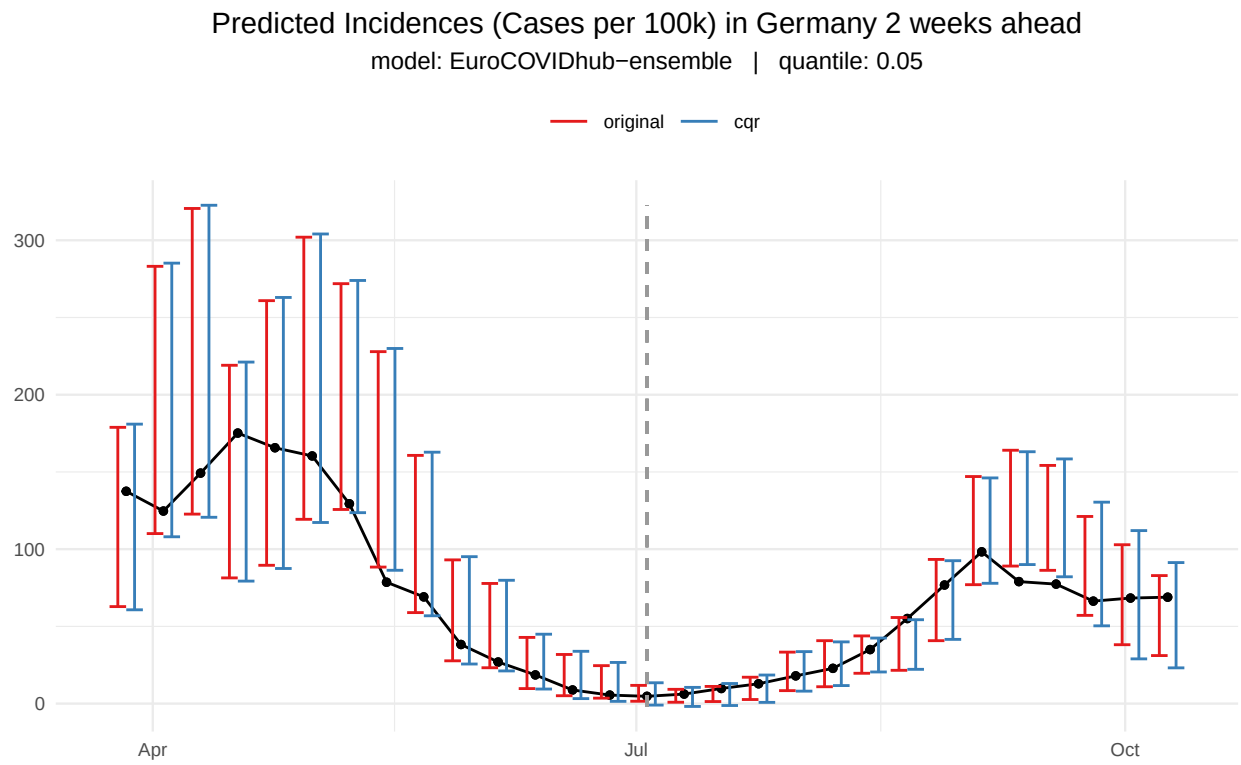

Figure 3: Original and CQR-adjusted Prediction Intervals for Covid-19 Cases in Germany

Indeed, the 2 weeks-ahead 90% prediction intervals for Cases in Germany are increased by CQR. The grey dashed line indicates the end of the training set within the time-series cross validation process.

Recall that uncertainty increases with larger horizons. Similarly, CQR adjustments also increase in size for forecasts that are submitted further in advance, which can be seen in Figure 4.

```
plot_intervals_grid(df_combined, facet_by = "horizon", quantiles = 0.05)
```

Interestingly, CQR expands the intervals only for Cases whereas the forecasts for Deaths are narrowed!

Besides the target type (Cases or Deaths), it is also useful to compare CQR effects across forecast horizons or quantiles. Quite intuitively, CQR generally has a stronger *relative* benefit for large time horizons and extreme quantiles, where the original forecaster faced a greater uncertainty. Figure 5 illustrates how, in special cases like this one, the effect on the validation set can show rather mixed trends due to disadvantageous adjustments for the two and three weeks-ahead 98% prediction intervals.

```
df_eval <- eval_methods(df_combined, summarise_by = c("quantile", "horizon"))
plot_eval(df_eval)
```

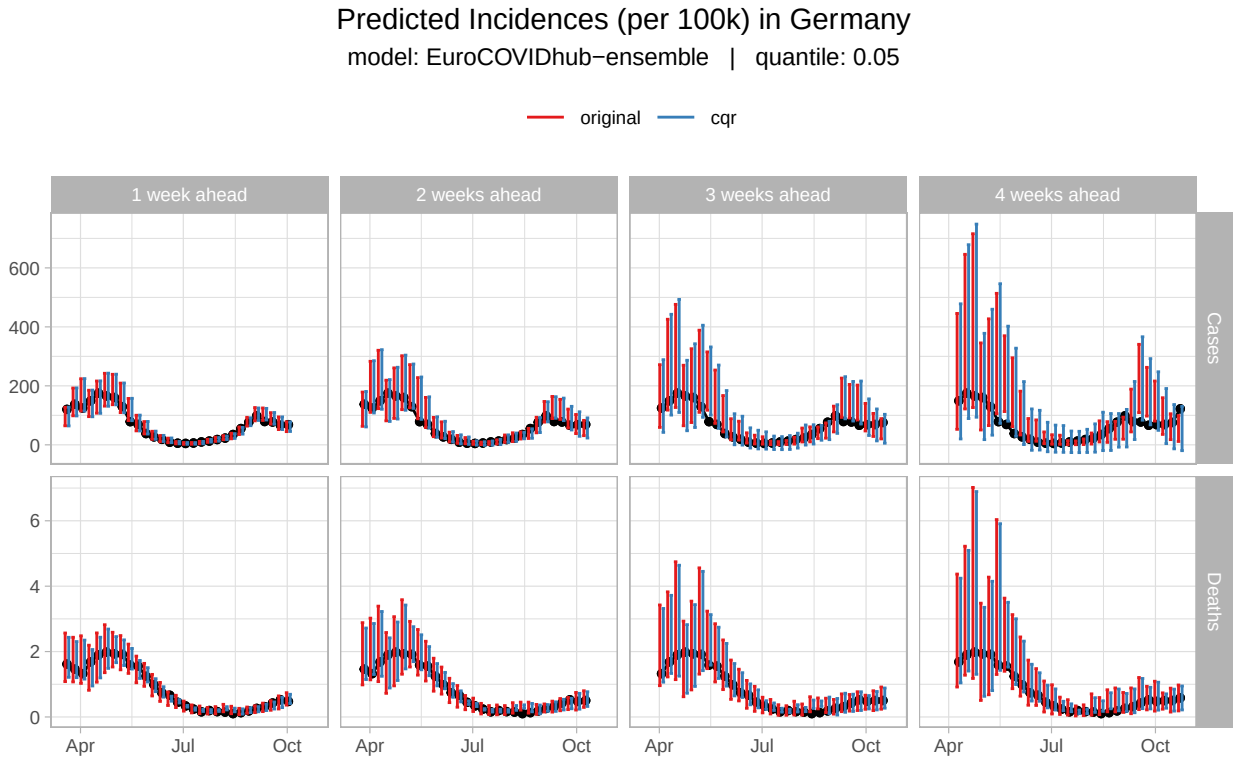

Figure 4: Original and CQR-adjusted Prediction Intervals for different Forecast Horizons

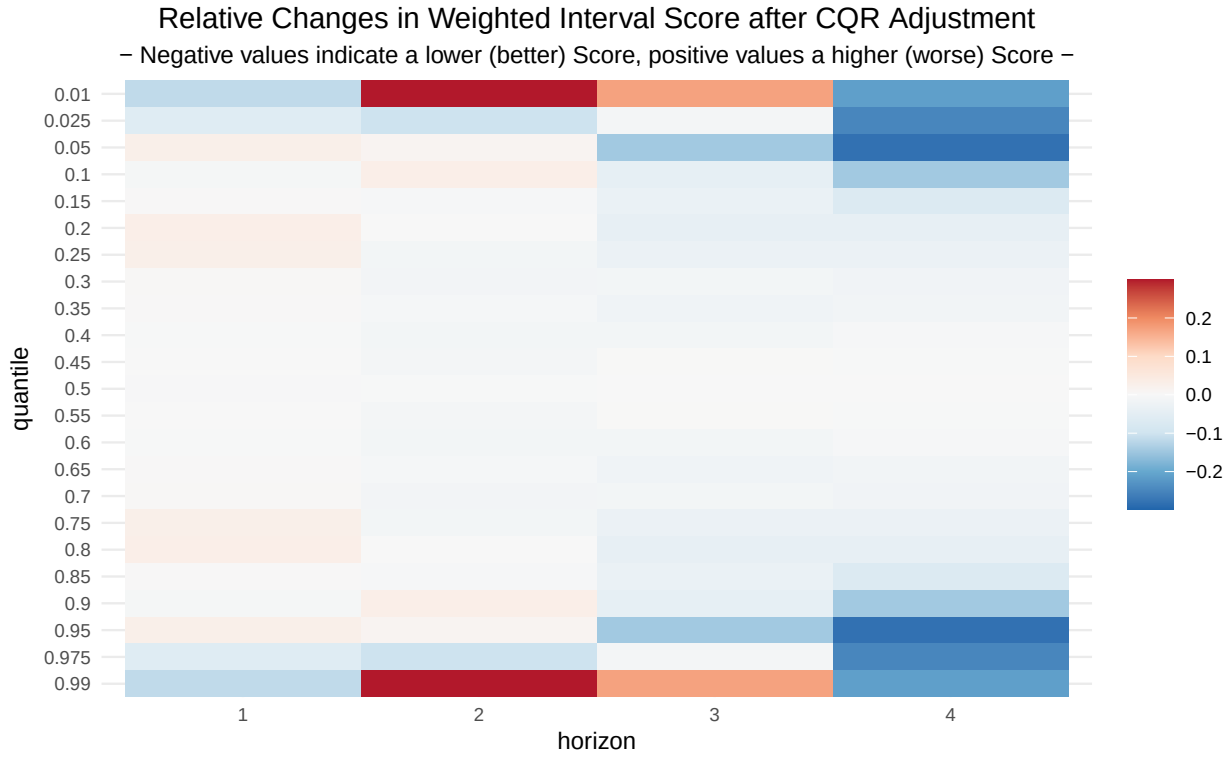

Figure 5: Relative Changes in the WIS through CQR for all Quantile-Horizon combinations

2 Conformalized Quantile Regression

3 Quantile Spread Adjustment

The general idea behind the Quantile Spread Adjustment (QSA), is to adjust the spreads of each forecasted quantile by some factor. Quantile spreads are defined as the distance between the respective quantile and some basis. As basis three different points in the forecasting spectrum come into question: the median, the next inner neighbor and the symmetric interval quantile. The quantile spread for the different basis are illustrated in Figure 6.

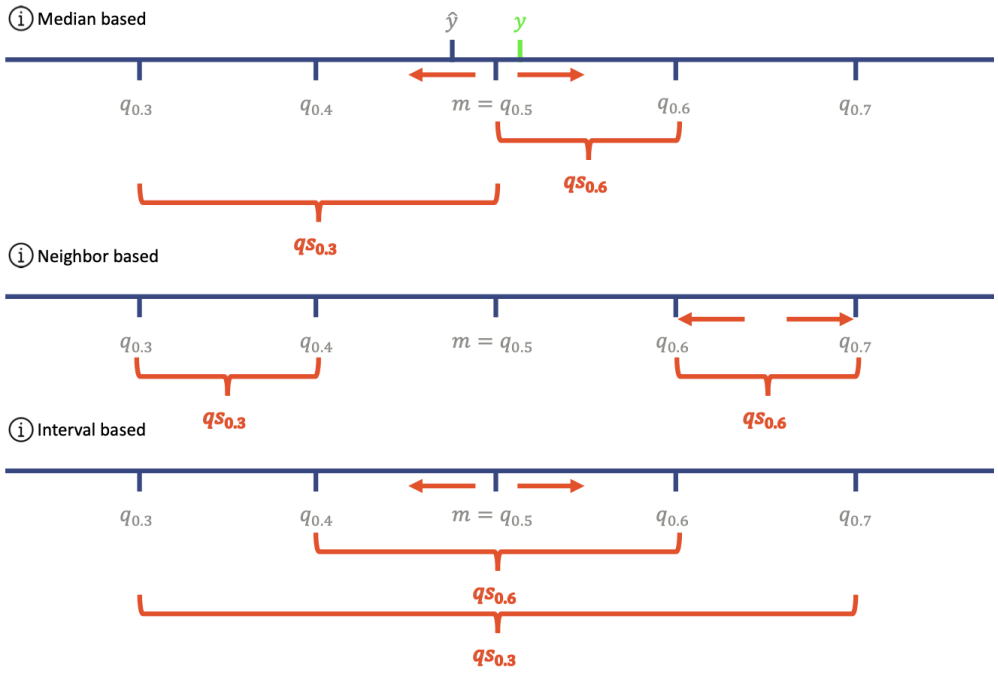

Figure 6: Quantile Spreads for different Basis

We choose the median based definition of the quantile spreads for two main reasons. First, in contrast to the neighborhood based definition, the median basis has the advantage that different quantile spreads are independent of one another. This property makes finding the optimal quantile spread adjustments for a large set of quantiles much simpler. However it comes at the cost that theoretically adjustments can lead to quantile crossing, which would not be the case for neighborhood based adjustments. Our second reason to use the median basis is that it doesn't restrict adjustments to be symmetric for quantile pairs, as would is the case for the interval based approach.

3.1 Theory

Using the median based definition, the next step is to determine how to optimally adjust the quantile spreads. As target function, QSA uses the Weighted Interval Score (reference). Equation (reference) shows how the QSA weights $\mathbf{w}$ influence the WIS.

$$\begin{aligned}\mathbf{w}^* &= \arg\min_{\mathbf{w}\in\mathbb{R}^p} WIS_\alpha(\mathbf{y}) \\ &= \arg\min_{\mathbf{w}\in\mathbb{R}^p} \sum_{i=1}^{p} \frac{\alpha}{2} \sum_{j=1}^{n} (u^*_{i,j} - l^*_{i,j}) + \frac{2}{\alpha}\cdot(l^*_{i,j} - y_j)\cdot\mathbf{1}(y_j \le l^*_{i,j}) + \frac{2}{\alpha}\cdot(y_j - u^*_{i,j})\cdot\mathbf{1}(y_j \ge u^*_{i,j}) \\ \text{s.t.} \quad & l^*_{i,j} = l_{i,j} + (l_{i,j} - m)\cdot(w^l_i - 1) \quad \text{and} \quad u^*_{i,j} = u_{i,j} + (u_{i,j} - m)\cdot(w^u_i - 1)\end{aligned}$$

For a given prediction interval level of α, by varying the QSA factor w^l_i for the lower and w^u_i for the upper bound, QSA moves the quantiles from there original values $l_{i,j}$ and $u_{i,j}$ to there adjusted values $l^*_{i,j}$ and $u^*_{i,j}$.

QSA factor values larger than 1 lead to an increase in the prediction interval, thus $w_i^l > 1$ reduces the value of $l_{i,j}^*$ and $w_i^u > 1$ increases the value of$u_{i,j}^*$. These changes have two effects, on the one side an increase in w_i^l and w_i^u reduces the sharpness and increases the WIS, on the other side the increased interval may capture more observation which reduces the under- and overprediction penalities in the WIS. Thus depending on the positions of the observed values and predicted quantiles, QSA will either increase or decrease the interval size in order to minimize the WIS.

Next step is to describe the different flavors....

4 Method Comparison

This chapter aims to compare the effectiveness of all Post-Processing methods that were introduced in the previous chapters. In particular, we investigate if some methods consistently outperform other procedures across a wide range of scenarios, i.e. different data sets and different covariate combinations.

Further, it will be interesting to observe the *types* of adjustments to the original forecasts: Some methods might improve the Weighted Interval Score by *extending* the interval width and thus increasing coverage whereas others might yield a similar final score by *shrinking* the prediction intervals leading to a higher precision. One can imagine even more variants: Moving the interval bounds farther apart or closer together can happen in a *symmetric* or *asymmetric* way and the interval's midpoint might stay *fixed* or get *shifted* throughout the post-processing process.

Before jumping into the analysis, we propose one additional model that in contrast to those we have covered so far, does not add any new information to the equation. Instead, it *combines* the predictions from existing post-processing methods to build an *ensemble* prediction. The idea is that leveraging information from multiple independent algorithms can stabilize estimation since the ensemble learns to focus on a model with a strong performance for one particular covariate set while the same model might perform badly for an opposing covariate set and, thus, make little contributions to the ensemble in that case.

Next, we explain the mathematical reasoning behind the ensemble model in more detail.

4.1 Ensemble Model

There exist various options how to combine multiple building blocks into one ensemble. We chose an approach that can be efficiently computed by well-understood algorithms on the one hand and is highly interpretable on the other hand. Each quantile prediction of our ensemble model is a *convex combination* of the individual models, i.e. a linear combination where all weights are contained in the unit interval and sum up to one. Hence, the resulting value lives on the same scale as the original predictions and each weight can be interpreted as the *fractional contribution* of each building block model.

Consider one particular covariate combination of **model**, **location**, **horizon**, **target type** and **quantile**. Let n specify the number of observations in the training set within this combination, $\mathbf{y} \in \mathbb{R}^n$ the vector of true values and $\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_k \in \mathbb{R}^n$ vectors of adjusted predictions from k different post-processing procedures.

Then, for each such combination, the ensemble model computes weights $\mathbf{w}^* \in [0,1]^k$ by solving the following nonlinear constrained optimization problem:

$$\begin{aligned}
\mathbf{w}^* = \underset{\mathbf{w}\in[0,1]^k}{\arg\min}\, WIS_\alpha(\mathbf{y}) = \underset{\mathbf{w}\in[0,1]^k}{\arg\min} (\mathbf{u}-\mathbf{l}) + \frac{2}{\alpha}\cdot(\mathbf{l}-\mathbf{y})\cdot\mathbb{1}(\mathbf{y}\le\mathbf{l}) + \frac{2}{\alpha}\cdot(\mathbf{y}-\mathbf{u})\cdot\mathbb{1}(\mathbf{y}\ge\mathbf{u}),\\
\text{with} \qquad \mathbf{l} = \sum_{j=1}^k w_j \mathbf{l}_j, \quad \mathbf{u} = \sum_{j=1}^k w_j \mathbf{u}_j\\
\text{s.t.} \qquad \sum_{j=1}^k w_j = 1,
\end{aligned}$$

where all operations for vector inputs $\mathbf{l}$, $\mathbf{u}$ and $\mathbf{y}$ are understood elementwise and the *same* weights w_j, $j = 1, \dots, k$ are chosen for lower and upper quantiles.

Hence, we choose the (nonlinear) Weighted Interval Score as our objective function that we minimize subject to linear constraints. The optimization step is implemented with the nloptr[6] package (Ypma and Johnson 2022), which describes itself as *an R interface to NLopt, a free/open-source library for nonlinear optimization.*

Note that, technically, the weight vector has to be denoted by $\mathbf{w}^*_{m,l,h,t,q}$ since the computed weights are generally different for each combination. We omit the subscripts at this point to keep the notation clean.

[6] https://cran.r-project.org/web/packages/nloptr/index.html

The Weighted Interval Score always considers *pairs* of quantiles α and $1 - \alpha$ that constitute a $(1 - 2 \cdot \alpha)$ prediction interval. The best results are achieved when a separate weight vector for each quantile pair is computed. Since our data sets contains 11 quantile pairs, 2 target types, 4 horizons and we consider 6 different forecasters, the ensemble model requires solving $11 \cdot 2 \cdot 4 \cdot 6 = 528$ nonlinear optimization problems for each location, which amounts to $18 \cdot 528 = 9504$ optimzation problems for the European Hub Data Set.

Due to this high computational cost the *maximum number of iterations* within each optimization was an important hyperparameter that balanced the trade-off between computational feasibilty and sufficient convergence. Here, we ultimately settled with 10.000 maximum steps which could ensure convergence with respect to a *tolerance level* of 10^{-8} in the vast majority of cases.

Finally, it is worth noting that the weight vector of the ensemble model $\mathbf{w}^*$ is learned on a *training set* such that a fair comparison with all individual post-processing methods on a *validation set* is possible.

5 Conclusion

A First Appendix

B Second Appendix

References

Bosse, Nikos, Sam Abbott, and Hugo Gruson. 2022. *Scoringutils: Utilities for Scoring and Assessing Predictions*.

Bracher, Johannes, Evan L. Ray, Tilmann Gneiting, and Nicholas G. Reich. 2021. "Evaluating Epidemic Forecasts in an Interval Format." *PLOS Computational Biology* 17 (2): e1008618. https://doi.org/10.1371/journal.pcbi.1008618.

Gneiting, Tilmann, and Adrian E Raftery. 2007. "Strictly Proper Scoring Rules, Prediction, and Estimation." *Journal of the American Statistical Association* 102 (477): 359–78. https://doi.org/10.1198/016214506000001437.

Hyndman, Rob, and George Athanasopoulos. 2021. *Forecasting: Principles and Practice*. Third. OTexts. https://otexts.com/fpp3/.

Ypma, Jelmer, and Steven G. Johnson. 2022. *Nloptr: R Interface to NLopt*. https://CRAN.R-project.org/package=nloptr.