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 **2**

1 Analysis Tools **2**
- 1.1 Data & Methodology . . . 2
- 1.2 The `postforecasts` Package . . . 4

2 Conformalized Quantile Regression **7**
- 2.1 Traditional CQR . . . 7
- 2.2 Asymmetric CQR . . . 13
- 2.3 Multiplicative CQR . . . 17

3 Quantile Spread Adjustment **21**
- 3.1 Theory . . . 21
- 3.2 Optimization . . . 23

4 Method Comparison **24**
- 4.1 Ensemble Model . . . 24
- 4.2 Comparison of CQR, QSA & Ensemble . . . 25

5 Conclusion **28**

A First Appendix **29**

B Second Appendix **29**

References **30**

Introduction

1 Analysis Tools

Data Analysis is inherently build upon two fundamental 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.

Section 1.1 introduces the two data sets we worked with while Section 1.2 provides an overview about the `postforecasts` package, a unified framework to apply and analyze various post-processing techniques.

1.1 Data & Methodology

This section first introduces the two data sources that all of our analysis is based on. The final paragraphs continue with a description of our evaluation procedure from a theoretical point of view.

1.1.1 UK Covid-19 Crowd Forecasting Challenge

As part of an ongoing research project by the EpiForecasts[1] group at the London School of Hygiene & Tropical Medicine, the UK Covid-19 Crowd Forecasting Challenge[2] consisted of submitting weekly predictions of Covid-19 Cases and Deaths in the United Kingdom in 2021. The challenge was not restricted to experienced researchers in the field but rather intended to collect quantile predictions for the upcoming four weeks by non-expert individuals.

One of the main motivations was to gather evidence for or against the hypothesis that humans are highly capable of 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 very specific quantiles of the predictive distribution 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[3].

1.1.2 European Covid-19 Forecast Hub

According to their webpage[4] 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.

The overall structure of the two data sets introduced above is very similar. Since we will refer to some particularly important columns by name frequently throughout the next chapters, they are briefly described here:

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

- `location`: The country for which the forecasts were submitted. Equals `GB` for the UK data. Our analysis for the European Forecast Hub data selects 18 different European countries.
- `model`: The forecaster (group). Mostly (non-expert) individuals for the UK data and international research groups for the European Forecast Hub.
- `target_type`: Either Covid-19 Cases or Covid-19 Deaths.
- `horizon`: The time horizon how far in advance the predictions were submitted. Ranges from 1 week-ahead to 4 weeks-ahead.
- `forecast_date`: The exact date when the forecasts were submitted.
- `target_end_date`: The exact date for which the forecasts were submitted.
- `quantile`: One of 23 different quantile values ranging from 0.01 to 0.99.
- `prediction`: The predicted value for one specific combination of the variables above.
- `true_value`: The actual, observed number of Covid-19 Cases or Deaths. This value is repeated 23 times, once for each quantile value.

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 in favour of own interests. It combines measures for interval *sharpness* as well as *overprediction* and *underprediction* and can thus be understood as a trade-off between interval *coverage* and *precision*.

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 **Interval Score** according to Bracher et al. (2021) is computed as

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

The penalties $\frac{2}{\alpha}\cdot(l-y)$ and $\frac{2}{\alpha}\cdot(y-u)$ for over- and underprediction are thus influenced by two components:

- The penalty gets larger for increasing distance of the true value y to the lower or upper bound, given that y is not contained in the interval.
- The penalty gets larger for smaller values of α, i.e. for higher nominal coverage levels $(1-\alpha)$.

A set of K Interval Scores with different quantile levels $\alpha_1, \ldots, \alpha_K$ can be aggregated to a single number, the **Weighted Interval Score**:

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

where m represents the predicted median.

Thus, as the name suggests, the Weighted Interval Score is a weighted sum of the individual Interval Scores. The weights w_k are typically chosen as $w_k = \frac{\alpha_k}{2}$ and the weight w_0 for the deviation of the median is usually set to 0.5.

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.

Our evaluation procedure can therefore be split into 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[5].

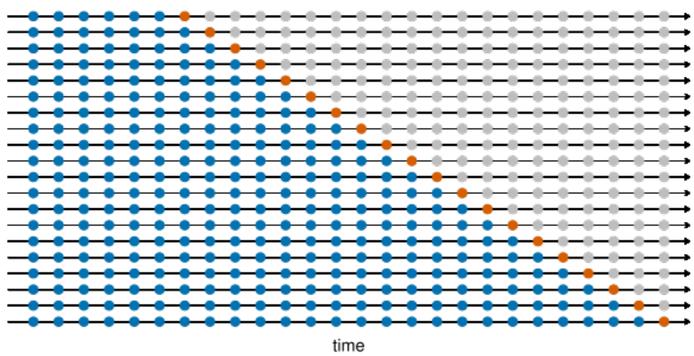


Figure 1: Time Series Cross Validation

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 package in the statistical programming language `R` (R Core Team 2021) 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. The source code is publicly available on GitHub[6].

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 Covid-19 Cases and Deaths over time as well as corresponding original and post-processed quantile predictions.

[5] Image Source: https://otexts.com/fpp3/tscv.html (Hyndman and Athanasopoulos 2021).
[6] https://github.com/nikosbosse/post-processing-forecasts

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 where the first element is given by 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 of the scoringutils[7] package (Bosse, Sam Abbott, and Gruson 2022). If multiple post-processing methods are applied, an ensemble model of all selected methods can be added via the `add_ensemble()` function, which lets the user access both the weighted ensemble predictions and a data frame with the corresponding weights. The ensemble approach will be further 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 quantity 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 visualized by the `plot_eval()` function.

The following section demonstrates the complete workflow described above to give an impression of the *interaction* between all implemented functions.

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.

```r
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, hence, the interval width increases with growing forecast horizons. 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:

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

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

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

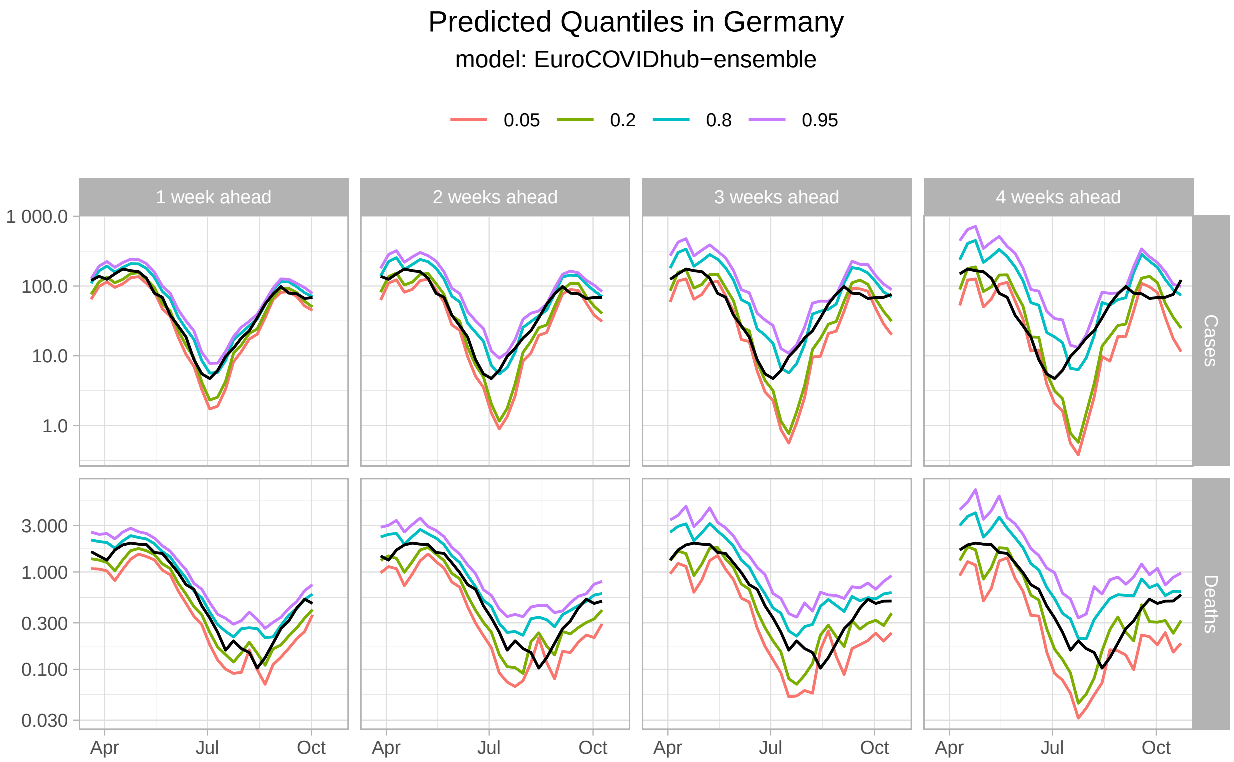


Figure 2: Original Quantile Predictions for Covid-19 Cases and Deaths in Germany 2021

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

method	target type	interval score	dispersion
cqr	Cases	13.40	5.07
original	Cases	13.78	3.81
cqr	Deaths	0.06	0.01
original	Deaths	0.05	0.03

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.

Moreover, the `update_predictions()` function automatically takes care of *quantile crossing* (Bassett and Koenker 1982) by reordering the output predictions in increasing quantile order. The `cv_init_training` parameter specifies the fraction of observations that is used for the pure training set before starting the Time Series Cross Validation process.

As seen in Table 1 CQR increases the *dispersion* of the predictions for Cases significantly. One example of these wider intervals is visualized in Figure 3.

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

Indeed, the 2 weeks-ahead 90% prediction intervals for Covid Cases in Germany are expanded by CQR.

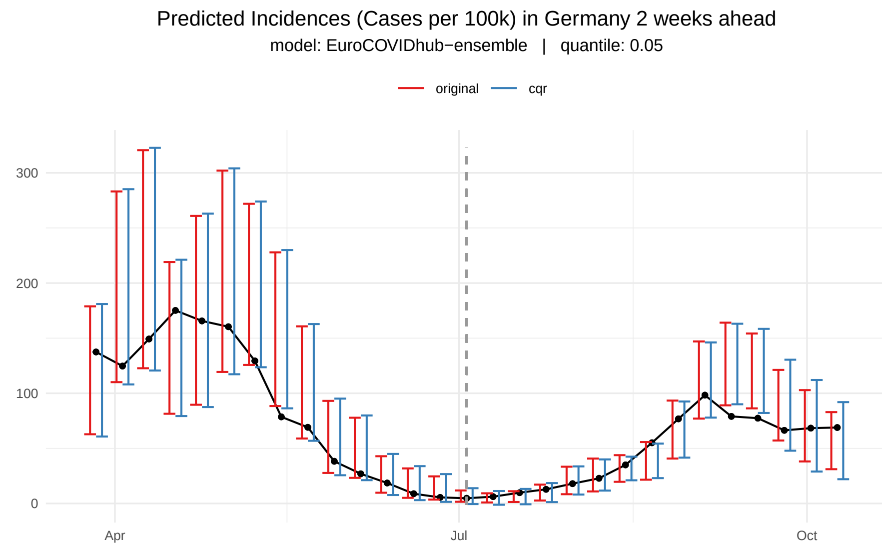


Figure 3: Original and CQR-adjusted Prediction Intervals for Covid-19 Cases in Germany

The grey dashed line indicates the end of the training set within the Cross Validation as specified by the `cv_init_training` parameter.

Recall from Figure 2 that prediction uncertainty increases with larger forecast horizons. Similarly, CQR adjustments also increase in size for forecasts that are submitted further in advance, which can be seen in Figure 4 along the horizontal dimension. Interestingly, CQR expands the intervals only for Cases whereas the forecasts for Deaths are narrowed!

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

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)
```

2 Conformalized Quantile Regression

This chapter introduces *Conformalized Quantile Regression (CQR)* as the first of two main Post-Processing procedures which are implemented in the `postforecasts` package.

Section 2.1 explains the original Conformalized Quantile Regression algorithm as proposed by Romano, Patterson, and Candès (2019). The underlying more general concept of *Conformal Inference* is motivated by Tibshirani (2019). We highlight potential limitations of the traditional implementation that could potentially be diminished by more flexible variants of CQR that are discussed in Section 2.2 and Section 2.3.

2.1 Traditional CQR

All derivations in this section are taken from the original paper (Romano, Patterson, and Candès 2019). The authors motivate Conformalized Quantile Regression by stating two criteria that the ideal procedure for generating prediction intervals should satisfy:

- It should provide valid coverage in finite samples without making strong distributional assumptions.

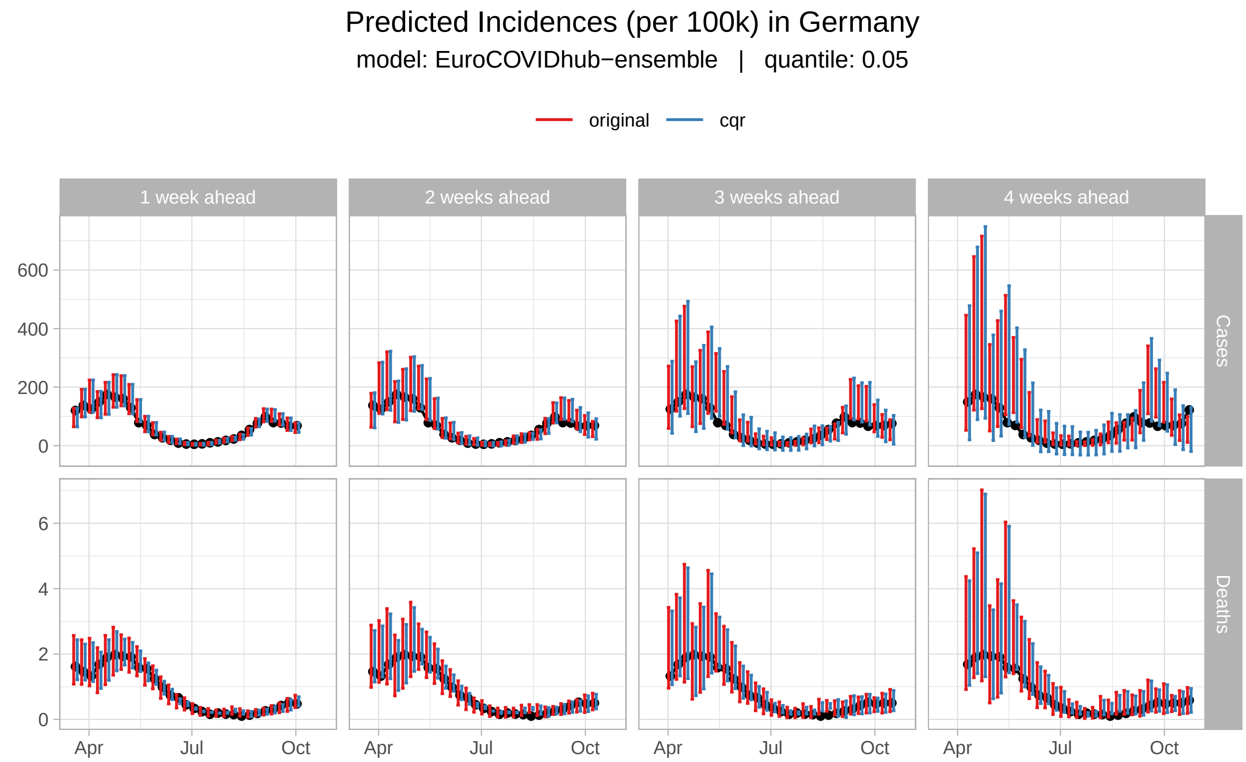


Figure 4: Original and CQR-adjusted Prediction Intervals for different Forecast Horizons

- The resulting intervals should be as narrow as possible at each point in the input space.

According to the authors CQR performs well on both criteria while being distribution-free and adaptive to heteroscedasticity.

2.1.1 Statistical Validity

The algorithm that CQR is build upon is statistically supported by Theorem 2.1. The term *conformity scores* is defined in Section 2.1.2.

Theorem 2.1. *If* $(X_i, Y_i), i = 1, \ldots, n+1$ *are exchangeable, then the* $(1-\alpha) \cdot 100\%$ *prediction interval* $C(X_{n+1})$ *constructed by the CQR algorithm satisfies*

$$P\left(Y_{n+1} \in C(X_{n+1})\right) \geq 1 - \alpha.$$

Moreover, if the conformity scores E_i *are almost surely distinct, then the prediction interval is nearly perfectly calibrated:*

$$P\left(Y_{n+1} \in C(X_{n+1})\right) \leq 1 - \alpha + \frac{1}{|I_2| + 1},$$

where I_2 *denotes the calibration (validation) set.*

Thus, the first statement of Theorem 2.1 provides a *coverage guarantee* in the sense that the adjusted prediction interval is *lower-bounded* by the desired coverage level. The second statement adds an *upper-bound* to the coverage probability which gets tighter with increasing sample size and asymptotically converges to the desired coverage level $1 - \alpha$ such that lower bound and upper bound are asymptotically identical.

2.1.2 Algorithm

The CQR algorithm is best described as a multi-step procedure.

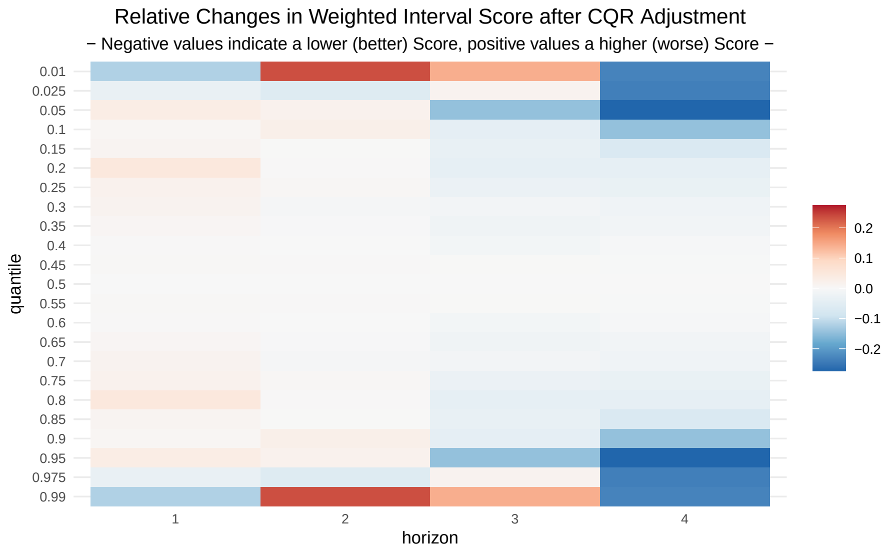


Figure 5: Relative Changes in the WIS through CQR for all Quantile-Horizon combinations

Step 1:
Split the data into a training and validation (here called *calibration*) set, indexed by I_1 and I_2, respectively.

Step 2:
For a given quantile α and a given quantile regression algorithm $\mathcal{A}$, compute the original lower and upper quantile predictions on the training set:

$$\{\hat{q}_{\alpha,low},\ \hat{q}_{\alpha,high}\} \leftarrow \mathcal{A}\left(\{(X_i, Y_i) : i \in I_1\}\right).$$

Note that the algorithm does *not* make any assumptions about the structural form of $\mathcal{A}$ which, in theory, could be a highly nonlinear function like a Deep Neural Network.

Step 3:
Compute *conformity scores* on the calibration set:

$$E_i := \max\left\{\hat{q}_{\alpha,low}(X_i) - Y_i,\ Y_i - \hat{q}_{\alpha,high}(X_i)\right\} \quad \forall\ i \in I_2$$

Thus, for each i, the corresponding score E_i is *positive* if Y_i is *outside* the interval $[\hat{q}_{\alpha,low}(X_i),\ \hat{q}_{\alpha,high}(X_i)]$ and *negative* if Y_i is *inside* the interval.

Step 4:
Compute the *margin* $Q_{1-\alpha}(E, I_2)$ given by the $(1-\alpha)(1+\frac{1}{1+|I_2|})$-th empirical quantile of the score vector E in the calibration set. For small sample sizes and small quantiles α this procedure might result in quantiles greater than 1. In this case we simply select the maximum value of the score vector.

Step 5:
On the basis of the original lower and upper quantile prediction $\hat{q}_{\alpha,low}(X_i)$ and $\hat{q}_{\alpha,high}(X_i)$, the new *post-processed* prediction interval for Y_i is given by

$$C(X_{n+1}) = \left[\hat{q}_{\alpha,low}(X_i) - Q_{1-\alpha}(E, I_2),\ \hat{q}_{\alpha,high}(X_i) + Q_{1-\alpha}(E, I_2)\right].$$

Note that the *same* margin $Q_{1-\alpha}(E, I_2)$ is subtracted from the original lower bound and added to the original upper bound This limitation is adressed in Section 2.2.

2.1.3 Results

We now investigate how well the algorithm performs for post-processing Covid-19 forecasts. Thereby we start UK Covid-19 Forecasting Challenge data set and briefly point out recurrent trends within CQR adjustments. Then, we continue with a more detailed discussion of the findings on the larger European Forecast Hub data.

Table 2: WIS improvement by CQR for one particular feature combination on the validation set.

method	model	target type	horizon	quantile	interval score	dispersion
cqr	seabbs	Cases	3	0.1	157.32	87.49
original	seabbs	Cases	3	0.1	210.62	38.14

Table 3: Overall WIS improvement by CQR on the validation set.

method	interval score	dispersion
cqr	62.15	24.1
original	65.74	12.0

One common characteristic that applies to almost all feature combinations is that CQR *expands* the original forecast intervals. As stated in **Step 5** of Section 2.1.2 it moves the original lower and upper bounds in a *symmetric* way either inwards or outwards by using the *same* margin. This implies that the interval *midpoint* remains unchanged when applying the traditional CQR algorithm.

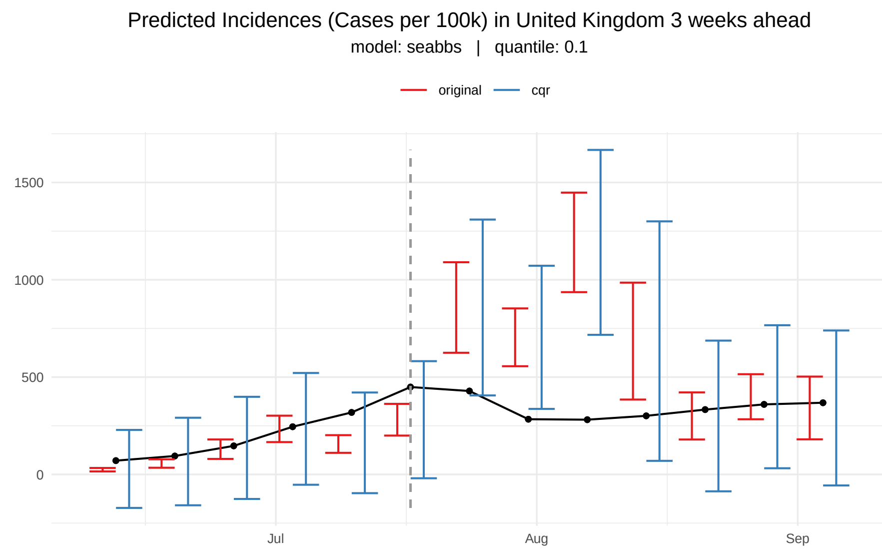


Figure 6: CQR tends to make prediction intervals larger, here for the `seabbs` forecasting model

One extreme example of this behaviour is shown in Figure 6. Since the `seabbs` forecasts are submitted by a single individual, we find evidence for the hypothesis of Section 1.1 that humans tend to be too confident in their own predictions resulting in too narrow uncertainty bounds. By extending the intervals symmetrically CQR maintains *pointwise* information from the original forecasts while simultaneously increasing interval coverage.

Yet, the pure effect of increasing coverage does not automatically imply that the Weighted Interval Score has improved as well due to the trade-off between coverage and precision. Thus, we explicitly compute the WIS, once for the specific covariate combination of Figure 6 in Table 2 and once aggregated over all *models*, *target types*, *horizons* and *quantiles* in Table 3.

Both tables confirm the visual impression of Figure 6: CQR improves the WIS by increasing the *dispersion* value, a measure for the interval *spread*. This effect is particularly strong in case of the `seabbs` model but still applies to a more moderate extent to most of the other forecasting models.

Since many of the general findings for traditional CQR coincide between the UK data and the EU Forecast

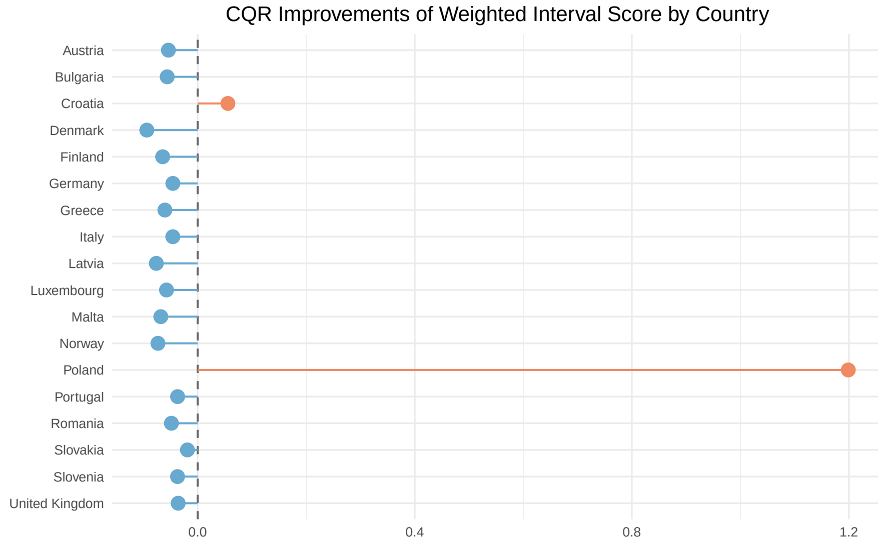


Figure 7: CQR proves to be beneficial for the vast majority of countries, with the major exception of Poland

Table 4: Weighted Interval Score for Poland by Model on Training and Validation Set

method	**model**	**training score**	**validation score**
cqr	epiforecasts-EpiNow2	22.84	1.94
original	epiforecasts-EpiNow2	23.71	1.32
cqr	EuroCOVIDhub-ensemble	29.44	2.34
original	EuroCOVIDhub-ensemble	31.37	0.96
cqr	IEM_Health-CovidProject	56.54	4.68
original	IEM_Health-CovidProject	62.04	0.91

Hub data, we jump straight to the latter for the following analysis. First, we investigate if CQR is equally effective across all countries. Figure 7 indicates that this is clearly *not* the case: CQR is beneficial on out of sample data in almost all of the 18 selected countries. The largest effect size, however, is linked to Poland in *negative* direction.

At first sight this finding seems like a data entry error, there is no obvious reason why such a general algorithm like Conformalized Quantile Regression might not work for one specific location. The large negative effect is also interesting in light of Theorem 2.1: We know that CQR *always* improves the forecast intervals on the training set which, of course, applies to Poland as well. We can confirm this theoretical guarantee empirically by evaluating the Weighted Interval Score for Poland on the training set only. Table 4 collects the training and validation scores for three selected forecasting models separately.

Indeed, CQR improves the WIS for all three models in-sample whereas the out-of-sample performance drops dramatically. This finding provides evidence that the observations used for the initial training phase must be *fundamentally different* to those encountered during the Cross Validation process. More specifically, it suggests a *distribution shift* of the true observed values and/or the original quantile predictions right at the split of training and validation phase. Further, the *scale* of training and validation scores is quite different, which usually stems from different magnitudes of the observed incidences within each stage.

Figure 8 confirms our hypothesis for 1 week-ahead forecasts of 90% prediction intervals for Covid-19 Cases. The left plot displays the development of observed and predicted values for the outlier Poland compared to the same setting for Germany where CQR performs just fine. A few weeks before the training-validation split, which is highlighted by the grey dashed line, the true incidences plummeted in Poland. In strong

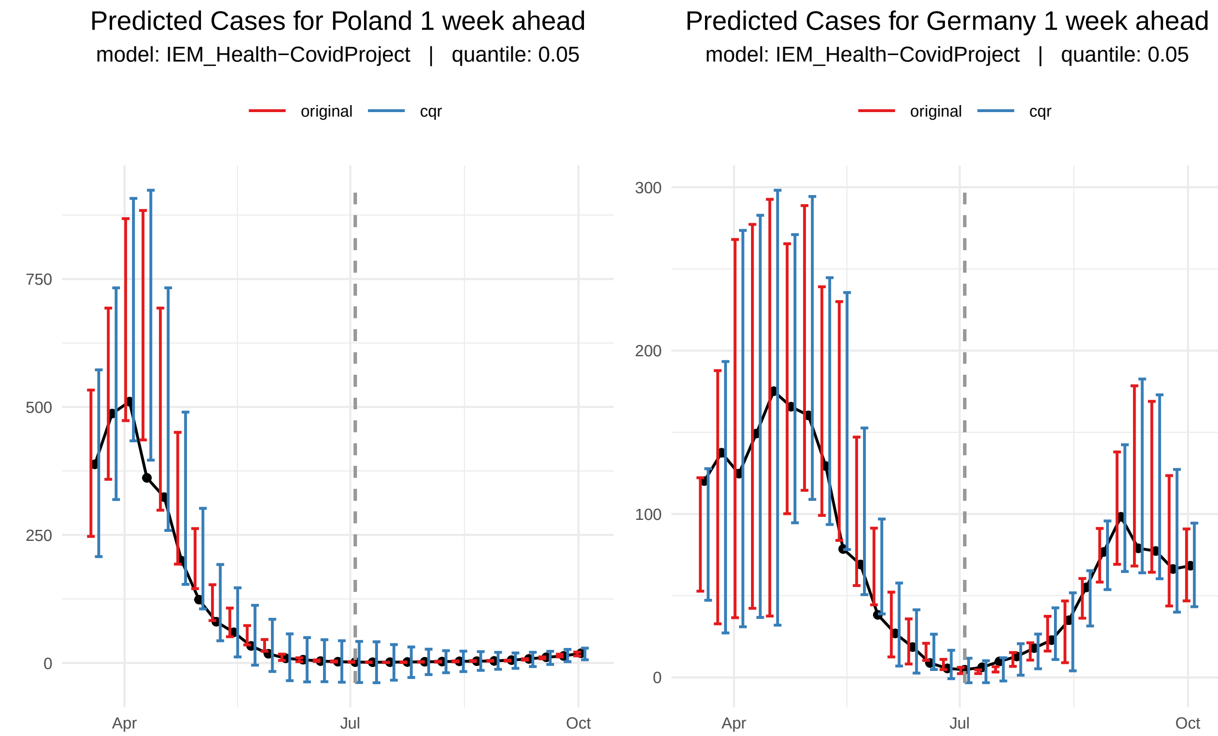


Figure 8: Development of Covid-19 Cases in 2021 in Poland and Germany

contrast to Germany, where the Covid-19 situation relaxed during the summer months of 2021 as well, the incidences *remain* low until late autumn in Poland (according to the collected data of the European Forecast Hub). Thus, the incidences are indeed much lower on average in the validation set which explains the scale discrepancy in Table 4.

The consistently low incidences are connected to decreased uncertainty margins of the original forecasts that were submitted only one week in advance. The forecasters were well aware of the current Covid-19 situation and were able to quickly react with lower point forecasts and narrower prediction intervals. CQR is not capable of competing with this flexibility and requires a long time span to adapt to irregular behaviour. The reasons for these slow adjustments, which reveal a major downside of CQR, follow immediately from the underlying statistical theory and are explained in detail in Section 2.2.2.

Lastly, we summarize the performance of vanilla CQR across different *quantiles*, *target types* and *horizons*. To obtain more informative visual illustrations we exclude Poland from the further analysis.

The left plot of Figure 9 shows the performance of CQR for all 23 quantile levels in the data set. Although the effect size varies by country, the general trend holds unanimously: Extreme quantiles in the tails of the predictive distribution benefit most from post-processing with a gradual decline towards centered quantiles. The same trend can be observed to an even larger extent for non-expert forecasts in the UK Covid-19 Forecasting Challenge data set.

Similar to quantiles there exist obvious tendencies for different forecast *horizons* as well. The right plot of Figure 9 shows the performance of CQR across horizons, again stratified by country. Although the effects are more diverse compared to the analysis across quantiles, CQR generally works better for larger forecast horizons. Exceptions of this rule are Croatia, which is the only country besides Poland with a negative effect of CQR, and Malta, where the trend is actually reversed and CQR corrections are most beneficial for short-term forecasts.

Both of the previous figures suggest that post-processing with Conformalized Quantile Regression is worthwhile whenever the uncertainty is comparably high, which is the case for both quantiles in the tails and large

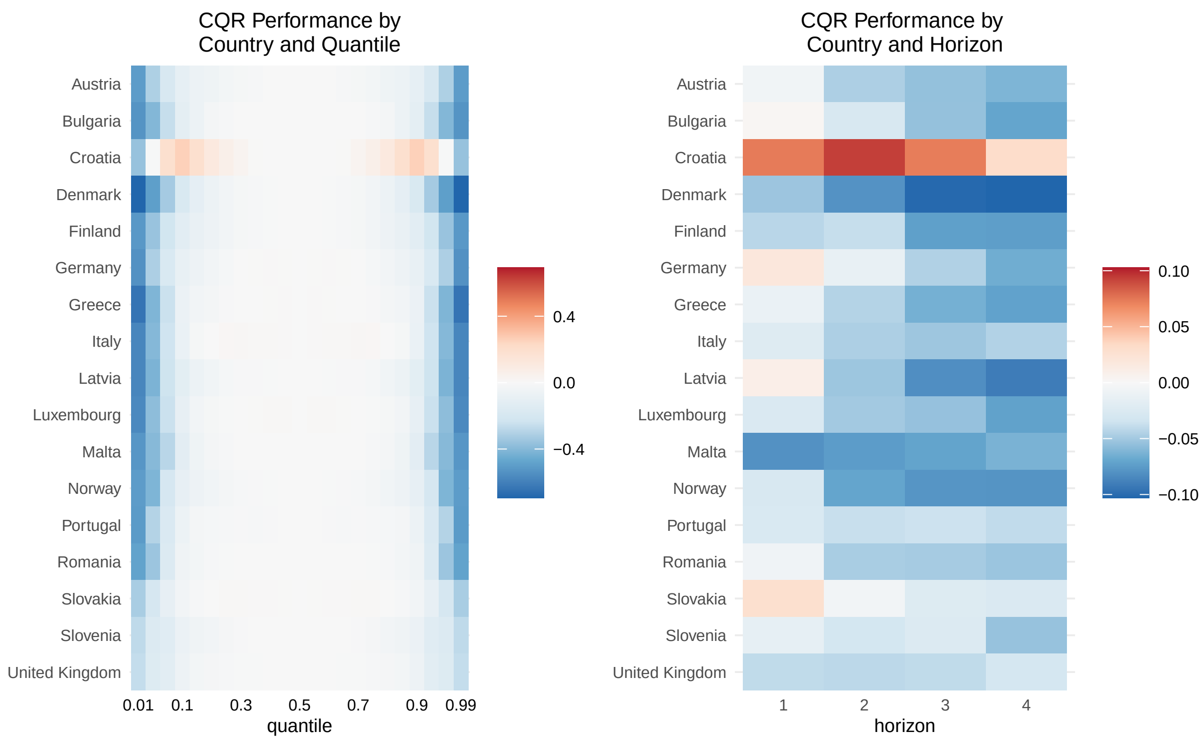


Figure 9: CQR is most beneficial for extreme Quantiles and large Forecast Horizons

Table 5: CQR Improvements by Target Type for European Forecast Hub Data excluding Poland

method	target type	interval score	dispersion
cqr	Cases	59.26	19.55
original	Cases	62.69	15.49
cqr	Deaths	0.32	0.15
original	Deaths	0.32	0.13

forecast horizons.

Lastly, Table 5 aggregates Weighted Interval Scores on the validation set by *target type*. Interestingly, the effect directions disagree for the first time: While forecasts for Covid-19 Cases benefit significantly, forecasts for Deaths become slightly worse through CQR adjustments.

2.2 Asymmetric CQR

2.2.1 Theory

This section proposes a first extension to the original CQR algorithm by relaxing the symmetry assumption. Instead of limiting ourselves to choose the *same* margin $Q_{1-\alpha}(E, I_2)$ on a *single* score vector E for adjusting the original lower and upper quantile predictions, we allow for individual and, thus, generally different margins $Q_{1-\alpha,low}(E_{low}, I_2)$ and $Q_{1-\alpha,high}(E_{high}, I_2)$ such that the post-processed prediction interval is given by

$$C(X_{n+1}) = [\hat{q}_{\alpha,low}(X_i) - Q_{1-\alpha,low}(E_{low}, I_2),\ \hat{q}_{\alpha,high}(X_i) + Q_{1-\alpha,high}(E_{high}, I_2)]\,.$$

This asymmetric version additionally requires a change in the computation of the conformity scores. Instead of considering the elementwise maximum of the differences between observed values Y_i and original bounds, we simply compute two separate score vectors:

$$E_{i,low} := \hat{q}_{\alpha,low}(X_i) - Y_i \quad \forall\, i \in I_2$$
$$E_{i,high} := Y_i - \hat{q}_{\alpha,high}(X_i) \quad \forall\, i \in I_2$$

2.2.2 CQR Downsides

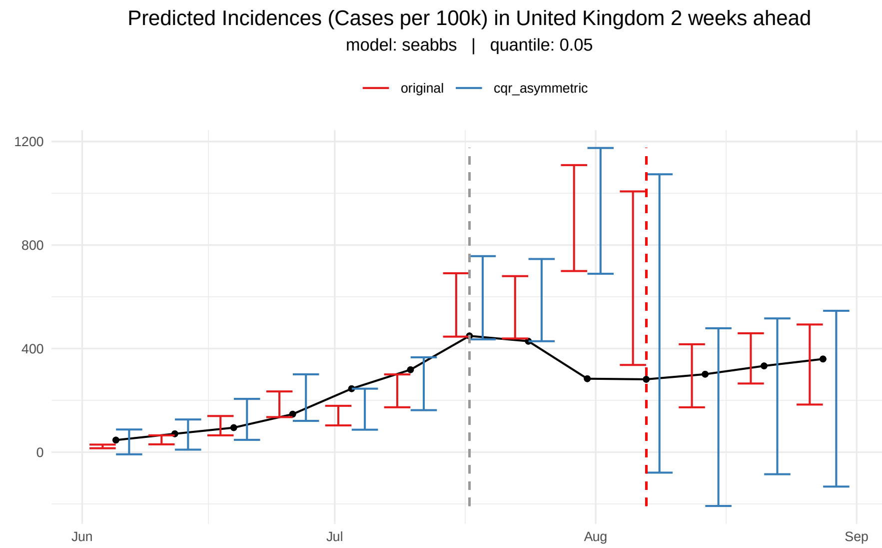


Figure 10: Illustration of CQR's slow reaction process

Figure 10 nicely demonstrates the key characteristics of asymmetric CQR: Adjustments of the lower and upper interval bounds are independent from each other. Considering the last interval on the far right the lower bound is adjusted downwards by a large amount whereas the upper bound is increased only slightly. This behaviour implies that, contrary to traditional CQR, original and corrected prediction intervals are generally *not* centered around the same midpoint.

The plot also illustrates what we have already seen in Section 2.1: Once the true value is not contained in the prediction interval and there is a large discrepancy towards the closest boundary, all CQR versions tend to *overcompensate* in the next time step. This jump can be observed from time step 9 to time step 10, where the latter is highlighted by the red dashed line. Even more problematic, the large correction margin only vanishes very gradually afterwards even if the observed Time Series has stabilized. In Figure 10 the lower quantile prediction of asymmetric CQR approaches the original lower quantile forecast very slowly after the jump in observed Cases. The following paragraphs aim to explain this inflexibility in detail and draw the connection to the underlying statistical algorithm of Section 2.1.2.

Going back to Section 2.2.1 asymmetric CQR computes two separate score vectors based on the original lower and upper quantile forecasts and the vector of observed values. To confirm our findings visually we now focus on the data subset of Figure 10.

Consider the intervals one step prior to the dashed red line. At this point in time the training set includes the first 9 elements of true values and predicted quantiles which are then used to compute a list of lower and upper scores:

```
scores_list <- compute_scores_asymmetric(
  true_values[1:9], quantiles_low[1:9], quantiles_high[1:9]
)
scores_list$scores_lower
## [1]  -31.443366  -40.808821  -29.765120  -11.289450 -141.757533 -145.173165
## [7]   -2.839344   10.514219  415.998372
```

The vector of lower scores E_{low} is given by $\hat{q}_{\alpha,low}(X) - Y$, i.e. elementwise differences of true values and predicted lower quantiles at each time step. Due to the jump from time point 9 to 10 the final element of the lower score vector has a large value of around 416.

Next, the (scalar) lower margin $Q_{1-\alpha,low}(E_{low}$ is computed:

```
margin <- compute_margin(scores_list$scores_lower, quantile)
margin
```

```
##     100%
## 415.9984
```

Due to the small sample size of 9 observations and the relatively small quantile level of 0.05 the margin is simply the *maximum* or 100% quantile of the lower scores. The *updated* lower quantile prediction for the 10th time point is simply $\hat{q}_{\alpha,low}(X_{10}) - Q_{1-\alpha,low}(E_{low})$, i.e. the original lower quantile prediction at time point 10 minus the margin:

```
quantiles_low[10] - margin
```

```
## [1] -79.18
```

which coincides with Figure 10.

The procedure now continues by consecutively adding the next elements to the vector of true values and original quantile predictions. Since the differences of observed incidences and predicted lower bounds are all much smaller for the remaining time steps, the *same* value 416 remains the maximum of the lower score vector until the end! Thus, if like in the case above, the margin always equaled the maximum score, the adjustments would stay that large independent of the future development of the time series.

In fact, the only difference from that scenario to *Step 4* of Section 2.1.2 is that the quantile of the score vector that determines the value of the margin depends on the *size* of the score vector. Since the size increases by one with each time step during the Cross Validation process, this quantile slowly declines. For instance, the margin which is responsible for adjusting forecasts at time point 11 is not simply the maximum anymore:

```
scores_list <- compute_scores_asymmetric(
  true_values[1:10], quantiles_low[1:10], quantiles_high[1:10]
)
margin <- compute_margin(scores_list$scores_lower, quantile)
margin
```

```
##      99%
## 383.5547
```

In this case the 99% quantile is an interpolation of the largest and second largest score, as implemented by the `stats::quantile()` function. Hence, even though the score outlier is not chosen directly, it strongly impacts the margins of future time steps.

The cycle proceeds in this way until the end. The conclusion of this brief case study is that all modifications of the traditional CQR algorithm suffer from a slow reaction time towards distribution shifts and particularly sudden jumps within observed values and original forecasts. This major downside of Conformalized Quantile Regression is an immediate consequence of the *margin* computation which finally determines the magnitude of forecast adjustments.

2.2.3 Results

Contrary to traditional CQR, the effect of asymmetric CQR highly depend on the underlying data set. Table 6 shows that this first CQR modification is beneficial for the UK data set by improving the out-of-sample Weighted Interval Score, yet the opposite is the case for the European Forecast Hub.

Table 6: Performance of asymmetric CQR on Validation Set

method	uk interval score	hub interval score
cqr_asymmetric	63.97	34.37
original	65.74	29.84

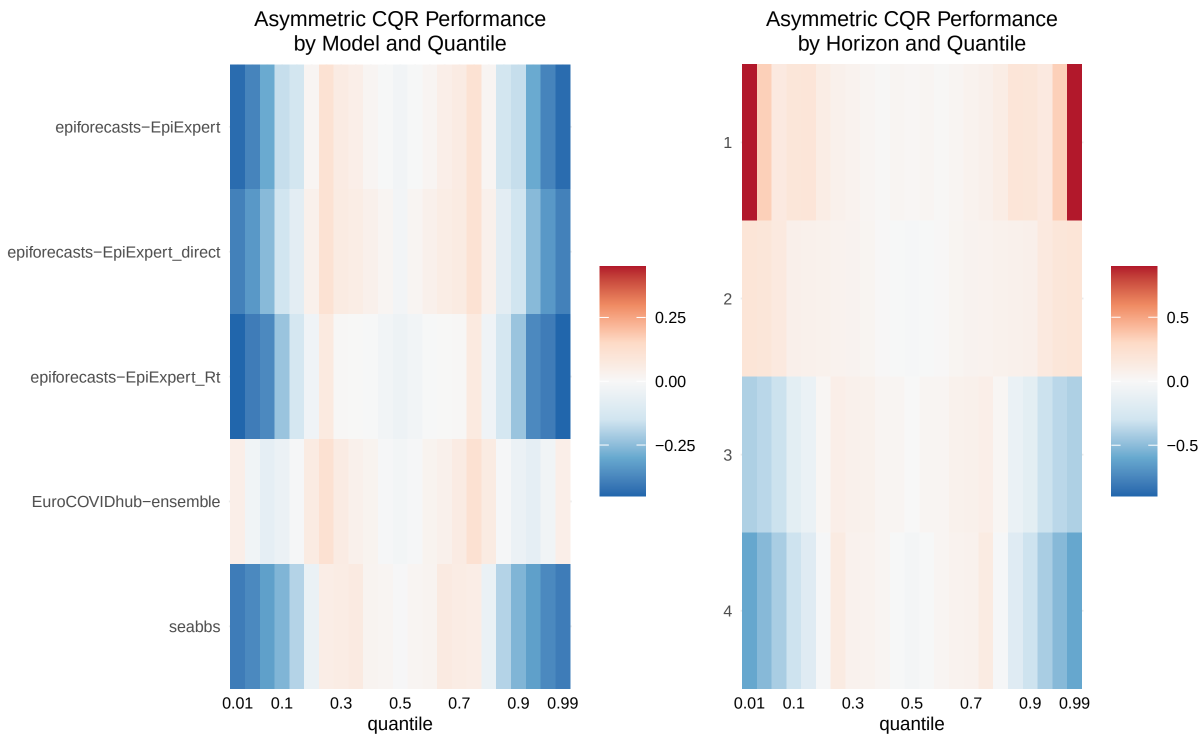


Figure 11: Mixed Results of asymmetric CQR on UK data

To get a better intuition which circumstances contribute to a positive outcome, we analyze the effects in more granularity. Figure 12 illustrates the relative improvements by asymmetric CQR for different forecasting models and different forecasting horizons stratified by the quantile level for the UK data. We exclude the `EuroCOVIDhub-baseline` model where the adjustments lead to a much *worse* score across all quantile levels.

The general trends are similar to vanilla CQR: Areas of higher uncertainty profit more from post-processing. While the effect is still positive for quantiles less than 0.15 or greater than 0.85, the *original* predictions are more accurate for the center quantiles across all models. The same statement holds for three or four week-ahead predictions. For short term forecasts, however, the effect is negative across *all* quantile levels.

Recall that traditional CQR improved performance for almost all European countries with the huge outlier Poland where the opposite effect could be observed. In light of the discussion in Section 2.2.2 it is not surprising that Poland keeps its outlier role for asymmetric CQR as well, since the slow reaction process to distribution shifts is coupled with the core of the CQR algorithm and not diminished by merely relaxing the symmetry assumption. In contrast to Figure 7, however, the relative effect of asymmetric CQR is *negative* for almost all of the remaining countries.

Thus, we detect first evidence that the (at least partly) promising results for the smaller UK data set do *not* transfer to the larger European Forecast Hub in this case. Figure 13 convincingly shows that the performance indeed dropped for each quantile and horizon category. While the asymmetric adjustments still result in slightly better predictions for intervals with large nominal coverage level, the left plot is dominated by the negative effect for centered quantiles, except for the median prediction which remains untouched by all CQR

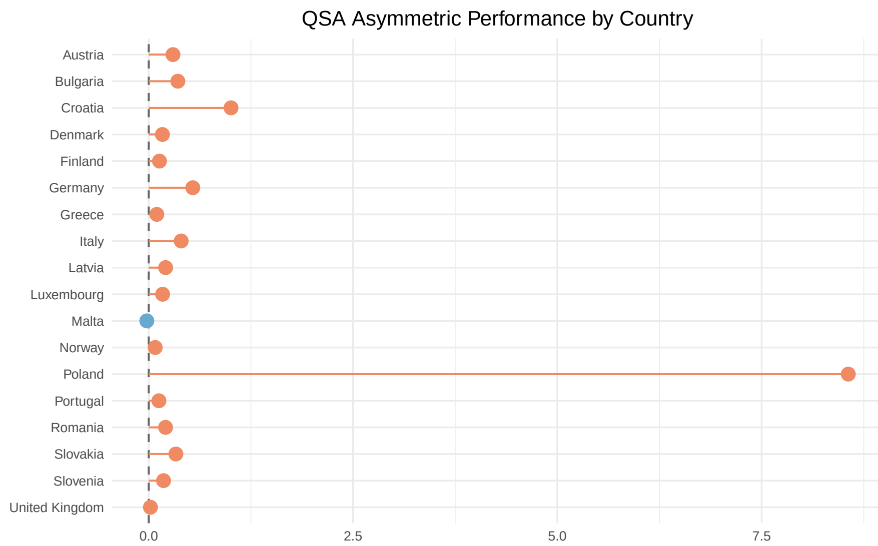


Figure 12: Asymmetric CQR has negative effects on almost all countries

Table 7: Performance of asymmetric CQR for Covid-19 Cases and Deaths

method	target type	uk interval score	hub interval score
cqr_asymmetric	Cases	127.78	70.59
original	Cases	131.29	62.69
cqr_asymmetric	Deaths	0.16	0.50
original	Deaths	0.18	0.32

versions. The right plot suggests that corrections with asymmetric CQR should be avoided altogether when only grouping by forecast horizons and not considering quantile levels separately.

Finally, Table 7 summarizes the dissimilar effects on the two data sets very clearly: Aggregated over all other categories, asymmetric CQR *does* improve the WIS for Covid-19 Cases and Deaths in the UK data. In strong contrast, the post-processed intervals perform *much* worse than the original forecasts across both target types in the European Forecast Hub data set.

In conclusion, asymmetric Conformalized Quantile Regression can lead to improved prediction intervals as it is the case for the UK data set. However, the vast majority of countries in the European Forecast Hub does not benefit from the first proposed CQR variant. Compared to the traditional CQR algorithm, giving up on symmetry leads to a worse performance across both data sets. It is worth noting that allowing for separate lower and upper margins does *not* cause significant overfitting as one might assume, the original CQR algorithm outperforms the asymmetric version even on the training set! The *magnitude* of the performance differences between the two methods is analyzed in Section 4.

2.3 Multiplicative CQR

2.3.1 Theory

On top of the asymmetric CQR modification described in Section 2.2, we can extend the CQR algorithm further. So far, the adjustments to the original prediction interval were always chosen in *additive* form. It may be useful to leverage the *magnitude* of the original bounds more explicitly by using *relative* or *multiplicative* adjustments.

Hence, we again compute separate margins $Q_{1-\alpha,low}(E_{low}, I_2)$ and $Q_{1-\alpha,high}(E_{high}, I_2)$ which are now

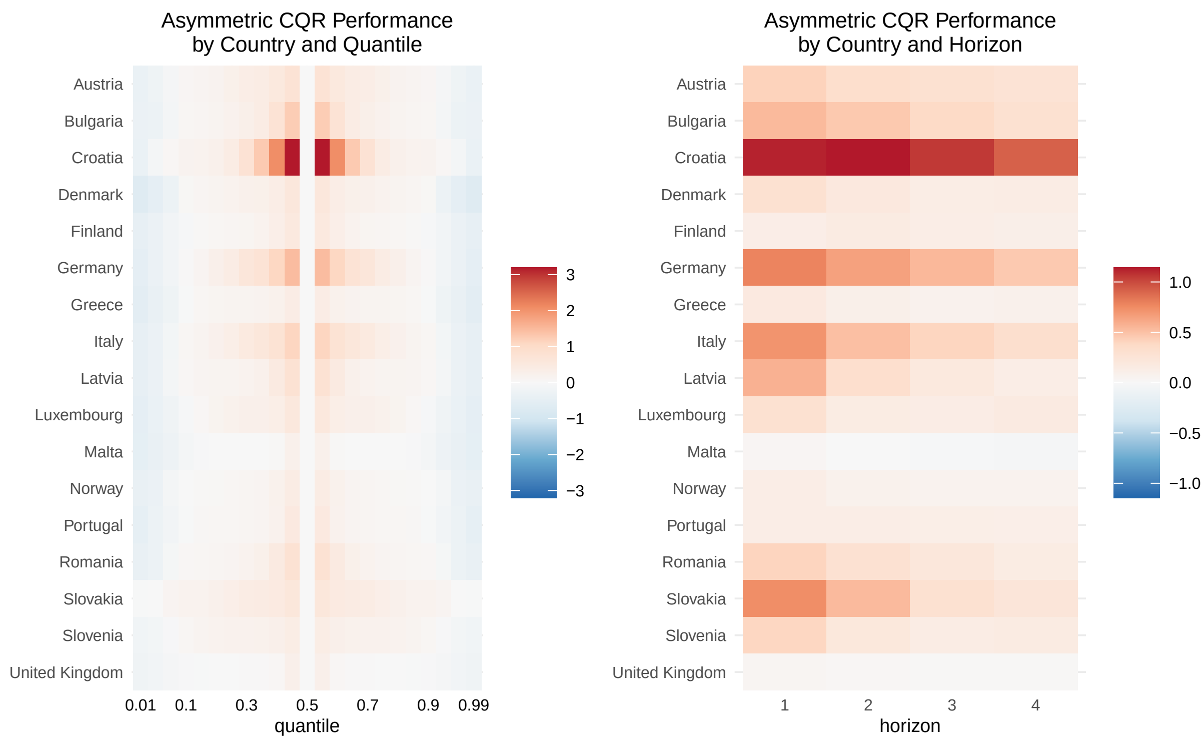


Figure 13: Out-of-sample performance of asymmetric CQR on European Forecast Hub data

multiplied with the existing forecasts. The post-processed prediction interval is thus given by

$$C(X_{n+1}) = \left[\hat{q}_{\alpha,low}(X_i) \cdot Q_{1-\alpha,low}(E_{low}, I_2),\ \hat{q}_{\alpha,high}(X_i) \cdot Q_{1-\alpha,high}(E_{high}, I_2)\right].$$

Just like the asymmetric version, the computation of the score vectors is changed accordingly to respect the new multiplicative relationship:

$$E_{i,low} := \frac{Y_i}{\hat{q}_{\alpha,low}(X_i)} \quad \forall\, i \in I_2$$

$$E_{i,high} := \frac{Y_i}{\hat{q}_{\alpha,high}(X_i)} \quad \forall\, i \in I_2,$$

where we have to exclude original predictions with the value 0. Since all Covid-19 Cases and Deaths are non-negative, we threshold the scores at zero such that $E_{i,low}$ equals 0 whenever $\hat{q}_{\alpha,low}(X_i) \leq 0$.

Note that the actual limiting value

$$\lim_{\hat{q}_{\alpha,low}(X_i)\to 0} \frac{Y_i}{\hat{q}_{\alpha,low}(X_i)} = \infty$$

does not make sense here since infinite scores would cause infinite lower margins $Q_{1-\alpha,low}(E_{low}, I_2)$, which in return causes infinite updated lower bounds. Thus, the value 0 is deliberately chosen to minimize the influence of negative original forecasts and keep the corrected lower quantile predictions always nonnegative.

2.3.2 Regularization

While the idea of multiplicative correction terms is appealing, it turns out that the approach above is flawed in two ways:

1. Recall that the (lower) margin $Q_{1-\alpha,low}(E_{low}, I_2)$ basically *picks* a value of the score vector E_{low} at a given quantile level. The score vectors are computed for each combination of *location*, *model*, *target type*, *horizon* and *quantile*, i.e. the number of values in the score vector is identical to the number of

distinct time points in the training set. For short time series such as our small UK data set, the margin selects the *largest* value in the score vector for small levels of α such as 0.01 or 0.05, where each such value represents a *ratio* of observed Y_i and original prediction $\hat{q}_{\alpha,low}(X_i)$.

As one might guess, these factors frequently get very large for small initial quantile predictions $\hat{q}_{\alpha,low}(X_i)$ such that the computed margin $Q_{1-\alpha,low}(E_{low}, I_2)$ for post-processing is unreasonably large. In fact, the margin can remain huge if there exists a *single* outlier in the score vector. In particular, this naive multiplicative version frequently adjusts the lower quantile prediction to a higher value than its upper quantile counterpart, leading to (an extreme form of) quantile crossing.

We counteract this sensitivity to outliers by *reducing the spread* of the score vector to make it more well behaved. Since we deal with multiplicative factors it makes no sense to standardize them to zero mean and unit variance. Instead, we regularize the score vector by pulling all values closer to 1, while keeping all values nonnegative and respecting their *directions*, i.e. values smaller than 1 remain smaller than 1 and prior values greater than 1 remain greater than 1.

This goal is achieved by a *root transformation*. Since a greater spread of the score vector should lead to larger regularization we settled on the corrections

$$E_{i,low}^{reg} = E_{i,low}^{\left(\frac{1}{\sigma_{E_{low}}}\right)}, \quad E_{i,high}^{reg} = E_{i,high}^{\left(\frac{1}{\sigma_{E_{high}}}\right)},$$

where σ_E denotes the standard deviation of the corresponding score vector.

Remark: We first restricted the scaling to the case $\sigma_{E_{low}}, \sigma_{E_{high}} > 1$, i.e. the spread of the score vector should only get reduced. However, the above correction empirically proved to be beneficial even for $\sigma_{E_{low}}, \sigma_{E_{high}} < 1$ in which case the score variance gets *increased*. Therefore we removed the original restriction and only handled the (unlikely) cases of constant score vectors with $\sigma_{E_{low}} = 0$ or $\sigma_{E_{high}} = 0$ separately.

2. Chances are high that at least *one* of the original true values Y_i is larger than its corresponding lower quantile prediction $\hat{q}_{\alpha,low}(X_i)$ such that the maximum of the (regularized) score vector is still larger than 1. Thus, the lower bound for small quantiles α is almost *always* pushed upwards. The same logic applies to the upper bound in which case the *entire interval* is shifted to the top. This behaviour is usually not desired.

   To prevent interval shifts, we add the additional constraint that the lower and upper margin must multiply to 1, i.e.

$$Q_{1-\alpha,low} \cdot Q_{1-\alpha,high} \stackrel{!}{=} 1.$$

   Hence, when the *lower* bound is adjusted upwards ($Q_{1-\alpha,low} > 1$), the upper bound must decrease ($Q_{1-\alpha,high} < 1$) and the interval becomes smaller. Similarly, when the *upper* bound is adjusted upwards ($Q_{1-\alpha,high} > 1$), the lower bound must decrease ($Q_{1-\alpha,low} < 1$) leading to larger intervals overall after post-processing.

2.3.3 Results

As noted in Section 2.3.2, *naive* multiplicative Conformalized Quantile Regression without any regularization is useless for post-processing quantile predictions. Typically, one would observe strong overfitting on the training set such that the training performance indicated promising effects, yet the scores on the validation set would be *much* worse than the original forecasts. Further, the adjusted intervals would be shifted upwards and usually be too large.

Before numerically evaluating the performance of *regularized* CQR, it is instructive to look at a visual comparison of all three CQR modifications for one specific feature combination as shown in Figure 14.

The effect of scaling the score vectors in step one of the regularization procedure and constraining lower and upper margins in the second step can be detected immediately: Similar to vanilla CQR, the multiplicatively corrected intervals are now centered around the same midpoint as the original forecasts. In strong contrast

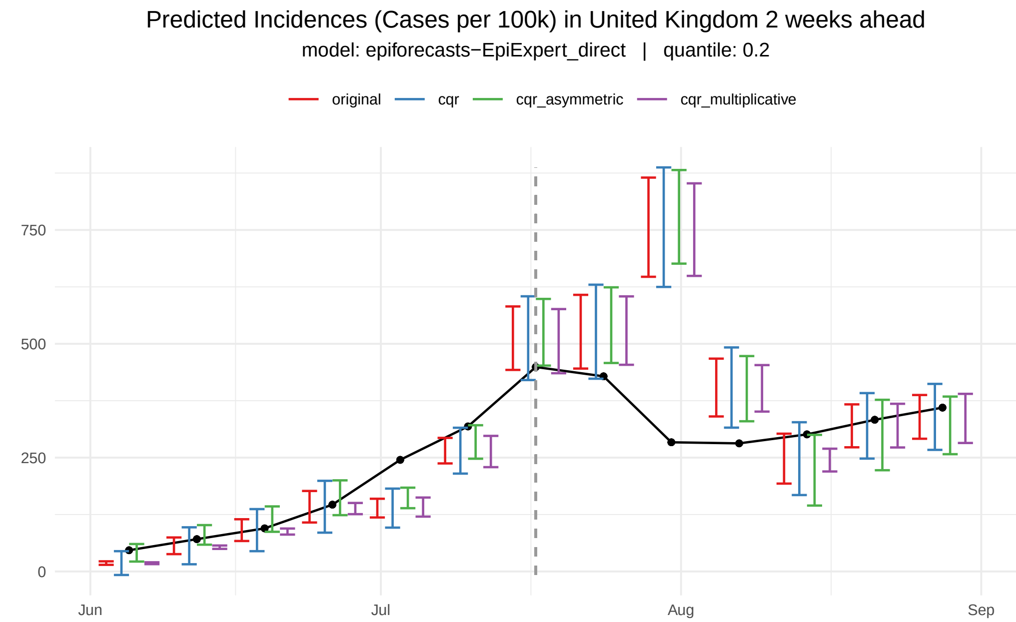


Figure 14: Comparison of CQR variations on the UK data set

Table 8: Performance of Multiplicative CQR on the Training Set

method	interval score	dispersion	underprediction	overprediction
cqr_multiplicative	24.49	5.98	18.01	0.51
original	23.62	4.16	18.88	0.58

to the additive CQR versions, however, the issue of interval explosion has not only been diminished by downscaling the scores, but rather *reversed* such that the interval widths now actually *decreased* at most time points and generally appear too narrow.

Moreover, we no longer have any theoretical guarantees of improved forecasts on the training set since Theorem 2.1 only applies to the original additive and symmetric version of CQR. This fact is confirmed empirically by Table 8 which shows the Weighted Interval Score aggregated over all categories of `model`, `target_type`, `horizon` and `quantile`. Indeed, the multiplicative adjustments result in a slightly worse Weighted Interval Score on the training set.

Recall that this behaviour is different from the unregularized version, which performed better in-sample than the original forecasts across almost all feature combinations. On the flipside, the out-of-sample performance improved dramatically compared to the naive implementation, even though it ultimately does *not* lead to a score improvement as shown in Table 9, stratified by the forecasting `model`. Interestingly, multiplicative CQR indicates the best *relative* performance for the `EuroCOVIDhub-baseline` model where the additive CQR algorithms struggle the most. Overall the score differences across different forecasting models appear to be smoothed out compared to the previous CQR versions which also results from the regularization component that is unique to the multiplicative modification.

The impression of too narrow adjusted intervals does not generalize to the entire data set. The *dispersion* column in Table 9 shows that the intervals are downsized only for some models such as `epiforecasts-EpiExpert` whereas for others like `epiforecasts-ensemble` the distance between lower and upper bound gets larger on average.

Table 10 suggests a connection of the dispersion change by multiplicative CQR with the `quantile` level. Aggregated over all models, target types and horizons the dispersion value is increased by a large amount

Table 9: Performance of Multiplicative CQR by Model on the Validation Set

method	model	interval score	dispersion
cqr_multiplicative	epiforecasts-EpiExpert	71.60	10.05
original	epiforecasts-EpiExpert	67.74	12.07
cqr_multiplicative	EuroCOVIDhub-baseline	29.85	17.95
original	EuroCOVIDhub-baseline	29.61	5.92
cqr_multiplicative	EuroCOVIDhub-ensemble	61.24	15.48
original	EuroCOVIDhub-ensemble	56.07	14.00
cqr_multiplicative	seabbs	98.18	9.14
original	seabbs	95.11	14.03

Table 10: Dispersion of Multiplicative CQR by Quantile on the Validation Set

method	0.01	0.025	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45
cqr_multiplicative	8.79	7.40	11.98	16.83	18.26	18.82	18.24	17.53	14.60	10.12	5.22
original	2.82	5.82	9.65	14.86	17.84	18.99	18.83	17.44	14.71	10.97	6.08

for extreme quantiles but remains in a similar range to before for quantiles in the center of the predictive distribution. This behaviour is in line with the previously seen additive correction methods and emphasizes that Figure 14 is not representative for the entire UK data set.

Overall, we must conclude that the original CQR algorithm as described by Romano, Patterson, and Candès (2019) can *not* be modified towards multiplicative margins in any straightforward way. For this reason, we neither extend the analysis of multiplicative CQR to the European Forecast Hub data set nor include it in the method comparison in Section 4.

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 15.

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), with the number of confidence intervals p, the certainty level of a confidence interval α_p and the number of observations n, shows how the QSA weights $\mathbf{w}$ influence the WIS.

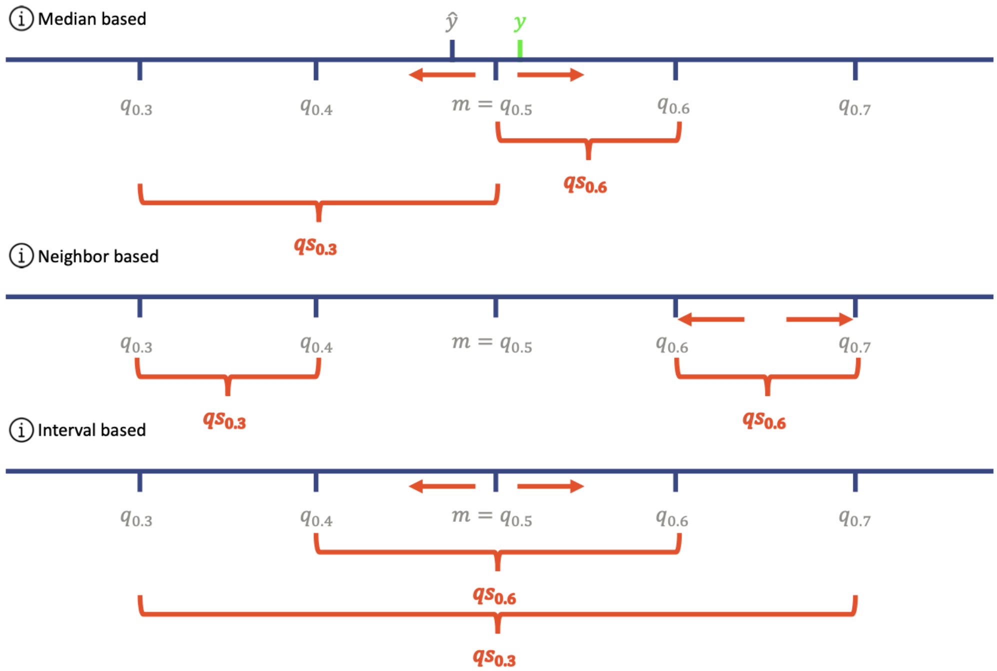


Figure 15: Quantile Spreads for different Basis

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

For a given prediction interval level of α_i, by varying the QSA factor w^l_i for the lower and w^u_i for the upper bound, QSA moves the quantiles from their original values $l_{i,j}$ and $u_{i,j}$ to their adjusted values $l^*_{i,j}$ and $u^*_{i,j}$. QSA factor values larger than 1 lead to an increase in the prediction interval, thus $w^l_i > 1$ reduces the value of $l^*_{i,j}$ and $w^u_i > 1$ increases the value of $u^*_{i,j}$. These changes have two effects, on the one side an increase in w^l_i and w^u_i 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.

The `postforecasts` package implements the QSA optimization in three, the weight vector $\mathbf{w}$ restricting, flavors:`qsa_uniform`, `qsa_flexible_symmetric` and `qsa_flexible`. These are listed in equations (reference).

$$
\begin{aligned}
uniform: w_i = c \quad & i \in [0, 1, \ldots, p-1, p], \quad c \in \mathbb{R} \\
flexibel\_symmetric: w_i = w_{p-i} \quad c_i \quad & i \in [0, 1, \ldots, \frac{p}{2} - 1], \quad c_i \in \mathbb{R} \\
& flexibel: w_i \in \mathbb{R}
\end{aligned}
$$

`qsa_uniform` restricts all weight vector values to be identical. `qsa_flexible_symmetric` only restricts pair wise adjustments to be identical. It essentially represents unrestricted QSA with interval based adjustments. Finally `qsa_flexible` is completely unrestricted as each quantile is adjusted separately.

In addition to different flavors, the `postforecasts` package also provides the option to regularize the optimization. Equation (reference) depicts the penalization term that is added to the WIS. It is designed to penalize differences between weight vector values by adding a factor proportional to the sum of squared deviation of the weight vector values from there mean. It therefor regularizes towards the `qsa_uniform` method and only has an effect for the `qsa_flexible_symmetric` and `qsa_flexible` flavors.

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^p}{\arg\min}\ WIS_\alpha(\mathbf{y}) + r \cdot Pen(\mathbf{w}), \quad Pen(\mathbf{w}) = \sum_{i=1}^{p}(w_i - \bar{w})^2$$
$$\text{s.t.} \qquad \bar{w} = \frac{1}{p}\sum_{i=1}^{p} w_i$$

3.2 Optimization

Underneath the hood, `postforecasts` accesses the `optim` function from the R package stats[8]. From the available optimization methods, `BFGS` and `L-BFGS-B` turned out to be the most reliable for QSA. `BFGS` is named after Broyden, Fletcher, Goldfarb and Shanno and a quasi-Newton method. `L-BFGS-B`, is a limited memory version of `BFGS` and additionally also support box constraints. As default value we set the optimization method to `L-BFGS-B` as it converges faster than `BFGS` in our data set, due to its limited memory property. The time gain is especially important for the `qsa_flexible_symmetric` and `qsa_flexible` methods which take considerably longer than `qsa_uniform` for a large number of quantiles. Furthermore `L-BFGS-B` also has the advantage that we can lower bound the quantile spread factor to not drop below zero, hence we can exclude quantile crossing with the median. The optimization method can be accessed in the function `update_predictions` by means of the `optim_method` argument. For `L-BFGS-B`, the lower and upper bound box constraints can be set with the arguments `lower_bound_optim` and `upper_bound_optim`. Besides the use of `optim`, `postforecasts` also provides a line search optimization which is used by setting the `optim_method` to `line_search`. As the run time increases exponentially with the parameter spaces, this method is currently restricted to the `qsa_uniform`. Here, the method runs QSA for all values of the QSA factor within a sequence. This sequence is defined by its upper and lower values set with the arguments `lower_bound_optim` and `upper_bound_optim` as well as its step size set by `steps_optim`. Regarding the QSA optimization functions shape, there is a potential issue: Due to the trade-off between sharpness and coverage defining the WIS, it can happen that an interval of values for the QSA factor result in the same score. In other words, the WIS loss function has plateaus. This becomes less likelier the more observations and quantiles are available, nevertheless it still has to be kept in mind. The `line_search` optimization handles multiple optima by choosing the value closest to 1, hence the smallest possible adjustment of the quantiles. In essence this is a regularization. For the `BFGS` and `L-BFGS-B` plateaus means that both methods can converge to different optima while attaining the same WIS. In a future version of the package we aim to tackle this by adding a line search after the use of `BFGS` and `L-BFGS-B` in order to find the optima closest to 1 and thereby regularize the results. Furthermore, due to the long run times, especially for large data sets and small initial training periods, the `postforecasts` package also provides the option to run QSA in parallel. The parallelisation uses the R package foreach[9]. It can be activated by defining a parallel processing environment in R and then setting the argument `parallel=TRUE` within the `update_predictions` function. A parallel processing environment is defined by defining the number of cores available with the `registerDoParallel` function from the doParallel[10] package. For a device with two available cores the code is as follows:

```
library(postforecasts)
library(doParallel)
registerDoParallel(cores = 2)

#df_updated_parallel <- update_predictions(df,
#  methods = "qsa_flexible",
```

[8] https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/optim
[9] https://www.rdocumentation.org/packages/foreach/versions/1.5.2
[10] https://www.rdocumentation.org/packages/doParallel/versions/1.0.16

```
#   parallel = TRUE,
#   verbose = TRUE)
```

To observe the progress of the computations users can set the `verbose=TRUE` in order to print the logging output that the `foreach` function provides.

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. 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 variations: Moving the interval bounds farther apart or closer together can happen in *symmetric* or *asymmetric* manner 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 the strongest individual model within each area of the feature space. 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 methods, 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 method

Consider one particular feature 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, $\mathbf{l}_1, \ldots, \mathbf{l}_k \in \mathbb{R}^n$ vectors of original lower quantile predictions and $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{R}^n$ vectors of original upper quantile 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:

$$
\begin{aligned}
\mathbf{w}^* = \underset{\mathbf{w} \in [0,1]^k}{\arg\min}\, IS_\alpha(\mathbf{y}) &= \underset{\mathbf{w} \in [0,1]^k}{\arg\min} (\mathbf{u} - \mathbf{l}) + \frac{2}{\alpha} \cdot (\mathbf{l} - \mathbf{y}) \cdot \mathbb{1}(\mathbf{y} \leq \mathbf{l}) + \frac{2}{\alpha} \cdot (\mathbf{y} - \mathbf{u}) \cdot \mathbb{1}(\mathbf{y} \geq \mathbf{u}), \\
\text{with} \qquad \mathbf{l} &= \sum_{j=1}^{k} w_j \mathbf{l}_j, \quad \mathbf{u} = \sum_{j=1}^{k} w_j \mathbf{u}_j \\
\text{s.t.} \qquad \|\mathbf{w}\|_1 &= \sum_{j=1}^{k} w_j = 1,
\end{aligned}
$$

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

Hence, we choose the (nonlinear) Interval Score (Section 1.1.3) as our objective function that we minimize subject to linear constraints. The optimization step is implemented with the nloptr[11] package (Ypma and

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

Table 11: WIS of all Post-Processing Methods on Training and Validation Set on UK Data

method	validation score	training score	dispersion
ensemble	57.69	18.22	21.73
qsa_uniform	60.00	20.88	26.84
qsa_flexible	60.47	19.48	25.31
qsa_flexible_symmetric	60.92	20.49	33.22
cqr	62.15	20.82	24.10
cqr_asymmetric	63.97	14.46	17.99
original	65.74	23.62	12.00

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 feature combination. We omit the subscripts at this point to keep the notation clean.

The Interval Score always considers *pairs* of quantiles α and $1-\alpha$ as outer bounds of a $(1-2\alpha)\cdot 100\%$ prediction interval. The best results are achieved when a separate weight vector for each quantile pair is computed. Since our data sets contain 11 quantile pairs, 2 target types and 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 $18\cdot 528 = 9504$ optimization problems for the European Hub Data Set.

Due to this high computational cost the *maximum number of iterations* within each optimization is an important hyperparameter that balances the trade-off between computational feasibilty and sufficient convergence of the iterative optimization algorithm. 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 separate *validation set* is possible.

4.2 Comparison of CQR, QSA & Ensemble

Now that we have introduced *Conformalized Quantile Regression* in Section 2, *Quantile Spread Averaging* in Section 3 and the *Ensemble* Model in Section 4.1, the obvious question is which of the methods performs best. Thus, this section is dedicated to a detailed comparison across various feature combinations. Due to the high computational demands for Quantile Spread Averaging, we limit the discussion to the compact UK Covid-19 Forecasting Challenge data set. The results that constitute the starting point of the analysis can be generated with the following commands:

```r
library(postforecasts)

df_updated <- uk_data |>
  update_predictions(
    methods = c(
      "cqr", "cqr_asymmetric", "qsa_uniform", "qsa_flexible", "qsa_flexible_symmetric"
    ),
    cv_init_training = 0.5
  ) |>
  collect_predictions() |>
  add_ensemble()
```

Table 11 collects the Weighted Interval Scores for each method on the training and validation set, sorted by

Table 12: Fraction of Feature Combinations where largest Ensemble Weight exceeds Threshold

> 0.5	**> 0.9**	**> 0.99**
0.93	0.69	0.53

increasing validation score. There are a couple of interesting findings:

- All six custom methods improve out-of-sample performance compared to the original predictions on the UK data set.
- All three QSA versions lead to lower validation scores than any CQR variant. Thus, based on this first impression, the familiy of QSA post-processing methods clearly outperforms the CQR algorithm for the UK data.
- The ensemble model is the clear winner: Combining information from multiple QSA and CQR methods works better on new data than any individual method. This suggests that the five building block methods are not redundant in the sense that they have different strengths and weaknesses depending on the location in feature space.
- The asymmetric CQR method suffers most from *overfitting*. Compared to the European Forecast Hub data, where overfitting was not a major issue as described in Section 2.2, the more flexible CQR modification results in the lowest training but highest validation score for the small UK data set.
- In general, additional model *restrictions* such as identical weights in case of `qsa_uniform` and/or a symmetry assumption in case of `cqr` and `qsa_flexible_symmetric` have some kind of *regularization* effect which leads to better generalization to the validation set. Indeed, the *least* flexible version of both method frameworks indicate the best validation performance and yet, unsurprisingly, the highest training score.
- All methods improve the original forecasts by *expanding* the prediction intervals on average which is indicated by the larger *dispersion* values. `qsa_flexible_symmetric` produces by far the widest intervals on average, yet we can not observe a correlation of better validation scores and narrower or wider prediction intervals.

Table 11 convincingly demonstrates that the ensemble model leads to the best forecasts. Thus, we want to gain more insight how the ensemble predictions are created for this specific use case. Recall that the weights for each of the five building block methods are nonnegative and (in case of convergence) sum to one. A different set of weights is computed for each of the $6 \cdot 2 \cdot 4 \cdot 23 = 1104$ combinations of `model`, `target_type`, `horizon` and `quantile`. One question of interest is if the optimization algorithm tends more towards evenly distributed weights within each combination by assigning a positive weight to many component methods, or rather selects a single winning method with a weight of 1 and all remaining methods are discarded with a weight of 0.

Table 13 provides a first impression on the weight distribution of the ensemble model. In 93% of all feature combinations a single method has a larger weight than all of the competing methods combined. Further, in more than half of the optimization solutions the ensemble emulates one particular method by concentrating the entire weight mass on a single point. Hence, although we can not observe a strict *winner takes it all* procedure, the weight distribution is heavily skewed towards one contributing component at each location in feature space.

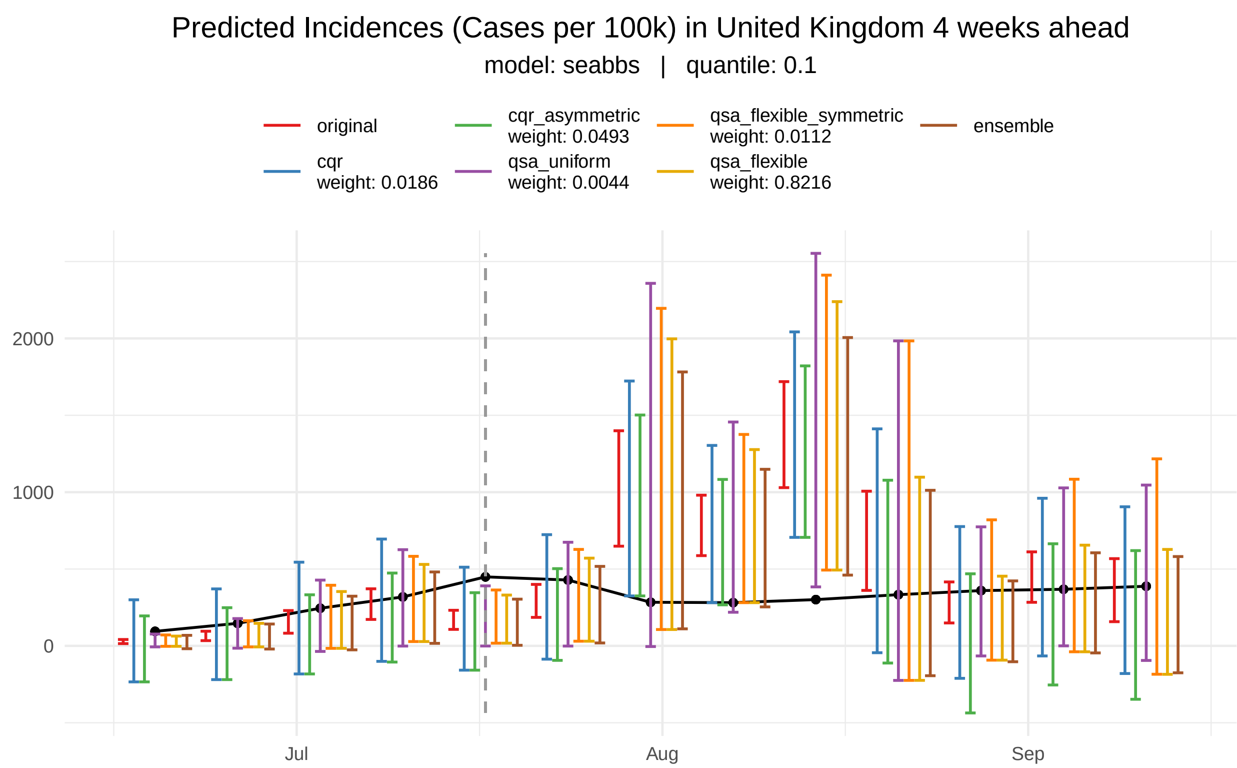
Now that we have discovered that for most covariate combinations there seems to be a single method that clearly outperforms its competition, we want to find out *which* of the five post-processing methods takes the winning trophy most often. Table 13 displays the frequency with which the ensemble assigns the largest weight to each method. The first row contains the absolute number of times and the second row the fraction of all 1104 optimization problems where the methods contributed most to the ensemble model.

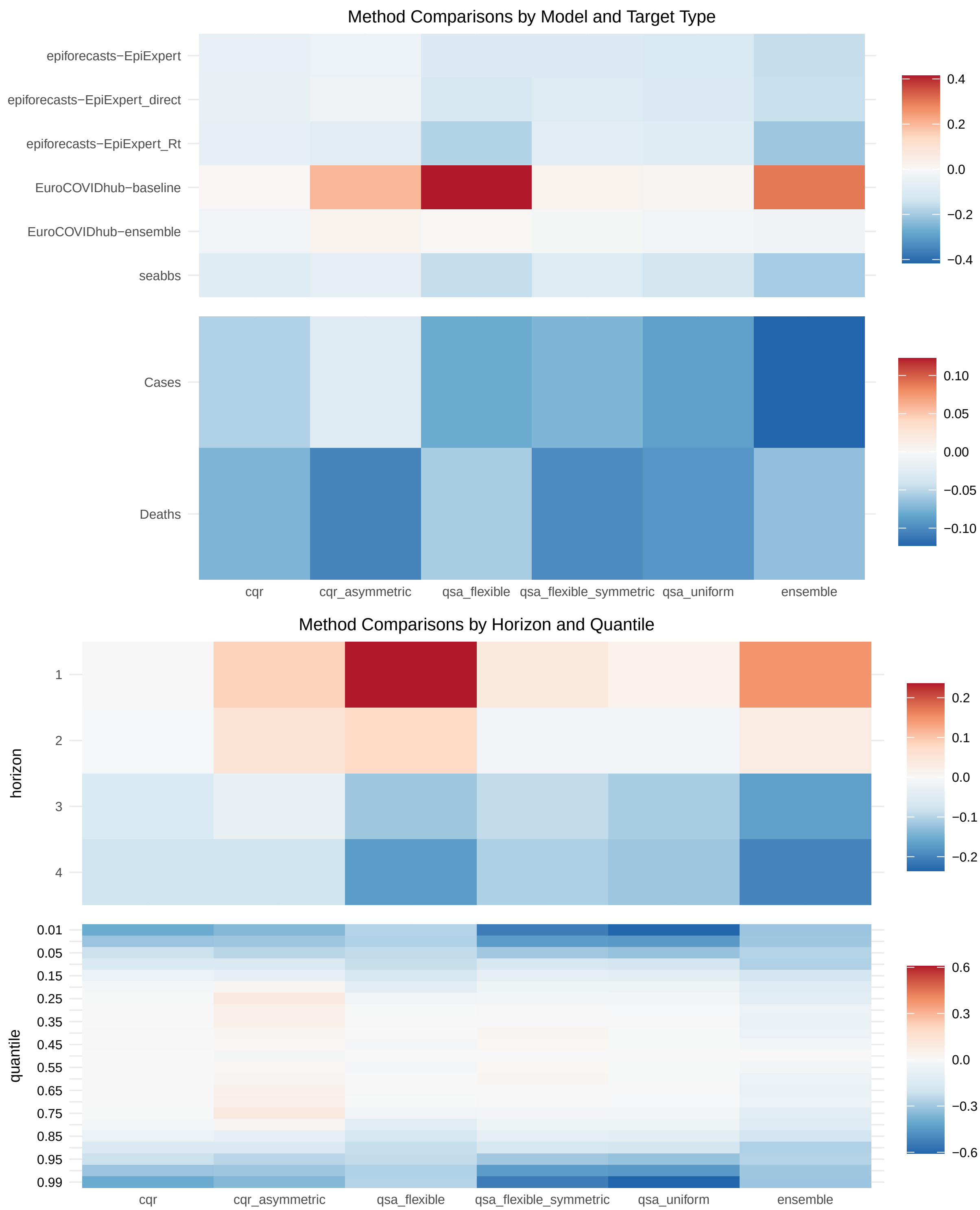
In almost 90% of cases the largest weight is given to either the asymmetric CQR or the flexible QSA method

Table 13: Number (Row 1) and Fraction (Row 2) of largest Ensemble Weights for each Method

cqr	cqr_asymmetric	qsa_uniform	qsa_flexible_symmetric	qsa_flexible
72.00	450.00	2	50.00	530.00
0.07	0.41	0	0.05	0.48

whereas the uniform QSA method almost never has the largest impact on the ensemble. This finding is particularly interesting in comparison with Table 11: Since the ensemble is fitted on the *training* set, it distributes the weights according to training set performance of each individual method. `cqr_asymmetric` and `qsa_flexible` indeed have the best training scores whereas `qsa_uniform` performs worst in the training set which corresponds exactly to the order of Table 13. In light of this connection it appears even more surprising that the ensemble method generalizes very well to out-of-sample data while simultaneously rewarding potentially overfitting methods like `cqr_asymmetric` during its own learning process.





5 Conclusion

A First Appendix

B Second Appendix

References

Bassett, Gilbert, and Roger Koenker. 1982. "An Empirical Quantile Function for Linear Models with | Operatornameiid Errors." *Journal of the American Statistical Association* 77 (378): 407. https://doi.org/10.2307/2287261.

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/.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Romano, Yaniv, Evan Patterson, and Emmanuel J. Candès. 2019. "Conformalized Quantile Regression." *arXiv:1905.03222 [Stat]*, May. http://arxiv.org/abs/1905.03222.

Tibshirani, Ryan. 2019. "Advances and Challenges in Conformal Inference." Carnegie Mellon University. https://www.stat.cmu.edu/~ryantibs/talks/conformal-2019.pdf.

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