1 Method Comparison

This chapter aims to compare the effectiveness of all post-processing methods that were introduced throughout 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 comparable final score by *shrinking* the prediction intervals leading to higher precision. One can imagine even more variations: Moving the interval bounds farther apart or closer together can happen *symmetricly* or *asymmetricly* and the interval's midpoint might stay *fixed* or get *shifted* by the post-processing algorithm.

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.

1.1 Ensemble Model

There exist various options how to generally combine multiple building blocks into one ensemble. We chose an approach that can be efficiently computed by well-understood optimization algorithms and, at the same time, is highly interpretable. 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 the corresponding 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, $\mathbf{l}_1, \dots, \mathbf{l}_k \in \mathbb{R}^n$ vectors of original lower quantile predictions and $\mathbf{u}_1, \dots, \mathbf{u}_k \in \mathbb{R}^n$ 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:

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

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

Hence, we choose the (nonlinear) Interval Score (**??**) as our objective function that we minimize subject to linear constraints. The optimization step is implemented with the nloptr[1] 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 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

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

computed. Since our data sets contain 11 quantile pairs, 2 target types, 4 horizons and we consider 6 different forecasting models, 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.

1.2 Comparison of CQR, QSA & Ensemble

Now that we have introduced *Conformalized Quantile Regression* in **??**, *Quantile Spread Averaging* in **??** and the *Ensemble* Model in Section 1.1, the obvious question is which of the methods performs best. This section conducts a detailed comparison across various feature combinations. Due to the high computational demands of 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:

```
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()
```

Figure 1 provides a visual illustration of original and adjusted prediction intervals of all post-processing methods including the ensemble. It is important to note that the relative differences between all methods highly depend on the selected feature combination. For demonstration purposes we chose the `seabbs` model in combination with a large forecast horizon and a small quantile. As we have seen on multiple occasions throughout the previous chapters, this configuration correlates with large adjustments to the original forecasts. For a random example of the feature space the effect sizes and the effect differences between the methods would likely be much smaller such that Figure 1 is not representative but still useful in order to gain some basic intuition.

Except for the last observations on the horizontal axis the forecasts of the two CQR versions are quite similar and significantly closer to the original predictions than the QSA intervals. Within the QSA family `qsa_flexible` and `qsa_flexible_symmetric` produce almost identical corrections whereas `qsa_uniform` behaves quite differently from all other methods and consistently causes the largest intervals. The side-by-side comparison nicely illustrates that the traditional CQR version is the only method which adjusts lower and upper bounds symmetrically by an equal amount. Since the ensemble method is a linear combination of the individual methods, its corrections are generally not symmetric and the produced intervals are usually centered around a different midpoint than the original forecasts.

Figure 1 corresponds to exactly one combination of `model`, `target_type`, `horizon` and `quantile` (pair). Thus, there exists exactly one optimization problem and one set of ensemble weights for this situation. The weights are identical for lower and upper bounds and across all time steps. The color legend displays the ensemble weights: In this case only the asymmetric CQR and the flexible (not symmetric) QSA methods contribute to the ensemble. As a simple weighted average with weights close to 0.5 the lower (upper) bounds

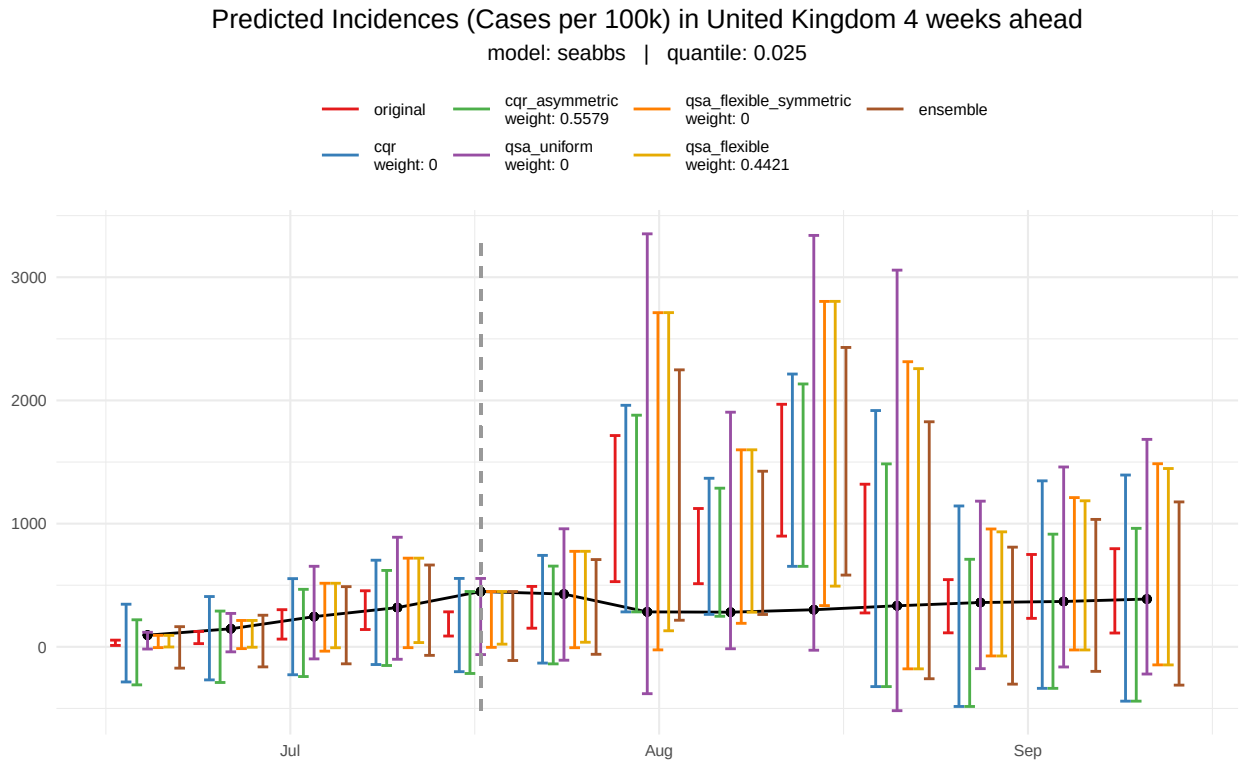

Figure 1: Comparison of Prediction Intervals for all Post-Processing Methods including the Ensemble

of the ensemble intervals are approximately halfway between the lower (upper) bounds of the `cqr_asymmetric` and `qsa_flexible` intervals.

Figure 1 shows that different methods can have significantly different effects, yet it does not provide any hints which method improves the Weighted Interval Score most. Table 1 collects the WIS for each method on the training and validation set, aggregated over all models, target types, horizons and quantiles and sorted by 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.

Table 1: 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

Table 2: Fraction of Feature Combinations where largest Ensemble Weight exceeds Threshold

> 0.5	> 0.9	> 0.99
0.97	0.73	0.55

Table 3: Number (Row 1) and Fraction (Row 2) of largest Ensemble Weights for each Method

cqr	**cqr_asymmetric**	**qsa_uniform**	**qsa_flexible_symmetric**	**qsa_flexible**
12.00	225.00	1	25.00	265.0
0.02	0.43	0	0.05	0.5

- The ensemble model is the clear winner: Combining information from multiple QSA and CQR methods works better on new data than any individual method on its own. 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 **??**, the more flexible CQR modification results in the lowest training but highest validation score for the small UK data set.
- In general, additional design restrictions such as identical weights in case of `qsa_uniform` and/or the 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 versions of both method frameworks indicate the best validation performance and yet, unsurprisingly, the worst training score.
- All methods improve the original forecasts by *expanding* the prediction intervals 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 either narrower or wider prediction intervals.

Table 1 convincingly demonstrates that the ensemble model leads to the best forecasts. Thus, we want to gain more insight how the ensemble predictions are created in this specific use case. Recall that the weights for each of the five building block methods are by construction nonnegative and (in case of convergence) sum to one. A different set of weights is computed for each of the $6 \cdot 2 \cdot 4 \cdot 11 = 528$ combinations of `model`, `target_type`, `horizon` and `quantile` (pairs). 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 (or two as in Figure 1) with a weight of 1 and all remaining methods are discarded with a weight of 0.

Table 2 provides a first insight into the weight distribution of the ensemble model. In 97% 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 there seems to be a single method that clearly outperforms its competition for most covariate combinations, we want to find out *which* of the five post-processing methods takes the winning trophy most often. Table 3 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 528 optimization problems where the methods contributed most to the ensemble model.

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

method whereas the uniform QSA method almost never has the largest impact on the ensemble. This finding is particularly interesting in comparison with Table 1: Since the ensemble is fitted on the *training* set, it distributes the weights according to the training set performance of each individual method. `cqr_asymmetric` and `qsa_flexible` indeed have the best training scores whereas `qsa_uniform` performs worst on the training set which exactly corresponds to the order of Table 3. With this connection in mind it seems 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.

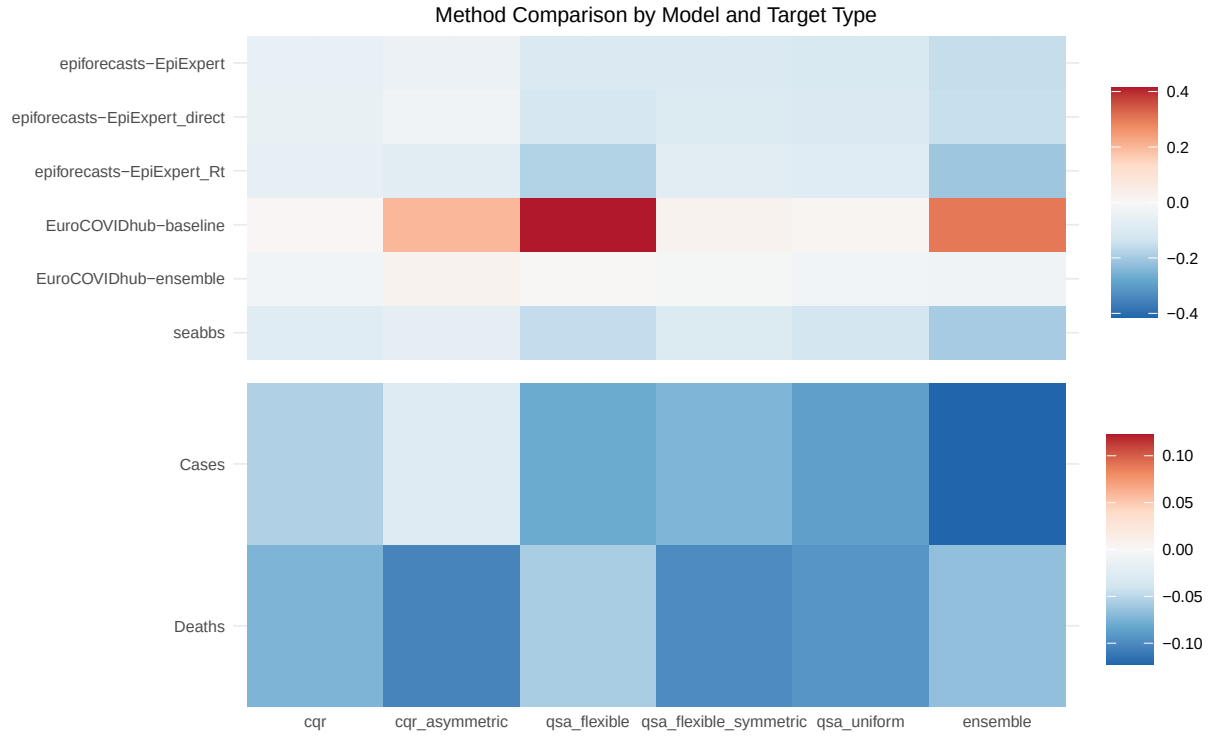

Figure 2: Method Comparison for each Forecasting Model and Target Type

Finally, we compare the different methods within each of the four major categories. Figure 2 shows relative improvements stratified either by forecasting model (top) or by target type (bottom). The picture is surprisingly diverse: Discriminating between the models first the `epiforecasts-EpiExpert_Rt` and the `seabbs` model benefit most overall whereas the `EuroCOVIDhub-baseline` model shows a negative effect for `cqr_symmetric` and `qsa_flexible`. In confirmation of Table 3 the latter two are exactly the methods with the strongest impact on the ensemble which, indeed, cannot improve the original forecasts for the `EuroCOVIDhub-baseline` model either. For all other models the flexible non-symmetric QSA version and the ensemble consistently outperform the remaining alternatives.

The ambiguity of the results is even more highlighted by the bottom plot. Based on the heatmap alone we cannot draw any conclusions if Cases or Deaths benefit more from post-processing in general, the effects simply vary too much across different methods. Comparing within the columns `qsa_flexible` and the ensemble are the only methods which indicate stronger benefits for Covid-19 Cases, all alternatives perform better for Deaths. Within the rows Figure 2 reveals the ensemble model as a good post-processing choice for Covid Cases, while `cqr_asymmetric` and `qsa_flexible_symmetric` show promising results for Covid Deaths.

When stratifying by forecast horizons and quantiles in Figure 3, the similarities between the methods are much stronger than in Figure 2. As we already know from previous chapters, all methods generally work better for larger horizons and quantiles in the tails. Aus usual the ensemble predictions are closest to `qsa_flexible` and lead to the largest performance gains for three and four week-ahead predictions while simultaneously

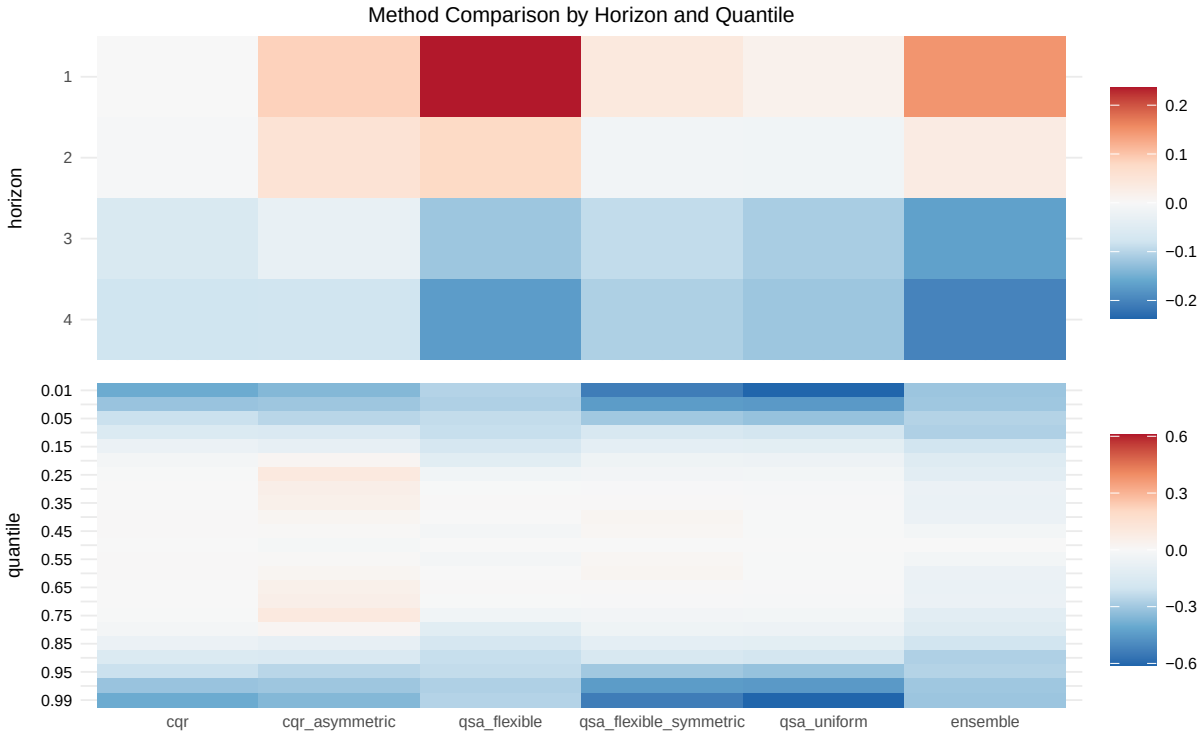

Figure 3: Method Comparison for each Forecast Horizon and Quantile Level

indicating the *worst* results for short-term one week ahead forecasts. The classical `cqr` method is the only method that reliably works for any forecast horizon, yet the effect sizes are quite small.

The bottom plot of Figure 3 provides a deeper understanding on the interaction between post-processing method and quantile level. `qsa_flexible_symmetric` and `qsa_uniform` have the largest positive effect for very small or very large quantiles with a significant drop in performance towards the center. In contrast, the `qsa_flexible` and the ensemble method cause more balanced improvements that are less sensitive to the location of the predictive distribution. The asymmetric CQR version is the only method with a (partially) negative impact and should thus be avoided for post-processing forecasts of centered quantiles.

In summary, Figure 2 and Figure 3 do not allow generic recommendations which post-processing method should be used. The ensemble model as the overall winner from Table 1 tends to have the largest effect sizes, both in positive and negative direction (except for the quantile plot). It thus offers a good choice if upsides and downsides contribute equally to the user's cost function. In case of risk aversion the traditional CQR method might be preferable, it is the only method which *exclusively* leads to performance improvements for all category levels across both figures.

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