Post Processing Covid-19 Forecasts

Matthias Herp, Joel Beck

supervised by

Nikos Bosse

Chair of Statistics, University of Göttingen

15.03.2022

Table of Contents

Introduction

1 Analysis Tools

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

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:

- `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.

[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

- `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**:

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

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].

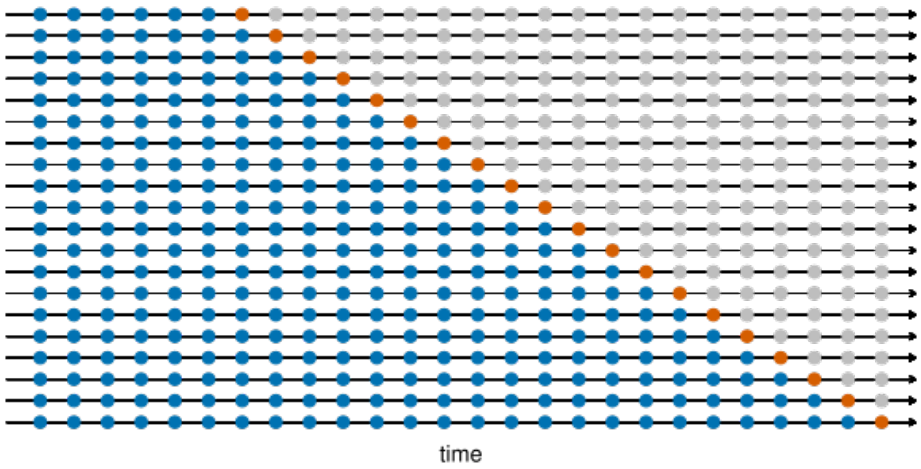

Figure 1: Time Series Cross Validation

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.

2. Model Fitting

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

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.

```
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:

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

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

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

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

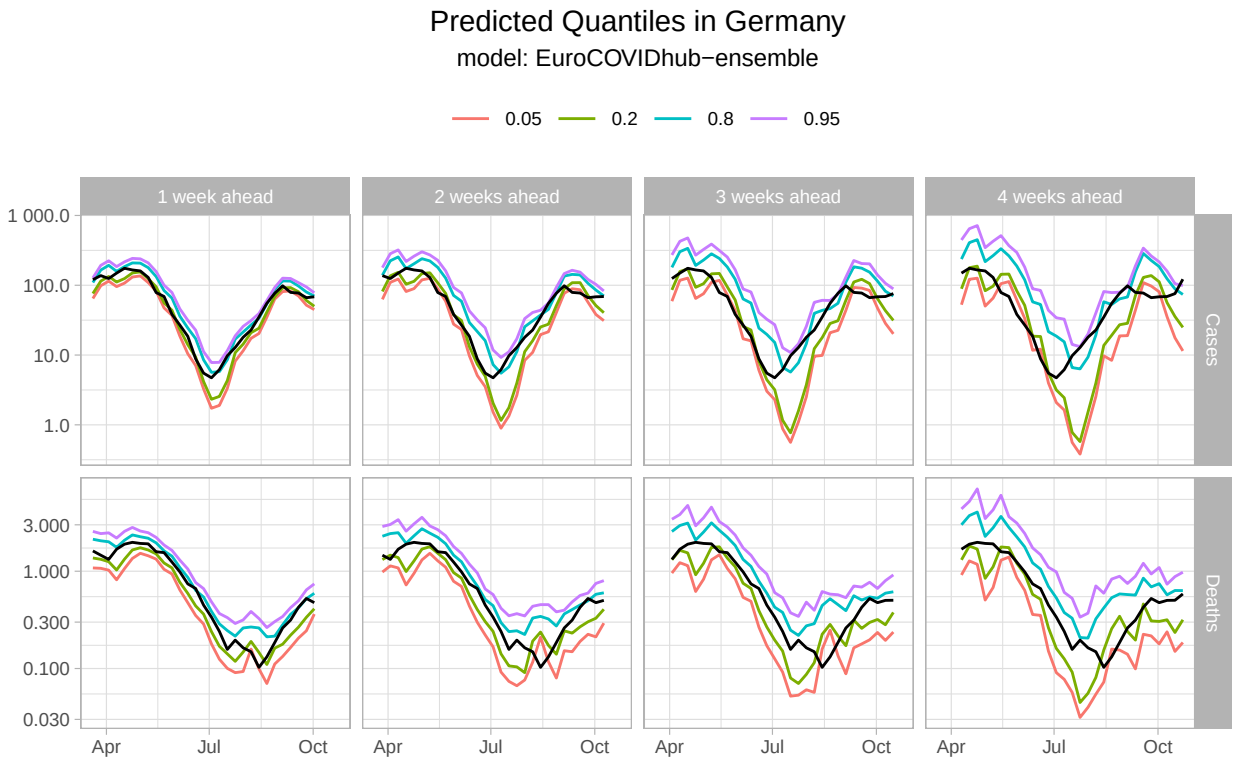

Figure 2: Original Quantile Predictions for Covid-19 Cases and Deaths in Germany 2021

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

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)
```

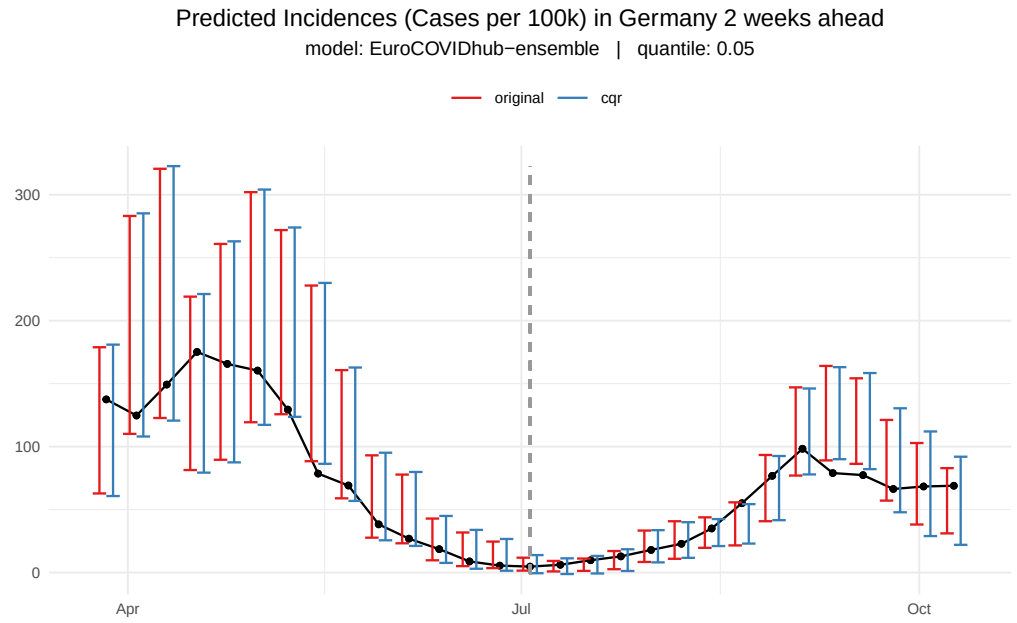

Figure 3: Original and CQR-adjusted Prediction Intervals for Covid-19 Cases in Germany

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

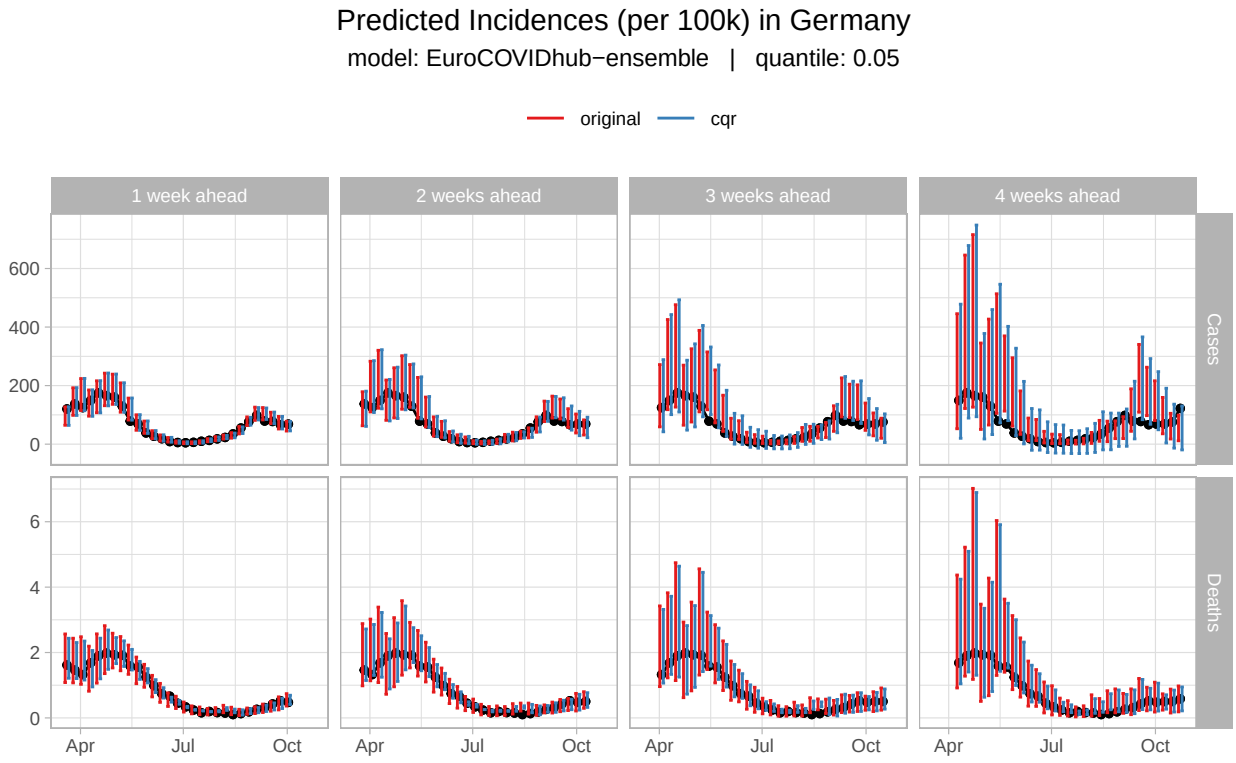

Figure 4: Original and CQR-adjusted Prediction Intervals for different Forecast Horizons

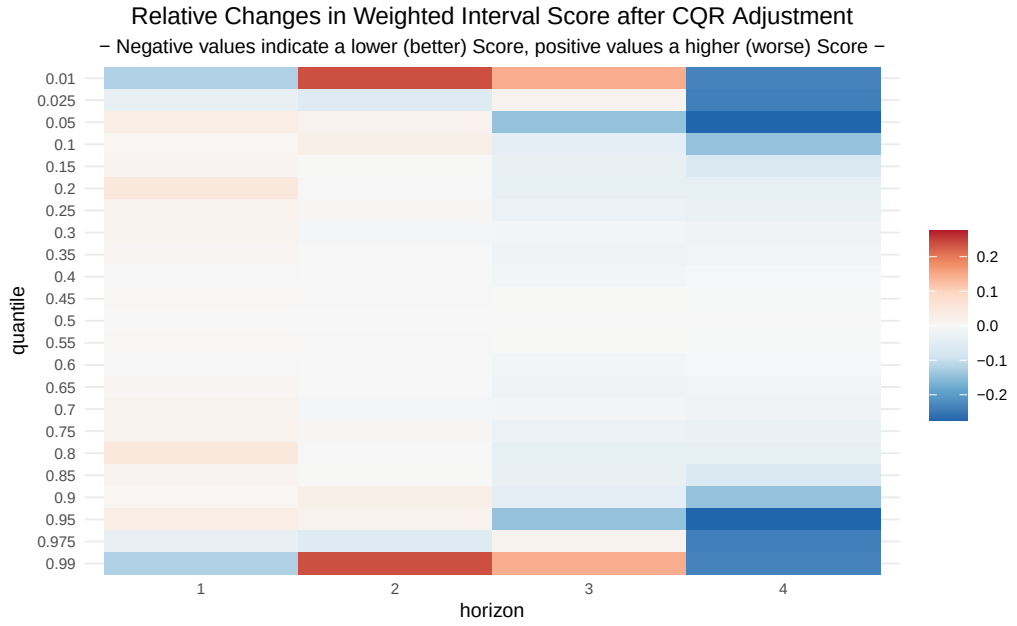

Figure 5: Relative Changes in the WIS through CQR for all Quantile-Horizon combinations

2 Conformalized Quantile Regression

This chapter introduces *Conformalized Quantile Regression (CQR)* as the first of two main post-processing procedures that we implemented in the `postforecasts` package.

Section 2.1 explains the original Conformalized Quantile Regression algorithm as proposed in Romano, Patterson, and Candès (2019). There, we highlight potential limitations of the traditional implementation that could potentially be fixed by more flexible modifications, which are discussed in Section 2.2 and Section 2.3.

2.1 Traditional CQR

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

- It should provide valid coverage in finite samples without making strong distributional assumptions
- The resulting intervals should be as short 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 the following Theorem:

Theorem 2.1. *If $(X_i, Y_i), i = 1, \dots, n+1$ 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 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.

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 $\mathcal{A}$, calculate lower and upper interval bounds on the training set:

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

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

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

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 scores E_i in the calibration set. For small sample sizes and small quantiles α the quantile above can be greater than 1 in which case it is simply set to 1 such that the maximum value of the score vector is selected.

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

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

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

2.2 Asymmetric CQR

2.2.1 Theory

As noted in Section 2.1 this section suggests a first extension to the original CQR algorithm. Instead of limiting ourselves to choose the *same* margin $Q_{1-\alpha}(E, I_2)$ for adjusting the original lower and upper quantile predictions, we allow for individual and, thus, generally different margins $Q_{1-\alpha,low}(E, I_2)$ and $Q_{1-\alpha,high}(E, I_2)$ 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 Results

To avoid repetitions with respect to Section 2.1 we only briefly mention where asymmetric and vanilla CQR have similar effects and rather focus on the differences.

Figure 6 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, in contrast 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 6 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 algorithm that was developed ealier in this chapter.

2.2.3 CQR downsides

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 select the data subset of Figure 6.

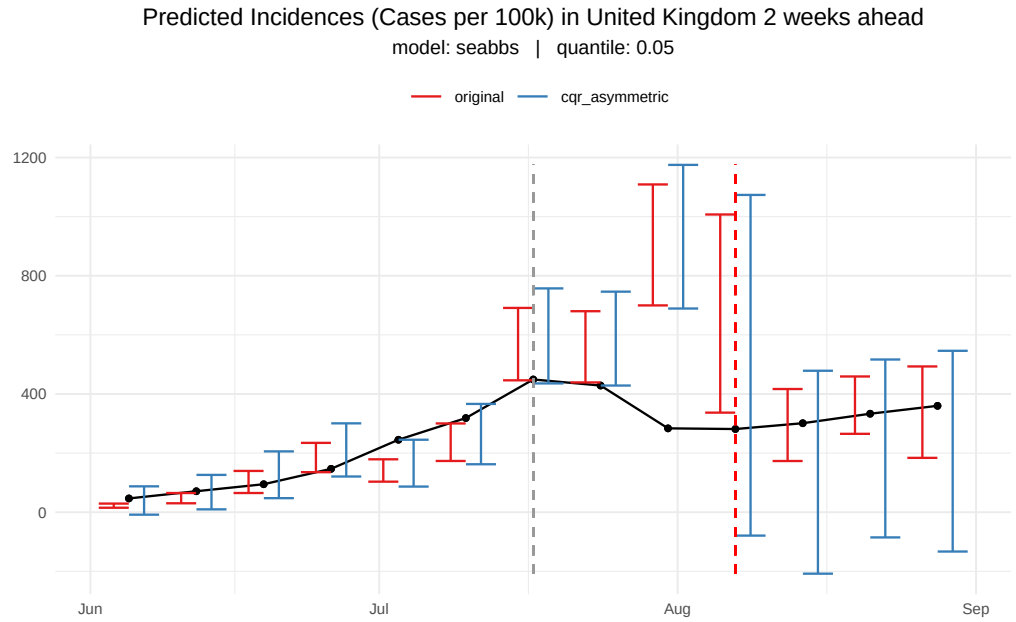

Figure 6: Illustration of CQR's slow reaction process

Consider the intervals one step prior to the red vertical 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 is calculated by the difference of true values and lower predicted 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 416.

Next, the (scalar) lower margin 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 thus the original lower quantile prediction at time point 10 minus the margin, i.e.

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

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

which coincides with Figure 6.

The procedure now continues by consecutively adding the next elements to the vector of true values and original quantile predictions. Since the differences in observed incidence of Cases and predicted lower bound

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 ine with each time step during the Time Series Cross Validation process, this quantile slowly declines. For instance, the margin which is responsible for adjusting forcasts 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.

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.

TODO: Add Global Results that show differences to traditional CQR

2.3 Multiplicative CQR

Theory

On top of the asymmetric CQR version introduced 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 $Q_{1-\alpha,low}(E, I_2)$ and $Q_{1-\alpha,high}(E, I_2)$ which are now *multiplied* with the existing forecasts. The post-processed prediction interval is then 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 in our application of Covid-19 Cases and Deaths all values are non-negative, we threshold the scores at zero such that $E_{i,low}$ equals 0 whenever $\hat{q}_{\alpha,low}(X_i) \leq 0$.

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 $Q_{1-\alpha,low}(E, I_2)$ 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 selected margin 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.

   We counteract this extreme sensitivity to outliers by *reducing the spread* inside 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 that reduce the interval width keep doing so but to a lesser extent than before and, analogously, prior values greater than one remain to be 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.

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.

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

   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.

Results

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

Before numerically evaluating the performance of *regularized* CQR, it is instructive to look at a visual comparison of the original and post-processed forecasts of all three CQR modifications for one specific feature combination, which is shown in Figure 7.

The effect of scaling the score vectors in step 1 of the regularization procedure and constraining lower and upper margins in the second step can be detected immediately: Similar to vanilla CQR, the corrected intervals are now centered around the same midpoint as the original forecasts. In strong contrast to the additive CQR versions, however, the issue of interval explosion has not only been diminished by downscaling the scores,

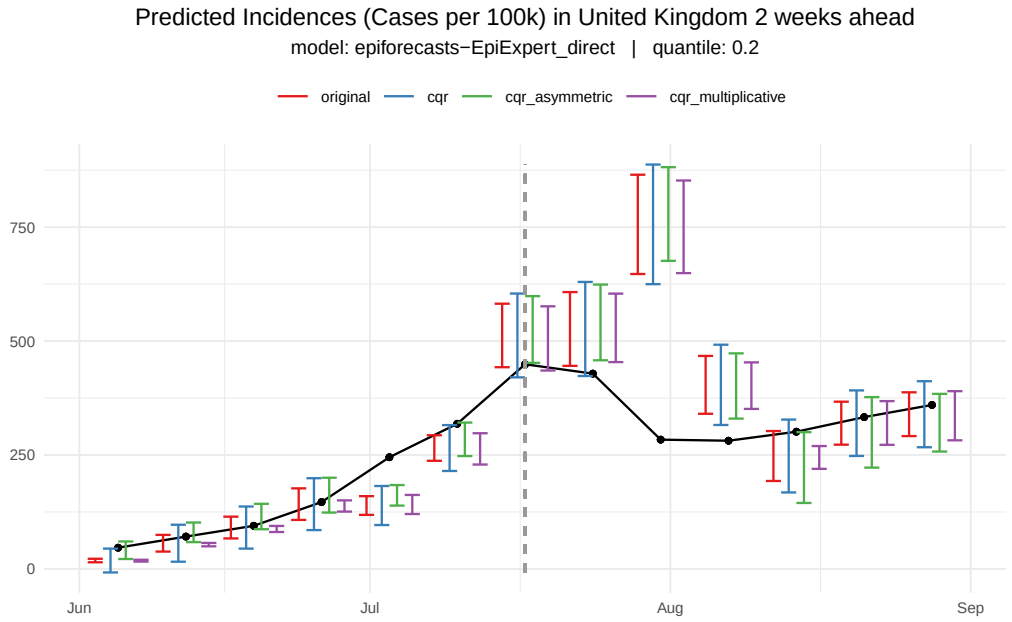

Figure 7: Comparison of CQR variations on the UK data set

Table 2: 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

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 2 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.

Recall that this behaviour is different from the unregularized version, which performed better than the original forecasts across almost all feature combinations. On the flipside, the performance on the validation set improved dramatically compared to the naive implementation, even though it does *not* lead to a score improvement in absolute terms as is shown in Table 3 separated by the forecasting `model`. Interestingly, multiplicative CQR indicates the strongest *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 3 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 is larger on average.

Table 4 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 for extreme quantiles and remains in a similar range to before towards the quantiles in the center of the

Table 3: 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 4: 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

predictive distribution. This behaviour is in line with the previously seen additive correction methods and underlines that Figure 7 is not representative for all feature combinations in the UK data set.

Overall, we must conclude that the original CQR algorithm as described in (Romano, Patterson, and Candès 2019) can *not* be modified towards multiplicative margins in any straightforward way. For this reason, we do not extend the analysis of multiplicative CQR to the European Forecast Hub data set and do not include it in the detailed 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 8.

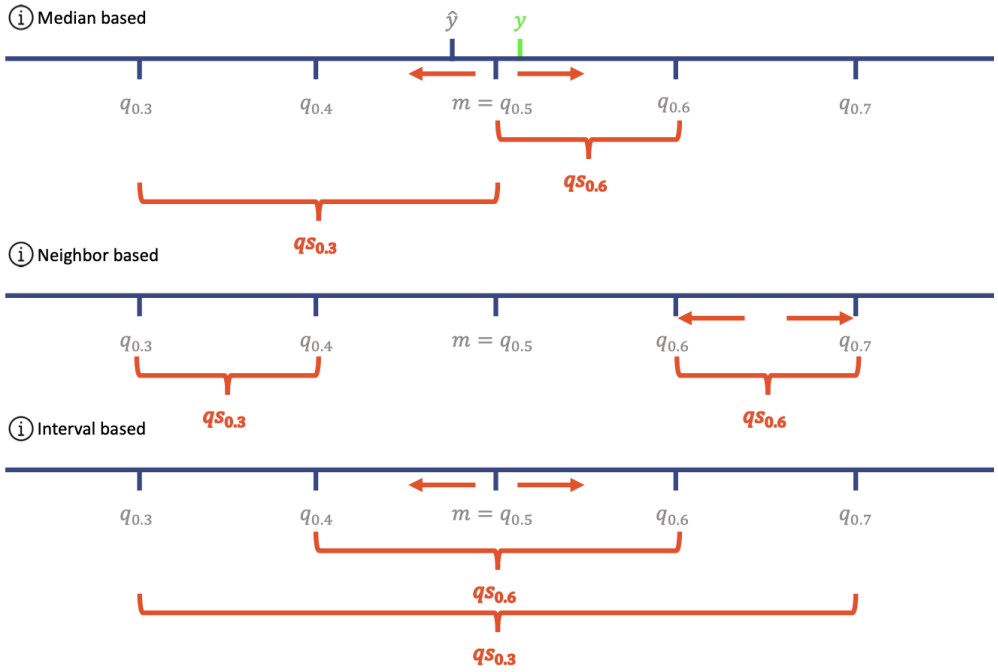

Figure 8: Quantile Spreads for different Basis

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

3.1 Theory

Using the median based definition, the next step is to determine how to optimally adjust the quantile spreads. As target function, QSA uses the Weighted Interval Score (reference). Equation (reference), 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.

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

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

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.

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

3.2 Optimization

Underneath the hood, `postforecasts` accesses the `optim` function from the R package stats[8] package. From the in `optim` 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` functions by setting the `optim_method` argument. For `L-BFGS-B`, the lower and upper bound box constraints can be set with the arguements `lower_bound_optim` and `upper_bound_optim`. Besides the use of `optim`, `postforecasts` also provides a line search optimization which is used by passing `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

[8] https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/optim

score. 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 potentially 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` this simply 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..

4 Method Comparison

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

Further, it will be interesting to observe the *types* of adjustments to the original forecasts: Some methods might improve the Weighted Interval Score by *extending* the interval width and thus increasing coverage whereas others might yield a similar final score by *shrinking* the prediction intervals leading to a higher precision. One can imagine even more 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 a model with a strong performance for one particular covariate set while the same model might perform worse for a different covariate set and, thus, make little contributions to the ensemble in that case.

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

4.1 Ensemble Model

There exist various options how to combine multiple building blocks into one ensemble. We chose an approach that can be efficiently computed by well-understood algorithms on the one hand and is highly interpretable on the other hand. Each quantile prediction of our ensemble model is a *convex combination* of the individual 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, $\mathbf{l}_1, \dots, \mathbf{l}_k \in \mathbb{R}^n$ 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 (Section 1.1.3) as our objective function that we minimize subject to linear constraints. The optimization step is implemented with the nloptr[9] package (Ypma and Johnson 2022), which describes itself as "an R interface to NLopt, a free/open-source library for nonlinear optimization".

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

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 covariate combinations as well as both the UK Forecasting Challenge data and the European Forecast Hub data.

Except for some minor modifications for computational efficiency, the results that constitute the starting point of the analysis in this chapter can be generated with the following commands, where `df` represents either the UK or the European Forecast Hub data set:

```
library(postforecasts)

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

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.

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