1 Analysis Tools

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

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

1.1 Data

1.2 The `postforecasts` Package

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

1.2.1 Overview

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

1. Exploratory

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

2. Model Fitting

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

   All list elements can be analyzed separately or collectively by stacking them into one large data frame with the `collect_predictions()` function. The combined data frame is designed to work well with analysis functions that are provided by the `scoringutils` package. Finally, an ensemble model of all selected methods can be appended which will be explained in chapter **??**.

3. Evaluation

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

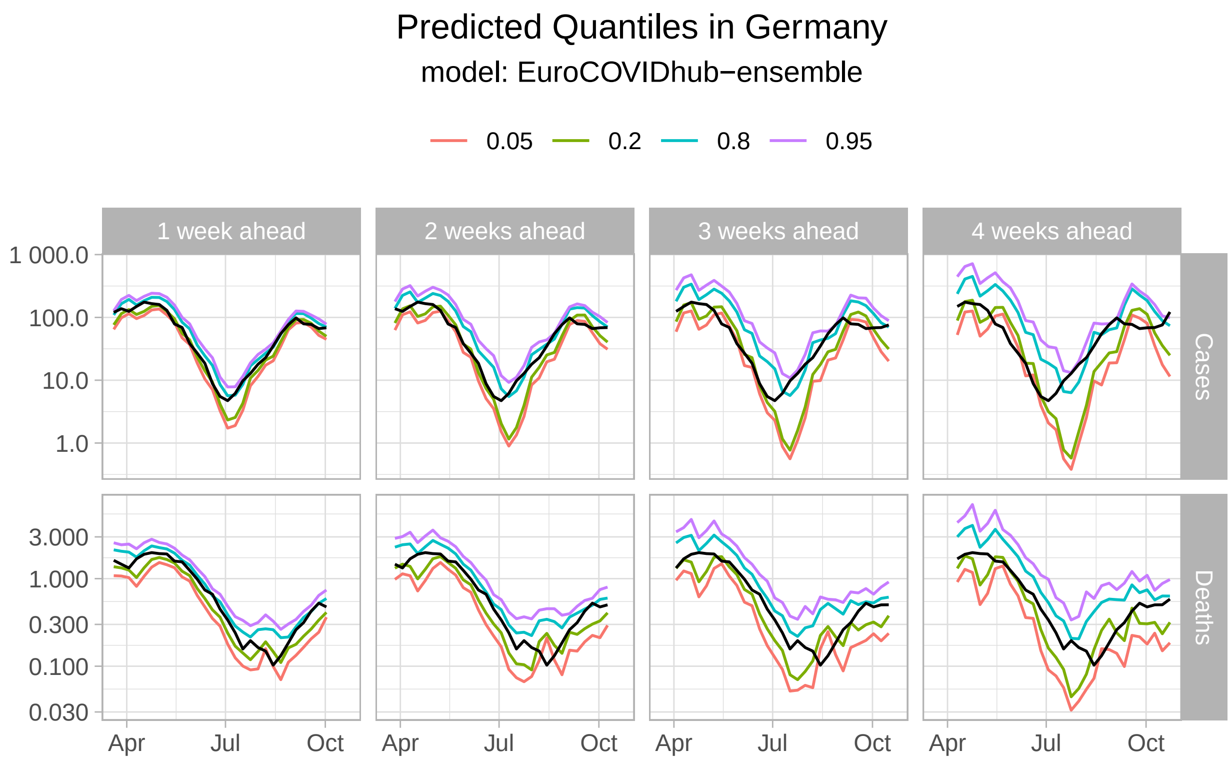
The following section demonstrates the complete workflow described above to give an impression how all these functions interact.

1.2.2 Workflow

We use the Covid19 data for Germany in 2021 that is provided by the European Forecast Hub.

The following plot illustrates the 5%, 20% 80% and 95% quantile predictions of the EuroCOVIDhub-ensemble during the summer months of 2021 in Germany.

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



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

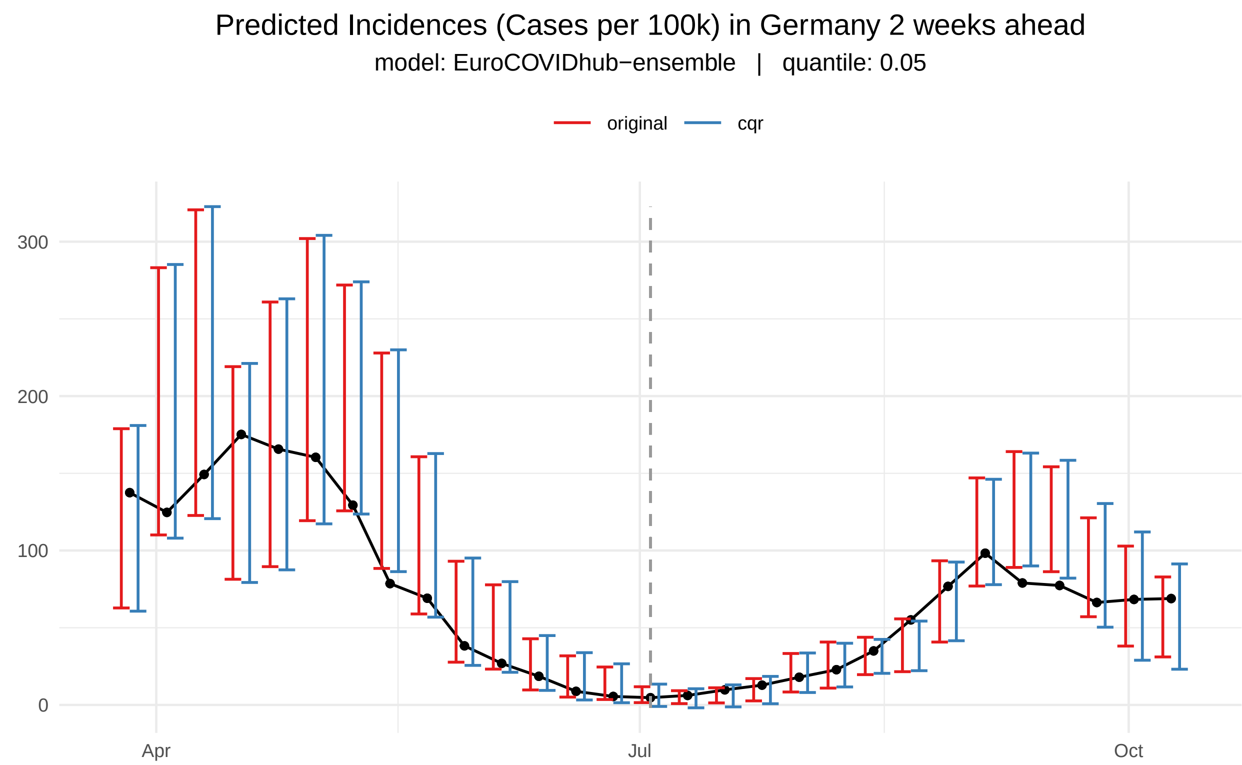
```
df_updated <- update_predictions(
  hub_germany,
  methods = "cqr", models = "EuroCOVIDhub-ensemble", cv_init_training = 0.5
)
df_combined <- collect_predictions(df_updated)
```

```
df_combined |>
  extract_validation_set() |>
  scoringutils::score() |>
  scoringutils::summarise_scores(by = c("method", "target_type")) |>
  dplyr::select(method:dispersion) |>
  dplyr::arrange(target_type)
## # A tibble: 4 x 4
##   method   target_type interval_score dispersion
##   <chr>    <chr>                <dbl>      <dbl>
## 1 cqr      Cases                 13.4       5.05
## 2 original Cases                 13.8       3.81
```

```
## 3 cqr      Deaths                0.0520     0.0138
## 4 original Deaths                0.0510     0.0253
```

On the validation set CQR improved the Weighted Interval Score for Covid Cases, whereas the predictive performance for Covid Deaths dropped slightly. Note that CQR increased the *dispersion* of the predictions for Cases significantly. We can visualize these wider intervals for specific covariate combinations:

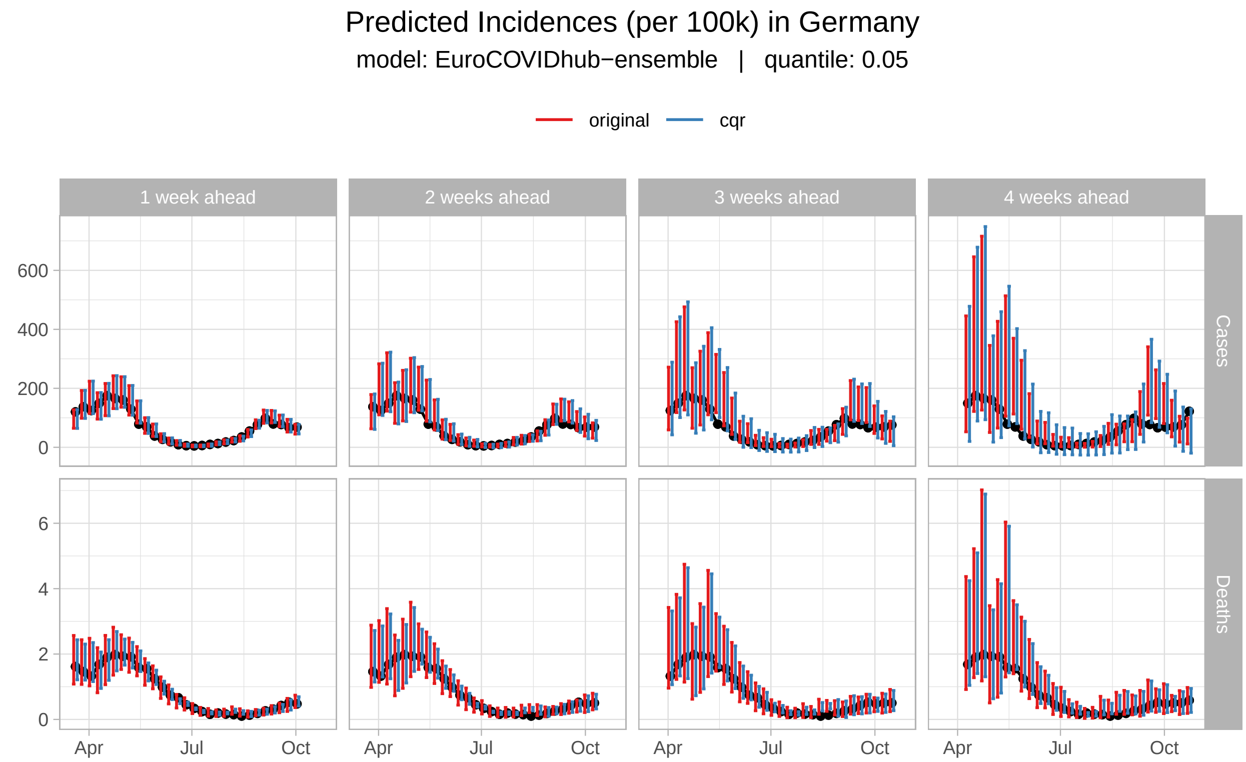
```
plot_intervals(df_combined, target_type = "Cases", horizon = 2, quantile = 0.05)
```



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

Recall that uncertainty increases with larger horizons. Similarly, CQR adjustments also increase in size for forecasts that are submitted further in advance:

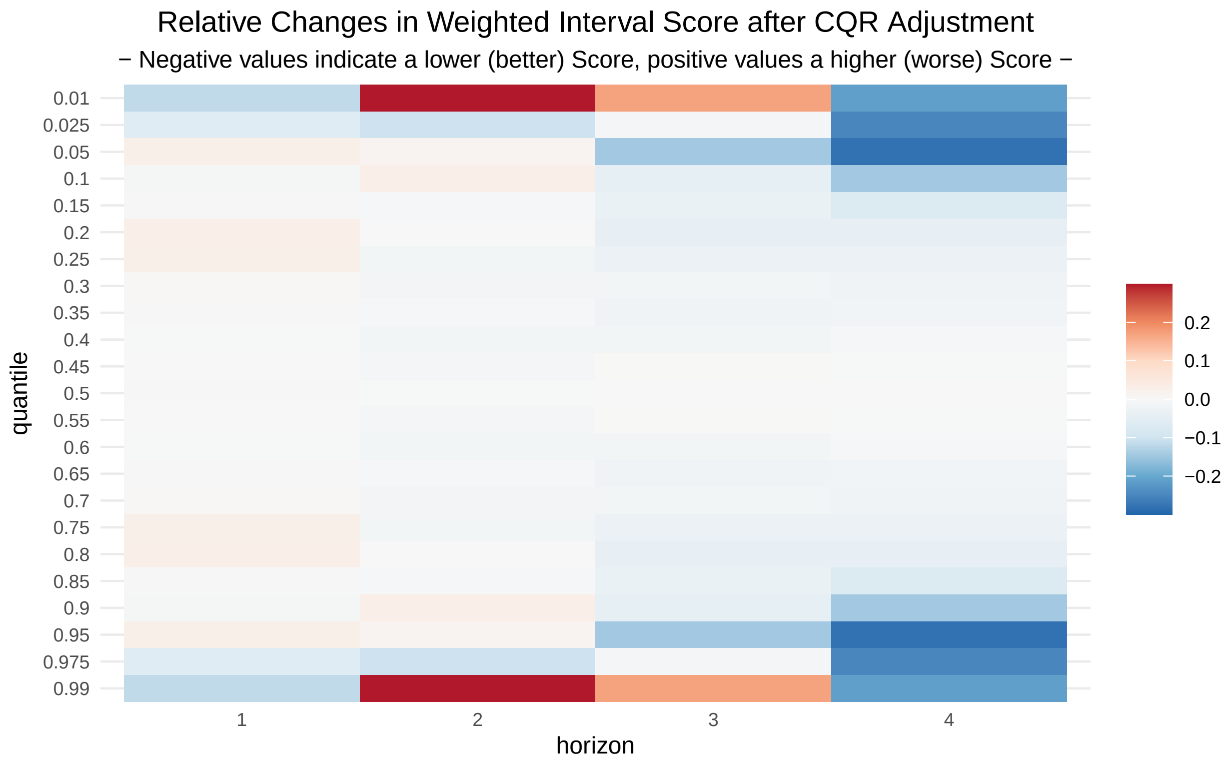
```
plot_intervals_grid(df_combined, facet_by = "horizon", quantiles = 0.05)
```



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

Besides the target type (Cases or Deaths), it is also useful to compare CQR effects across forecast horizons or quantiles. Quite intuitively, CQR generally has a stronger *relative* benefit for large time horizons and extreme quantiles, where the original forecaster faced a greater uncertainty. 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)
```


Relative Changes in Weighted Interval Score after CQR Adjustment
− Negative values indicate a lower (better) Score, positive values a higher (worse) Score −
quantile
0.01
0.025
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.55
0.6
0.65
0.7
0.75
0.8
0.85
0.9
0.95
0.975
0.99
1
2
3
4
horizon
0.2
0.1
0.0
−0.1
−0.2