Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Evaluating Covid-19 Short-Term Forecasts using scoringutils in R

Nikos I. Bosse
London School of Hygiene and Tropical Medicine

Second Author
Plus Affiliation

Abstract

Forecasts play an important role in a variety of fields. Their role in informing public policy has attracted increased attention from the general public with the emergence of the Covid-19 pandemic. Much theoretical work has been done on the development of proper scoring rules and other scoring metrics that can help evaluate these forecasts. Making these theoretical insights available to end users, however, has been less of a focus. In this paper we introduce **scoringutils**, an R package that facilitates automated forecast evaluation. It gives the user access to a wide range of scoring metrics for various types of forecasts as well as a variety of ways to visualise the evaluation. We give an overview of the evaluation process and the metrics implemented in **scoringutils** and show an example evaluation of forecasts for COVID-19 cases and deaths submitted to the European Forecast Hub between May und September 2021.

Keywords: JSS, style guide, comma-separated, not capitalized, R.

1. Introduction

Good forecasts are of great interest to decision makers in various fields like finance (Timmermann) [CITATION], weather predictions [CITATION] or infectious disease modeling (Funk et al. a) [CITATIONS]. An integral part of assessing and improving their usefulness is forecast evaluation. For decades, researchers have developed and refined an arsenal of techniques not only to forecast, but also to evaluate these forecasts (see e.g. Bracher et al., Funk et al. (b), Gneiting, Balabdaoui, and Raftery, and Gneiting and Raftery). Yet even with this rich body of research available, implementing a complete forecast evaluation in R is not trivial.

Some packages exist that bundle different scoring metrics together, but none offer the user a standalone solution to forecast evaluation. The **scoringRules** package (Jordan, Krüger, and Lerch 2019) offers a very extensive collection of proper scoring rules. Its exclusive focus on proper scoring rules and the fact that all functions are implemented using vectors and matrices instead of data.frames, however, make it more suitable for experienced users or as a building block in larger application. It also lacks features like pairwise comparisons between forecast models and plotting functionality that are invaluable in the evaluation process. Other packages like **Metrics** [https://cran.r-project.org/web/packages/Metrics/Metrics.pdf] **MLmetrics** [https://cran.r-project.org/web/packages/MLmetrics/MLmetrics.pdf] are geared towards machine learning problems and don't implement the set of metrics and scoring rules desired for forecast evaluation. The **scoringutils** package aims to bring forth a standardised and tested toolkit. It offers convenient automated forecast evaluation in a data.table format, but also provides experienced users with a set of reliable lower-level scoring metrics they can build upon in other applications. In addition it implements a wide range of flexible plots that are able to cover most day-to-day use cases.

This paper provides an overview of the fundamental ideas behind forecast evaluation, gives a detailed explanation of the evaluation metrics in **scoringutils** and discusses what to think about when applying them in practice. It then presents a case study based on the evaluation of Covid-19 related short-term forecasts in the UK (Funk et al. a).

1.1. Forecast types and forecast formats

In its most general sense, a forecast is the forecaster's stated belief about the future (Gneiting and Raftery) that can come in many different forms. Quantitative forecasts are either point forecasts or probabilistic in nature and can make statements about continuous, discrete or binary outcome variables. Point forecasts only give one single number for the most likely outcome, but do not quantify the forecaster's uncertainty. This limits their usefulness, as a very certain forecast may, for example, warrant a very different course of actions than does a very uncertain one. Probabilistic forecasts, in contrast, by definition provide a full predictive distribution. This makes them much more useful in any applied setting, as we learn about the forecaster's uncertainty and their belief about all aspects of the underlying data-generating distribution (including e.g. skewness or the width of its tails). Probabilistic forecasts are therefore the focus of this paper as well as the **scoringutils** package.

The predictive distribution of a probabilistic forecast can be represented in different ways with implications for the appropriate evaluation approach. For most forecasting problems, predictive distributions are not readily available in a closed form (and the **scoringutils** package therefore does not support scoring them directly). Instead, predictive distributions are often represented by a set of quantiles or predictive samples. Predictive samples require a lot of storage space and also come with a loss of precision that is especially pronounced in the tails of the predictive distribution, where quite a lot of samples are needed to accurately characterise the distribution. For that reason, often quantiles or central prediction intervals are reported instead [citation FORECAST HUBS]. For binary or multinomial prediction targets, common in many classification problems, a probabilistic forecasts is represented by the probability that an outcome will come true. Table 2 summarises the different forecast types and formats. While specific metrics may differ depending on the forecast type or format, the general forecasting paradigm Gneiting, Balabdaoui, and Raftery that guides the evaluation process is the same.

Forecast type	**Target type**	**Representation of the predictive distribution**
Point forecast	continuous discrete binary	one single number for the predicted outcome
Probabilistic forecast	continuous discrete	predictive samples quantiles closed analytical form
	binary	binary probabilities

Table 2: Forecasts can be distinguished by whether they are probabilistic in nature, or a point forecast only. Depending on the type of the target (discrete, continuous or binary) different representation of the predictive distribution are possible.

1.2. The forecasting paradigm

Any forecaster should aim to minimise the difference between the (cumulative) predictive distribution F and the unknown true data-generating distribution G (Gneiting, Balabdaoui, and Raftery). For an ideal forecast, we therefore have

$$F = G,$$

where F and G are both cumulative distribution functions. As we don't know the true data-generating distribution, we cannot assess the difference between the two distributions directly. Gneiting, Balabdaoui, and Raftery instead suggest to focus on two central aspects of the predictive distribution, calibration and sharpness (illustrated in Figure 1. Calibration refers to the statistical consistency (i.e. absence of systematic deviations) between the predictive distribution and the observations. Sharpness is a feature of the forecast only and describes how concentrated the predictive distribution is, i.e. how precise the forecasts are. The general forecasting paradigm states that we should maximise sharpness of the predictive distribution subject to calibration. A model that made very precise forecasts would at best be useless if the forecasts were wrong most of the time. On the other hand, a model may be well calibrated, but not sharp enough to be useful. Take a weather forecast that would assign 30 percent rain probability for every single day. It may be (marginally) calibrated when looking at the average rainfall over the course of a year, but it doesn't give much guidance on a day to day basis. Gneiting, Balabdaoui, and Raftery discuss different forms of calibration in more detail.

2. Scoring metrics implemented in scoringutils

Some of the metrics in **scoringuitls** focus only on sharpness or on calibration. Others, called proper scoring rules, combine both aspects into a single number. The former can be helpful to learn about specific model aspects and improve them, the latter are especially useful to assess and rank predictive performance of a forecaster. The following gives an introduction to how these metrics can be used to evaluate forecasts. Table **??** shows an overview of the metrics implemented in **scoringutils**. Table 5 in the Appendix gives a mathematical definition

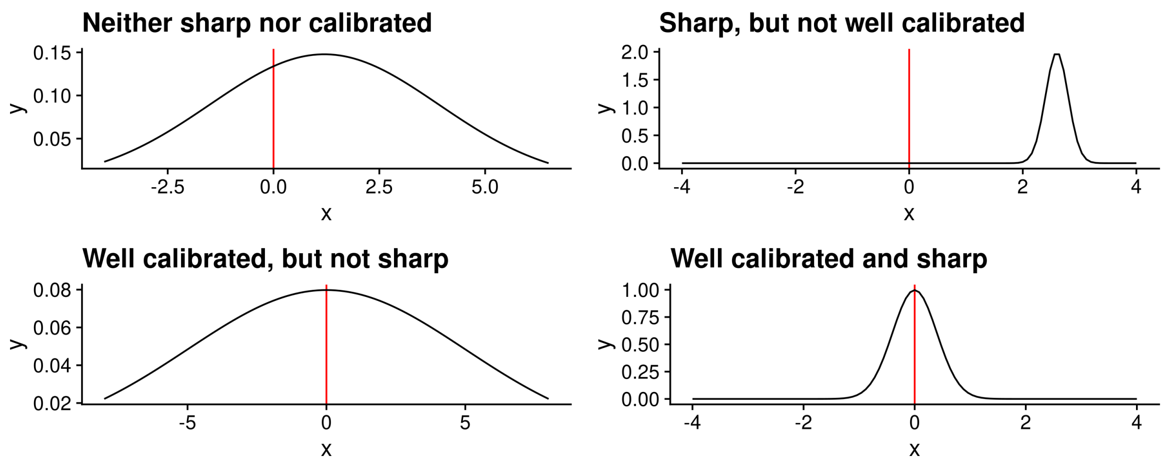
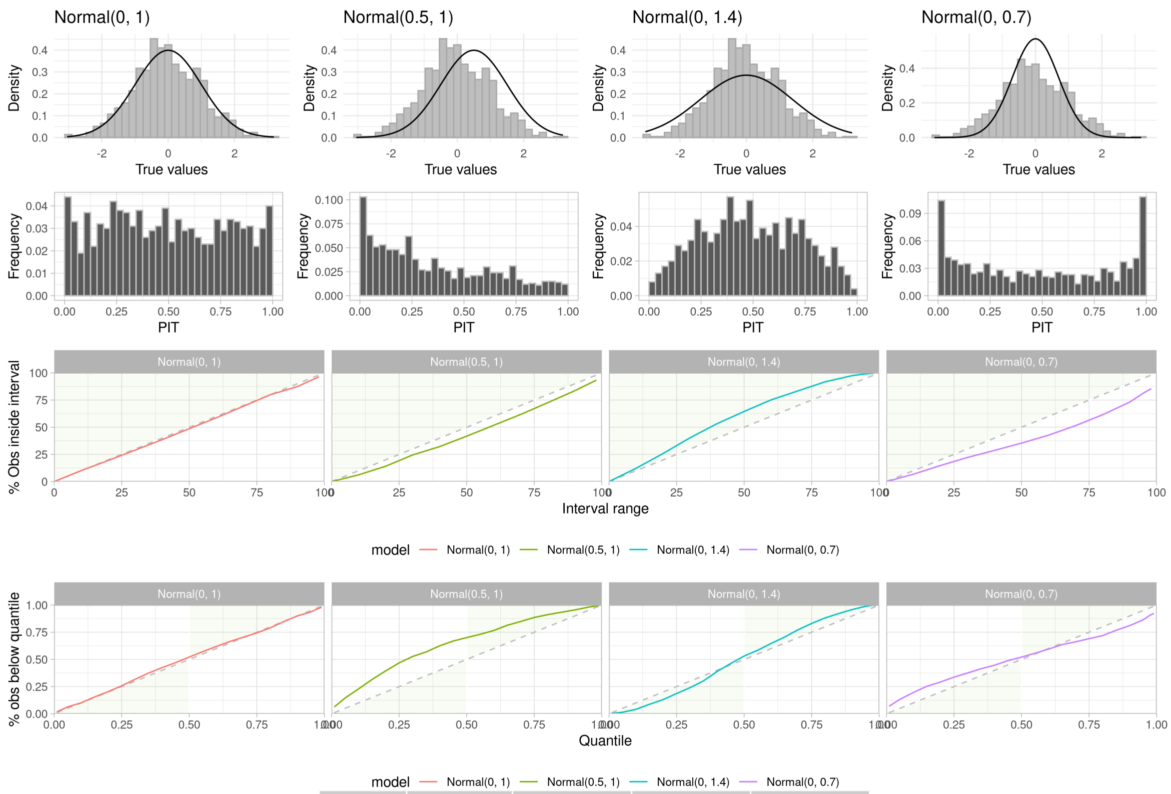


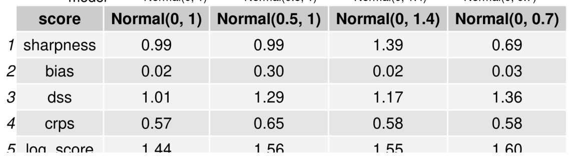
Figure 1: Schematic illustration of calibration and sharpness. True value are represented in red, the predictive distribution is shown in black

and thorough explanation of all the metrics. Figure 2 gives an applied example of different scoring metrics and visualisations for simple toy data.

Metric	Discrete	Continuous	Binary	Closed-form	Samples (approx.)	Quantiles	Properties
(Continuous) ranked probability score (CRPS)	✓	✓		✓	✓		proper scoring rule, global, stable handling of outliers
Log score (logS)		✓		✓	✓		proper scoring rule, log of predictive density evaluated at observed value, local, unstable for outliers
(Weighted) interval score (WIS)	✓	✓			THINK	✓	proper scoring rule, global, stable handling of outliers, converges to crps for an increasing numbre of equally spaced intervals
Dawid-Sebastiani score (DSS)	✓	✓		✓	✓		proper scoring rule, somewhat global, somewhat stable handling of outliers
Brier score (BS)			✓		✓		proper scoring rule
Interval coverage	✓	✓			THINK	✓	measure for calibration
Quantile coverage	✓	✓			THINK	✓	measure for calibration
Probability integral transform (PIT)	✓	✓		✓	✓	✓	assesses calibration
Sharpness	✓	✓		✓	✓	✓	measures forecast dispersions
Bias	✓	✓	✓???	✓	✓	✓	captures tendency to over-or underpredict (aspect of calibration)
Mean score ratio	∼	∼	∼	∼	∼	∼	compares performance of two models. Properties depend on the metric chosen for the comparison.
Relative skill	∼	∼	∼	∼	∼	∼	Ranks models based on pairwise comparisons. Properties depend on the metric chosen for the comparison.

Table 4: Summary table of scores available in scoringutils





	score	Normal(0, 1)	Normal(0.5, 1)	Normal(0, 1.4)	Normal(0, 0.7)
1	sharpness	0.99	0.99	1.39	0.69
2	bias	0.02	0.30	0.02	0.03
3	dss	1.01	1.29	1.17	1.36
4	crps	0.57	0.65	0.58	0.58
5	log_score	1.44	1.56	1.55	1.60

Figure 2: Calibration plots for different forecast distributions. I plan to add interval and quantile coverage plots, but these still have to be made.

2.1. Evaluating calibration and sharpness independently

Evaluating calibration and sharpness independently is helpful for model diagnostics. To that end **scoringutils** makes numerous metrics available that aim to capture different aspects of sharpness and calibration.

Assessing calibration

Calibration means consistency between forecasts and observed values, but there are various ways in which a forecast can systematically deviate from the observations (see Gneiting, Balabdaoui, and Raftery for a discussion of different forms of calibration MAYBE IT WOULD BE A GOOD IDEA TO INCLUDE THESE DIFFERENT FORMS OF CALIBRATION HERE). **scoringutils** allows the user to examine three different sub-aspects of calibration: bias, empirical coverage, and the probability integral transform (PIT).

Bias, i.e. systematic over- or underprediction, is a very common form of miscalibration which therefore deserves separate attention. The bias metric (with slightly different versions for the various forecast types and formats) captures a general tendency to over- and underpredict that is bound to be between minus one (underpredicton) and one (overprediction), where zero is ideal. It is derived by looking at how much of the probability mass of the predictive distribution is below or above the true observed value. For quantile forecasts we have second alternative approach available to assess over- and underprediction - by simply looking at the corresponding components of the weighted interval score. What is different between the over- and underprediction components and bias as described above is its sensitivity to outliers. The former are derived from absolute differences, while the latter is bound and rather captures a general tendency to be biased.

Another way to look at calibration (precisely: probabilistic calibration in Gneiting, Balabdaoui, and Raftery) is to compare the proportion of observed values covered by different parts of the predictive distribution with the nominal coverage implied by the CDF of the distribution. This is most easily understood in the context of quantile forecasts, but can easily be transferred to sample-based continuous and discrete forecasts as well. To assess empirical coverage at a certain interval range, we simply measure the proportion of true observed values that fall into corresponding range of the predictive distribution. If the 0.05, 0.25, 0.75, and 0.95 quantiles are given, then 50% of the true values should fall between the 0.25 and 0.75 quantiles and 90% should fall between the 0.05 and 0.95 quantiles. We can calculate and plot these values to inspect how well different parts of the forecast distribution are calibrated. To get an even more precise picture, we can also look at the percentage of true values below every single quantile of the predictive distribution. This allows to diagnose issues in the lower and upper tails of the prediction intervals separately. A similar way to visualise the same information is a PIT histogram. In order to conveniently assess deviations between the predictive distribution and the true data-generating distribution we can transform the observed values using the probability integral transformation (PIT) (Dawid) (see more details in Table 5). If both distributions are equal, the transformed values will follow a uniform distribution. A histogram of the transformed values can help to diagnose systematic differences between the predictions and the observed values. Figure 2 exemplifies the characteristic shape of certain systematic deviations of the predicitve distribution from the true data-generating distribution. In the PIT histograms, bias leads to a triangular shape, overdispersion results in a hump shaped form and underdispersion in a U-shape. ADD INTERPRETATION FOR QUANTILE AND INTERVAL COVERAGE PLOTS HERE.

Assessing sharpness

Sharpness is the ability to produce narrow forecasts. It does not depend on the actual observations and is a quality of the forecast only **??**. Sharpness is therefore only useful subject to calibration, as exemplified above in Figure **??**. We may be willing to trade off a little calibration for a lot more sharpness, but usually not much. For sample-based forecasts, **scoringutils** calculates sharpness as the normalised median absolute deviation about the median (MADN) [funkAssessingPerformanceRealtime2019] (for details see Table **??**). For quantile forecasts, we take the sharpness component of the WIS which corresponds to a weighted average of the individual interval widths.

WHAT WOULD BE REALLY COOL IS TO DETERMINE A WAY TO FIND THE OPTIMAL SHARPNESS OF A FORECAST. AS IS, I FEEL THIS PARAGRAPH IS SOMEWHAT USELESS.

2.2. Proper scoring rules

Proper scoring rules (Gneiting and Raftery) jointly assess sharpness and calibration and assign a single numeric value to a forecast. A scoring rule is proper if a perfect forecaster (the predictive distribution equals the data-generating distribution) receives the lowest score on average. This makes sure that a forecaster evaluated by a proper scoring rule is always incentivised to state their best estimate. As summarised in Table **??**, not all proper scoring rules are suitable for all forecast formats.

2.3. Proper scoring rules for sample-based forecasts (CRPS, logS and DSS)

For sample-based forecasts, the **scoringutils** provides the following proper scoring rules: the (continuous) ranked probability score (CRPS) [CITATION], the log score (logS) [CITATION], and the Dawid-Sebastiani-score (DSS) [CITATION] (formal definitions are given in Table 5). The corresponding functions are imported from the **scoringRules** package and exposed to the user through a slightly adapted interface. Other, non-sample-based variants of the CRPS, logS and DSS are available directly in the **scoringRules** package, but not in **scoringutils**. All three scores are in principle applicable to continuous as well as discrete forecasts. The **scoringRules** implementation of the log score, however, requires a kernel density estimation that may be inappropriate for discrete values (see also Table **??**). The logS is therefore not computed for discrete predictions in **scoringutils**.

When scoring forecasts in a sample-based format, the choice is usually between the logS and the CRPS. The DSS is much less commonly used. It is easier to compute, but apart from that does not have immediate advantages over the former two. CRPS and logS differ in three important aspects: sensitivity to distance Winkler et al., sensitivity to outlier predictions, and sensitivity to the order of magnitude of the forecasted quantity.

The CRPS is a so-called global scoring rule, which means that the entire predictive distribution is taken into account when scoring a single forecast. The log score, on the other hand is local. The resulting score does not depend on the overall distance between the observed value and the distribution, but only on the probability density assigned to the actual outcome. Imagine two forecasters, A and B, who forecast the number of goals scored by a team in a football match. If both forecasters assigned the same probability to the true outcome (e.g. 2 points), but A assigned higher probability to extreme outcomes far away from the actually observed outcome (e.g. A stated 10 goals may be quite likely, while B's predictions are concentrated around 1, 2 or 3 points), then A will receive a worse score than B. The log score, in contrast, is a local scoring rule that only scores the probability assigned to the actual outcome and ignores the rest of the predictive distribution. Judged by the log score, A and B would receive exactly the same score. Sensitivity to distance (taking the entire predictive distribution into account) may be an advantage in most settings that involve decision making. Forecaster A's prediction that assigns high probability to results far away from the observed value is arguably less useful than B's forecast that assigns higher probability to values closer to it (the probability assigned to the actual outcome being equal for both forecasts). The log score is only implicitly sensitive to distance if we assume that values close to the observed value are actually more likely to occur. The logS may, however, be more appropriate for inferential purposes (see Winkler et al.) and is commonly used in Bayesian statistics [CITATION].

A second important difference is how forecasts are treated that deviate strongly from the observed outcome. The CRPS can be thought of as a generalisation of the absolute error to a predictive distribution. It therefore scales linearly with the distance between forecast distribution and true value. The log score, however, is the log of the predictive density evaluated at the observed value. It can quickly go to negative infinity if the probability assigned to the observed outcome is close to zero. The CRPS is therefore considered more stable than the log score. The behaviour of the DSS is in between the two. Whether or not harsh punishment of bad predictions is desirable or not depends of course on the setting. Bracher et al. exemplify that in practice there may indeed be substantial differences between how the CRPS and log score judge the same forecast.

As the CRPS is a generalisation of the absolute value, overall scores depend on the order of magnitude of the quantity we try to forecast. This makes it harder to compare forecasts for very different targets, or assess average performance if the quantity of interest varies substantially over time. Average scores are dominated by forecasts for targets with high absolute numbers. This may be desirable,if we care most about forecasts in situations where numbers are high, but usually it is not. LogS and DSS are more robust against this effect. Another way to address this issue by using pairwise comparisons will be introduced later.

2.4. Proper scoring rules for non-sample-based forecasts (WIS and BS)

For forecasts in an interval or quantile format, **scoringutils** offers the weighted interval score (WIS) (Bracher et al.). The WIS has very similar properties to the CRPS and can be thought of as a quantile-based approximation. For an increasing number of equally-spaced prediction intervals the WIS converges to the CRPS. One additional benefit of the WIS is that it can easily be decomposed into three additive components: an uncertainty penalty (called dispersion or sharpness) for the width of a prediction interval and penalties for over- and underprediction (if a value falls outside of a prediction interval). This can be very helpful in diagnosing model problems. It may even be useful to convert samples into quantiles and use the WIS instead of the CRPS to make use of this decomposition for the purpose of model diagnostics.

Binary forecasts can be scored using the Brier score (BS) [CITATION], which corresponds to the squared difference between the given probability and the outcome (either 0 or 1).

2.5. Pairwise comparisons

If what we care about is to determine which model performs best, pairwise comparisons between models are a suitable approach [CITATION CRAMER et al.]. In turn, each pair of models is evaluated based on the targets that both models have predicted. The mean score by one model is divided by the mean score of the other model to obtain the mean score ratio (see Table **??**, a measure of relative performance. To obtain an overall relative skill score for a model, we take the geomatric mean of all mean score ratios that involve that model (omitting comparisons where there is no overlapping set of forecasts). This gives us an indicator of performance relative to all other models. The orientation depends on the score used. For the proper scoring rules described above, smaller is better and a relative skill score smaller than 1 indicates that a model is performing better than the average model. We can obtain a scaled relative skill score by dividing a model's relative skill by the relative skill of a baseline model. A scaled relative skill smaller than one then means that the model in question performed better than the baseline.

It is in principle possible to obtain p-values that help determine whether two models perform significantly differently. **scoringutils** allows to compute these using eitab:score-tablether the Wilcoxon rank sum test or a permutation test. In practice, this is slightly complicated by the fact that both tests assume independent observations. In reality, however, forecasts by a model may be correlated across time or another dimension (e.g. if a forecaster has a bad day, they will likely perform badly across different targets for a given forecast date). P-values may therefore be too quick to suggest significant differences where there aren't any. One way to mitigate this is to aggregate observations over a category where one suspects correlation. A test that is performed on aggregate scores will likely be more conservative.

3. Evaluating UK short-term forecasts

The following section shows an example evaluation of short-term predictions for COVID-19 cases and deaths submitted to the European Forecast Hub [CITATION]. The full oficial hub evaluations, which also use **scoringutils**, can be seen at https://covid19forecasthub.eu/. The European Forecast Hub each week collates, aggregates and evaluates one to four week ahead predictions of different COVID-19 related targets submitted by different research groups. Forecasts are submitted in a quantile-based format with a set of 22 quantiles plus the median $(0.01, 0.025, 0.05, ..., 0.5, ...0.95, 0.975, 0.99)$. The following example evaluation is based on a subset of forecasts from the European Forecast Hub which includes one to three week ahead forecasts made between May and September 2021 for COVID-19 cases and deaths from four different models.

This example data follows a quantile-based format, but the evaluation process would work analogously for forecasts in a different format. We will point out differences to scoring different forecast formats where appropriate. The example data set (as well as versions of it in different formats) is also included in the **scoringutils** package.

The evaluation process looks as follows: First we load, prepare and visualise (some of) the data. Then we obtain forecast scores and a model ranking based on pairwise-comparisons, followed by a more detailed analysis of calibration and sharpness.

3.1. Checking the data

```r
> # load packages
> library(scoringutils)
> library(dplyr)
> library(data.table)
> library(kableExtra)
>
> # load forecasts
> data(example_quantile)
```

All higher level functions that allow for convenient evaluation of forecasts expect a `data.frame` or similar. Lower-level functions (discussed in a later section) accept inputs as vectors and matrices.

We omit `NA` values as the data set also contains entries for which we have an observed value, but no forecast.

```
> example_quantile |>
+     na.omit() |>
+     glimpse()

## Rows: 20,401
## Columns: 10
## $ location        <chr> "DE", "DE", "DE", "DE", "DE", "DE", "DE", "DE", "DE", ~
## $ target_end_date <date> 2021-05-08, 2021-05-08, 2021-05-08, 2021-05-08, 2021-~
## $ target_type     <chr> "Cases", "Cases", "Cases", "Cases", "Cases", "Cases", ~
## $ true_value      <dbl> 106987, 106987, 106987, 106987, 106987, 106987, 106987~
## $ location_name   <chr> "Germany", "Germany", "Germany", "Germany", "Germany",~
## $ forecast_date   <chr> "2021-05-03", "2021-05-03", "2021-05-03", "2021-05-03"~
## $ quantile        <dbl> 0.010, 0.025, 0.050, 0.100, 0.150, 0.200, 0.250, 0.300~
## $ prediction      <int> 82466, 86669, 90285, 95341, 99171, 102990, 105962, 108~
## $ model           <chr> "EuroCOVIDhub-ensemble", "EuroCOVIDhub-ensemble", "Eur~
## $ horizon         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
```

For forecasts in a quantile-based format it is necessary to have columns called `true_value`, `prediction` and `quantile`. It is recommended (although not necessary except in order to do pairwise comparisons) to have a column called "model" with an identifier for the forecaster. Table **??** shows the columns expected for different input formats.

```
> requirements <- data.table(Format = c("quantile-based", "sample-based",
+     "binary", "pairwise-comparisons"), `Required columns` = c("'true_value', 'prediction
+     "'true_value', 'prediction', 'sample'", "'true_value', 'prediction'",
+     "additionally a column 'model'"))
>
> requirements |>
+     kable(format = "latex")
```

Format	Required columns
quantile-based	'true_value', 'prediction', 'quantile'
sample-based	'true_value', 'prediction', 'sample'
binary	'true_value', 'prediction'
pairwise-comparisons	additionally a column 'model'

We can check whether the data conforms to the requirements by running `check_forecasts()`.

```
> check_forecasts(example_quantile)

## $target_type
## [1] "integer"
##
## $prediction_type
## [1] "quantile"
##
## $forecast_unit
## [1] "location"        "target_end_date" "target_type"     "location_name"
## [5] "forecast_date"   "model"           "horizon"
##
## $unique_values
##                    model location target_end_date target_type true_value
## 1: EuroCOVIDhub-ensemble        4              12           2         96
## 2: EuroCOVIDhub-baseline        4              12           2         96
## 3:  epiforecasts-EpiNow2        4              12           2         95
## 4:       UMass-MechBayes        4              12           1         48
##    location_name forecast_date quantile prediction horizon
## 1:             4            11       23       3969       3
## 2:             4            11       23       3733       3
## 3:             4            11       23       3903       3
## 4:             4            11       23       1058       3
##
## $warnings
## [1] "Some values for `prediction` are NA in the data provided"
##
##
## Based on your input, scoringutils thinks:
## Forecasts are for a `integer` target using a `quantile` prediction format.
## The unit of a single forecast is defined by `location`, `target_end_date`, `target_type
## $unique_values shows how many unique values there are per column per model (across the
##
## You should be aware of the following warnings:
## Some values for `prediction` are NA in the data provided
```

This returns a list with different entries giving information about what **scoringutils** infers from the data. `target_type` `prediction_type` refer to the forecast format. `forecast_unit` contains a vector of the columns which **scoringutils** thinks denote the unit of a single forecast. This means that in this instance a single forecast (with a set of 23 quantiles) can uniquely be identified by the values in the columns "location", "target_end_date", "target_type", "location_name", "forecast_date", "model", "horizon". In this example, having "location" as well as "location_name" included does not make a difference, as they contain duplicated information. In general, however, it is strongly advised to remove all unnecessary columns that do not help identify a single forecast. `unique_values` gives an overview of the number of unique values per column across the entire data set, providing a first hint as to whether the forecast set is complete. `warnings` shows potential warnings about the data. In this example, **scoringutils** warns that there are observed values present for which there is no corresponding forecast. These warnings can often be ignored, but may provide important information. If there are errors that cannot be ignored, a list entry `errors` will appear.

It is helpful to start the evaluation process by visualising data availability, as missing forecasts can impact the evaluation if missingness is not random, but instead correlates with performance. The function `show_avail_forecasts()` returns a heatmap with the number of available forecasts. By default, the function treats a set of different quantiles or samples as one forecast. However, the user can specify manually which elements to treat as one forecast and which categories to sum over to count the number of available forecasts. The forecasts

```
> show_avail_forecasts(data = example_quantile, x = "target_end_date",
+     show_numbers = FALSE, legend_position = "bottom", facet_formula = ~target_type)
```

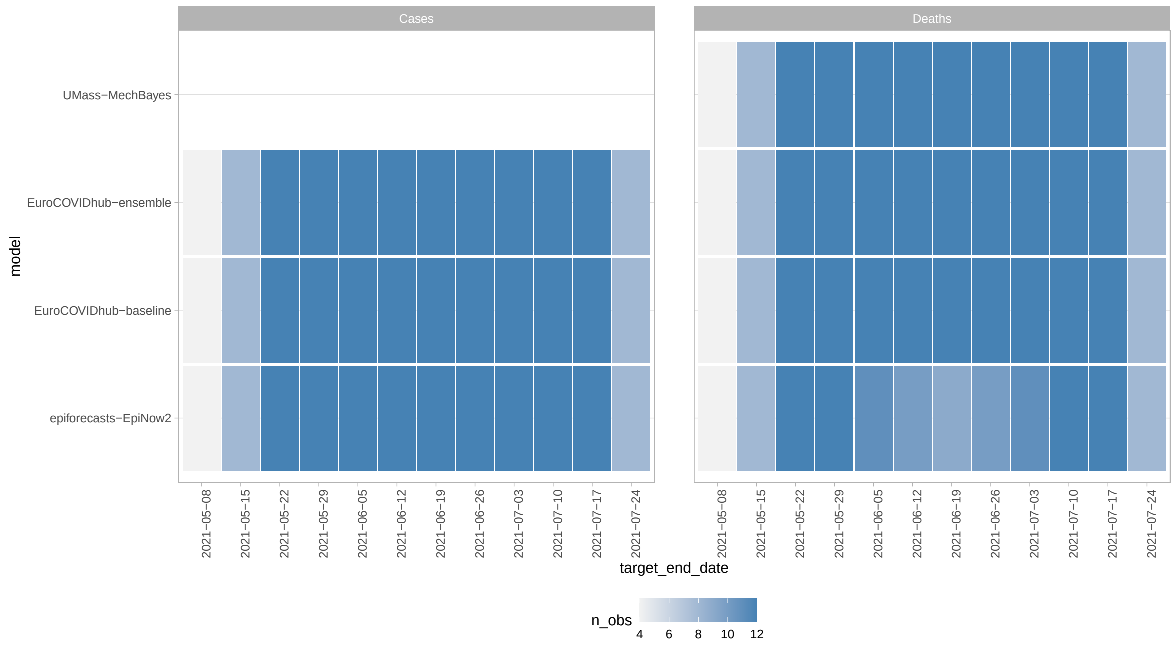


Figure 3: Overview of the number of available forecasts

and observed values themselves can be visualised using the `plot_predictions()` function. Data visualisation is of course highly context-dependent, but the function tries to accommodate most practical use-cases. In addition to basic plotting functionality it offers the user an optional ad hoc way to filter both forecasts and observed values. This makes it possible to tweak the plot in a beginner-friendly way without having to manipulate the data separately. Forecasts and observed values can be passed in separately (and are merged internally) or as a single data.frame. Conditions to filter on need to be provided as a list of strings, where each of the strings represents an expression that can be evaluated to filter the data. To display, for example, short-term forecasts for COVID-19 cases and deaths made by the EuroCOVIDhub-ensemble model on June 28 2021 as well as 5 weeks of prior data, we can call: The output is shown in Figure 4.

3.2. Scoring forecasts with `eval_forecasts()`

The actual scoring of forecasts based on observed values can be performed using the function `eval_forecasts()`. The function automatically detects the forecast type and format (as shown in the output of `check_forecasts()`), applies the appropriate scoring metrics and can

```
> plot_predictions(data = example_quantile, x = "target_end_date",
+     filter_truth = list("target_end_date <= \"2021-07-15\"",
+         "target_end_date > \"2021-05-22\""), filter_forecasts = list("model == 'EuroCOVI
+         "forecast_date == \"2021-06-28\""), facet_formula = target_type ~
+         location, ncol = 4) + ggplot2::theme(legend.position = "bottom")
```

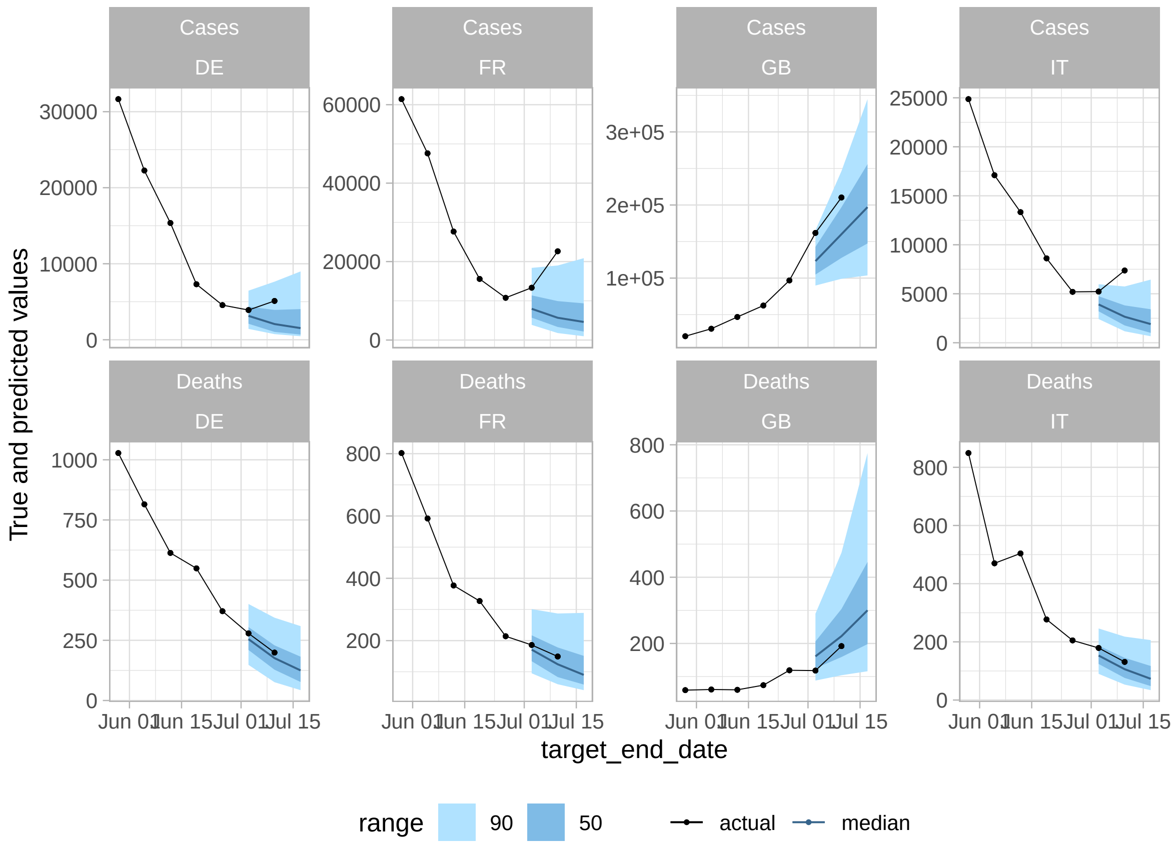


Figure 4: Short-term forecasts for COVID-19 cases and deaths made by the EuroCOVIDhub-ensemble model on June 28 2021.

in addition also aggregate results as well as perform pairwise comparisons between models. Internally, operations are handled using **data.table** to allow for fast and efficient computation.

```
> scores <- eval_forecasts(example_quantile)
> glimpse(scores)

## Rows: 887
## Columns: 14
## $ location            <chr> "DE", "DE", "DE", "DE", "DE", "DE", "DE", "DE", "DE~
## $ target_end_date     <date> 2021-05-08, 2021-05-08, 2021-05-08, 2021-05-08, 20~
## $ target_type         <chr> "Cases", "Cases", "Cases", "Deaths", "Deaths", "Dea~
## $ location_name      <chr> "Germany", "Germany", "Germany", "Germany", "German~
## $ forecast_date      <chr> "2021-05-03", "2021-05-03", "2021-05-03", "2021-05-~
## $ model              <chr> "EuroCOVIDhub-baseline", "EuroCOVIDhub-ensemble", "~
## $ horizon            <dbl> 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 2, ~
## $ interval_score     <dbl> 16925.04696, 7990.85478, 25395.96087, 46.79304, 53.~
## $ dispersion         <dbl> 1649.22087, 5440.98522, 8173.70000, 44.66261, 53.27~
## $ underprediction    <dbl> 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.60869~
## $ overprediction     <dbl> 15275.826087, 2549.869565, 17222.260870, 2.130435, ~
## $ coverage_deviation <dbl> -0.38521739, 0.04956522, -0.29826087, 0.22347826, 0~
## $ bias               <dbl> 0.95, 0.50, 0.90, 0.30, -0.10, -0.50, 0.10, 1.00, 0~
## $ aem                <dbl> 25620, 12271, 44192, 15, 14, 208, 24, 67622, 45731,~
```

The above produces one score for every forecast. However, we usually like to summarise scores to learn about average performance.

We can either run a separate function to summarise the scores,

```
> summarised_scores <- summarise_scores(scores = scores, summarise_by = c("model",
+     "target_type"))
```

or we score and summarise in one function call:

```
> summarised_scores <- eval_forecasts(example_quantile, summarise_by = c("model",
+     "target_type"))
```

```
> summarised_scores

##                     model target_type interval_score dispersion underprediction
## 1: EuroCOVIDhub-baseline       Cases    28483.57465 4102.50094    10284.972826
## 2: EuroCOVIDhub-ensemble       Cases    17943.82383 3663.52458     4237.177310
## 3:  epiforecasts-EpiNow2       Cases    20831.55662 5664.37795     3260.355639
## 4: EuroCOVIDhub-baseline      Deaths      159.40387   91.40625        2.098505
## 5: EuroCOVIDhub-ensemble      Deaths       41.42249   30.18099        4.103261
## 6:        UMass-MechBayes      Deaths       52.65195   26.87239       16.800951
## 7:  epiforecasts-EpiNow2      Deaths       66.64282   31.85692       15.893314
##    overprediction coverage_deviation        bias         aem
## 1:   14096.100883        -0.11211957  0.09796875 38473.60156
## 2:   10043.121943        -0.09785326 -0.05640625 24101.07031
## 3:   11906.823030        -0.06660326 -0.07890625 27923.81250
## 4:      65.899117         0.11614130  0.33906250   233.25781
## 5:       7.138247         0.19528533  0.07265625    53.13281
## 6:       8.978601        -0.02312500 -0.02234375    78.47656
## 7:      18.892583        -0.04287176 -0.00512605   104.74790
```

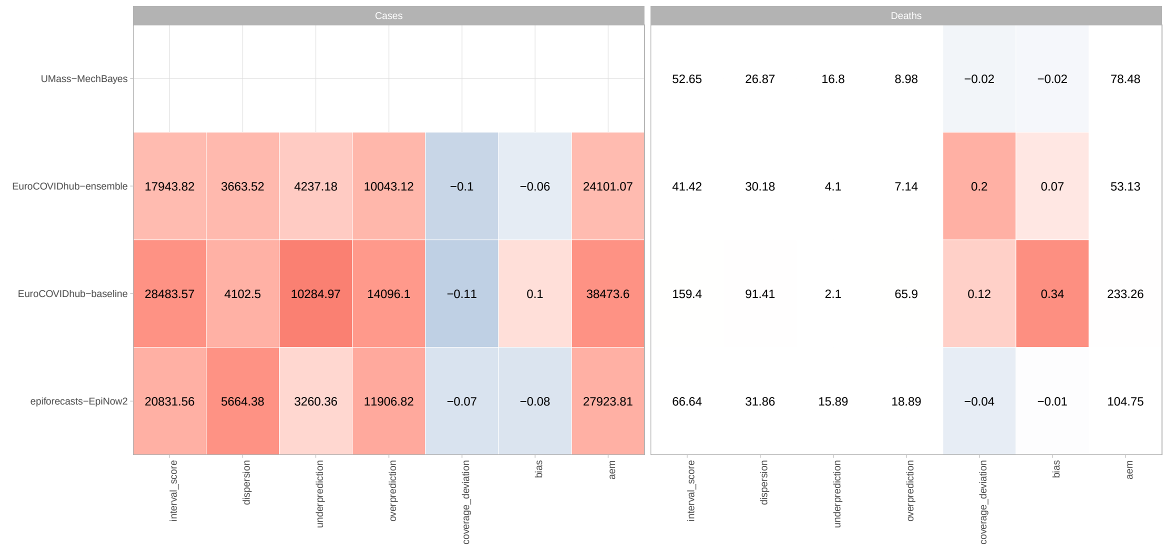
`eval_forecasts()` and `summarise_scores()` aggregate scores by taking the mean over according to the grouping specified in `summarise_by`. In the above example, if `summarise_by`

`= c("model", "target_type")`, then scores would be averaged over all other categories to obtain one score per model and forecast target type. For a more detailed analysis we could for example specify `summarise_by = c("mode", "target_type", "location")` to additionally stratify by country. Summarised scores can be visualised using the function `scores_table()` as is shown in Figure 5.

If we wanted to have one score per quantile or one per prediction interval range, we could use something like `summarise_by = c("model", "quantile", "range")`. This would allow us to examine interval or quantile coverage or makes it possible to analyse the accuracy of the tails of the forecasts. In addition to the mean, we can also obtain the standard deviation of the scores over which we average, as well as any desired quantile, by specifying `sd = TRUE` and for example `quantiles = c(0.5)` for the median.

The user must, however, exercise some caution when aggregating scores. As explained above, many of the metrics are absolute and scale with the magnitude of the quantity to forecast. This makes it sometimes ill-advised to average over them. In the given example, looking at one score per model (i.e. specifying `summarise_by = c("model")`) is problematic, as overall aggregate scores would be dominated by case forecasts, while errors on death forecasts would have little influence. Similarly, aggregating over different forecast horizons is often not a good idea as the mean will be dominated by further ahead forecast horizons.

```
> score_table(summarised_scores, y = "model", facet_formula = ~target_type)
```



	Cases							Deaths						
	interval_score	dispersion	underprediction	overprediction	coverage_deviation	bias	aem	interval_score	dispersion	underprediction	overprediction	coverage_deviation	bias	aem
UMass−MechBayes								52.65	26.87	16.8	8.98	−0.02	−0.02	78.48
EuroCOVIDhub−ensemble	17943.82	3663.52	4237.18	10043.12	−0.1	−0.06	24101.07	41.42	30.18	4.1	7.14	0.2	0.07	53.13
EuroCOVIDhub−baseline	28483.57	4102.5	10284.97	14096.1	−0.11	0.1	38473.6	159.4	91.41	2.1	65.9	0.12	0.34	233.26
epiforecasts−EpiNow2	20831.56	5664.38	3260.36	11906.82	−0.07	−0.08	27923.81	66.64	31.86	15.89	18.89	−0.04	−0.01	104.75

Figure 5: Coloured table to visualise the computed scores

In order to obtain a model ranking, we recommend to look at the relative skill in terms of an appropriate proper scoring rule instead of the raw score. Relative skill scores can be aggregated more easily across different forecast targets as they are less influenced by the order of magnitude of the quantity to forecast than e.g. the WIS or the CRPS.

Again there are two possible ways to do pairwise comparisons. First, the function `pairiwse_comparison()` can be called to compare models against each other. Here `summarise_by` denotes what relative skill scores shall be computed for. `summarise_by = "model"` means there will be one relative skill score per model, while `summarise_by = c("target_type", "target_type")` would mean there is one relative skill score per model, computed completely separately for different target types.

When using `pairwise_comparison()`, it is important that the scores used as input have not been summarised in any way.

```
> pairwise <- pairwise_comparison(scores, baseline = "EuroCOVIDhub-baseline")
>
> pairwise2 <- pairwise_comparison(scores, summarise_by = c("model",
+     "target_type"), baseline = "EuroCOVIDhub-baseline")
```

Pairwise comparisons between models [CITATION] can be obtained in two different ways. First, relative skill scores based on pairwise comparisons are by default returned from `eval_forecasts()`. These will be computed separately for the categories defined in the `summarise_by` argument (excluding the category 'model'). Alternatively, a set of scores can be post-processed using the separate function `pairwise_comparison()`. This approach is to be used for visualisation and if p-values for the pairwise comparisons are needed, as those are not returned from `eval_forecasts()`. Usually, one would compute scores without specifying a `summarise_by` argument, but sometimes it may be sensible to average over certain scores, for example for predictions generated at a certain date. This allows to reduce the correlation between observations that enter the computation of p-values, which in turn makes the test less liberal. Using the function `plot_pairwise_comparison()` we can visualise the mean score ratios between all models as well as the

The result is a `data.table` with different scores and metrics in a tidy format that can easily be used for further manipulation and plotting.

4. Visualisation and interpretation of evaluation results

4.1. Visualising aggregate scores and rankings

A good starting point for an evaluation is the following score table that visusalises the scores we produced above. We can facet the table to account for the different forecast targets:

The most informative metric in terms of model ranking is the relative_skill. However, interpretation is not always straightforward and has to be done carefully. We can see that performance varied quite a bit across different metrics, where some models did well on one target, but poorly on another. Especially the Exponential growth/decline model stands out as it received the lowest relative skill score for hospital admissions, but the highest for the total number of beds occupied. Looking back at Figure 3, we see that the model has only submitted very few forecasts over all. It may therefore be sensible to require all models to have submitted forecasts for at least 50% of all forecast targets in order to enter the pairwise comparisons. For similar reasons, the interval score may be deceiving if looked at in isolation. As can be seen, the DetSEIRwithNB MLE model received a lower relative skill score, but a higher interval score than the DetSEIRwithNB MCMC model. This, again, can be explained by the fact that they forecasted different targets. The interval score, as an absolute metric,

```
> plot_pairwise_comparison(pairwise2) + ggplot2::facet_wrap(~target_type,
+     scales = "free_x")
```

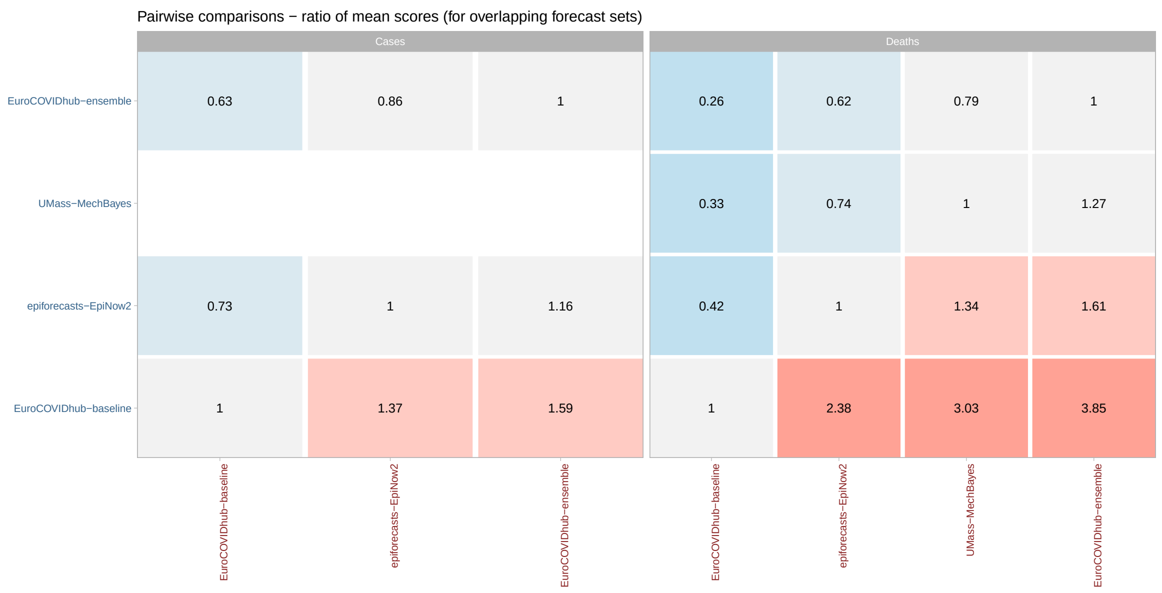


Figure 6: Ratios of mean scores based on overlapping forecast sets. If a tile is blue, then the model on the y-axis performed better. If it is red, the model on the x-axis performed better in direct comparison.

is highly influenced by the absolute value of the quantity that is forecasted. For the same reason, one should be careful when summarising interval scores from different locations or forecast targets, as the average score will be dominated by outliers as well as differences in the absolute level. Assuming a large enough set of available overlapping forecasts, the relative skill score is more robust. It therefore is reasonable to assume that the DetSEIRwithNB MLE forecasted quantities with a higher absolute value, but tended to perform worse than the DetSEIRwithNB MCMC model as far as we can tell based on the set of all pariwise comparisons. This can be confirmed for the direct comparison between the two by looking at the mean score ratios from the pairwise comparisons. These can be obtained by calling

In terms of actually understanding *why* one model performs well or badly, the other metrics shown in Figure 5 provide additional insight. We turn to them in the following.

4.2. Visual model diagnostics

For forecasts in an interval format, looking at the components of the weighted interval score separately is a natural next step. We can see in Figure 7 that the majority of penalties come from over-and underprediction, instead of the sharpness component. We also see that most models tended to either over- or underpredict actual numbers.

We can have a closer look at calibration using the functions `interval_coverage()` and `quantile_coverage()`. The interval coverage plot shows the proportion of all true values that

```
> wis_components(summarised_scores, facet_formula = ~target_type,
+     scales = "free_x", relative_contributions = TRUE, x_text_angle = 0) +
+     ggplot2::coord_flip() + ggplot2::theme(legend.position = "bottom")
```

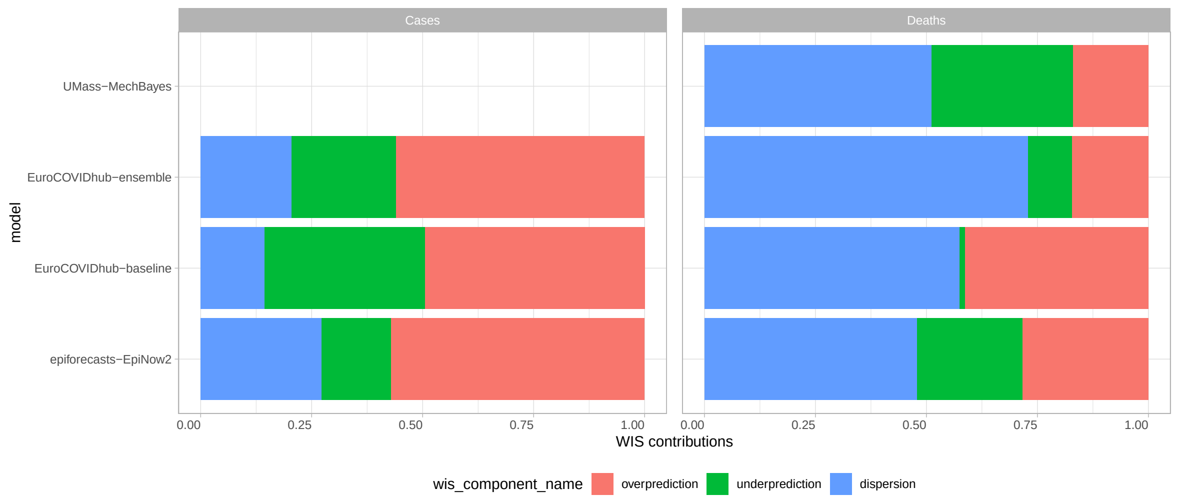
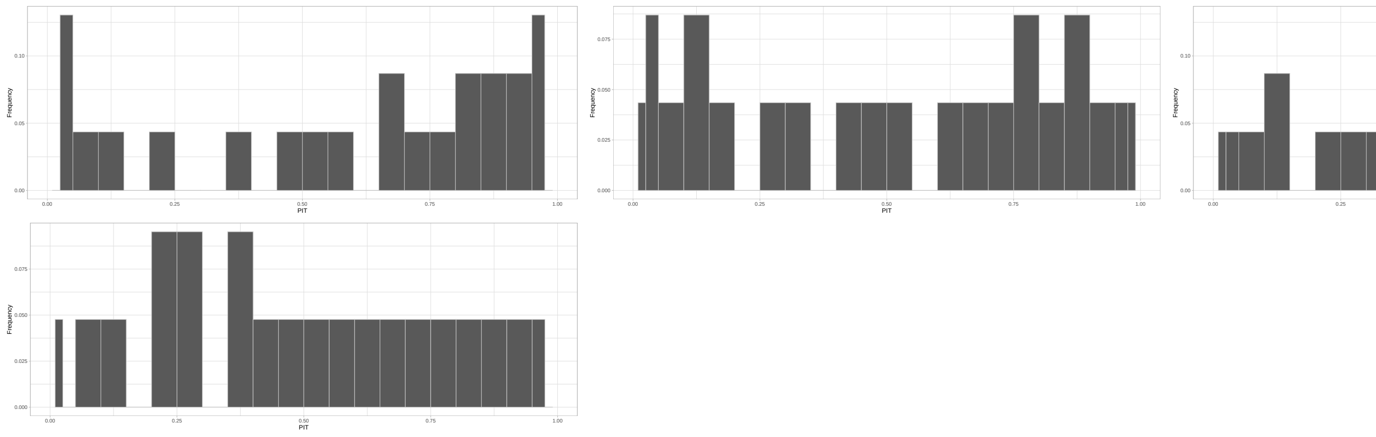


Figure 7: p-values and ratios together.

fall within all the different prediction intervals. This gives a visual impression of probabilistic calibration **??**. Ideally, x percent of true values should be covered by the $x\%$-prediction intervals, resulting in a 45° line. Areas shaded in green indicate that the model is covering more true values than it actually should, while areas in white indicate that the model fails to cover the desired proportion of true values with its prediction intervals. The majority of the models were too confident in their predictions, while some showed showed good calibration. The quantile coverage plot shows the proportion of all true values below certain predictive quantiles. While this plot is slightly harder to interpret, it also includes information about bias as and allows to separate the lower and upper boundaries of the prediction intervals. We can see, for example, that the Exponential growth/decline model was consistently biased downwards. Figure 8

```
> pit <- pit_df(example_quantile, summarise_by = "model", "target")
>
> hist_PIT(pit)

## $`EuroCOVIDhub-baseline`
##
## $`EuroCOVIDhub-ensemble`
##
## $`epiforecasts-EpiNow2`
##
## $`UMass-MechBayes`
```



DO I WANT TO INCLUDE PIT PLOTS AS WELL? I GUESS? NEED TO LOOK AT THE IMPLEMENTATION FOR QUANTILE FORECASTS

Look at e.g. bias by location? Figure 9

WHAT IS NEEDED HERE IS A BIT OF THINKING WITH REGARDS TO WHAT VISUALISATION I WANT TO SHOW AND IN HOW MUCH DETAIL I WANT TO ANALYSE THE MODELS.

5. Summary and discussion

COMING SOON.

Acknowledgments

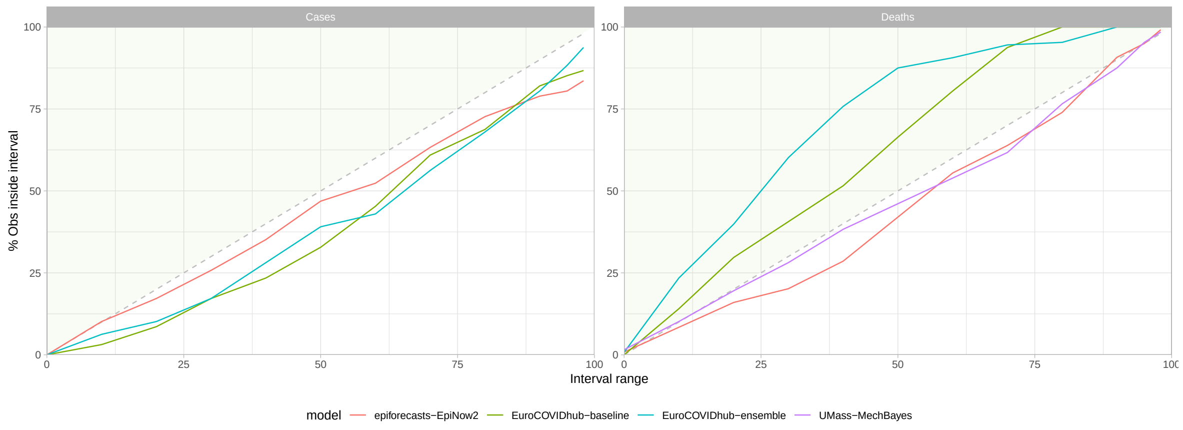
References

Bracher J, Ray EL, Gneiting T, Reich NG (????). "Evaluating Epidemic Forecasts in an Interval Format." **17**(2), e1008618. ISSN 1553-7358. doi:10.1371/journal.pcbi.1008618. 33577550.

Dawid AP (????). "Present Position and Potential Developments: Some Personal Views Statistical Theory the Prequential Approach." **147**(2), 278–290. ISSN 2397-2327. doi:10.2307/2981683. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.2307/2981683.

Funk S, Abbott S, Atkins BD, Baguelin M, Baillie JK, Birrell P, Blake J, Bosse NI, Burton J, Carruthers J, Davies NG, Angelis DD, Dyson L, Edmunds WJ, Eggo RM, Ferguson NM, Gaythorpe K, Gorsich E, Guyver-Fletcher G, Hellewell J, Hill EM, Holmes A, House TA, Jewell C, Jit M, Jombart T, Joshi I, Keeling MJ, Kendall E, Knock ES, Kucharski AJ, Lythgoe KA, Meakin SR, Munday JD, Openshaw PJM, Overton CE, Pagani F, Pearson J, Perez-Guzman PN, Pellis L, Scarabel F, Semple MG, Sherratt K,

```
> cov_scores <- eval_forecasts(example_quantile, summarise_by = c("model",
+     "target_type", "range", "quantile"))
>
> scoringutils::interval_coverage(cov_scores, facet_formula = ~target_type)
```



```
> scoringutils::quantile_coverage(cov_scores, facet_formula = ~target_type)
```

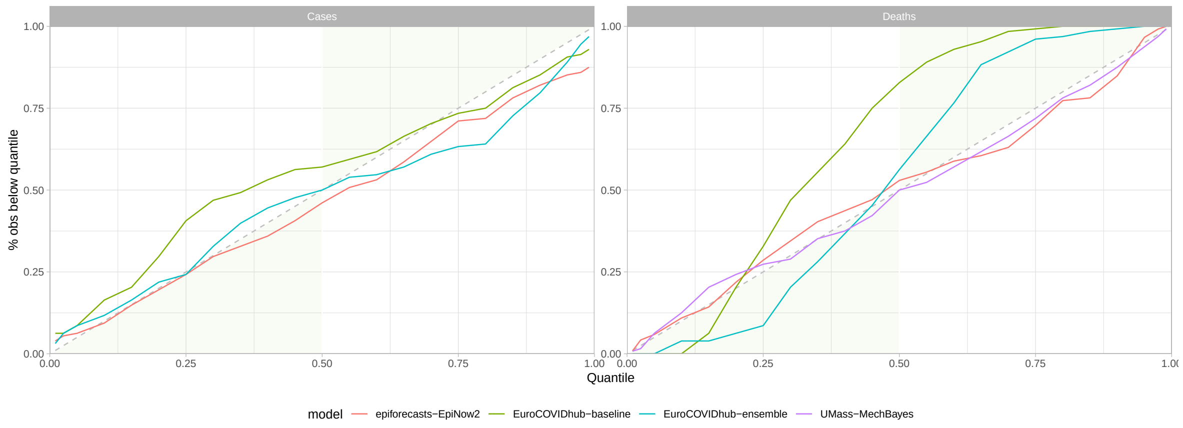


Figure 8: quantile and interval coverage

Tang M, Tildesley MJ, Leeuwen EV, Whittles LK, Group CCW, Team ICCR, Investigators I (????a). "Short-Term Forecasts to Inform the Response to the Covid-19 Epidemic in the UK." p. 2020.11.11.20220962. doi:10.1101/2020.11.11.20220962. URL https://www.medrxiv.org/content/10.1101/2020.11.11.20220962v1.

Funk S, Camacho A, Kucharski AJ, Lowe R, Eggo RM, Edmunds WJ (????b). "Assessing the Performance of Real-Time Epidemic Forecasts: A Case Study of Ebola in the Western Area Region of Sierra Leone, 2014-15." **15**(2), e1006785. ISSN 1553-7358. doi:10.1371/journal.pcbi.1006785. URL https://journals.plos.org/ploscompbiol/

```
> scores <- eval_forecasts(example_quantile, summarise_by = c("model",
+     "target_type", "location_name"), compute_relative_skill = FALSE)
> scoringutils::score_heatmap(scores, metric = "interval_score",
+     x = "location_name", facet_formula = ~target_type, scale = "free_x")
```

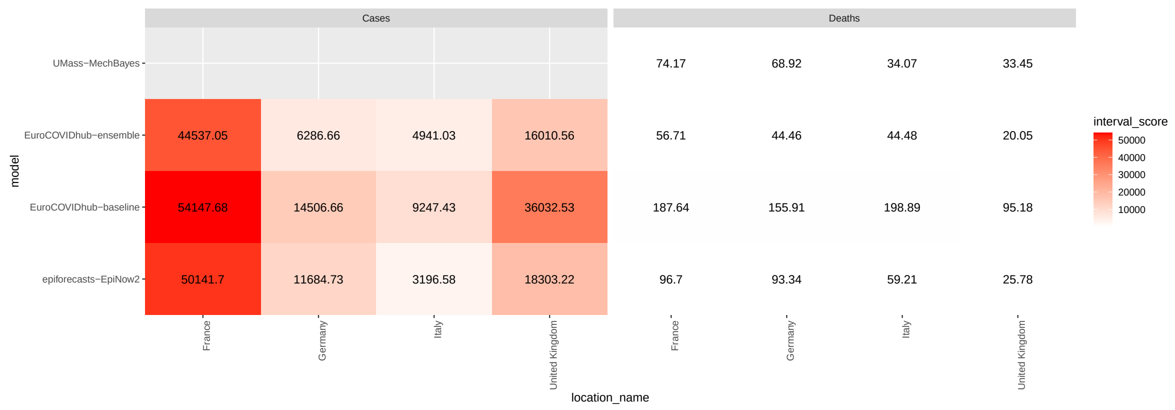


Figure 9: bias by location

article?id=10.1371/journal.pcbi.1006785.

Gneiting T, Balabdaoui F, Raftery AE (????). "Probabilistic Forecasts, Calibration and Sharpness." **69**(2), 243–268. ISSN 1467-9868. doi:10.1111/j.1467-9868.2007.00587.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2007.00587.x.

Gneiting T, Raftery AE (????). "Strictly Proper Scoring Rules, Prediction, and Estimation." **102**(477), 359–378. ISSN 0162-1459, 1537-274X. doi:10.1198/016214506000001437. URL http://www.tandfonline.com/doi/abs/10.1198/016214506000001437.

Jordan A, Krüger F, Lerch S (2019). "Evaluating Probabilistic Forecasts with scoringRules." *Journal of Statistical Software*, **90**(12), 1–37. doi:10.18637/jss.v090.i12.

Timmermann A (????). "Forecasting Methods in Finance." **10**(1), 449–479. doi:10.1146/annurev-financial-110217-022713. URL https://doi.org/10.1146/annurev-financial-110217-022713.

Winkler RL, Muñoz J, Cervera JL, Bernardo JM, Blattenberger G, Kadane JB, Lindley DV, Murphy AH, Oliver RM, Ríos-Insua D (????). "Scoring Rules and the Evaluation of Probabilities." **5**(1), 1–60. ISSN 1863-8260. doi:10.1007/BF02562681. URL https://doi.org/10.1007/BF02562681.

A. Appendix section

Table 5: Explanation of all the scores

Affiliation:

Nikos Bosse
Centre for Mathematical Modelling of Infectious Diseases
London School of Hygiene and Tropical Medicine
Keppel Street
London WC1E 7HT
E-mail: nikos.bosse@lshtm.ac.uk

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd