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. Its role in informing public policy has attracted increased attention from the general public with the emergence of the Covid-19 pandemic. Even with a plethora of tools and models available to make forecasts, evaluating the predictions is not trivial. Even though scoring methods such as proper scoring rules have been widely studied in the past, there is a lack of software implementation that allows a forecaster to conveniently evaluate their forecasts. 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 rationale behind forecast evaluation and the metrics implemented in **scoringutils** and show a full evaluation of a set of short-term forecasts of public health related targets made by SPI-M during the 2020 Covid-19 epidemic in the United Kingdom.

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 (), weather predictions or infectious disease modeling (Funk, Abbott, Atkins, Baguelin, Baillie, Birrell, Blake, Bosse, Burton, Carruthers, Davies, Angelis, Dyson, Edmunds, Eggo, Ferguson, Gaythorpe, Gorsich, Guyver-Fletcher, Hellewell, Hill, Holmes, House, Jewell, Jit, Jombart, Joshi, Keeling, Kendall, Knock, Kucharski, Lythgoe, Meakin, Munday, Openshaw, Overton, Pagani, Pearson, Perez-Guzman, Pellis, Scarabel, Semple, Sherratt, Tang, Tildesley, Leeuwen, Whittles, Group, Team, and Investigators 2020). An integral part of assessing and improving their usefulness is forecast evaluation. For decades, researchers therefore have developed and refined an arsenal of techniques not only to forecast, but also to evaluate these forecasts (see e.g. Bracher, Ray, Gneiting, and Reich (2020), **?**, Gneiting, Balabdaoui, and Raftery

(2007), and Gneiting and Raftery (2007)). Yet even with this rich body of research available, implementing a complete forecast evaluation in R is not trivial. We therefore present the **scoringutils** package. The goal of the scoringuitls package is to facilitate the evaluation process and to allow even inexperienced users to perform a thorough evaluation of their forecasts. In this paper we give a quick introduction of the fundamental ideas behind forecast evaluation, explain the evaluation metrics implemented in **scoringutils** and present a full example evaluation of Covid-19 related short-term forecasts in the UK (Funk *et al.* 2020).

1.1. Forecast types

In its most general sense, a forecast is the forecaster's stated belief about the future (Gneiting and Raftery 2007). For conceptual clarity, it is however useful to distinguish between different types of forecasts. In terms of the quantitative forecasts that can be evaluated with **soringutils**, we can distinguish point forecasts from probabilistic forecasts. A point forecast states a single number and is the simplest form of a forecast. It is limited in its usefulness, as it does not give information about the uncertainty of the forecast. A very certain forecast may, for example, warrant a very different course of actions than does a very uncertain one. A probabilistic forecast, in contrast, is a forecast made in terms of the entire predictive distribution. Providing the entire predictive distribution allows the forecaster to express 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.

1.2. Forecast formats

Not only can we distinguish different forecast types, but also different formats in which predictions may be reported. This distinction is important, as different formats and prediction types require slightly different evaluation approaches. Forecasts can be expressed analytically in a closed form, or more commonly through either predictive samples or quantiles. Predictive samples are useful if no closed-form predictive distribution is available, but require a lot of storage space. They may 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 are reported instead [citation FORECAST HUBS]. As an alternative representation of a quantile forecast, one can also express predictions using central prediction intervals. In this case, the forecaster reports a range to denote the interval as well as a value for the lower and upper bound. A forecaster could also in principle state their forecasts in a binary way by defining an outcome and assigning a probability that the outcome will come true. This type of forecasting is common in many classification problems. If we think of the outcome of some number being larger or smaller than a certain value, we can immediately see the connection between binary predictions and the concept of a cumulative distribution functions (CDF). All of these forecast types can be evaluated using **scoringutils**. The general forecasting paradigm [GNEITING] that guides the evaluation process is the same irrespective of the reporting format, even though specific scoring metrics differ.

1.3. The forecasting paradigm

Any forecaster should aim to minimise the difference between the predictive distribution and the unknown true data-generating distribution [@gneitingProbabilisticForecastsCalibration2007]. For an ideal forecast, we therefore have

$$P_t = F_t,$$

where P_t is the the cumulative density function (CDF) of the predictive distribution at time t and F_t is the CDF of the true, unknown data-generating distribution. As we don't know the true data-generating distribution, we cannot assess the difference between the two distributions directly. [CITATIONS] instead suggest to focus on two central aspects of the predictive distribution, calibration and sharpness. Calibration refers to the statistical consistency between the predictive distribution and the observations. A well calibrated forecast does not systematically differ deviate from the observed values. For an in-depth discussion of different ways in which a forecast can be miscalibrated, we refer to [GneitingProbabilisticForecastsCalibration2007]. 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. Take for example the task of predicting rainfall in a city like London. A model that made very precise forecasts would not be useful if the forecasts were wrong most of the time. On the other hand, a model that predicts the same rainfall probability for every day can be correct on average [To be precise, this model would be marginally calibrated according to @gneitingProbabilisticForecastsCalibration2007], but is also less useful than a model that were able to accurately predict the weather every single day.

2. Scoring metrics implemented in scoringutils

The metrics included in the **scoringuitls** package can be divided into two categories. The first consists of metrics that aim to capture different aspects of sharpness and calibration, the second comprises various proper scoring rules. We begin with the latter.

2.1. Proper scoring rules

Proper scoring rules [CITATION] 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 state their true best belief. The following scoring rules are implemented in **scoringutils**: The (continuous) ranked probability score (crps) [CITATION], the log score (logs) [CITATION], the Dawid-Sebastiani-score (dss) [CITATION], the (weighted) interval score (wis), and the Brier score (bs). The first three proper scoring rules are implemented as wrappers around functions from the **scoringRules** package. They are suitable for any sample-based prediction format. The Log Score, however, is not applied if integer-valued forecasts are supplied, as the implementation in **scoringRules** requires a kernel density estimation that may be inappropriate for integer values. The interval score is appropriate for forecasts in a quantile formats. For every central prediction interval, the interval score is computed as the sum of separate penalties for overprediction, underprediction and sharpness (i.e. width) of the forecast. By default, the weighted interval score is returned. The overall score for a forecast is obtained as the mean of all scores for the individual central prediction intervals, weighted according to the range of the prediction interval. This ensures that the weighted interval score converges to the continuous ranked probability score for an increasing number of available prediction intervals. DO I WANT FORMULAS FOR ALL THE SCORES? The Brier score is used for binary predictions.

TALK ABOUT CAVEATS (LATER?) E.G. THAT THE MEAN WIS IS INFLUENCED BY THE ABSOLUTE VALUE OF THE TARGET?

2.2. Evaluating calibration and sharpness independently

In addition to the proper scoring rules outlined above, **scoringutils** makes numerous metrics available to evaluate calibration and sharpness independently. This is especially helpful for model diagnostics.

Assessing calibration

Several strategies have been proposed to detect systematic deviations of the predictive distributions from the observations (see e.g. @funkAssessingPerformanceRealtime2019; @gneitingProbabilisticForecastsCalibration2007; @gneitingStrictlyProperScoring2007). Using **scoringutils**, we can look at three different 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. For continuous forecasts, assessing whether a predictive distribution has a tendency to over- or underpredict can be very easily achieved by simply evaluating the predictive distribution at the true observed value. This metric is a generalisation of the integer-valued one @funkAssessingPerformanceRealtime2019 have proposed. It is also closely related to the probability integral transform (PIT) discussed later in this chapter. To improve the interpretability of the score we can transform it to a value between -1 (under-prediction) and 1 (over-prediction). Consequently, we measure bias as

$$B_t(P_t, x_t) = 1 - 2 \cdot (P_t(x_t)),$$

where P_t is the cumulative distribution function of the predictive distribution for the true value x_t. When using predictive samples, $P_t(x_t)$ is simply the fraction of predictive samples for x_t that are smaller than the true observed x_t.

For integer valued forecasts, we use the metric proposed by @funkAssessingPerformanceRealtime2019:

$$B_t(P_t, x_t) = 1 - (P_t(x_t) + P_t(x_t + 1)).$$

Bias can again assume values between -1 (under-prediction) and 1 (over-prediction) and is 0 ideally.

For quantile forecasts, we propose the following metric to assess bias:

$$\begin{aligned} B_t =& (1 - 2 \cdot \max\{i | q_{t,i} \in Q_t \land q_{t,i} \leq x_t\}) \mathbb{1}(x_t \leq q_{t,0.5}) \\ &+ (1 - 2 \cdot \min\{i | q_{t,i} \in Q_t \land q_{t,i} \geq x_t\}) \mathbb{1}(x_t \geq q_{t,0.5}), \end{aligned}$$

where Q_t is the set of quantiles that form the predictive distribution at time t. They represent our belief about what the true value x_t will be. For consistency, we define Q_t such that it always includes the element $q_{t,0} = -\infty$ and $q_{t,1} = \infty$. $\mathbb{1}()$ is the indicator function that is 1 if the condition is satisfied and 0 otherwise. In clearer terms, B_t is defined as the maximum percentile rank for which the corresponding quantile is still below the true value, if the true value is smaller than the median of the predictive distribution. If the true value is above the median of the predictive distribution, then B_t is the minimum percentile rank for which the corresponding quantile is still larger than the true value. If the true value is exactly the median, both terms cancel out and B_t is zero. For a large enough number of quantiles, the percentile rank will equal the proportion of predictive samples below the observed true value, and this metric coincides with the one for continuous forecasts. For quantile forecasts, an alternative approach is to look at the over- and underprediction components of the weighted interval score. These however, capture the bias in absolute terms, while the above proposed metric captures a tendency to over- and underpredict that is less sensitive to outliers.

I ASSUME ALL OF THIS IS A BIT TOO LONG?

Another way to look at calibration[precisely: probabilistic calibration in @gneitingProbabilisticForecastsCalibration2007] 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 in principle be transferred to continuous and integer 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.

A third way to look at calibration is to look a the probability integral tranform. As explained previously, the CDF of predictice distribution P_t should ideally be equal to the CDF of the true unknown distribution F_t that generated the observed value x_t. In order to assess whether there are substantial deviations between the two, @dawidPresentPositionPotential1984 suggested to transform the observed values using the probability integral transform (PIT). Agreement between the forecasts and the observed values can then be examined by observing whether or not the transformed values follow a uniform distribution. The PIT is given by

$$u_t = P_t(x_t),$$

where u_t is the transformed variable and $P_t(x_t)$ is the predictive distribution evaluated at the true observed value x_t. If $P_t = F_t$ at all times t, then $u_t, t = 1 \dots T$ follows a uniform distribution (for a proof see e.g. @angusProbabilityIntegralTransform1994).

In the case of discrete outcomes, the PIT is no longer uniform even when forecasts are ideal. As @funkAssessingPerformanceRealtime2019 suggest, we use a randomised PIT instead by redefining

$$u_t = P_t(x_t) + vt \cdot (P_t(x_t) - P_t(x_t - 1)),$$

where x_t is again the observed value at time t, $P_t()$ is the CDF of the predictive distribution function, $P_t(-1) = 0$ by definition, and v_t is a standard uniform variable independent of x_t.

If P_t is equal to the true data-generating distribution function, then u_t is standard uniform. @czadoPredictiveModelAssessment2009 also propose a non-randomised version of the PIT for count data that could be used alternatively.

One can then plot a histogram of u_t values to look for deviations from uniformity. U-shaped histograms often result from predictions that are too narrow, while hump-shaped histograms indicate that predictions may be too wide. Biased predictions will usually result in a triangle-shaped histogram.

Assessing sharpness

Sharpness is the second property central to model evaluation. The ability to produce narrow forecasts is a quality of the forecasts only and does not depend on the observations. Sharpness is therefore only of interest conditional on calibration: a very precise forecast is not useful if it is clearly wrong. Again we need to take slightly different approaches for continuous, integer and quantile forecasts. For continuous and integer forecasts we follow the suggestion from @funkAssessingPerformanceRealtime2019. For quantile forecasts we propose a novel metric.

For continuous and integer forecasts, @funkAssessingPerformanceRealtime2019 suggest to measure sharpness as the normalised median absolute deviation about the median (MADN), i.e.

$$S_t(P_t) = \frac{1}{0.675} \cdot \text{median}(|y - \text{median(y)}|),$$

where y is the vector of all predictive samples and $\frac{1}{0.675}$ is a normalising constant that ensures that sharpness will equal the standard deviation of the predictive distribution if P_t is the CDF of a normal distribution.

MAKE THE FOLLOWING PARAGRAPH SHORTER AND JUST SAY WE USE THE SHARPNESS COMPONENT OF THE WIS.

For quantile forecasts, we propose to measure sharpness as a weighted mean of the width of the interval ranges. Let Q_t be a set of predicted quantiles for a true x_t at time t. This set of quantiles is assumed to be symmetric, such that there exist K corresponding pairs of elements $q_{t,\frac{\alpha}{2}}$ and $q_{t,1-\frac{\alpha}{2}}$. These K corresponding pairs of quantiles cover a $(1-\alpha)\cdot 100$ prediction interval. We can, accordingly, also denote $q_{t,\frac{\alpha}{2}}$ as l_t the lower bound of the prediction interval at time t and $q_{t,1-\frac{\alpha}{2}}$ as u_t, the upper bound. We measure the sharpness of a quantile forecast at time t as

$$\begin{aligned} \text{sharpness}_t &= \frac{1}{K}\sum_{\alpha}\frac{\alpha}{2}(q_{t,1-\frac{\alpha}{2}} - q_{t,\frac{\alpha}{2}}) \\ &= \frac{1}{K}\sum_{\alpha}\frac{\alpha}{2}(u_t - l_t). \end{aligned}$$

Weighting the width of different intervals with $\frac{\alpha}{2}$ ensures that the score does not grow indefinitely for very large prediction intervals, and correspondingly, very small α. We also argue that this sharpness metric for quantile forecasts is a natural choice for quantile forecasts as it corresponds to the sharpness component of the Weighted Interval Score.

2.3. Pairwise comparisons

If what we care about is to determine which model performs best, pairwise comparisons between models are a suitable approach [CITATION]. In turn, each pair of models is evaluated based on the targets that that both models have predicted. The mean score by one model is divided by the mean score of the other model to obtain a measure of relative performance. Relative skill scores [citation (not sure there is something published, preprint fron Estee Cramer and Nick Reich)] for every model can then be obtained by taking the geometric mean of all pairwise ratios calculated for each model (omitting comparisons where there is no overlapping set of forecasts). ADD FORMULAS. A skill score smaller than 1 indicates that a model is performing better than the average model. One can also scale the skill by a reference baseline. A scaled relative skill smaller than one then means that the model in question performed better than the baseline.

By applying a permutation test or non-parametric test like the Wilcoxon rank sum test we can also obtain p-values that help determine whether or not models perform significantly differently. THERE IS AN ISSUE WITH CORRELATION: THE TEST WOULD TREAT ALL OBSERVATIONS AS INDEPENDENT, WHICH THEY ARENT'T, SO I WOULD ASSUME THAT IT IS TOO LIBERAL. CURRENTLY, NO ADJUSTMENTS ARE IMPLEMENTED, I'M NOT REALLY SURE WHETHER TO DISCUSS THIS HERE OR NOT.

3. Evaluating UK short-term forecasts

3.1. The data

To illustrate the evaluation process with **scoringutils** we use short-term predictions of four different Covid-19 related targets in the UK made between March 31 and July 13 2020. Forecasts were produced by six groups in the UK, and submitted to the Scientific Pandemic Influenza Group on Modelling (SPI-M). The forecasts aimed to assess the likely future burden the UK healthcare system would face from the Covid-19 pandemic. Predictions were then aggregated and used to inform UK government health policy through the Strategic Advisory Group of Experts (SAGE). The data, as well as the individual forecast models are discussed in more depth in [FUNK ET AL].

- timing (weekly?) and number of forecast dates - the four targets - the models. - is the set complete?

We first need to obtain the data by installing and loading the **covid19.forecasts.uk** using the following commands:

```
R> # install and load data pacakge from external repository
R> # remotes::install_github("sbfnk/covid19.forecasts.uk")
R>
R> # load packages
R> library(covid19.forecasts.uk)
R> library(dplyr)
R> library(scoringutils)
R> # load truth data
R> data(covid_uk_data)
R> # head(covid_uk_data)
R>
R> # load forecasts
```

```
R> data(uk_forecasts)
R> # head(uk_forecasts)
```

Let us take a first look a the data:

```
R> glimpse(covid_uk_data)

Rows: 4,174
Columns: 6
$ geography  <fct> London, London, London, London, London, London, ...
$ value_type <fct> hospital_inc, hospital_inc, hospital_inc, hospit...
$ value_desc <fct> Hospital admissions, Hospital admissions, Hospit...
$ truncation <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
$ value_date <date> 2020-03-20, 2020-03-21, 2020-03-22, 2020-03-23,...
$ value      <dbl> 18, 232, 280, 241, 313, 353, 516, 637, 677, 546,...
```

```
R> glimpse(uk_forecasts)

Rows: 1,254,513
Columns: 8
$ model         <fct> EpiSoon, EpiSoon, EpiSoon, EpiSoon, EpiSoon, ...
$ geography     <chr> "East of England", "East of England", "East o...
$ value_type    <fct> hospital_inc, hospital_inc, hospital_inc, hos...
$ creation_date <date> 2020-03-31, 2020-03-31, 2020-03-31, 2020-03-...
$ value_date    <date> 2020-04-01, 2020-04-02, 2020-04-03, 2020-04-...
$ quantile      <dbl> 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.0...
$ value         <dbl> 70, 64, 53, 42, 30, 19, 4, 0, 0, 0, 0, 0, 0, ...
$ value_desc    <fct> Hospital admissions, Hospital admissions, Hos...
```

To bring the forecasts into the format needed for the evaluation, some minor changes need to be made to the data. The names of the columns that hold the forecasts and the true observed values need to be changed to `prediction` and `true_value`. While we could also proceed with separate data sets for the evaluation, we merge the two data sets in order to remove all instances where the forecasts, but not the true observations were made public. The **scoringutils** package provides a function that attempts to merge the data sets in a sensible way.

```
R> uk_forecasts <- rename(uk_forecasts, prediction = value)
R> covid_uk_data <-rename(covid_uk_data, true_value = value)
R> combined <- merge_pred_and_obs(uk_forecasts, covid_uk_data)
```

Before we start with scoring the forecasts, it makes sense to start the evaluation process by visualising the data. To get a feeling for how complete the data set is, we can run the following code to obtain a heatmap with the number of available forecasts: Missing forecasts can have a large impact on the forecast evaluation, if forecasts are not missing at random, but instead missingness correlates with performance. By default, the function treats a set of different quantiles or samples as one forecast. However, the user can specify manually which elements

```
R> show_avail_forecasts(combined,
+                        x = "creation_date",
+                        show_numbers = FALSE,
+                        legend_position = "bottom",
+                        facet_formula = ~ value_desc)
```

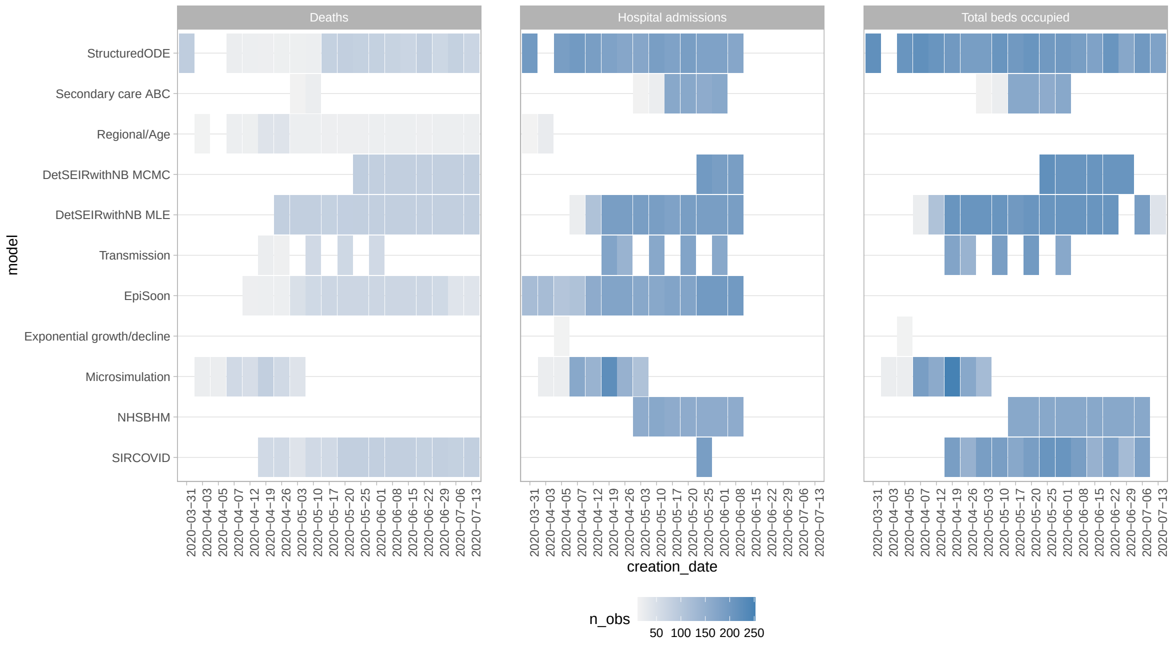


Figure 1: Overview of the number of forecasts available

to treat as one forecast and which categories to sum over to count the number of available forecasts.

Forecasts can be visualised using the `plot_predictions()` function. The function accepts either a single combined data set or two separate truth data sets that can be merged together. Forecasts and observed values can be filtered independently, to show for example only predictions from a certain model or a specific number of weeks of true data before the forecast. 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 obtain, for example, a specific forecast from a model we are interest in, we can call: The output is shown in Figure 2.

3.2. Scoring forecasts with `eval_forecasts()`

A full evaluation of all forecasts based on observed values can be performed using the function `eval_forecasts()`. This requires a `data.frame` or similar which has at least a column called "prediction" and one called "true_value". Depending on the exact input format, additional columns like "sample", "quantile", or "range" and "boundary" are needed. Additional columns may be present to indicate a grouping of forecasts, for example forecasts made in different locations or over different forecast horizons. We will not discuss all possible input formats

```
R> locations <- 'c("Scotland", "Wales", "Northern Ireland", "England")'
R> plot_predictions(truth = covid_uk_data,
+                   forecasts = uk_forecasts,
+                   filter_both = list(paste("geography %in%", locations, collapse = "")),
+                   filter_truth = list('value_date <= "2020-06-22"',
+                                       'value_date > "2020-06-01"'),
+                   filter_forecasts = list('model == "SIRCOVID"',
+                                           'creation_date == "2020-06-22"'),
+                   x = "value_date",
+                   facet_formula = geography ~ value_type) +
+    ggplot2::theme(legend.position = "bottom")
```

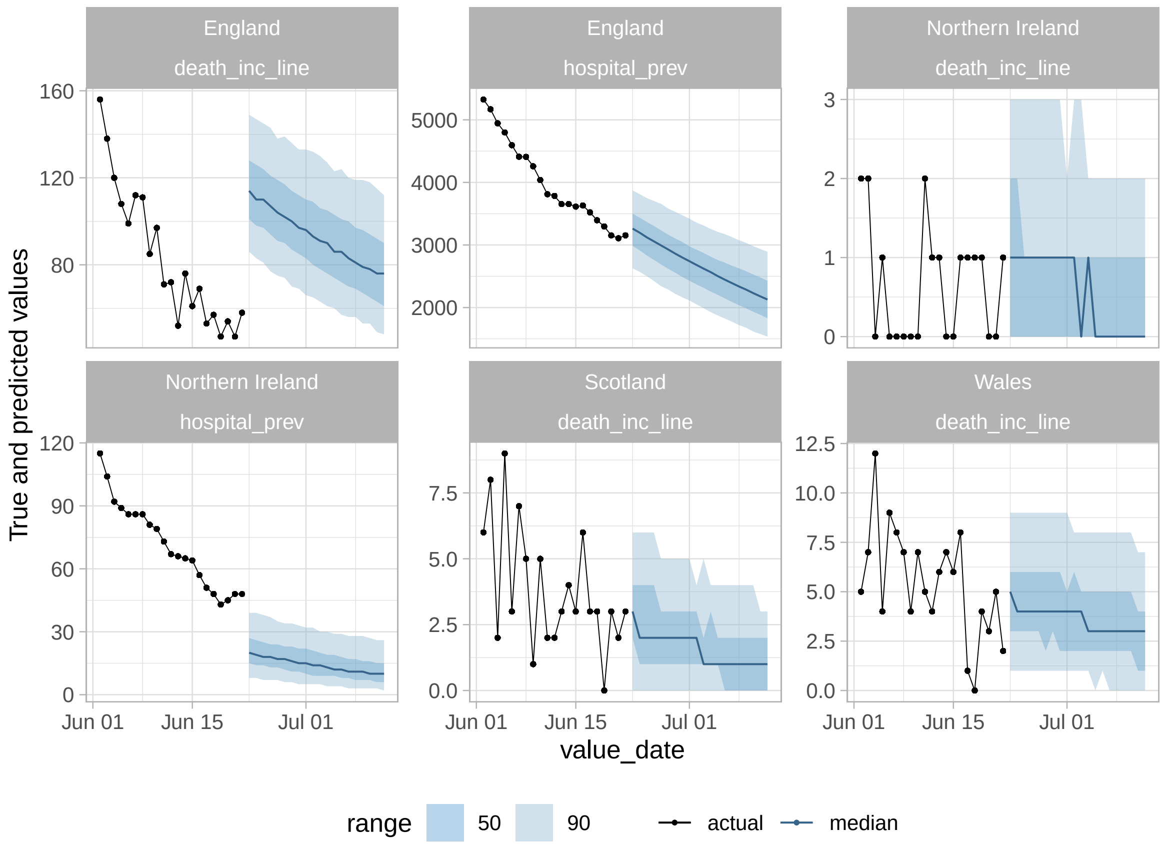


Figure 2: Short-term forecasts made by the SIRCOVID model on June 22 2020.

here, but instead refer to the example data for each format that is provided with the package. Where possible, **scoringutils** also provides functionality to transform between various formats, e.g. from a sample based format to a quantile format. The `eval_forecasts()` function automatically recognises the prediction type and input format, applies the appropriate scoring metrics and aggregates results as desired by the user. Internally, operations are handled using **data.table** to allow for fast and efficient computation.

As a first start for the evaluation of UK short-term forecasts, it makes sense to look at the scores achieved by every model separate for all prediction targets. This can be achieved by calling

```
R> scores <- eval_forecasts(combined,
+                             summarise_by = c("model", "value_desc"))
R> glimpse(scores)
```

```
Rows: 29
Columns: 11
$ model              <chr> "EpiSoon", "StructuredODE", "Microsimula...
$ value_desc         <fct> Hospital admissions, Hospital admissions...
$ interval_score     <dbl> 38.56510, 61.75914, 100.71914, 35.44876,...
$ sharpness          <dbl> 22.886153, 6.133974, 33.985553, 10.36508...
$ underprediction    <dbl> 3.66782766, 24.82255866, 33.12350598, 8....
$ overprediction     <dbl> 12.0111222, 30.8026045, 33.6100814, 16.5...
$ coverage_deviation <dbl> 0.034531405, -0.268749094, -0.095652174,...
$ bias               <dbl> 0.26194170, -0.27368253, -0.39217045, -0...
$ aem                <dbl> 45.96333, 73.38096, 138.14395, 53.22563,...
$ relative_skill     <dbl> 0.9085917, 1.2479010, 0.7405995, 0.82232...
$ scaled_rel_skill   <dbl> NaN, NaN, NaN, NaN, NaN, NaN, NaN, NaN, ...
```

If a more detailed analysis is desired, the level of aggregation can of course be changed to show for example separate scores for the different locations as well. This can be achieved using the `summarise_by` argument. To additionally stratify by location, we could specify `summarise_by = c("mode", "value_type", "geography")`. If we wanted to have one score per quantile or one per prediction interval range, we could specify something like `summarise_by = c("model", "quantile")` or `summarise_by = c("model", "quantile", "range")`. This can be useful if we, for example, want to analyse what proportion of true values are covered by certain interval ranges, or if we want to analyse the accuracy of the tails of the forecasts. When aggregating, `eval_forecasts()` takes the mean according to the group defined in `summarise_by`. In the above example, if `summarise_by = c("model", "value_type")`, then scores would be averaged over all creation dates, forecast horizons (as represented by the the value dates), locations and quantiles to yield one score per model and forecast target. 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, still exercise some caution when aggregating scores, as many of the metrics are absolute and scale with the magnitude of the quantity to forecast. Looking at one score per model (i.e. specifying `summarise_by = c("model")`) may not be so useful in this instance, as overall aggregate scores would be dominated by hospital admissions, while errors on death forecasts would have little influence.

In the above example, we did not have to explicitly specify the `by` argument, but this may be necessary if additional columns are present in the data that do not indicate a grouping of forecasts. The `by` argument must then be used to denote the unit of a single forecast. In the above example, the unit of a single forecast would be `by = c("model", "geography", "value_type", "creation_date", "value_date", "value_desc")`. Quantiles should not be included, as several quantiles make up one forecast (and similarly for samples). If we had additional columns that do not serve to group forecasts (like for example the number of inhabitants in a certain location over time), these should also not be included. By default, if `by = NULL`, `eval_forecasts()` will automatically use all present columns to determine the unit of a single forecast.

3.3. Pairwise comparisons

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

```
R> # unsummarised scores
R> unsummarised_scores <- eval_forecasts(combined)
R> pairwise <- pairwise_comparison(unsummarised_scores,
+                                  summarise_by = "value_desc")
```

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.

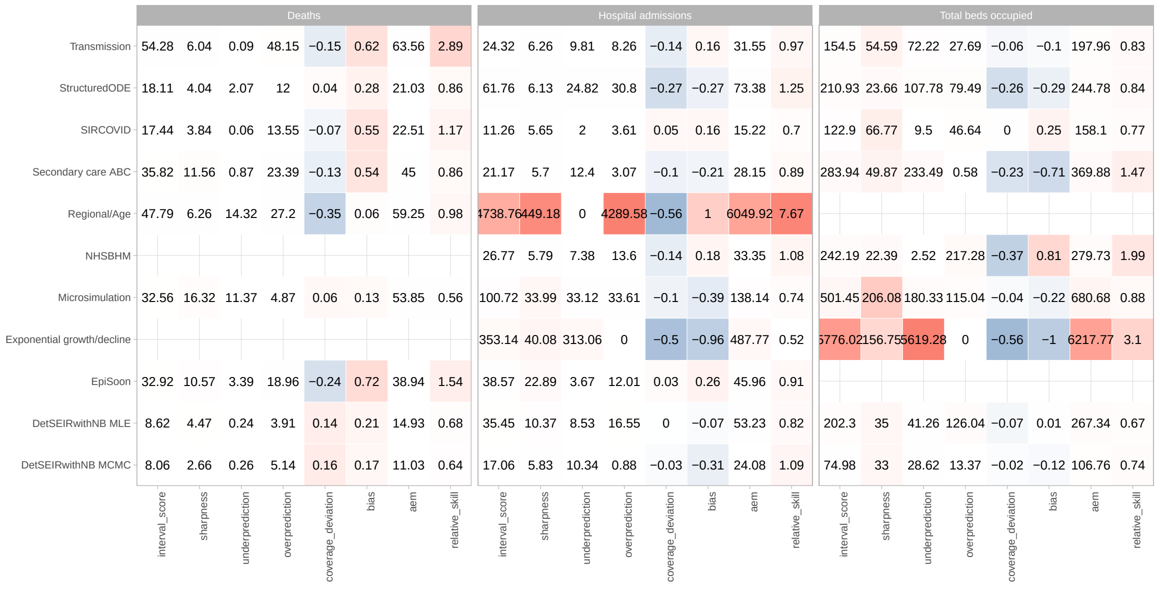
4. Visualisation and interpretation of evaluation results

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 1, 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

```
R> score_table(scores, y = "model", facet_formula = ~ value_desc)
```



Deaths

	interval_score	sharpness	underprediction	overprediction	coverage_deviation	bias	aem	relative_skill
Transmission	54.28	6.04	0.09	48.15	−0.15	0.62	63.56	2.89
StructuredODE	18.11	4.04	2.07	12	0.04	0.28	21.03	0.86
SIRCOVID	17.44	3.84	0.06	13.55	−0.07	0.55	22.51	1.17
Secondary care ABC	35.82	11.56	0.87	23.39	−0.13	0.54	45	0.86
Regional/Age	47.79	6.26	14.32	27.2	−0.35	0.06	59.25	0.98
NHSBHM								
Microsimulation	32.56	16.32	11.37	4.87	0.06	0.13	53.85	0.56
Exponential growth/decline								
EpiSoon	32.92	10.57	3.39	18.96	−0.24	0.72	38.94	1.54
DetSEIRwithNB MLE	8.62	4.47	0.24	3.91	0.14	0.21	14.93	0.68
DetSEIRwithNB MCMC	8.06	2.66	0.26	5.14	0.16	0.17	11.03	0.64

Hospital admissions

	interval_score	sharpness	underprediction	overprediction	coverage_deviation	bias	aem	relative_skill
Transmission	24.32	6.26	9.81	8.26	−0.14	0.16	31.55	0.97
StructuredODE	61.76	6.13	24.82	30.8	−0.27	−0.27	73.38	1.25
SIRCOVID	11.26	5.65	2	3.61	0.05	0.16	15.22	0.7
Secondary care ABC	21.17	5.7	12.4	3.07	−0.1	−0.21	28.15	0.89
Regional/Age	4738.76	449.18	0	4289.58	−0.56	1	6049.92	7.67
NHSBHM	26.77	5.79	7.38	13.6	−0.14	0.18	33.35	1.08
Microsimulation	100.72	33.99	33.12	33.61	−0.1	−0.39	138.14	0.74
Exponential growth/decline	353.14	40.08	313.06	0	−0.5	−0.96	487.77	0.52
EpiSoon	38.57	22.89	3.67	12.01	0.03	0.26	45.96	0.91
DetSEIRwithNB MLE	35.45	10.37	8.53	16.55	0	−0.07	53.23	0.82
DetSEIRwithNB MCMC	17.06	5.83	10.34	0.88	−0.03	−0.31	24.08	1.09

Total beds occupied

	interval_score	sharpness	underprediction	overprediction	coverage_deviation	bias	aem	relative_skill
Transmission	154.5	54.59	72.22	27.69	−0.06	−0.1	197.96	0.83
StructuredODE	210.93	23.66	107.78	79.49	−0.26	−0.29	244.78	0.84
SIRCOVID	122.9	66.77	9.5	46.64	0	0.25	158.1	0.77
Secondary care ABC	283.94	49.87	233.49	0.58	−0.23	−0.71	369.88	1.47
Regional/Age								
NHSBHM	242.19	22.39	2.52	217.28	−0.37	0.81	279.73	1.99
Microsimulation	501.45	206.08	180.33	115.04	−0.04	−0.22	680.68	0.88
Exponential growth/decline	5776.02	156.75	5619.28	0	−0.56	−1	6217.77	3.1
EpiSoon								
DetSEIRwithNB MLE	202.3	35	41.26	126.04	−0.07	0.01	267.34	0.67
DetSEIRwithNB MCMC	74.98	33	28.62	13.37	−0.02	−0.12	106.76	0.74

Figure 3: Coloured table to visualise the computed scores

by the fact that they forecasted different targets. The interval score, as an absolute metric, 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

we can also look at p-values in Figure 5 PROBABLY REMOVE THAT FROM THE PAPER

Both plots can be combined in one: PLOT NEEDS FIXING AND ALSO PROBABLY REMOVE THAT FROM THE PAPER? Figure 6

In terms of actually understanding *why* one model performs well or badly, the other metrics shown in Figure 3 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

```
R> plot_pairwise_comparison(pairwise) +
+    ggplot2::facet_wrap(~ value_desc, scales = "free_x")
```

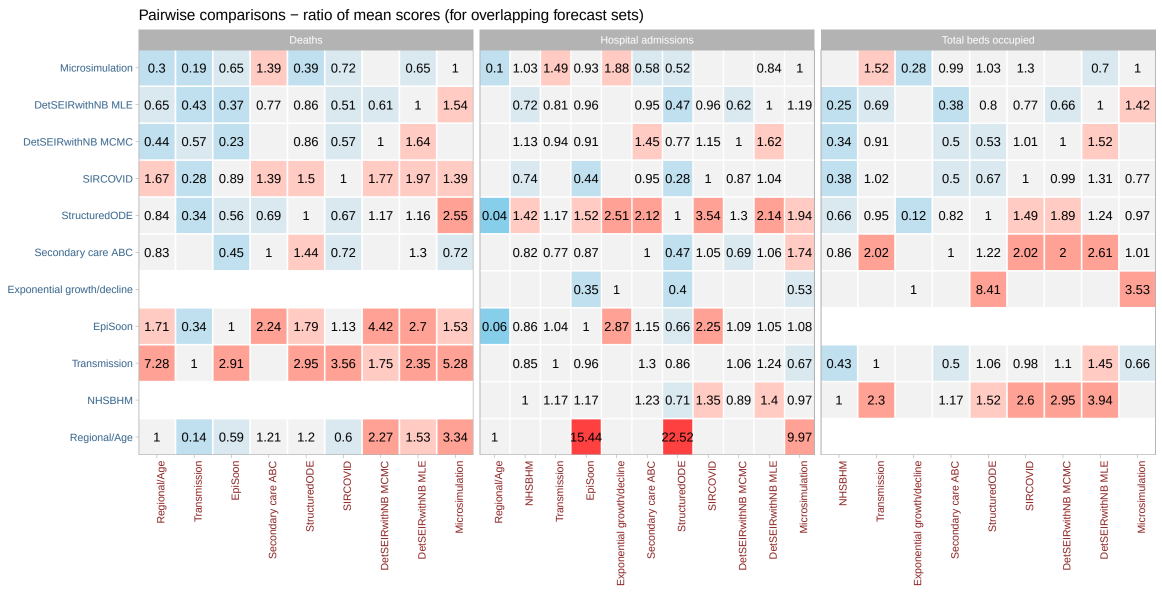


Figure 4: Ratios of mean scores based on overlapping forecast sets. If a tile is blue, then the model on the y-axis performed better. If it is red, the model on the x-axis performed better in direct comparison.

`quantile_coverage()`. The interval coverage plot shows the proportion of all true values that fall within all the different prediction intervals. This gives a visual impression of probabilistic calibration [GNEITING]. 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

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.

.

```
R> plot_pairwise_comparison(pairwise, type = "pval") +
+    ggplot2::facet_wrap(~ value_desc, scales = "free_x")
```

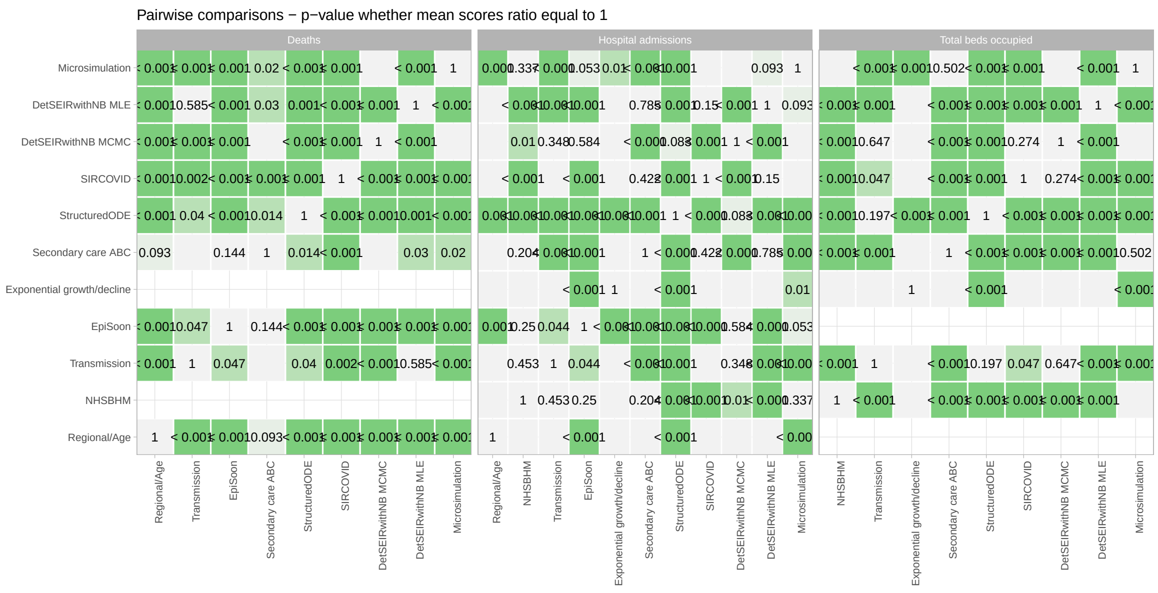


Figure 5: XXXX

```
R> plot_pairwise_comparison(pairwise, type = "together") +
+    ggplot2::facet_wrap(~ value_desc, scales = "free_x")
```

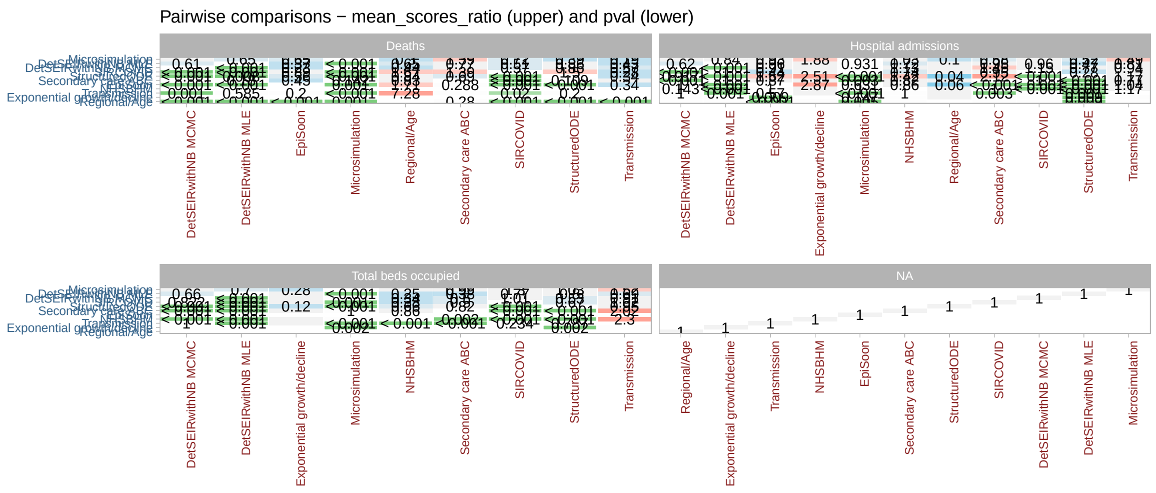


Figure 6: p-values and ratios together.

5. Summary and discussion

COMING SOON.

```
R> wis_components(scores,
+                 facet_formula = ~ value_desc,
+                 scales = "free_x") +
+    ggplot2::coord_flip() +
+    ggplot2::theme(legend.position = "bottom")
```

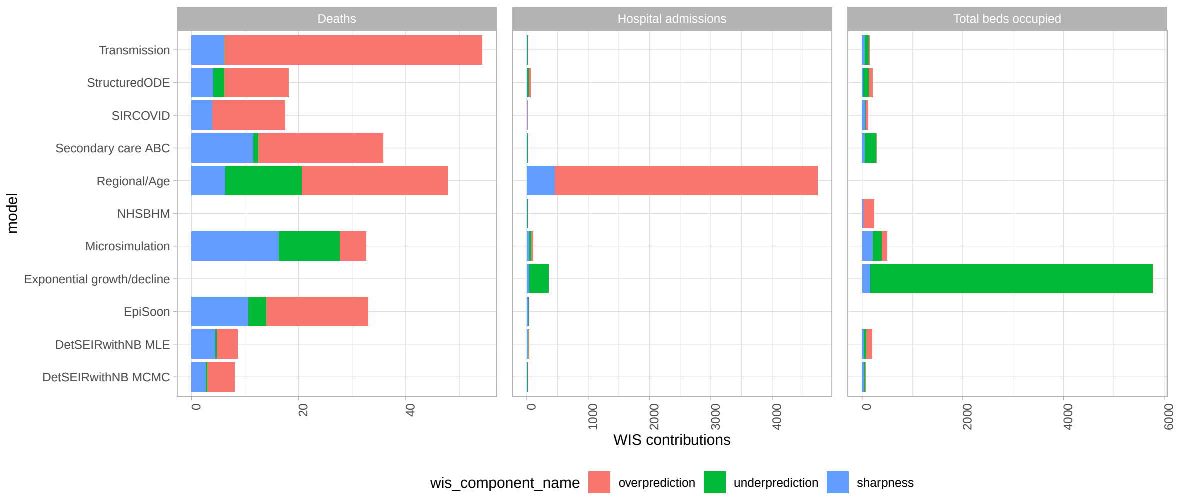


Figure 7: p-values and ratios together.

Acknowledgments

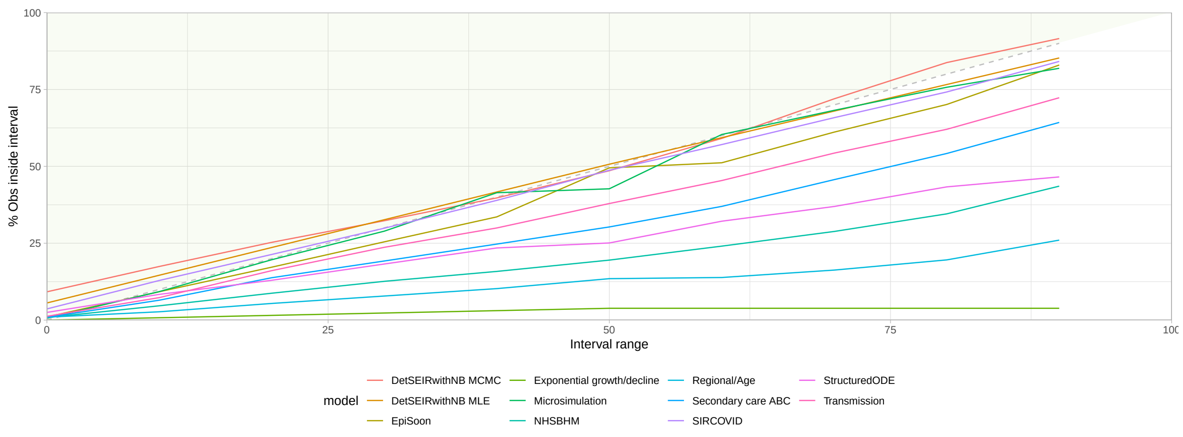
References

Bracher J, Ray EL, Gneiting T, Reich NG (2020). "Evaluating epidemic forecasts in an interval format." *arXiv:2005.12881 [q-bio, stat]*. ArXiv: 2005.12881, URL `http://arxiv.org/abs/2005.12881`.

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, Tang M, Tildesley MJ, Leeuwen EV, Whittles LK, Group CCW, Team ICCR, Investigators I (2020). "Short-term forecasts to inform the response to the Covid-19 epidemic in the UK." *medRxiv*, p. 2020.11.11.20220962. `doi:10.1101/2020.11.11.20220962`. Publisher: Cold Spring Harbor Laboratory Press, URL `https://www.medrxiv.org/content/10.1101/2020.11.11.20220962v1`.

Gneiting T, Balabdaoui F, Raftery AE (2007). "Probabilistic forecasts, calibration and sharpness." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*,

```
R> cov_scores <- eval_forecasts(combined,
+                                summarise_by = c("model",
+                                                 "range", "quantile"))
R> scoringutils::interval_coverage(cov_scores)
R> scoringutils::quantile_coverage(cov_scores)
```



```
R> cov_scores <- eval_forecasts(combined,
+                                summarise_by = c("model",
+                                                 "range", "quantile"))
R> scoringutils::quantile_coverage(cov_scores)
```

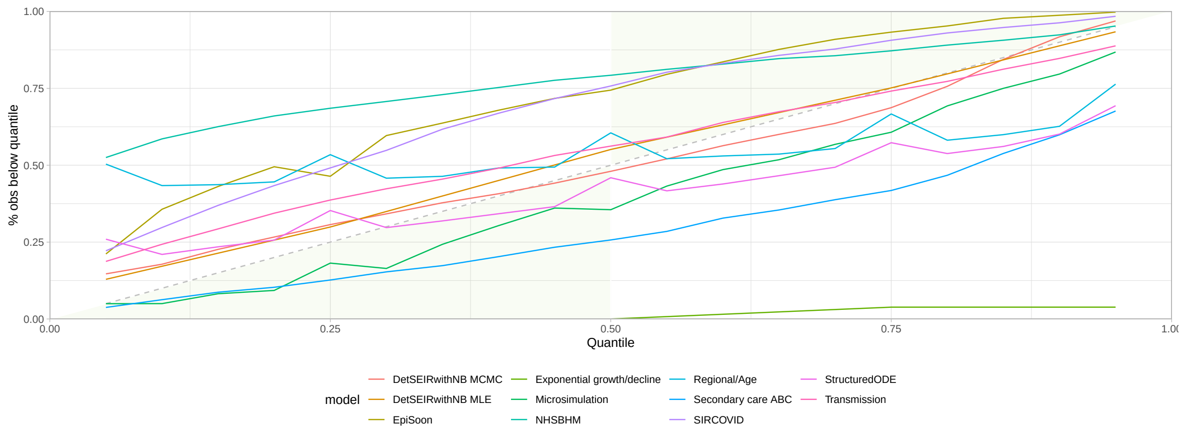


Figure 8: quantile and interval coverage

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 (2007). "Strictly Proper Scoring Rules, Prediction, and Estimation." *Journal of the American Statistical Association*, **102**(477), 359–378. ISSN 0162-1459, 1537-274X. doi:10.1198/016214506000001437. URL http://www.tandfonline.com/doi/abs/10.1198/016214506000001437.

```
R> scores <- eval_forecasts(combined,
+                           summarise_by = c("model",
+                                            "value_desc",
+                                            "geography"),
+                           compute_relative_skill = FALSE)
R> scoringutils::score_heatmap(scores, metric = "bias",
+                              x = "geography", facet_formula = ~ value_desc,
+                              scale = "free_x")
```

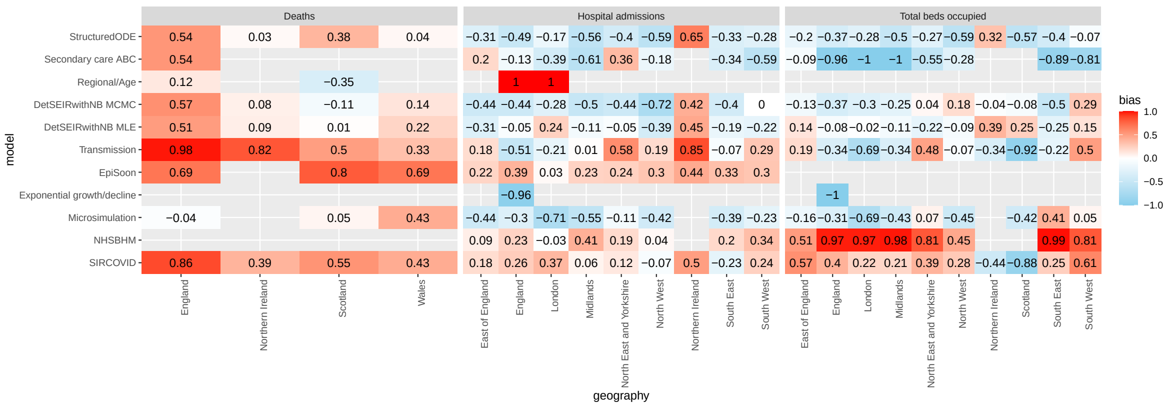


Figure 9: bias by location

A. Appendix section

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