Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

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 a role in many scientific fields from finance to meteorology and epidemiology. With the emergence of Covid-19 the role of forecasting to inform public policy has again attracted increased attention. In this paper we introduce a complete framework for evaluating different types of forecasts using the R package scoringutils. We discuss different appropriate evaluation metrics and apply the framework to XXXXX.

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

1. Introduction

Good forecasts are of great interest to decision makers in almost every field. 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 [some citations]. Yet even with this rich body of research available, implementing a consistent framework in R to evaluate forecasts is not trivial. We therefore present the **scoringutils** pacakge. 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.

1.1. Forecast types

In its most general sense, a forecast is the forecaster's stated belief about the future. For conceptual clarity, it is however useful to distinguish between different times of forecasts. Forecasts can be either quantitative or qualitative, but **scoringutils** only deals with the for-

mer. In terms of quantitative forecasts we can distinguish point forecasts from probabilistic forecasts. A point forecast states a single number and is the simplest form of a forecast. A point forecast is limited in its usefulness, as it does not give information about the uncertainty of the foreacst. 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 are usually 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, usually 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 [GneitingProbabilisticForecastsCal-

ibration2007]. 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 @gneiting-ProbabilisticForecastsCalibration2007], but is also less useful than a model that were able to accurately predict the weather every single day. Figure ref(fig:forecast-paradigm) illustrates the concepts of calibration and sharpness once again. DO I WANT THAT FIGURE?

2. Data: UK short-term forecasts

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?

PLOT WITH OVERVIEW OF MISSING FORECASTS LIKE THE ONE THEY MADE FOR THE FORECAST HUB.

3. Running an automated evaluation

A full evaluation of all forecasts based on observed values can be performed using the function `eval_forecasts()`. The function automatically recognises the prediction type and format and applies the appropriate scoring metrics. Internally, operations are handled using **data.table** to allow for fast and efficient computation.

All it requires is a `data.frame` or similar that has 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. To get a sense of the required input formats, example data for each format is provided with the package and shown in the package vignette. We also included functionality to transform between various formats. Additional columns may be present to indicate a grouping of forecasts. One common use case is a set of forecasts made by different models for various locations at different time points, each for several weeks into the future. In this case, we would expect additional columns that could be called "model", "date", "forecast_date", "forecast_horizon" and "location". Through the by argument, the user needs to to specify the unit of a single forecast. In the above example we would set by = c("model", "date", "forecast_date", "forecast_horizon", "location"). Columns such as quantile or sample should not be included, because several quantiles or samples make up one forecast. By default, by = NULL and scoringutils will automatically use all appropriate

present columns.

The `summarise_by` argument allows the user to choose categories to aggregate over. If we were only interested in scores for the different models, we would for example specify `summarise_by = c("model")`. If we wanted to have scores for every model in every location, we would specify `summarise_by = c("model", "location")`. 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")`.

When aggregating, `eval_forecasts` takes the mean according to the group defined in `summarise_by`. In the above example, if `summarise_by = c("model", "location")`, scores would be averaged over all forecast dates, forecast horizons and quantiles to yield one score per model and location. 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.

To evaluate the UK short term forecast data with scoringutils, 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> library(covid19.forecasts.uk)
R> # load truth data
R> data(covid_uk_data)
R> head(covid_uk_data)

# A tibble: 6 x 5
  value_date value_type    geography value_desc          value
  <date>     <fct>         <fct>     <fct>               <dbl>
1 2020-03-19 hospital_inc  England   Hospital admissions   425
2 2020-03-20 hospital_inc  England   Hospital admissions   651
3 2020-03-20 hospital_prev England   Total beds occupied  1559
4 2020-03-21 hospital_inc  England   Hospital admissions   703
5 2020-03-21 hospital_prev England   Total beds occupied  2104
6 2020-03-22 hospital_inc  England   Hospital admissions   702

R> # load forecasts
R> data(uk_forecasts)
R> head(uk_forecasts)

# A tibble: 6 x 8
  model geography value_type creation_date value_date quantile value
  <fct> <chr>     <fct>      <date>        <date>        <dbl> <dbl>
1 EpiS... East of ... hospital_... 2020-03-31    2020-04-01     0.05    70
2 EpiS... East of ... hospital_... 2020-03-31    2020-04-02     0.05    64
3 EpiS... East of ... hospital_... 2020-03-31    2020-04-03     0.05    53
4 EpiS... East of ... hospital_... 2020-03-31    2020-04-04     0.05    42
5 EpiS... East of ... hospital_... 2020-03-31    2020-04-05     0.05    30
6 EpiS... East of ... hospital_... 2020-03-31    2020-04-06     0.05    19
# ... with 1 more variable: value_desc <fct>
```

We also need to make some some minor changes to the data. Both data sets need to be joined and variable names for predictions and forecasts need to be changed. This can be achieved using the following commands:

```
R> combined <- dplyr::inner_join(uk_forecasts %>%
+                                  dplyr::rename(prediction = value),
+                                covid_uk_data %>%
+                                  dplyr::rename(true_value = value))
```

MAYBE USE DATA.TABLE INSTEAD OF DPLYR IN ORDER TO AVOID MIXING? ALSO MAYBE JUST IMPLEMENT THE JOINING WITHIN SCORINGUTILS DIRECTLY

The scoringutils package has some built-in functionality for visualisation (also see section [visualisation]). In order to plot predictions we are interested in, we could call

```
R> example_subset <- combined %>%
+    dplyr::filter(model == "SIRCOVID",
+                  creation_date == "2020-06-22")
R> additional_observations <- covid_uk_data %>%
+    dplyr::rename(true_value = value) %>%
+    dplyr::filter(value_date <= "2020-06-22",
+                  value_date > "2020-06-01",
+                  value_type %in% unique(example_subset$value_type))
R> scoringutils::plot_predictions(example_subset,
+                                  additional_observations,
+                                  x = "value_date",
+                                  facet_formula = geography ~ value_type)
```

The output is shown in Figure 1.

MAYBE CHANGE THE SCORINGUTILS CODE TO MAKE THAT A BIT SLICKER. FOR EXAMPLE: HAVE THE ENTIRE DATA SET AND RESTRICT FORECASTS TO EVERYTHING AFTER A GIVEN DATE.

An evaluation that returns scores for every model and every prediction target can now be performed by simply calling

```
R> library(scoringutils)
R> scores <- eval_forecasts(combined,
+                           summarise_by = c("model", "value_type"))
R> dplyr::glimpse(scores)

Rows: 35
Columns: 11
$ model              <fct> SIRCOVID, SIRCOVID, SIRCOVID, SIRCOVID, ...
$ value_type         <fct> hospital_inc, hospital_prev, death_inc_l...
$ interval_score     <dbl> 13.022238, 137.574205, 8.845734, 5.42272...
$ sharpness          <dbl> 5.105571, 70.048708, 3.647750, 1.586862,...
$ underprediction    <dbl> 1.589048e+00, 1.068817e+01, 7.949136e-01...
```

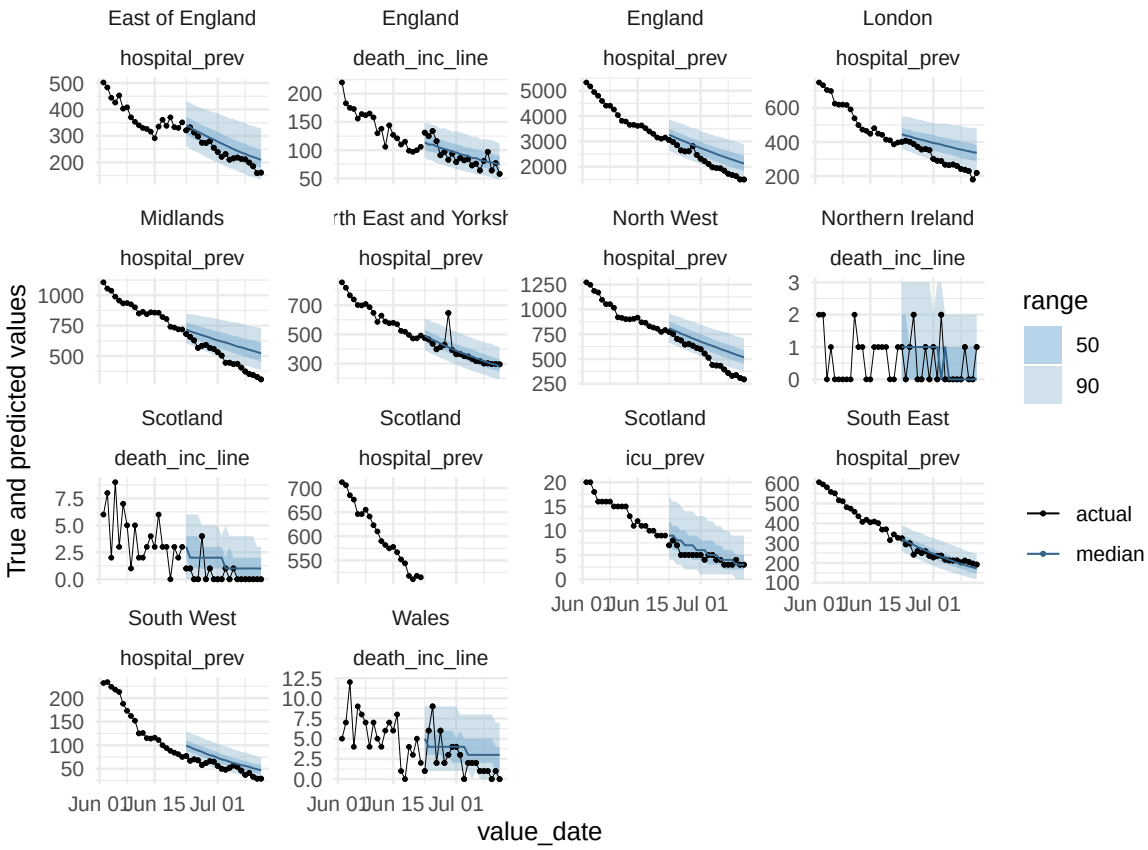

Figure 1: Short-term forecasts made by the SIRCOVID model on June 22 2020.

```
$ overprediction     <dbl> 6.327619, 56.837324, 4.403071, 3.798621,...
$ coverage           <dbl> 0.4219048, 0.4527387, 0.4642994, 0.46620...
$ coverage_deviation <dbl> -0.028095238, 0.002738730, 0.014299424, ...
$ bias               <dbl> 0.2971428571, 0.3348521571, 0.3913627639...
$ aem                <dbl> 17.166667, 174.513330, 12.309981, 7.3172...
$ quantile_coverage  <dbl> 0.66333333, 0.66992244, 0.79510557, 0.79...
```

Pairwise comparisons between models [CITATION] can be made using

```
R> pairwise <- pairwise_comparison(scores)
```

This will be explained in more detail in section [visualisation and interpretation of the evaluation].

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. 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.

4.1. 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?

4.2. 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:

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

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.

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 described in the following section.

5. Visualisation and interpretation of evaluation results

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:

```
R> scoringutils::score_table(scores, y = "model", facet_formula = ~ value_type)
```

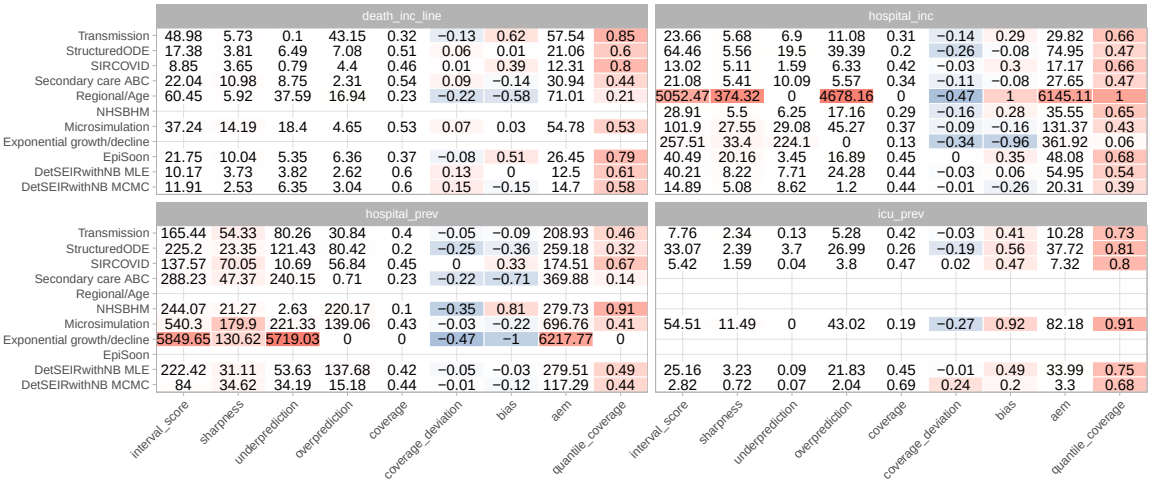

Figure 2: Coloured table to visualise the computed scores

If all we care about is which model performed best, the pairwise comparison is most suitable. This approach compares all possible pairs of models based on an overlapping set of forecasts.

.

6. Summary and discussion

```
R> scoringutils::plot_pairwise_comparison(pairwise)
```

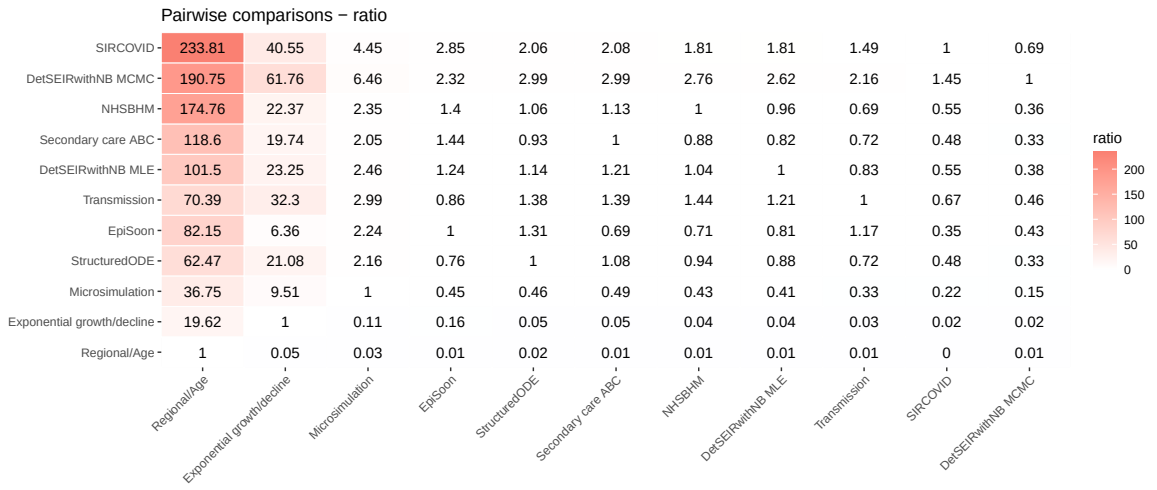

Figure 3: XXXX

Acknowledgments

References

Cameron AC, Trivedi PK (2013). *Regression Analysis of Count Data*. 2nd edition. Cambridge University Press, Cambridge.

Chambers JM, Hastie TJ (eds.) (1992). *Statistical Models in S*. Chapman & Hall, London.

Jackman S (2015). ***pscl**: Classes and Methods for R Developed in the Political Science Computational Laboratory, Stanford University*. R package version 1.4.9, URL https://CRAN.R-project.org/package=pscl.

McCullagh P, Nelder JA (1989). *Generalized Linear Models*. 2nd edition. Chapman & Hall, London. doi:10.1007/978-1-4899-3242-6.

Mullahy J (1986). "Specification and Testing of Some Modified Count Data Models." *Journal of Econometrics*, **33**(3), 341–365. doi:10.1016/0304-4076(86)90002-3.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Stasinopoulos DM, Rigby RA (2007). "Generalized Additive Models for Location Scale and Shape (GAMLSS) in R." *Journal of Statistical Software*, **23**(7), 1–46. doi:10.18637/jss.v023.i07.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York. doi:10.1007/978-0-387-21706-2.

Wood SN (2006). *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC, Boca Raton.

Yee TW (2010). "The **VGAM** Package for Categorical Data Analysis." *Journal of Statistical Software*, **32**(10), 1–34. doi:10.18637/jss.v032.i10.

Zeileis A, Kleiber C, Jackman S (2008). "Regression Models for Count Data in R." *Journal of Statistical Software*, **27**(8), 1–25. doi:10.18637/jss.v027.i08.

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

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd