Journal of Statistical Software

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

Evaluating Forecasts with scoringutils in R

Nikos I. Bosse
London School of Hygiene & Tropical Medicine (LSHTM)

Hugo Gruson
LSHTM

Anne Cori
Imperial College London

Edwin van Leeuwen
UK Health Security Agency, LSHTM

Sebastian Funk
LSHTM

Sam Abbott
LSHTM

Abstract

Evaluating forecasts is essential in order to understand and improve forecasting and make forecasts useful to decision-makers. Much theoretical work has been done on the development of proper scoring rules and other scoring metrics that can help evaluate forecasts. In practice, however, conducting a forecast evaluation and comparison of different forecasters remains challenging. In this paper we introduce **scoringutils**, an R package that aims to greatly facilitate this process. It is especially geared towards comparing multiple forecasters, regardless of how forecasts were created, and visualising results. The package is able to handle missing forecasts and is the first R package to offer extensive support for forecasts represented through predictive quantiles, a format used by several collaborative ensemble forecasting efforts. The paper gives a short introduction to forecast evaluation, discusses the metrics implemented in **scoringutils** and gives guidance on when they are appropriate to use, and illustrates the application of the package using example data of forecasts for COVID-19 cases and deaths submitted to the European Forecast Hub between May and September 2021.

Keywords: forecasting, forecast evaluation, proper scoring rules, scoring, R.

1. Introduction

Good forecasts are of great interest to decision makers in various fields like finance (Timmermann 2018; Elliott and Timmermann 2016), weather predictions (Gneiting and Raftery 2005;

Kukkonen *et al.* 2012) or infectious disease modelling (Reich *et al.* 2019; Funk *et al.* 2020; Cramer *et al.* 2021; Bracher *et al.* 2021b; European Covid-19 Forecast Hub 2021). Throughout the COVID-19 pandemic, forecasting has garnered widespread interest from policy makers and the general public, with several collaborative forecasting efforts ("Forecast Hubs") being established (Reich *et al.* 2019; Cramer *et al.* 2020; European Covid-19 Forecast Hub 2021; **?**). Forecast evaluation is an integral part of assessing and improving the usefulness of forecasts. For decades, researchers, especially in the field of weather forecasting, have therefore developed and refined an arsenal of techniques to evaluate predictions (see for example Good (1952), Epstein (1969); Murphy (1971); Matheson and Winkler (1976), Gneiting, Balabdaoui, and Raftery (2007), Funk, Camacho, Kucharski, Lowe, Eggo, and Edmunds (2019), Gneiting and Raftery (2007), Bracher, Ray, Gneiting, and Reich (2021a)).

Various R (R Core Team 2021) packages cover a wide variety of scoring rules, plots and other metrics that are useful in assessing the quality of a forecast. Existing packages are well suited to evaluate the quality of a single forecast or to compare the performance of several models on a single target variable (we'll be using the terms "model" and "forecaster" interchangeably). One particular challenge, however, is dealing with multidemsionality: evaluating and comparing forecasts from several models across multiple dimensions such as time, space, and different types of targets.

Packages such as **scoringRules** (Jordan, Krüger, and Lerch 2019), **Metrics** (Hamner and Frasco 2018), **MLmetrics** (Yan 2016), **verification** (Laboratory 2015), or **SpecsVerification** (Siegert 2020), which provide an extensive collection of scoring rules and metrics, usually operate on vectors and matrices. Applying them to multiple forecasts across several dimensions can therefore be cumbersome. Some packages like **scoring** (Merkle and Steyvers 2013) operate on a data.frame and use a formula interface, making this task easier. However, **scoring** only exports a few scoring rules and does not provide a general interface or framework that would allow users to conduct forecast evaluations in which they can apply arbitrary scoring rules to data.

Some packages such as **tscount** (Liboschik, Fokianos, and Fried 2017), **topmodels** (Zeileis and Lang 2022), **GLMMadaptive** (Rizopoulos 2023) or **cvGEE** (Rizopoulos 2019) provide useful scoring metrics and plots, but those are only accessible if the forecasts were generated in a particular way. **tscount**, for example, requires an object of class `tsglm`, **topmodels** requires an object of class `lm`, `glm`, `crch` or `disttree`, **GLMMadaptive** requires an object of class `MixMod` and **cvGEE** requires an object of class `geeglm`. These tools are by their nature not generally applicable to all use cases practitioners might encounter.

yardstick (Kuhn, Vaughan, and Hvitfeldt 2023a), which builds on the **tidymodels** (Kuhn and Wickham 2020) framework, is the most general other forecast evaluation package. It allows users to apply arbitrary scoring rules to a data.frame of forecasts, regardless of how they were created. The frameworks is flexible and easily extendable, and also addresses the issue of scoring forecasts across various dimensions. However, **yardstick** is primarily focused on point forecasts and classification tasks. It currently lacks general support for probabilistic forecasts (forecasts in the form of a full predictive distribution, represented e.g. by a set of quantiles or samples from the forecast distribution). Probabilistic forecasts are desirable, as they allow decision makers to take into account the uncertainty of a forecast (Gneiting *et al.* 2007), and are widely used, e.g. in Meteorology or Epidemiology. **fabletools** (O'Hara-Wild, Hyndman, and Wang 2023) offers some functionality to evaluate probabilistic forecasts, but is not fully general as scoring is tied to specific object classes and users cannot easily supply their own scoring rules.

scoringutils aims to fill the gap in the ecosystem by providing a general-purpose tools as well as a framework for the evaluation of (probabilistic) forecasts across multiple dimensions using wide variety of user-provided scoring rules. Notably, **scoringutils** is the first package to offer extensive support for probabilistic forecasts in the form of predictive quantiles, a format that is currently used by several infectious disease Forecast Hubs (Reich *et al.* 2019; Cramer *et al.* 2020; European Covid-19 Forecast Hub 2021; **?**). The package provides wide functionality to check the data and diagnose issues, to visualise forecasts and missing data, to transform data before scoring (Bosse, Abbott, Cori, van Leeuwen, Bracher, and Funk 2023), to apply scoring rules to data, to handle missing forecasts, to aggregate scores and to visualise the results of the evaluation. **scoringutils** makes extensive use of **data.table** (Dowle and Srinivasan 2023) to ensure fast and memory-efficient computations. The package aims to be flexible and extendable, providing multiple generics and methods that users can expand on.

Many practitioners who develop models are not experts in forecast evaluation and do not have a deep understanding of the various scoring rules or know which one to apply. **scoringutils** aims to make the task of evaluating forecasts more accessible to inexperienced users by providing sensible defaults in addition to extensive function documentation. Explanations of the default metrics, as well as vignettes and case studies help users to understand how to use the package and interpret results.

Currently, there exists a large number of packages with partly overlapping functionality. In addition to the already mentioned packages, **surveillance** (Meyer, Held, and Höhle 2017), **predtools** (Sadatsafavi, Safari, and Lee 2023), **probably** (Kuhn, Vaughan, and Ruiz 2023b) are notable for the scoring rules, plots, and tools they provide to verify and handle forecasts. For inexperienced practitioners, the large number of packages and functions can be overwhelming. **scoringutils** aims to bridge the gap between existing packages and to make it easier to work with different packages in the ecosystem a coherent way. It does this firstly by providing extensive vignettes and case studies that illustrate how functions from different packages can be used together. Secondly, it provides helper functions to convert to and from the formats used in different packages, where those are not immediately compatible, making it easy to use different packages in a single workflow. In the future, we also aim to provide interfaces to common modelling packages such as **odin** in order to allow for a closer integration of modelling and evaluation.

scoringutils currently does not handle all possible kinds of prediction formats supported by other packages. For example, it does not support multiclass predictions, or scoring forecasts that are specified as a closed-form distribution. However, we plan to add support for these to the package in the near future, as well as support for scoring multivariate forecasts that specify a joint distribution across targets.

The remainder of this section will provide an overview of the fundamental ideas behind forecast evaluation. Section **??** will give a detailed theoretical explanation of the evaluation metrics in **scoringutils** and when to use them. Section **??** will demonstrate how to conduct an evaluation in **scoringutils** using forecasts of COVID-19 submitted to the European Forecast Hub (European Covid-19 Forecast Hub 2021) as a case study. In the following we will use the words "model" and "forecaster" interchangeably, regardless of how forecasts were actually generated.

2. Package overview

The following section gives an overview of the main functionality of **scoringutils**. Everything will be illustrated using the example data. An overview of all existing functions can be seen on `https://epiforecasts.io/scoringutils`.

2.1. Basic workflow

scoringutils supports two basic workflows (see Figure **??**). One option is for users to score their forecasts directly using the scoring rules exported by **scoringutils** in a format based on vectors and matrices. Those functions that can be called directly on a set of vectors and matrices and return a single vector of scores will consistently be called "scoring rules" in the following. Most users, however, will interact with **scoringutils** through a more convenient framework based on a `data.frame`-like structure (internally, **data.table** (Dowle and Srinivasan 2023) is used).

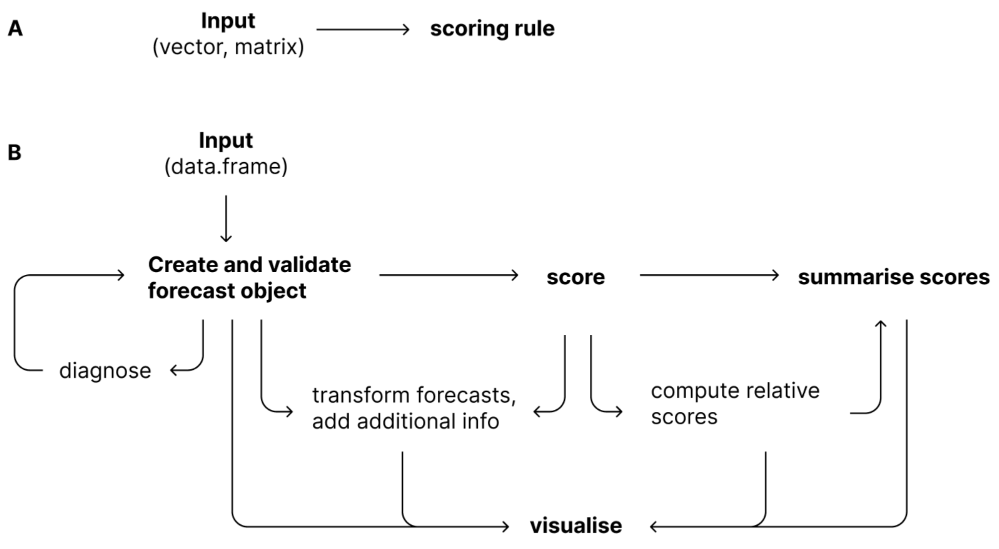


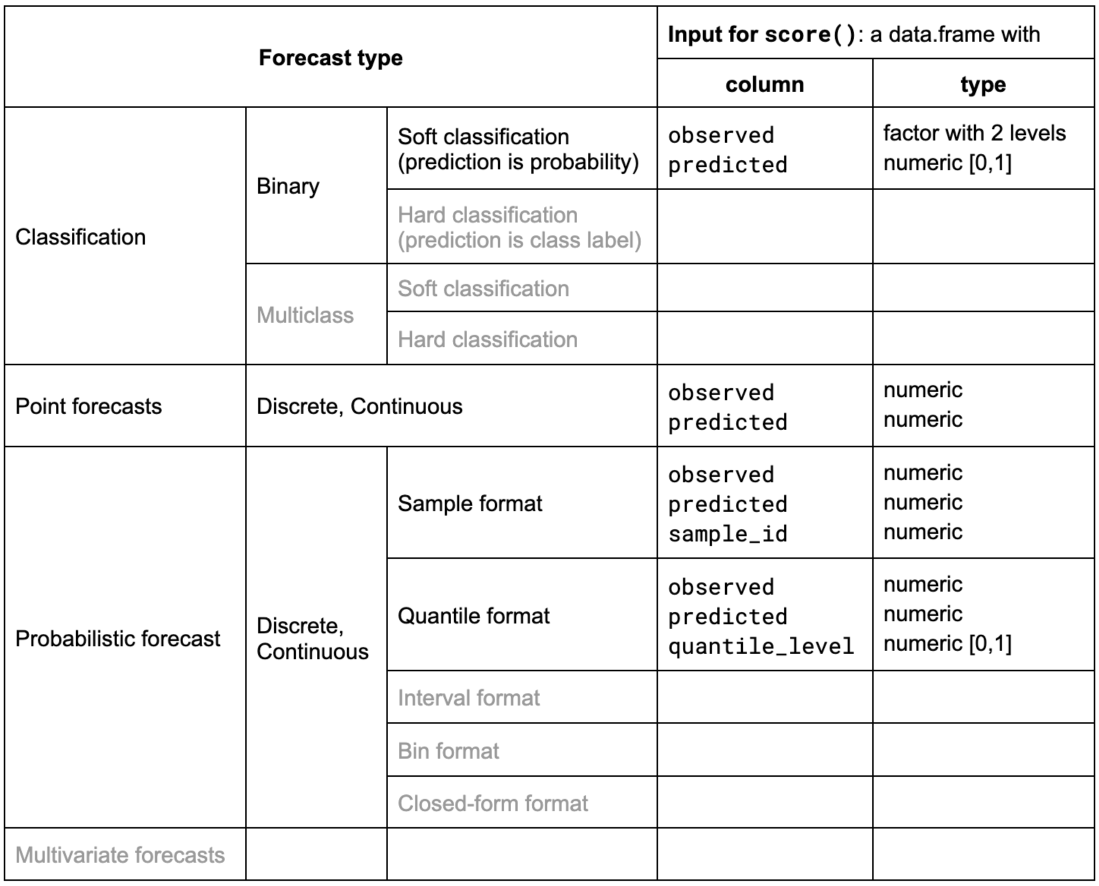
Figure 1: Illustration of the workflow for working with scoringutils.

2.2. Forecast types

Forecasts differ in the exact prediction task and in how the forecaster chooses to represent their prediction. To distinguish different kinds of forecasts, `scoringutils` uses the term "forecast type" (which is more a convenient classification than a formal definition). At the moment, `scoringutils` distinguishes four different forecast types: "binary", "point", "quantile" and "sample" forecasts (support for more forecast types is planned for the future).

"Binary" denotes a probability forecast for a binary (yes/no) outcome variable. This is sometimes also called "soft binary classification". "Point" denotes a forecast for a continuous or discrete outcome variable that is represented by a single number. "Quantile" or "quantile-based" is used to denote a probabilistic forecast for a continuous or discrete outcome variable, with the forecast distribution represented by a set of predictive quantiles. While a single quantile would already satisfy the requirements for a quantile-based forecast, most scoring rules expect a set of quantiles which are symmetric around the median (thus forming the lower and upper bounds of central “prediction intervals”) and will return `NA` if this is not the case. “Sample” or “sample-based” is used to denote a probabilistic forecast for a continuous or discrete outcome variable, with the forecast represented by a finite set of samples drawn from the predictive distribution. A single sample technically suffices, but would to very imprecise results.

Forecast types (in the `data.frame` framework) are determined based on the names and the type of the columns present in the input. Table 2 shows the expected input format for each forecast type. Forecast types that are planned, but not currently supported, are greyed out. Input formats for the scoring rules that can be called directly follow the same convention, with inputs expected to be vectors or matrices.



Forecast type			**Input for `score()`**: a data.frame with	
			column	**type**
Classification	Binary	Soft classification (prediction is probability)	`observed` `predicted`	factor with 2 levels numeric [0,1]
		Hard classification (prediction is class label)		
	Multiclass	Soft classification		
		Hard classification		
Point forecasts	Discrete, Continuous		`observed` `predicted`	numeric numeric

Probabilistic forecast	Discrete, Continuous	Sample format	`observed`
			`predicted`
			`sample_id`
				numeric
				numeric
				numeric

		Quantile format	`observed`
			`predicted`
			`quantile_level`
				numeric
				numeric
				numeric [0,1]

		Interval format		
		Bin format		
		Closed-form format		
Multivariate forecasts				

Figure 2: Table with different input formats.

2.3. Classes

scoringutils uses separate classes that correspond to the different forecast types. Those are currently `forecast_binary`, `forecast_point`, `forecast_quantile`, and `forecast_sample`.

These classes are used to allow for dedicated input checks and sensible defaults for the scoring rules to be applied to the forecasts.

They come with a constructor, `new_forecast()`, a generic validator, `validate_forecast()` (which dispatches to a specialised validator method depending on the class of the input), and a convenient wrapper function `as_forecast()`. `as_forecast()` determines the forecast type of the input, constructs the class and validates the input. The process is illustrated in Figure 3. All classes also have a `print()` method designed to provide helpful information to the user.

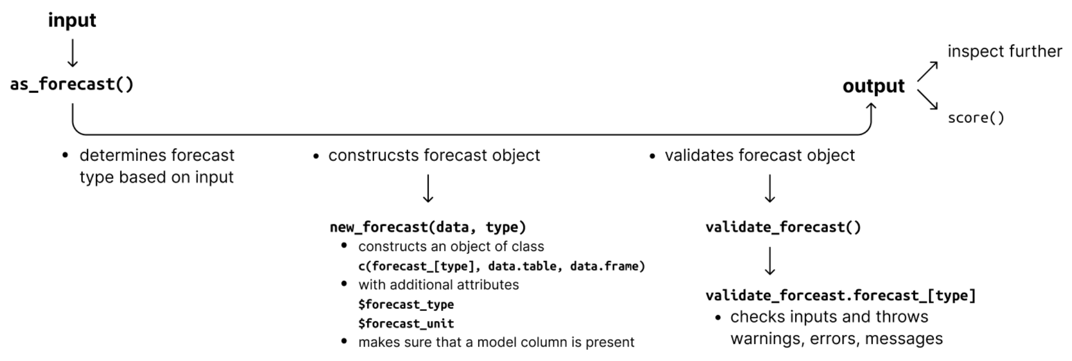

input
as_forecast()
output
inspect further
score()
• determines forecast type based on input
• construcsts forecast object
• validates forecast object
new_forecast(data, type)
• constructs an object of class c(forecast_[type], data.table, data.frame)
• with additional attributes $forecast_type $forecast_unit
• makes sure that a model column is present
validate_forecast()
validate_forceast.forecast_[type]
• checks inputs and throws warnings, errors, messages

Figure 3: Illustration of the process of creating a ‘forecast‘ object.

2.4. Example data

The example data included in the package and used in this paper consists of one to three week ahead forecasts made between May and September 2021 for COVID-19 cases and deaths from four different forecasting models. It represents a small subset of short-term predictions for COVID-19 cases and deaths submitted to the European Forecast Hub (European Covid-19 Forecast Hub 2021). One to four week ahead predictions of different COVID-19 related targets were submitted to the Forecast Hub each week in a quantile-based format. The full official hub evaluations, which also use **scoringutils**, can be seen at `https://covid19forecasthub.eu/`. The example data was converted to all the different forecast types used in **scoringutils**, in order to illustrate the workflows and provide users with a reference for the correct input formats. The example data also contains observations without corresponding forecasts, both for the purposes of plotting the data and to make the example data more realistic. The stored data sets are called `example_quantile`, `example_continuous`, `example_integer`, `example_point` and `example_binary`.

Here is the output of validating the example data:

```
R> library(scoringutils)
R> print(example_quantile, 2)
```

```
     location target_end_date target_type observed location_name
  1:       DE      2021-01-02       Cases   127300       Germany
  2:       DE      2021-01-02      Deaths     4534       Germany
```

```
   ---
20544:       IT      2021-07-24      Deaths       78          Italy
20545:       IT      2021-07-24      Deaths       78          Italy
       forecast_date quantile predicted                model horizon
    1:          <NA>       NA        NA                 <NA>      NA
    2:          <NA>       NA        NA                 <NA>      NA
   ---
20544:    2021-07-12    0.975       611 epiforecasts-EpiNow2       2
20545:    2021-07-12    0.990       719 epiforecasts-EpiNow2       2
```

```
R> validated <- validate(example_quantile)
R> class(validated)
```

```
[1] "scoringutils_quantile" "data.table"
[3] "data.frame"
```

2.5. Diagnostic functions that provide additional information about the data

In addition to printing the validated objects, users can call a variety of different functions to obtain more information about the data and to visualise it. Functions that provide this kind of additional information usually are named starting with `get_`.

`get_forecast_type()` infers the forecast type from a `data.frame` and returns a single string (one of "binary", "point", "sample" or "quantile"). `get_forecast_unit()` returns a vector with the names of the columns that uniquely define a single forecast (see Section 3.4 below for more information).

`get_forecast_counts()` returns forecast counts, which is helpful to obtain an overview of missing forecasts. This can impact the evaluation, if missingness correlates with performance. Users can specify the level of summary through the `by` argument. For example, to see how many forecasts there are per model and target_type, we can run

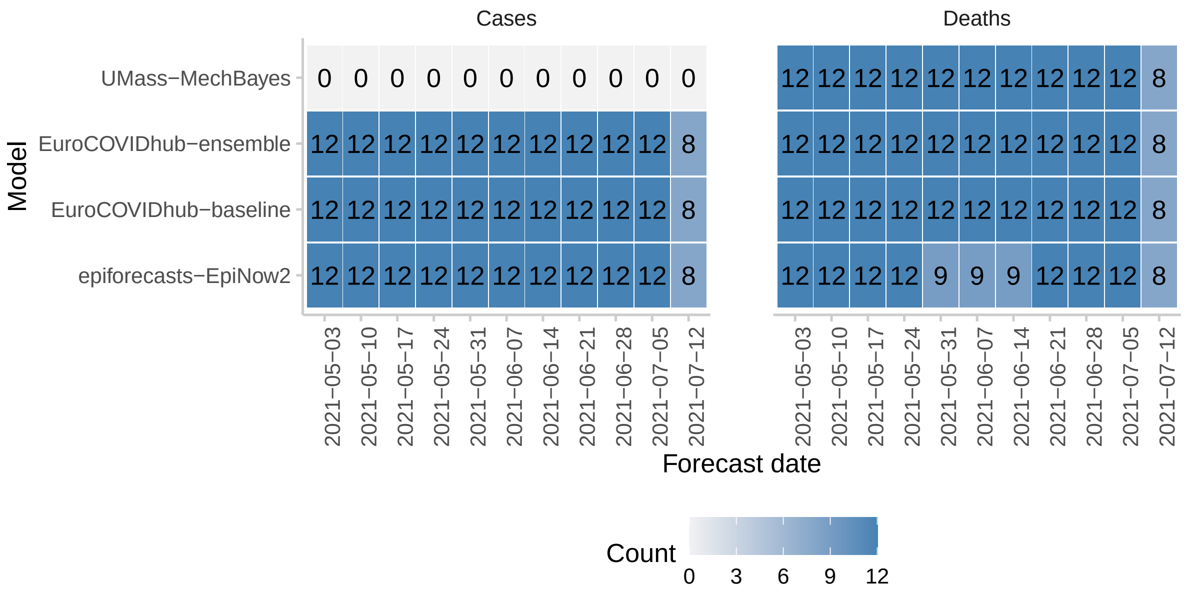
```
R> forecast_counts <- available_forecasts(
+   example_quantile, by = c("model", "target_type", "forecast_date")
+ )
```

This returns an object of class `forecast_counts`. We can visualise the results by calling `plot()` on the object (Figure 4).

```
R> library(ggplot2)
R> plot(forecast_counts, xvar = "forecast_date") +
+   facet_wrap(~ target_type) +
+   labs (y = "Model", x = "Forecast date")
```

2.6. Visualising the data

Work in progress



	Cases 2021−05−03	2021−05−10	2021−05−17	2021−05−24	2021−05−31	2021−06−07	2021−06−14	2021−06−21	2021−06−28	2021−07−05	2021−07−12	Deaths 2021−05−03	2021−05−10	2021−05−17	2021−05−24	2021−05−31	2021−06−07	2021−06−14	2021−06−21	2021−06−28	2021−07−05	2021−07−12
UMass−MechBayes	0	0	0	0	0	0	0	0	0	0	0	12	12	12	12	12	12	12	12	12	12	8
EuroCOVIDhub−ensemble	12	12	12	12	12	12	12	12	12	12	8	12	12	12	12	12	12	12	12	12	12	8
EuroCOVIDhub−baseline	12	12	12	12	12	12	12	12	12	12	8	12	12	12	12	12	12	12	12	12	12	8
epiforecasts−EpiNow2	12	12	12	12	12	12	12	12	12	12	8	12	12	12	12	9	9	9	12	12	12	8

Figure 4: Forecast counts for the example data.

```
R> example_quantile %>%
+   make_na(what = "truth",
+           target_end_date > "2021-07-15",
+           target_end_date <= "2021-05-22") %>%
+   make_na(what = "forecast",
+           model != "EuroCOVIDhub-ensemble",
+           forecast_date != "2021-06-28") %>%
+   plot_predictions(x = "target_end_date", by = c("target_type", "location")) +
+   aes(colour = model, fill = model) +
+   facet_wrap(target_type ~ location, ncol = 4, scales = "free_y") +
+   labs(x = "Target end date")
```

3. Scoring forecasts

scoringutils offers two ways of scoring forecasts: Users can either call different scoring rules directly on vectors and matrices or use the function `score()` on a data.frame (or similar) to apply multiple scoring rules at once.

3.1. score()

The `score()` is the workhorse of the package. It takes as input either an object of class `forecast_*` or a `data.frame` (or similar) with forecasts, as well as a list of functions (the scoring rules). The function then applies those scoring rules to the input. Additional arguments can be passed down to the scoring rules via `...`. `score()` is a generic function that dispatches to different methods depending on the class of the input.

`score.default()` is the default method that is used if `as_forecast()` has not yet been called on the input. `score.default()` calls `as_forecast()` and then calls `score()` again in order to dispatch to the appropriate method. The method then validates the input again, applies the scoring rules, and returns a `data.table` with the scores. The process is illustrated in Figure 5.

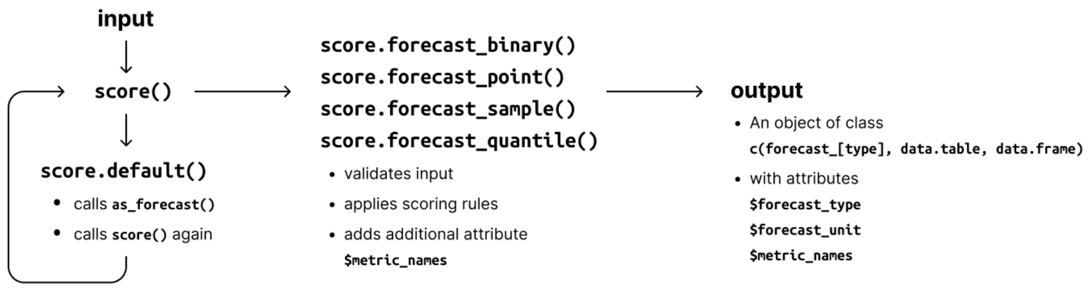


Figure 5: Flowchart for calling ‘score()‘

3.2. Scoring rules

Scoring rules are the functions the various scores and metrics. They are consistently named `name of the metric` + _ + `forecast type`. The return value is a vector with scores (only in the case of `wis()` is there an optional argument that causes the function to return a list of vectors). The first argument of a scoring rule is always `observed`, and the second one is `predicted`. Scoring rules for quantile-based arguments require an additional vector that denotes the quantile levels of the predictive quantiles.

Scoring rules differ in the relationship between input and output. Some scoring rules have a one-to-one relationship between prediction and score, returning one value per value in `predicted`. This is the case for all scoring rules for binary and point forecasts. Other scoring rules have a many-to-one relationship, returning one value per multiple values in `predicted`. This is the case for all scoring rules for sample- and quantile-based forecasts. For sample- and quantile-based forecasts, `predicted` is therefore a matrix, with values in each row jointly forming a single forecast.

Input formats and return values are shown in more detail in Figure **??**.

3.3. Passing scoring rules to score()

The second argument to `score()` is a named list of scoring rules. Names of the list item will be used as names of the columns in the output corresponding to the respective scoring rule. The default list for every forecast type can be accessed by calling `metrics_binary()`, `metrics_point()`, `metrics_sample()` and `metrics_quantile()`.

Scoring rules need to be compatible with the type of the forecast that is scored and need to have compatible input formats and return values. Within `score()`, arguments are passed by position, meaning that users are not tied to the **scoringutils** naming convention, and can supply functions that take equivalent, but differently called arguments. The default metrics for point forecasts, for example, uses functions from the **Metrics** package, which use the names `actual` and `predicted` instead of `observed` and `predicted`.

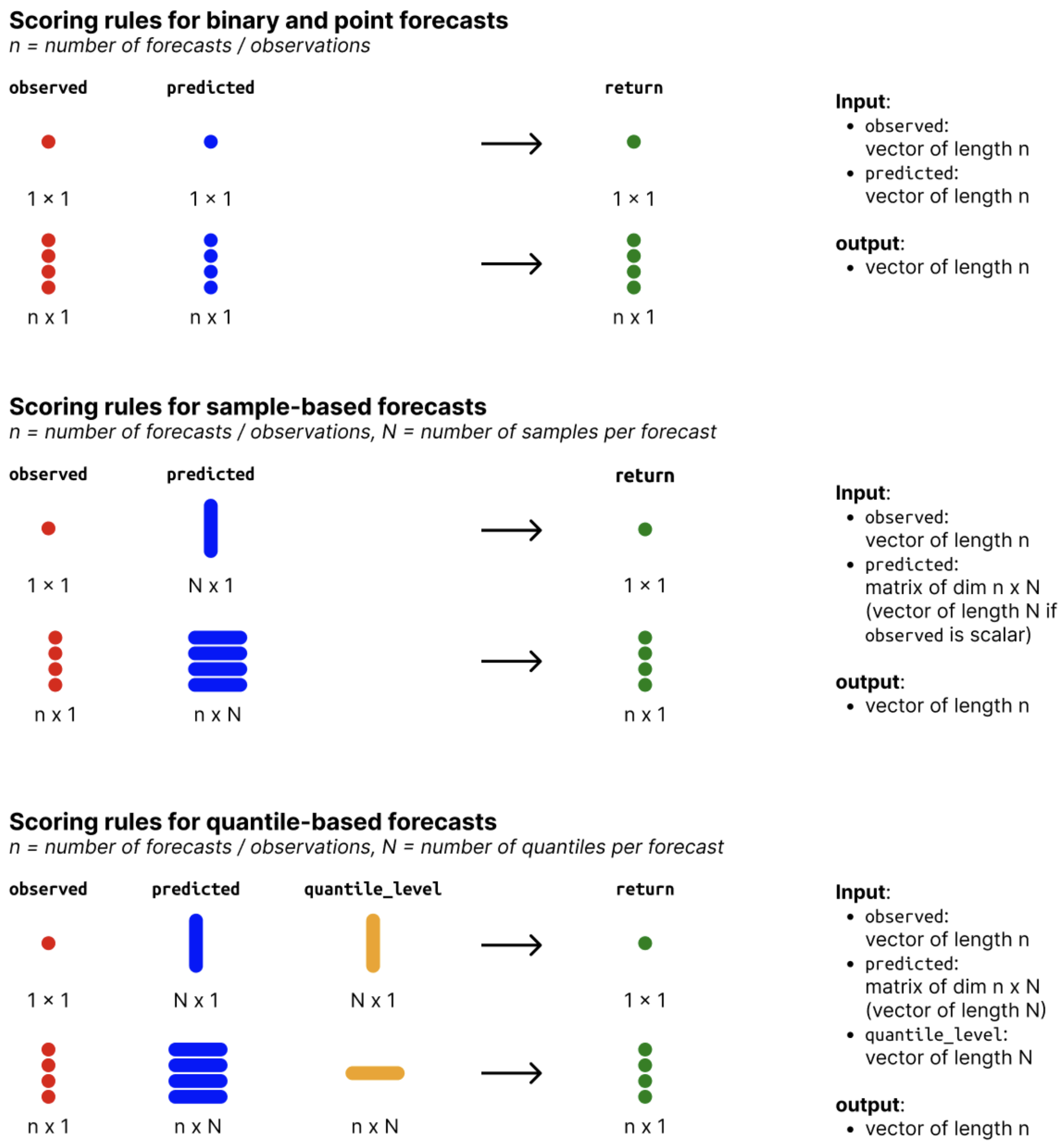

Scoring rules for binary and point forecasts
n = number of forecasts / observations
observed
predicted
return
1 × 1
1 × 1
1 × 1
n x 1
n x 1
n x 1
Input:
observed: vector of length n
predicted: vector of length n
output:
vector of length n
Scoring rules for sample-based forecasts
n = number of forecasts / observations, N = number of samples per forecast
observed
predicted
return
1 × 1
N x 1
1 × 1
n x 1
n x N
n x 1
Input:
observed: vector of length n
predicted: matrix of dim n x N (vector of length N if observed is scalar)
output:
vector of length n
Scoring rules for quantile-based forecasts
n = number of forecasts / observations, N = number of quantiles per forecast
observed
predicted
quantile_level
return
1 × 1
N x 1
N x 1
1 × 1
n x 1
n x N
n x N
n x 1
Input:
observed: vector of length n
predicted: matrix of dim n x N (vector of length N)
quantile_level: vector of length N
output:
vector of length n

Figure 6: Overview of the inputs and outputs of the scoring rules (scoring functions that can be called directly on a set of vectors / matrices)

```
R> score(example_point) |>
+   print(2)
```

```
     location target_end_date target_type observed location_name
  1:       DE      2021-05-08       Cases   106987       Germany
  2:       DE      2021-05-08       Cases   106987       Germany
 ---
886:       IT      2021-07-24      Deaths       78         Italy
```

```
887:         IT     2021-07-24      Deaths        78         Italy
     forecast_date predicted                 model horizon ae_point
  1:    2021-05-03    119258 EuroCOVIDhub-ensemble       1    12271
  2:    2021-05-03    132607 EuroCOVIDhub-baseline       1    25620
 ---
886:    2021-07-05       104  epiforecasts-EpiNow2       3       26
887:    2021-07-12       186  epiforecasts-EpiNow2       2      108
      se_point       ape
  1: 150577441 0.1146962
  2: 656384400 0.2394683
 ---
886:       676 0.3333333
887:     11664 1.3846154
```

3.4. The unit of a single forecast

In order to score forecasts, **scoringutils** needs to know which of the rows of the data belong together and jointly form a single forecasts. This is easy e.g. for point forecast, where there is one row per forecast. Probabilistic forecasts, however, are usually composed of several values and for quantile or sample-based forecasts, there are therefore multiple rows that belong to a single forecast.

The unit of a single forecast (or "forecast unit") is defined by the other columns of the data (except for a few protected columns). A single forecast should be uniquely defined by the combination of values in those columns. For example, consider forecasts made by different models in various locations at different time points and for different targets. A single forecast could then be uniquely described by the values in the columns "model", "location", "date", and "target", and the forecast unit would be `forecast_unit = c("model", "location", "date", "taret")`. As the forecast unit is internally determined based on the names of the existing columns, it is mandatory that no column is present that is unrelated to the forecast unit. As a very simplistic example, consider an additional row, "even", that is one if the row number is even and zero otherwise. The existence of this column would change results, as `score()` assumes it was relevant to grouping the forecasts.

In order to avoid issues, we recommend using the function `set_forecast_unit()` to determine the forecast unit manually. The function simply drops unneeded columns, while making sure that all necessary, 'protected columns' like `predicted` or `observed` are retained. You can get a list of all protected columns by calling `get_protected_columns()`. You can get the current forecast unit by calling `get_forecast_unit()` on the data or by simply printing the object if it is of class `forecast_*`.

The function `get_duplicate_forecasts()` may be helpful in case `validate_forecast()` returns an error message about duplicate forecasts. This can occur if there is more than one predicted value for a single forecast unit. In the case of quantile- and sample-based forecasts, rows will only be considered duplicates if they have the same quantile level or sample id. Duplicate forecasts can be a sign of a wrongly defined forecast unit. For example, if we dropped the column `target_type` in the example data, we would obtain an error about duplicate forecasts. Even though the predicted values differ, there are now two different forecasts for the exact same forecast unit, as far as **scoringutils** can tell. `get_duplicate_forecasts()` finds duplicates and returns them for the user to inspect.

```
R> rbind(example_quantile, example_quantile[1001:1002]) |>
+   get_duplicate_forecasts()

   location target_end_date target_type observed location_name
1:       DE      2021-05-22      Deaths     1285       Germany
2:       DE      2021-05-22      Deaths     1285       Germany
3:       DE      2021-05-22      Deaths     1285       Germany
4:       DE      2021-05-22      Deaths     1285       Germany
   forecast_date quantile predicted                model horizon
1:    2021-05-17    0.975      1642 epiforecasts-EpiNow2       1
2:    2021-05-17    0.990      1951 epiforecasts-EpiNow2       1
3:    2021-05-17    0.975      1642 epiforecasts-EpiNow2       1
4:    2021-05-17    0.990      1951 epiforecasts-EpiNow2       1
```

3.5. Transforming forecasts

As suggested in (Bosse *et al.* 2023), users may want to transform forecasts before scoring them. In an epidemiological context for example, the continuous ranked probability score (CRPS) and the weighted interval score (WIS) are two commonly used scoring rules. Both measure the absolute distance between the forecast and the observation. This may not be desirable in the context of epidemiological forecasts, where infectious disease processes are usually modelled to occur on a multiplicative scale. Taking the logarithm of the forecasts and observations before scoring them makes it possible to evaluate forecasters based on how well they predicted the exponential growth rate.

The function `transform_forecasts()` allows users to apply arbitrary transformations to forecasts and observations. Users can specify a function via the argument `fun` (as well as supply additional function parameters). The default function is `log_shift()`, which is simply a wrapper around `log()` with an additional offset to deal with zeroes in the data, i.e. computing `log(x + offset)`. Users can specify to either append the transformed forecasts to the existing data by setting `append = TRUE` (the default behaviour, resulting in an additional column `scale`) or to replace the existing forecasts in place.

Before calling the logarithm, we need to make sure that there are no negative values in the data. When evaluating forecasts of epidemiological count data, one should perhaps simply remove negative observations altogether, but for illustrative purposes we will replace them with zeroes first before appending transformed counts.

```
R> example_quantile |>
+   transform_forecasts(fun = \(x) {pmax(x, 0)}, append = FALSE) |>
+   transform_forecasts(fun = log_shift, offset = 1) |>
+   print(2)

       location target_end_date target_type     observed
    1:       DE      2021-01-02       Cases 1.273000e+05
    2:       DE      2021-01-02      Deaths 4.534000e+03
   ---
41089:       IT      2021-07-24      Deaths 4.369448e+00
41090:       IT      2021-07-24      Deaths 4.369448e+00
       location_name forecast_date quantile predicted
    1:       Germany          <NA>       NA        NA
    2:       Germany          <NA>       NA        NA
   ---
41089:         Italy    2021-07-12    0.975  6.416732
41090:         Italy    2021-07-12    0.990  6.579251
                     model horizon   scale
    1:                <NA>      NA natural
    2:                <NA>      NA natural
   ---
41089: epiforecasts-EpiNow2       2     log
41090: epiforecasts-EpiNow2       2     log
```

3.6. Summarising scores

Usually, users will not be interested in the scores for each individual forecast, but rather in a summarised score. The function `summarise_scores()` allows users to aggregate scores across dimensions using an arbitrary function.

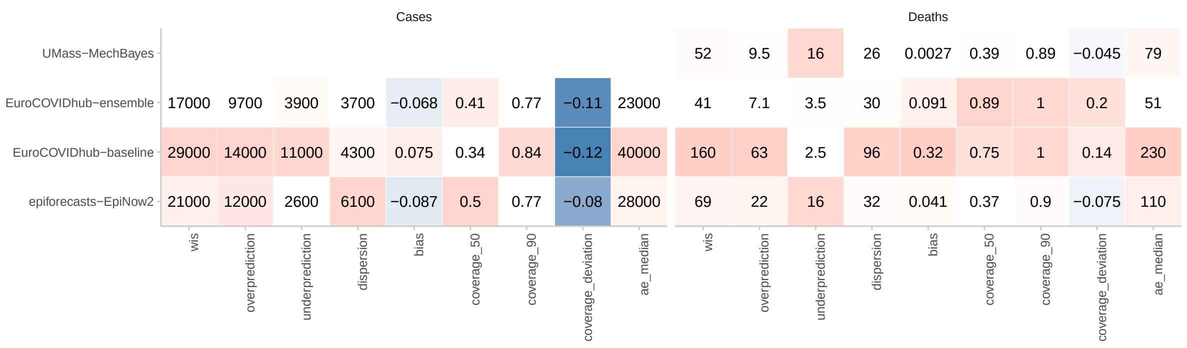
There are two different, but essentially equivalent ways of specifying the summary level. Users can either specify the columns that should be retained (using the argument `by`), or they can specify the columns that should be aggregated over (using the argument `across`).

```
R> example_quantile[horizon == 2] |>
+   score(metrics = list("wis" = wis)) |>
+   summarise_scores(by = c("model", "target_type"))
```

Summarised scores can then be visualised using the function `XXXscores_table()`. In order to display scores it is often useful to round the output, for example to two significant digits, which can be achieved with another call to `summarise_scores()`. The output of the following is shown in Figure 7:

```
R> example_quantile[horizon == 2] |>
+   score() |>
+   summarise_scores(by = c("model", "target_type")) |>
+   summarise_scores(fun = signif, digits = 2) |>
+   plot_score_table(y = "model", by = "target_type") +
+   facet_wrap(~ target_type)
```

While `summarise_scores()` accepts arbitrary summary functions, care has to be taken when using something else than `mean()`. Many scoring rules for probabilistic forecasts are so-called 'strictly proper scoring rules' (Gneiting and Raftery 2007). Strictly proper scoring rules are constructed such that they cannot always incentivise the forecaster to report her honest



	Cases									Deaths								
	wis	overprediction	underprediction	dispersion	bias	coverage_50	coverage_90	coverage_deviation	ae_median	wis	overprediction	underprediction	dispersion	bias	coverage_50	coverage_90	coverage_deviation	ae_median
UMass−MechBayes										52	9.5	16	26	0.0027	0.39	0.89	−0.045	79
EuroCOVIDhub−ensemble	17000	9700	3900	3700	−0.068	0.41	0.77	−0.11	23000	41	7.1	3.5	30	0.091	0.89	1	0.2	51
EuroCOVIDhub−baseline	29000	14000	11000	4300	0.075	0.34	0.84	−0.12	40000	160	63	2.5	96	0.32	0.75	1	0.14	230
epiforecasts−EpiNow2	21000	12000	2600	6100	−0.087	0.5	0.77	−0.08	28000	69	22	16	32	0.041	0.37	0.9	−0.075	110

Figure 7: Coloured table to visualise the computed scores. Red colours indicate that a value is higher than ideal, blue indicates it is lower than ideal and the opacity indicates the strength of the deviation from the ideal.

belief about the future and cannot be cheated. Let's assume that a forecaster's true belief about the future corresponds to a predictive distribution F. Then, if F really was the true data-generating process, a scoring rule would be proper if it ensures that no other forecast distribution G would yield a better expected score. If the scoring rule ensures that under F no other possible predictive distribution can achieve the same expected score as F, then it is called strictly proper. From the forecaster's perspective, any devation from her true belief F leads to a worsening of expected scores. When using summary functions other then the mean, however, scores may lose their propriety (the property of incentivising honest reporting) and become cheatable. For example, the median of several individual scores (individually based on a strictly roper scoring rule) is usually not proper. A forecaster judged by the median of several scores may be incentivised to misrepresent their true belief in a way that is not true for the mean score.

The user must exercise additional caution and should usually avoid aggregating scores across categories which differ much in the magnitude of the quantity to forecast, as forecast errors usually increase with the order of magnitude of the forecast target. 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 performance on deaths would have little influence. Similarly, aggregating over different forecast horizons is often ill-advised as the mean will be dominated by further ahead forecast horizons. In the previous function calls, we therefore decided to only analyse forecasts with a forecast horizon of two weeks.

3.7. Additional visualisations of scores

Heatmaps

To detect systematic patterns it may be useful to visualise a single metric across several dimensions. The function `plot_heatmap()` can be used to create a heatmap that achieves this. The following produces a heatmap of bias values across different locations and forecast targets (output shown in Figure 8).

```
R> score(example_continuous) |>
```

```
+   summarise_scores(by = c("model", "location", "target_type")) |>
+   plot_heatmap(x = "location", metric = "bias") +
+     facet_wrap(~ target_type)
```

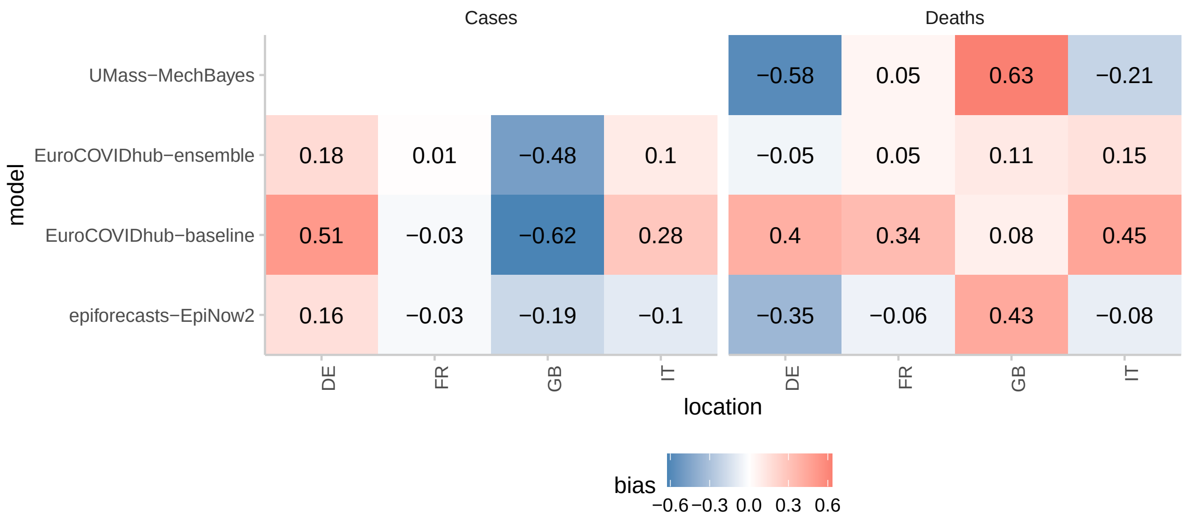


Figure 8: Heatmap of bias values for different models across different locations and forecast targets. Bias values are bound between -1 (underprediction) and 1 (overprediction) and should be 0 ideally. Red tiles indicate an upwards bias (overprediction), while blue tiles indicate a downwards bias (under-predicction)

Weighted interval score decomposition

For quantile-based forecasts, the weighted interval score (WIS, Bracher *et al.* 2021a) is commonly used and is a strictly proper scoring rule. The WIS treats the predictive quantiles as a set of symmetric prediction intervals and measures the distance between the observation and the forecast interval. It can be decomposed into a dispersion (uncertainty) component and penalties for over- and underprediction. For a single interval, the interval score is computed as

$$IS_\alpha(F,y) = \underbrace{(u-l)}_{\text{dispersion}} + \underbrace{\frac{2}{\alpha}\cdot(l-y)\cdot\mathbf{1}(y\leq l)}_{\text{overprediction}} + \underbrace{\frac{2}{\alpha}\cdot(y-u)\cdot\mathbf{1}(y\geq u)}_{\text{underprediction}},$$

where $\mathbf{1}()$ is the indicator function, y is the observed value, and l and u are the $\frac{\alpha}{2}$ and $1-\frac{\alpha}{2}$ quantiles of the predictive distribution F, i.e. the lower and upper bound of a single prediction interval. For a set of K prediction intervals and the median m, the score is computed as a weighted sum,

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

where w_k is a weight for every interval. Usually, $w_k = \frac{\alpha_k}{2}$ and $w_0 = 0.5$. It is helpful to visualise the decomposition of the weighted interval score into its components: dispersion, overprediction and underprediction. This can be achieved using the function `plot_wis()`, as shown in Figure 9

```
R> score(example_quantile) |>
+   summarise_scores(by = c("model", "target_type")) |>
+   plot_wis(relative_contributions = FALSE) +
+   facet_wrap(~ target_type,
+              scales = "free_x")
```

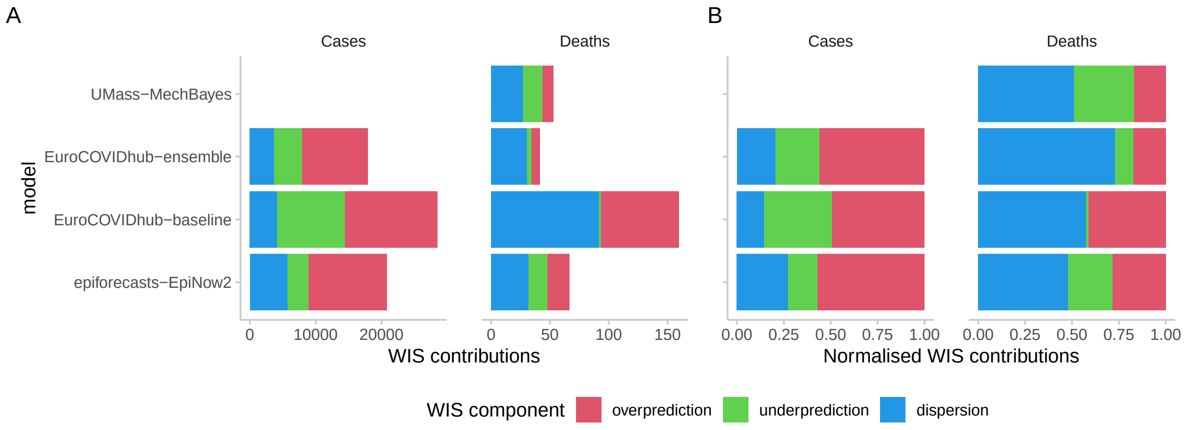


Figure 9: Decomposition of the weighted interval score (WIS) into dispersion, overprediction and underprediction. A: absolute contributions, B: contributions normalised to 1.

3.8. Correlations

Users can examine correlations between scores using the function `correlation()`. This produces an output of class XXX with its own plot method. The plot resulting from the following code is shown in Figure 10.

```
R> correlations <- example_quantile |>
+   score() |>
+   summarise_scores() |>
+   correlation()
R>
R> correlations |>
+   plot()
```

3.9. Pairwise comparisons

3.10. Pairwise comparisons

Raw scores for different forecasting models are not directly comparable in the case of missing forecasts, as forecasting targets usually differ in their characteristics (e.g., the scale of the forecast target, how difficult targets are to forecast etc.). One way to mitigate this are relative skill scores based on pairwise comparisons (Cramer *et al.* 2021).

Models enter a 'pairwise tournament', where all possible pairs of models are compared based on the overlapping set of available forecasts common to both models (omitting comparisons

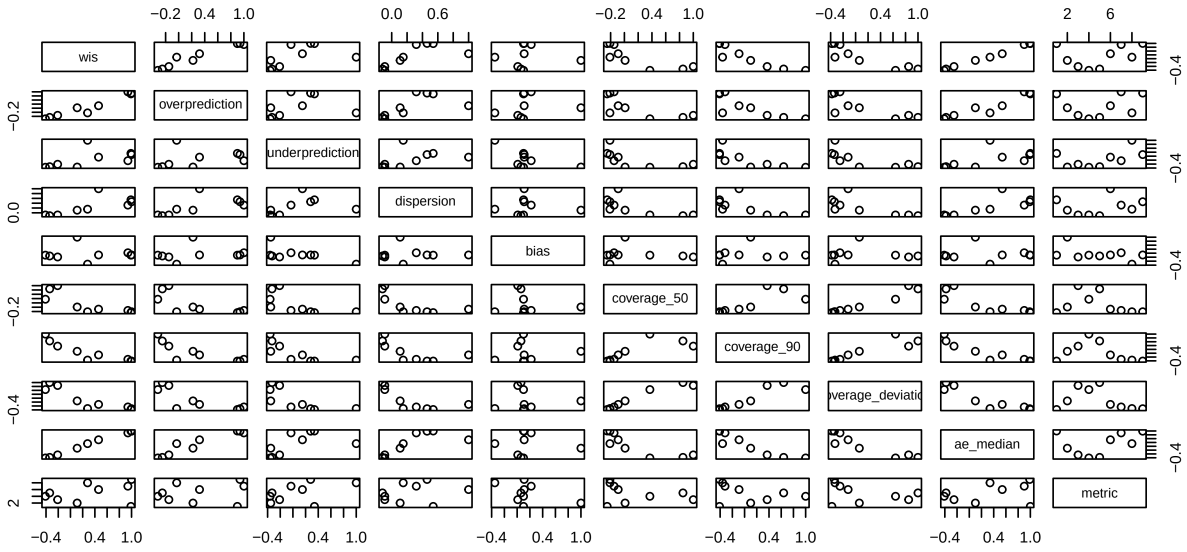


Figure 10: Correlation between different scores

where there is no overlapping set of forecasts). For every pair, the ratio of the mean scores of both models is computed. The relative skill score of a model is then the geometric mean of all mean score ratios which involve that model. This gives us an indicator of performance relative to all other models, with the orientation depending on the score used (if lower values are better for a particular scoring rule, then the same is true for the relative skill score computed based on that score). Two models can of course only be fairly compared if they have overlapping forecasts. One simple rule of thumb one could apply is to only compare models that have forecasts for at least 50% of the available targets, thereby ensuring that all models have overlapping sets of forecasts. Furthermore, pairwise comparisons between models for a given metric are only possible if all values have the same sign, i.e. all score values need to be either positive or negative. The process of pairwise comparisons is illustrated in Figure 11.

Relative skill scores via pairwise comparisons can be calculated in two different ways. The first one is by calling the function `pairiwse_comparison()`. It takes a `data.table` (or similar) of scores as input, and returns a `data.table` with the results of the pairwise tournament. It displays the mean scores ratio for every pair of models, a p-value for whether scores for one model are significantly different from scores for another model, and the relative skill score for every model. Users can also specify a baseline model, in which case a scaled relative skill scores is computed by dividing the relative skill score of every model by the relative skill score of the baseline model. Another option is to call the function `add_pairwise_comparison()` on the output of `score()` which will simply add additional columns for the relative skill and scaled relative skill.

In both cases, pairwise comparisons are computed according to the grouping specified in the argument `by`: internally, the `data.table` with all scores gets split into different `data.tables` according to the values specified in `by` (excluding the column 'model'). Relative scores are then computed for every individual group separately. In the example below we specify `by = c("model", "target_type")`, which means that there is one relative skill score per model, calculated separately for the different forecasting targets.

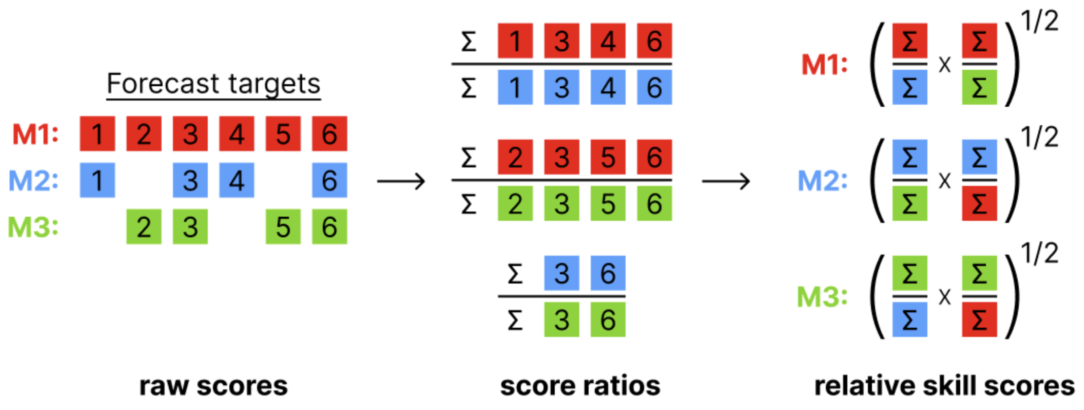


Figure 11: Illustration of the computation of relative skill scores through pairwise comparisons of three different forecast models, M1-M3..

```
R> s <- score(example_quantile[horizon == 2])
R> pairwise_comparison(s, by = c("model", "target_type")) |>
+    print(1:3)
```

The output of `pairwise_comparison()` is an object of class XXX with its own `plot()` method. An example is shown in Figure 12.

```
R> score(example_quantile) |>
+    pairwise_comparison(by = c("model", "target_type"),
+                        baseline = "EuroCOVIDhub-baseline") |>
+    plot_pairwise_comparison() +
+    facet_wrap(~ target_type)
```

It is in principle possible to compute p-values to determine whether two models perform significantly differently. **scoringutils** allows to compute these using either the Wilcoxon rank sum test (also known as Mann-Whitney-U test) (Mann and Whitney 1947) or a permutation test. In practice, this is 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 might perform badly across different targets for a given forecast date). P-values may therefore be too liberal in suggesting significant differences where there aren't any. One way to mitigate this is to aggregate observations over a category where one suspects correlation (for example averaging across all forecasts made on a given date) before making pairwise comparisons. A test that is performed on aggregate scores will likely be more conservative.

Pairwise comparisons should usually be made based on unsummarised scores (the function `pairwise_comparison()` internally summarises over samples and quantiles automatically, but nothing else), as summarising can change the set of overlapping forecasts between two models and distort relative skill scores. When using `pairwise_comparison()`, the function

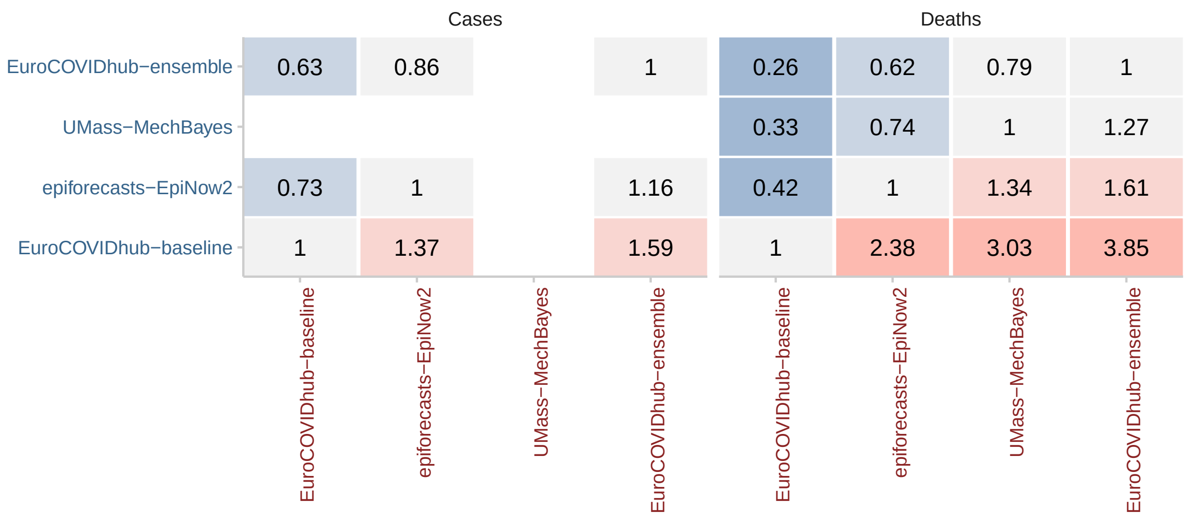


Figure 12: Ratios of mean weighted interval scores based on overlapping forecast sets. When interpreting the plot one should look at the model on the y-axis, and the model on the x-axis is the one it is compared against. If a tile is blue, then the model on the y-axis performed better. If it is red, the model on the x-axis performed better in direct comparison. In the example above, the EuroCOVIDhub-ensemble performs best (it only has values smaller than one), while the EuroCOVIDhub-baseline performs worst (and only has values larger than one). For cases, the UMass-MechBayes model is excluded as there are no case forecasts available and therefore the set of overlapping forecasts is empty.

`summarise_scores()` should therefore usually not be called beforehand. One potential exception to this is when one is interested in the p-values obtained from pairwise comparisons. As forecasts are usually highly correlated (which the calculation of p-values do not account for), it may be sensible to summaries over a few categories (provided there are no missing values within the categories summarised over) to reduce correlation and obtain more conservative p-values.

4. Calibration

Calibration refers to a statistical consistency (i.e., absence of systematic deviations) between the forecasts and the observations. It is possible to distinguish several forms of calibration which are discussed in detail by Gneiting *et al.* (2007). The form of calibration most commonly focused on is called probabilistic calibration (for other form of calibration, see Gneiting *et al.* (2007)). Probabilistic calibration means that the forecast distributions are consistent with the true data-generating distributions in the sense that on average, τ% of true observations will be below the corresponding τ-%-quantiles of the cumulative forecast distributions.

4.1. Interval coverage and quantile coverage

This of course, can be most easily verified for predictive distributions in a quantile-based format. For such forecasts, one can easily compare the proportion of observations that fall below the τ-quantiles of all forecasts ("empirical quantile coverage") to the nominal quantile coverage τ.

The above definition of probabilistic calibration also implies that the empirical coverage of the central prediction intervals formed by the predictive quantiles should be equal to the nominal interval coverage. For example, the central 50% prediction intervals of all forecasts should really contain around 50% of the observed values, the 90% central intervals should contain around 90% of observations etc. Forecasts that are too narrow and do not cover the required proportion of observations are called overconfident or underdispersed, while predictive distributions that are too wide are often called underconfident, overdispersed or conservative.

Users can obtain coverage values for quantile-based predictions in two different ways. The first is to call the functions XXX and XXX directly as scoring rules within `score()` to obtain coverage values for specific quantile-levels or central prediction intervals. A more comprehensive way is by using the function `add_coverage()` directly on the raw forecasts. It adds additional columns with the width of the central prediction interval corresponding to a given quantile level, quantile coverage, interval coverage, quantile coverage deviation and interval coverage deviation. Deviation here means the difference between nominal and empirical coverage. Coverage for a single quantile or interval is only ever `TRUE` or `FALSE`. Coverage values are therefore only meaningful when summarised over many forecasts. This can be done calling `summarise_scores()`. Results can then be visualised using the function `plot_interval_coverage()` (see row 3 in Figure 14) and `plot_quantile_coverage()` (row 4 in Figure 14). Both show nominal against empirical coverage. Ideally forecasters should lie on the diagonal line. For interval coverage plots, a shift to the left means a forecaster is too conservative and issues a predictive distribution that is too wide and covers more of the observed values than needed. A shift to the right means a forecaster is overconfident and the forecast distribution is too narrow. For *quantile coverage plots*, the interpretation depends on whether the quantile is above or below the median. For quantiles below the median, a line to the right of the diagonal (predictive quantiles lower than the quantiles of the data-generating distribution) means a forecaster is too conservative, while for quantiles above the median, a line to the left of the diagonal line (predictive quantiles higher than the quantiles of the data-generating distribution) implies conservative predictions. Areas that imply a conservative forecaster are shaded in green.

It is in principle possible to convert sample-based forecasts to quantile-based forecasts using the function `sample_to_quantile_forecastXXX()` to make use of `add_coverage()`. This should be done with caution, as the estimation of quantiles from predictive samples may be biased if the number of available samples is not sufficiently large.

4.2. Probability integral transform (PIT)

A more natural way to visualise probabilistic calibration for sample-based forecasts (and an alternative option for quantile-based ones) is the probability integral transform (PIT) histogram (Dawid 1984).

Observed values, y, are transformed using the CDF of the predictive distribution, F, to create a new variable u with $u = F(y)$. u is therefore simply the CDF of the predictive distribution evaluated at the observed value. If forecasts are probabilistically calibrated, then the transformed values will be uniformly distributed (for a proof see for example Angus (1994)). When plotting a histogram of PIT values (see row 2 in Figure 14), bias usually leads to a triangular shape, a U-shaped histogram corresponds to forecasts that are under-dispersed (too sharp) and a hump-shape appears when forecasts are over-dispersed (too wide). There exist different variations of the PIT to deal with discrete instead of continuous data (see e.g. Czado, Gneiting, and Held (2009) and Funk *et al.* (2019)). The PIT version implemented in `scoringutils` for discrete variables follows Funk *et al.* (2019).

...and create PIT histograms using the function `plot_pit()`. The output of the following is shown in Figure 13:

```
R> example_continuous |>
+   pit(by = c("model", "target_type")) |>
+   plot_pit() +
+   facet_grid(target_type ~ model)
```

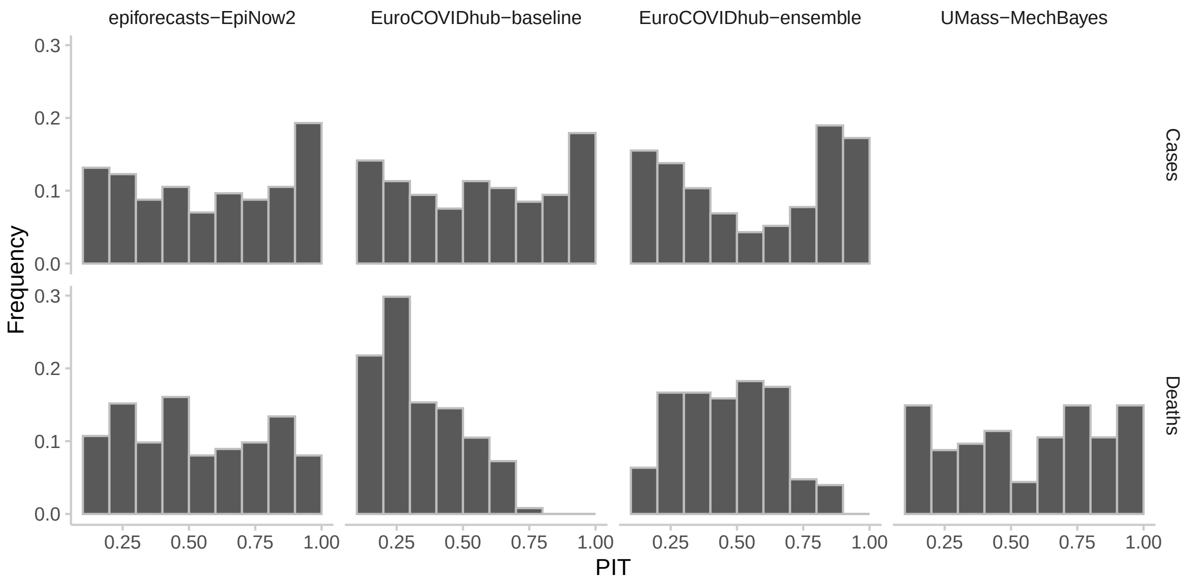
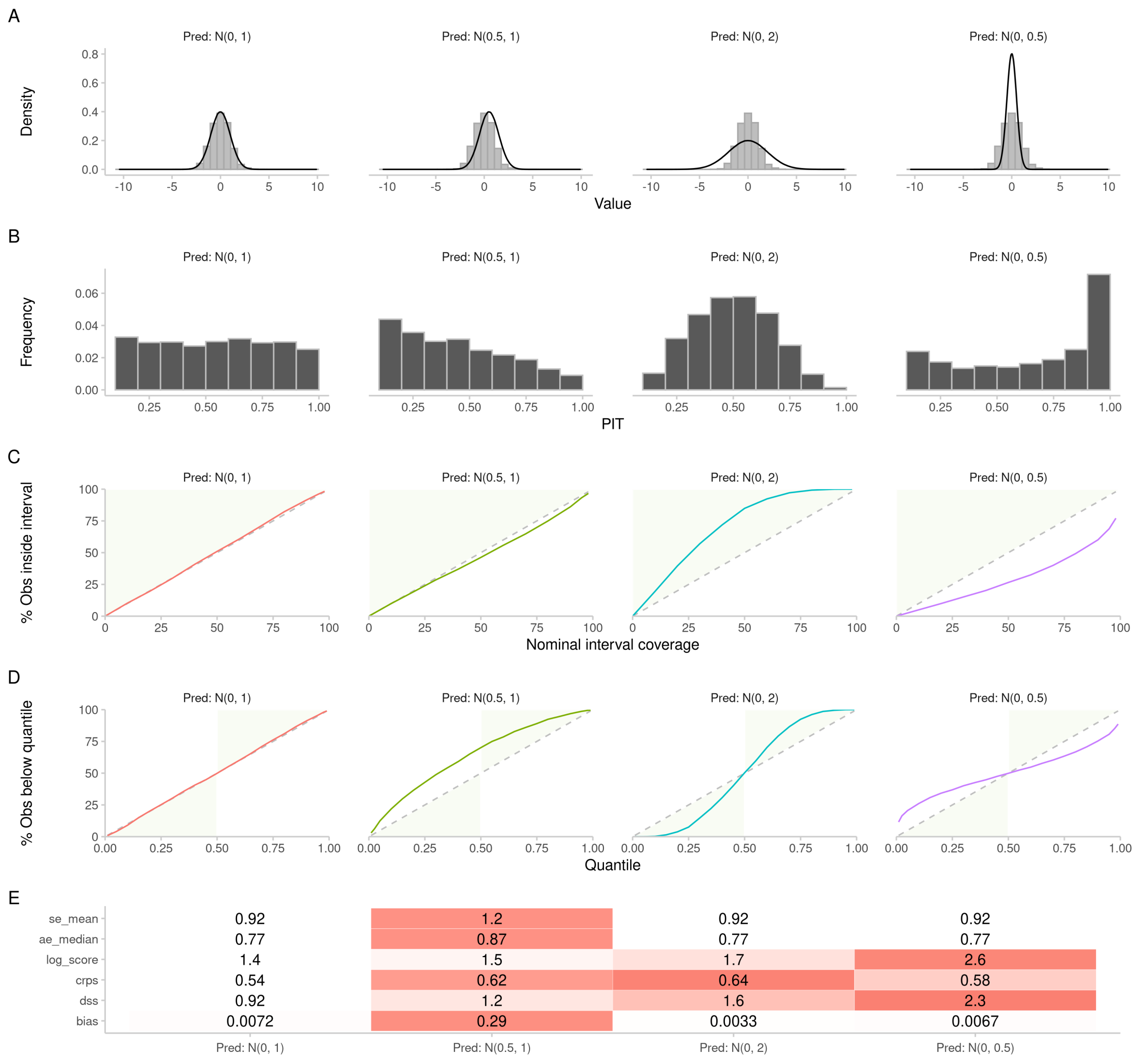


Figure 13: PIT histograms of all models stratified by forecast target. Histograms should ideally be uniform. A u-shape usually indicates overconfidence (forecasts are too narrow), a hump-shaped form indicates underconfidence (forecasts are too uncertain) and a triangle-shape indicates bias.

It is in theory possible to formally test probabilistic calibration, for example by employing an Anderson Darling test on the uniformity of PIT values. In practice this can be difficult as forecasts and therefore also PIT values are often correlated. Personal experience suggests that the Anderson Darling test is often too quick to reject the null hypothesis of uniformity. It is also important to note that uniformity of the PIT histogram (or a diagonal on quantile and interval coverage plots) indicates probabilistic calibration, but does not guarantee that forecasts are indeed calibrated in every relevant sense. Gneiting *et al.* (2007); Hamill (2001) provide examples with different forecasters who are clearly mis-calibrated, but have uniform PIT histograms.

```
R> cov_scores <- score(example_quantile) |>
+   summarise_scores(by = c("model", "target_type", "range", "quantile"))
```



	Pred: N(0, 1)	Pred: N(0.5, 1)	Pred: N(0, 2)	Pred: N(0, 0.5)
se_mean	0.92	1.2	0.92	0.92
ae_median	0.77	0.87	0.77	0.77
log_score	1.4	1.5	1.7	2.6
crps	0.54	0.62	0.64	0.58
dss	0.92	1.2	1.6	2.3
bias	0.0072	0.29	0.0033	0.0067

Figure 14: A: Different forecasting distributions (black) against observations sampled from a standard normal distribution (grey histograms). B: PIT histograms based on the predictive distributions and the sampled observations shown in A. C: Empirical vs. nominal coverage of the central prediction intervals for simulated observations and predictions. Areas shaded in green indicate that the forecasts are too wide (i.e., underconfident), covering more true values than they actually should, while areas in white indicate that the model generates too narrow predictions and fails to cover the desired proportion of true values with its prediction intervals. D: Quantile coverage values, with green areas indicating too wide (i.e., conservative) forecasts. E: Scores for the standard normal predictive distribution and the observations drawn from different data-generating distributions.

```
R>
R> plot_interval_coverage(cov_scores) +
+   facet_wrap(~ target_type)
```

```
R>
R> plot_quantile_coverage(cov_scores) +
+    facet_wrap(~ target_type)
```

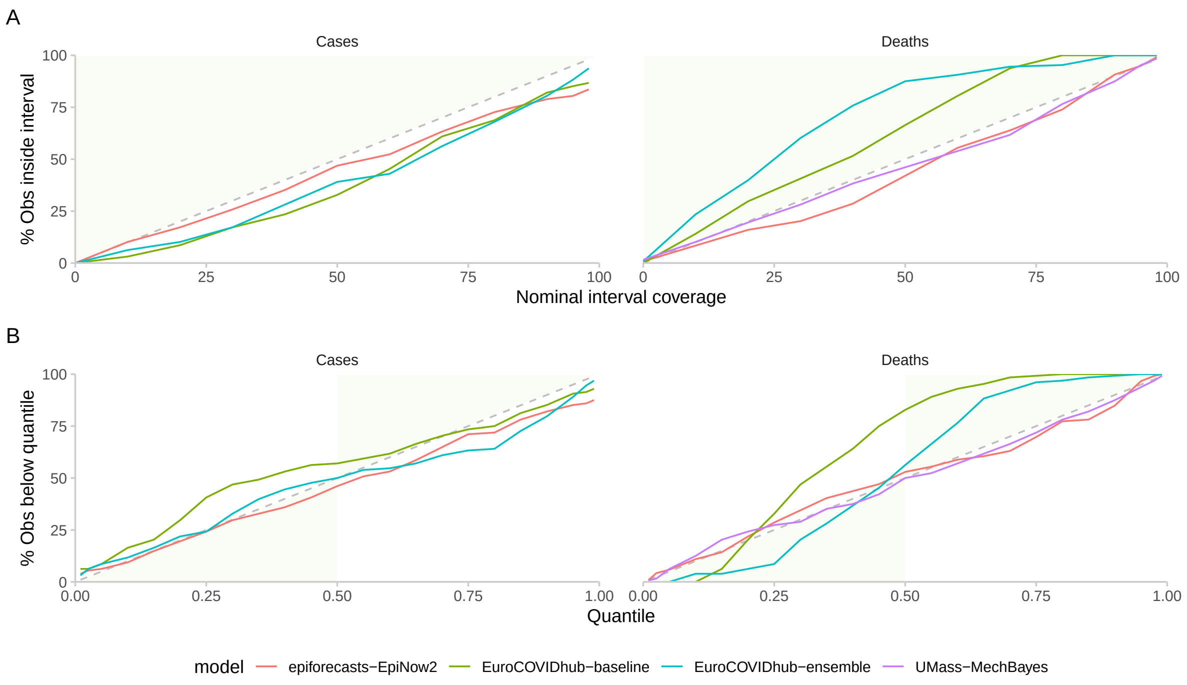


Figure 15: Interval coverage (A) and quantile coverage (B) plots. Areas shaded in green indicate that the forecasts are too wide (i.e., underconfident), while areas in white indicate that the model is overconfident and generates too narrow predictions intervals.

4.3. Bias

Another specific and very common form of miscalibration is bias, i.e. systematically over- or underpredicting the observed values. **scoringutils** exports a bias metric `bias_quantile()` and `bias_sample()`. The implementation follows Funk *et al.* (2019) and captures how much probability mass of the forecast was above or below the true value (mapped to values between -1 and 1, with 0 being ideal). Values represent a general tendency to over- or under-predict in relative terms. A value of -1 implies that the entire probability mass of the predictive distribution was below the observed value (and analogously above it for a value of 1).

For forecasts in a quantile format, bias is also reflected in the over- and underprediction components of the weighted interval score (a proper scoring rule explained in more detail in Section **??**). These measure over- and underprediction on an absolute scale (analogous to the absolute error of a point forecast), rather than a relative scale. It is important to note that it is not a priori clear what the decomposition 'should' look like - a forecast can be well calibrated and still have different amounts of over- and underprediction. High overprediction or underprediction values can therefore not immediately be interpreted as systematic bias.

5. Summary and discussion

Future development plans

Forecast evaluation is invaluable to understanding and improving current forecasts. The **scoringutils** package aims to facilitate this process and make it easier, even for less experienced users. It provides a fast, flexible and convenient evaluation framework based on `data.frame`s, but also makes a set of scoring functions available to more experienced users to be used in other packages or pipelines. A set of visualisations and plotting functions help with diagnosing issues with models and allow for thorough comparison between different forecasting approaches.

The package is still under active development and we warmly welcome contributions to **scoringutils**. In the future we hope to extend the number of scoring metrics supported. This includes spherical scoring rules (Gneiting and Raftery 2007; Jose 2009; Machete 2012), evaluation of multinomial prediction tasks, as well as a broader range of scoring metrics for point forecasts. We also plan to expand the plotting functionality and hope to make templates available for automated scoring reports.

6. Acknowledgments

Funding statements

NIB received funding from the Health Protection Research Unit (grant code NIHR200908). HG MISSING. AC acknowledges funding by the NIHR, the Sergei Brin foundation, USAID, and the Academy of Medical Sciences. EvL acknowledges funding by the National Institute for Health Research (NIHR) Health Protection Research Unit (HPRU) in Modelling and Health Economics (grant number NIHR200908) and the European Union's Horizon 2020 research and innovation programme - project EpiPose (101003688). SF's work was supported by the Wellcome Trust (grant: 210758/Z/18/Z), and the NIHR (NIHR200908). SA's work was funded by the Wellcome Trust (grant: 210758/Z/18/Z). This study is partially funded by the National Institute for Health Research (NIHR) Health Protection Research Unit in Modelling and Health Economics, a partnership between UK Health Security Agency and Imperial College London in collaboration with LSHTM (grant code NIHR200908); and acknowledges funding from the MRC Centre for Global Infectious Disease Analysis (reference MR/R015600/1), jointly funded by the UK Medical Research Council (MRC) and the UK Foreign, Commonwealth & Development Office (FCDO), under the MRC/FCDO Concordat agreement and is also part of the EDCTP2 programme supported by the European Union. Disclaimer: "The views expressed are those of the author(s) and not necessarily those of the NIHR, UKHSA or the Department of Health and Social Care. We thank Community Jameel for Institute and research funding

(APPENDIX) Detailed Information on Metrics

Metric	Explanation
CRPS (Continuous) ranked probability score	The crps is a proper scoring rule that generalises the absolute error to probabilistic forecasts. It measures the 'distance' of the predictive distribution to the observed data-generating distribution. The CRPS is given as $$\text{CRPS}(F, y) = \int_{-\infty}^{\infty} \left(F(x) - 1(x \geq y)\right)^2 dx,$$ where y is the true observed value and F the CDF of predictive distribution. Often An alternative representation is used: $$\text{CRPS}(F, y) = \frac{1}{2}\mathbb{E}_F\|X - X'\| - \mathbb{E}_P\|X - y\|,$$ where X and X' are independent realisations from the predictive distributions F with finite first moment and y is the true value. In this representation we can simply replace X and X' by samples sum over all possible combinations to obtain the CRPS. For integer-valued forecasts, the RPS is given as $$\text{RPS}(F, y) = \sum_{x=0}^{\infty}(F(x) - 1(x \geq y))^2.$$ **Usage and caveats** Smaller values are better. The crps is a good choice for most practical purposes that involve decision making, as it takes the entire predictive distribution into account. If two forecasters assign the same probability to the true event y, then the forecaster who assigned high probability to events far away from y will still get a worse score. The crps (in contrast to the log score) can at times be quite lenient towards extreme mispredictions. Also, due to it's similarity to the absolute error, the level of scores depend a lot on the absolute value of what is predicted, which makes it hard to compare scores of forecasts for quantities that are orders of magnitude apart.

Table 1: Detailed explanation of all the metrics.

Metric	Explanation
Log score	The Log score is a proper scoring rule that is computed as the negative log of the predictive density evaluated at the true observed value. It is given as $$\text{log score} = -\log f(y),$$ where f is the predictive density function and y is the true value. For integer-valued forecasts, the log score can be computed as $$\text{log score} = -\log p_y,$$ where p_y is the probability assigned to outcome p by the forecast F. **Usage and caveats**: Smaller values are better, but sometimes the sign is reversed. The log score is sensitive to outliers, as individual log score contributions quickly can become very large if the event falls in the tails of the predictive distribution, where $f(y)$ (or p_y) is close to zero. Whether or not that is desirable depends on the application. In **scoringutils**, the log score cannot be used for integer-valued forecasts, as the implementation requires a predictive density. In contrast to the crps, the log score is a local scoring rule: it's value only depends only on the probability that was assigned to the actual outcome. This property may be desirable for inferential purposes, for example in a Bayesian context (Winkler et al., 1996). In settings where forecasts inform decision making, it may be more appropriate to score forecasts based on the entire predictive distribution.

Table 1: Detailed explanation of all the metrics.

Metric	Explanation
WIS (Weighted) interval score	The (weighted) interval score is a proper scoring rule for quantile forecasts that converges to the crps for an increasing number of intervals. The score can be decomposed into a sharpness (uncertainty) component and penalties for over- and underprediction. For a single interval, the score is computed as $$IS_\alpha(F, y) = (u - l) + \frac{2}{\alpha} \cdot (l - y) \cdot 1(y \leq l) + \frac{2}{\alpha} \cdot (y - u) \cdot 1(y \geq u),$$ where 1() is the indicator function, y is the true value, and l and u are the $\frac{\alpha}{2}$ and $1 - \frac{\alpha}{2}$ quantiles of the predictive distribution F, i.e. the lower and upper bound of a single prediction interval. For a set of K prediction intervals and the median m, the score is computed as a weighted sum, $$WIS = \frac{1}{K + 0.5} \cdot (w_0 \cdot \lvert y - m\rvert + \sum_{k=1}^{K} w_k \cdot IS_\alpha(F, y)),$$ where w_k is a weight for every interval. Usually, $w_k = \frac{\alpha_k}{2}$ and $w_0 = 0.5$. **Usage and caveats**: Smaller scores are better. Applicable to all quantile forecasts, takes the entire predictive distribution into account. Just as the crps, the wis is based on measures of absolute error. When averaging across multiple targets, it will therefore be dominated by targets with higher absolute values. The decomposition into sharpness, over- and underprediction make it easy to interpret scores and use them for model improvement.
DSS Dawid-Sebastiani score	The Dawid-Sebastiani-Score is a proper scoring rule proposed that only relies on the first moments of the predictive distribution and is therefore easy to compute. It is given as $$\text{dss}(F, y) = \left(\frac{y - \mu}{\sigma}\right)^2 + 2 \cdot \log \sigma,$$ where F is the predictive distribution with mean μ and standard deviation σ and y is the true observed value. **Usage and caveats** The dss is applicable to continuous and integer forecasts and easy to compute. Apart from the ease of computation we see little advantage in using it over other scores.

Table 1: Detailed explanation of all the metrics.

Metric	Explanation
Brier score	Proper scoring rule for binary forecasts. The Brier score is computed as $$\text{Brier Score} = \frac{1}{N}\sum_{n=1}^{N}(f_n - y_n),$$ where f_n, with $n = 1, \dots, N$ are the predicted probablities that the corresponding events, $y_n \in (0, 1)$ will be equal to one.) **Usage**: Applicable to all binary forecasts.
Interval coverage	Interval coverage measures the proportion of observed values that fall in a given prediction interval range. Interval coverage for a single prediction interval range can be calculated as $$IC_\alpha = \text{nominal coverage} - \text{empirical coverage},$$ where nominal coverage is $1 - \alpha$ and empirical coverage is the proportion of true values actually covered by all $1 - \alpha$ prediction intervals. To summarise interval coverage over different over multiple interval ranges, we can compute coverage deviation defined as the mean interval coverage over all K interval ranges α_k with $k = 1, \dots, K$: $$\text{Coverage deviation} = \frac{1}{K}\sum_{k=1}^{K}\text{IC}_{\alpha_k}$$ **Usage**: Interval coverage for a set of chosen intervals, (e.g. 50% and 90%) gives a good indication of marginal calibration and is easy to interpret. Reporting coverage deviation has the advantage of summarising calibration in a single number, but loses some of the nuance.
Quantile coverage	Quantile coverage for a given quantile level is the proportion of true values smaller than the predictions corresponding to that quantile level. **Usage**: Quantile coverage is similar to interval coverage, but conveys more information. For example, it allows us to look at the 5% and 95% quantile separately, instead of jointly at the 90% prediction interval). This helps to diagnose whether it is the upper or lower end of a prediction interval that is causing problems. Plots of quantile coverage are conceptually very similar to PIT histograms.

Table 1: Detailed explanation of all the metrics.

Metric	Explanation
Probability integral transform (PIT)	The probability integral transform (PIT, Dawid 1984) represents a succinct way to visualise deviations between the predictive distribution F and the true data-generating distribution G. The idea is to transform the observed values such that agreement between forecasts and data can then be examined by observing whether or not the transformed values follow a uniform distribution. The PIT is given by $$u = F(y),$$ where u is the transformed variable and $F(y)$ is the predictive distribution F evaluated at the true observed value y. If $F = G$, then u follows a uniform distribution. For integer outcomes, the PIT is no longer uniform even when forecasts are ideal. Instead, a randomised PIT can be used: $$u = P(y) + v \cdot (P(y) - P(y-1)),$$ where y is again the observed value $P()$ is the cumulative probability assigned to all values smaller or equal to y (where $P(-1) = 0$ by definition, and v is a standard uniform variable independent of y. If P is equal to the true data-generating distribution function, then u is standard uniform. also propose a non-randomised version of the PIT for count data that could be used alternatively. **Usage**: One can plot a histogram of u 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. One can also test for deviations from normality, using for example an Anderson-Darling test. This, however, proves to be overly strict in practice and even slight deviations from perfect calibration are punished in a way that makes it very hard to compare models at all. In addition, errors from forecasts may be correlated (i.e. forecasts made on a given date), potentially violating the assumptions of the Anderson-Darling test. We therefore do not recommend it for most use cases.

Table 1: Detailed explanation of all the metrics.

Metric	Explanation
Sharpness	Sharpness is the ability to produce narrow forecasts and is a feature 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. As suggested by Funk et al. (2019), we measure sharpness for continuous and integer forecasts represented by predictive samples as the normalised median absolute deviation about the median (MADN)), i.e. $$S(F) = \frac{1}{0.675} \cdot \text{median}(\lvert x - \text{median(x)} \rvert),$$ where x is the vector of all predictive samples and $\frac{1}{0.675}$ is a normalising constant. If the predictive distribution F is the CDF of a normal distribution, then sharpness will equal the standard deviation of F. For quantile forecasts we can directly use the sharpness component of the weighted interval score. Sharpness is then simply the weighted mean of the widths of the central prediction intervals.

Table 1: Detailed explanation of all the metrics.

Metric	Explanation
Bias	Bias is a measure of the tendency of a forecaster to over- or underpredict. For continuous forecasts, bias is given as $$B(F, y) = 1 - 2 \cdot (F(y)),$$ where F is the CDF of the predictive distribution and y is the observed value. For integer-valued forecasts, bias can be calculated as $$B(P, y) = 1 - (P(y) + P(y + 1)),$$ where $P(y)$ is the cumulative probability assigned to all outcomes smaller or equal to y. For quantile forecasts, Bias can be calculated as the maximum percentile rank for which the prediction is smaller than y, 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 bias 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, bias 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. **Usage**: In contrast to the over- and underprediction penalties of the interval score it is bound between 0 and 1 and represents the tendency of forecasts to be biased rather than the absolute amount of over- and underprediction. It is therefore a more robust measurement, but harder to interpet. It largely depends on the application whether one is more interested in the tendency to be biased or in the absolute value of over- and underpredictions.

Table 1: Detailed explanation of all the metrics.

Metric	Explanation
Mean score ratio	The mean score ratio is used to compare two models on the overlapping set of forecast targets for which both models have made a prediction. The mean score ratio is calculated as the mean score achieved by the first model over the mean score achieved by the second model. More precisely, for two models i, j, we determine the set of overlapping forecasts, denoted by $\mathcal{A}_{ij}$ and compute the mean score ratio θ_{ij} as $$\theta_{ij} = \frac{\text{mean score model } i \text{ on } \mathcal{A}_{ij}}{\text{mean score model } j \text{ on } \mathcal{A}_{ij}}.$$ The mean score ratio can in principle be computed for any arbitrary score. **Usage**: Mean scores ratios are usually calculated in the context of pairwise comparisons, where a set of models is compared by looking at mean score ratios of all possible parings. Whether smaller or larger values are better depends on the orientation of the original score used
Relative skill	Relative skill scores can be used to obtain a ranking of models based on pairwise comparisons between all models. To compute the relative skill θ_i of model i, we take the geometric mean of all mean score ratios that involve model i, i.e. $$\theta_i = \left(\prod_{m=1}^{M} \theta_{im} \right)^{1/M},$$ where M is the number of models. **Usage and caveats**: Relative skill is a helpful way to obtain a model ranking. Whether smaller or larger values are better depends on the orientation of the original score used. It is in principle relatively robust against biases that arise when models only forecast some of the available targets and is a reasonable way to handle missing forecasts. One possible precautionary measure to reduces issues with missing forecasts is to only compare models that have forecasted at least half of all possible targets (this ensures that there is always an overlap between models). If there is no overlap between models, the relative skill implicitly estimates how a model would have forecasted on those missing targets.

References

Angus JE (1994). "The Probability Integral Transform and Related Results." *SIAM Review*, **36**(4), 652–654. ISSN 0036-1445. doi:10.1137/1036146.

Bosse NI, Abbott S, Cori A, van Leeuwen E, Bracher J, Funk S (2023). "Scoring Epidemiological Forecasts on Transformed Scales." *PLOS Computational Biology*, **19**(8), e1011393. ISSN 1553-7358. doi:10.1371/journal.pcbi.1011393.

Bracher J, Ray EL, Gneiting T, Reich NG (2021a). "Evaluating Epidemic Forecasts in an Interval Format." *PLoS computational biology*, **17**(2), e1008618. ISSN 1553-7358. doi:10.1371/journal.pcbi.1008618.

Bracher J, Wolffram D, Deuschel J, Görgen K, Ketterer JL, Ullrich A, Abbott S, Barbarossa MV, Bertsimas D, Bhatia S, Bodych M, Bosse NI, Burgard JP, Castro L, Fairchild G, Fuhrmann J, Funk S, Gogolewski K, Gu Q, Heyder S, Hotz T, Kheifetz Y, Kirsten H, Krueger T, Krymova E, Li ML, Meinke JH, Michaud IJ, Niedzielewski K, Ożański T, Rakowski F, Scholz M, Soni S, Srivastava A, Zieliński J, Zou D, Gneiting T, Schienle M (2021b). "Short-Term Forecasting of COVID-19 in Germany and Poland during the Second Wave – a Preregistered Study." *medRxiv*, p. 2020.12.24.20248826. doi:10.1101/2020.12.24.20248826.

Cramer E, Ray EL, Lopez VK, Bracher J, Brennen A, Rivadeneira AJC, Gerding A, Gneiting T, House KH, Huang Y, Jayawardena D, Kanji AH, Khandelwal A, Le K, Mühlemann A, Niemi J, Shah A, Stark A, Wang Y, Wattanachit N, Zorn MW, Gu Y, Jain S, Bannur N, Deva A, Kulkarni M, Merugu S, Raval A, Shingi S, Tiwari A, White J, Woody S, Dahan M, Fox S, Gaither K, Lachmann M, Meyers LA, Scott JG, Tec M, Srivastava A, George GE, Cegan JC, Dettwiller ID, England WP, Farthing MW, Hunter RH, Lafferty B, Linkov I, Mayo ML, Parno MD, Rowland MA, Trump BD, Corsetti SM, Baer TM, Eisenberg MC, Falb K, Huang Y, Martin ET, McCauley E, Myers RL, Schwarz T, Sheldon D, Gibson GC, Yu R, Gao L, Ma Y, Wu D, Yan X, Jin X, Wang YX, Chen Y, Guo L, Zhao Y, Gu Q, Chen J, Wang L, Xu P, Zhang W, Zou D, Biegel H, Lega J, Snyder TL, Wilson DD, McConnell S, Walraven R, Shi Y, Ban X, Hong QJ, Kong S, Turtle JA, Ben-Nun M, Riley P, Riley S, Koyluoglu U, DesRoches D, Hamory B, Kyriakides C, Leis H, Milliken J, Moloney M, Morgan J, Ozcan G, Schrader C, Shakhnovich E, Siegel D, Spatz R, Stiefeling C, Wilkinson B, Wong A, Gao Z, Bian J, Cao W, Ferres JL, Li C, Liu TY, Xie X, Zhang S, Zheng S, Vespignani A, Chinazzi M, Davis JT, Mu K, y Piontti AP, Xiong X, Zheng A, Baek J, Farias V, Georgescu A, Levi R, Sinha D, Wilde J, Penna ND, Celi LA, Sundar S, Cavany S, España G, Moore S, Oidtman R, Perkins A, Osthus D, Castro L, Fairchild G, Michaud I, Karlen D, Lee EC, Dent J, Grantz KH, Kaminsky J, Kaminsky K, Keegan LT, Lauer SA, Lemaitre JC, Lessler J, Meredith HR, Perez-Saez J, Shah S, Smith CP, Truelove SA, Wills J, Kinsey M, Obrecht RF, Tallaksen K, Burant JC, Wang L, Gao L, Gu Z, Kim M, Li X, Wang G, Wang Y, Yu S, Reiner RC, Barber R, Gaikedu E, Hay S, Lim S, Murray C, Pigott D, Prakash BA, Adhikari B, Cui J, Rodríguez A, Tabassum A, Xie J, Keskinocak P, Asplund J, Baxter A, Oruc BE, Serban N, Arik SO, Dusenberry M, Epshteyn A, Kanal E, Le LT, Li CL, Pfister T, Sava D, Sinha R, Tsai T, Yoder N, Yoon J, Zhang L, Abbott S, Bosse NI, Funk S, Hellewel J, Meakin SR, Munday JD, Sherratt K, Zhou M, Kalantari R, Yamana TK, Pei S, Shaman J, Ayer T, Adee M, Chhatwal J, Dalgic OO, Ladd MA, Linas

BP, Mueller P, Xiao J, Li ML, Bertsimas D, Lami OS, Soni S, Bouardi HT, Wang Y, Wang Q, Xie S, Zeng D, Green A, Bien J, Hu AJ, Jahja M, Narasimhan B, Rajanala S, Rumack A, Simon N, Tibshirani R, Tibshirani R, Ventura V, Wasserman L, O'Dea EB, Drake JM, Pagano R, Walker JW, Slayton RB, Johansson M, Biggerstaff M, Reich NG (2021). "Evaluation of Individual and Ensemble Probabilistic Forecasts of COVID-19 Mortality in the US." *medRxiv*, p. 2021.02.03.21250974. doi:10.1101/2021.02.03.21250974.

Cramer E, Reich NG, Wang SY, Niemi J, Hannan A, House K, Gu Y, Xie S, Horstman S, aniruddhadiga, Walraven R, starkari, Li ML, Gibson G, Castro L, Karlen D, Wattanachit N, jinghuichen, zyt9lsb, aagarwal1996, Woody S, Ray E, Xu FT, Biegel H, GuidoEspana, X X, Bracher J, Lee E, har96, leyouz (2020). "COVID-19 Forecast Hub: 4 December 2020 Snapshot." doi:10.5281/zenodo.3963371.

Czado C, Gneiting T, Held L (2009). "Predictive Model Assessment for Count Data." *Biometrics*, **65**(4), 1254–1261. ISSN 1541-0420. doi:10.1111/j.1541-0420.2009.01191.x.

Dawid AP (1984). "Present Position and Potential Developments: Some Personal Views Statistical Theory the Prequential Approach." *Journal of the Royal Statistical Society: Series A (General)*, **147**(2), 278–290. ISSN 2397-2327. doi:10.2307/2981683.

Dowle M, Srinivasan A (2023). *data.table: Extension of 'data.frame'.* R package version 1.14.8, URL https://CRAN.R-project.org/package=data.table.

Elliott G, Timmermann A (2016). "Forecasting in Economics and Finance." *Annual Review of Economics*, **8**(1), 81–110. doi:10.1146/annurev-economics-080315-015346.

Epstein ES (1969). "A Scoring System for Probability Forecasts of Ranked Categories." *Journal of Applied Meteorology*, **8**(6), 985–987. ISSN 0021-8952. doi:10.1175/1520-0450(1969)008<0985:ASSFPF>2.0.CO;2.

European Covid-19 Forecast Hub (2021). "European Covid-19 Forecast Hub." https://covid19forecasthub.eu/.

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, van Leeuwen E, 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.

Funk S, Camacho A, Kucharski AJ, Lowe R, Eggo RM, Edmunds WJ (2019). "Assessing the Performance of Real-Time Epidemic Forecasts: A Case Study of Ebola in the Western Area Region of Sierra Leone, 2014-15." *PLOS Computational Biology*, **15**(2), e1006785. ISSN 1553-7358. doi:10.1371/journal.pcbi.1006785.

Gneiting T, Balabdaoui F, Raftery AE (2007). "Probabilistic Forecasts, Calibration and Sharpness." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **69**(2), 243–268. ISSN 1467-9868. doi:10.1111/j.1467-9868.2007.00587.x.

Gneiting T, Raftery AE (2005). "Weather Forecasting with Ensemble Methods." *Science*, **310**(5746), 248–249. ISSN 0036-8075, 1095-9203. doi:10.1126/science.1115255.

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.

Good IJ (1952). "Rational Decisions." *Journal of the Royal Statistical Society. Series B (Methodological)*, **14**(1), 107–114. ISSN 0035-9246. 2984087.

Hamill TM (2001). "Interpretation of Rank Histograms for Verifying Ensemble Forecasts." *Monthly Weather Review*, **129**(3), 550–560. ISSN 1520-0493, 0027-0644. doi:10.1175/1520-0493(2001)129<0550:IORHFV>2.0.CO;2.

Hamner B, Frasco M (2018). *Metrics: Evaluation Metrics for Machine Learning.* R package version 0.1.4, URL https://CRAN.R-project.org/package=Metrics.

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.

Jose VR (2009). "A Characterization for the Spherical Scoring Rule." *Theory and Decision*, **66**(3), 263–281. ISSN 1573-7187. doi:10.1007/s11238-007-9067-x.

Kuhn M, Vaughan D, Hvitfeldt E (2023a). *yardstick: Tidy Characterizations of Model Performance.* R package version 1.2.0, URL https://CRAN.R-project.org/package=yardstick.

Kuhn M, Vaughan D, Ruiz E (2023b). *probably: Tools for Post-Processing Class Probability Estimates.* R package version 1.0.2, URL https://CRAN.R-project.org/package=probably.

Kuhn M, Wickham H (2020). *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.* URL https://www.tidymodels.org.

Kukkonen J, Olsson T, Schultz DM, Baklanov A, Klein T, Miranda AI, Monteiro A, Hirtl M, Tarvainen V, Boy M, Peuch VH, Poupkou A, Kioutsioukis I, Finardi S, Sofiev M, Sokhi R, Lehtinen KEJ, Karatzas K, San José R, Astitha M, Kallos G, Schaap M, Reimer E, Jakobs H, Eben K (2012). "A Review of Operational, Regional-Scale, Chemical Weather Forecasting Models in Europe." *Atmospheric Chemistry and Physics*, **12**(1), 1–87. ISSN 1680-7316. doi:10.5194/acp-12-1-2012.

Laboratory NRA (2015). *verification: Weather Forecast Verification Utilities.* R package version 1.42, URL https://CRAN.R-project.org/package=verification.

Liboschik T, Fokianos K, Fried R (2017). "tscount: An R Package for Analysis of Count Time Series Following Generalized Linear Models." *Journal of Statistical Software*, **82**(5), 1–51. doi:10.18637/jss.v082.i05.

Machete RL (2012). "Contrasting Probabilistic Scoring Rules." *arXiv:1112.4530 [math, stat]*. 1112.4530.

Mann HB, Whitney DR (1947). "On a Test of Whether One of Two Random Variables Is Stochastically Larger than the Other." *The Annals of Mathematical Statistics*, **18**(1), 50–60. ISSN 0003-4851, 2168-8990. doi:10.1214/aoms/1177730491.

Matheson JE, Winkler RL (1976). "Scoring Rules for Continuous Probability Distributions." *Management Science*, **22**(10), 1087–1096. ISSN 0025-1909. doi:10.1287/mnsc.22.10.1087.

Merkle EC, Steyvers M (2013). "Choosing a Strictly Proper Scoring Rule." *Decision Analysis*, **10**, 292–304.

Meyer S, Held L, Höhle M (2017). "Spatio-Temporal Analysis of Epidemic Phenomena Using the R Package surveillance." *Journal of Statistical Software*, **77**(11), 1–55. doi:10.18637/jss.v077.i11.

Murphy AH (1971). "A Note on the Ranked Probability Score." *Journal of Applied Meteorology and Climatology*, **10**(1), 155–156. ISSN 1520-0450. doi:10.1175/1520-0450(1971)010<0155:ANOTRP>2.0.CO;2.

O'Hara-Wild M, Hyndman R, Wang E (2023). *fabletools: Core Tools for Packages in the 'fable' Framework*. R package version 0.3.4, URL https://CRAN.R-project.org/package=fabletools.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Reich NG, Brooks LC, Fox SJ, Kandula S, McGowan CJ, Moore E, Osthus D, Ray EL, Tushar A, Yamana TK, Biggerstaff M, Johansson MA, Rosenfeld R, Shaman J (2019). "A Collaborative Multiyear, Multimodel Assessment of Seasonal Influenza Forecasting in the United States." *Proceedings of the National Academy of Sciences*, **116**(8), 3146–3154. ISSN 0027-8424, 1091-6490. doi:10.1073/pnas.1812594116.

Rizopoulos D (2019). *cvGEE: Cross-Validated Predictions from GEE*. R package version 0.3-0, URL https://CRAN.R-project.org/package=cvGEE.

Rizopoulos D (2023). *GLMMadaptive: Generalized Linear Mixed Models using Adaptive Gaussian Quadrature*. R package version 0.9-0, URL https://CRAN.R-project.org/package=GLMMadaptive.

Sadatsafavi M, Safari A, Lee TY (2023). *predtools: Prediction Model Tools*. R package version 0.0.3, URL https://CRAN.R-project.org/package=predtools.

Siegert S (2020). *SpecsVerification: Forecast Verification Routines for Ensemble Forecasts of Weather and Climate*. R package version 0.5-3, URL https://CRAN.R-project.org/package=SpecsVerification.

Timmermann A (2018). "Forecasting Methods in Finance." *Annual Review of Financial Economics*, **10**(1), 449–479. doi:10.1146/annurev-financial-110217-022713.

Yan Y (2016). *MLmetrics: Machine Learning Evaluation Metrics*. R package version 1.1.1, URL https://CRAN.R-project.org/package=MLmetrics.

Zeileis A, Lang MN (2022). *topmodels: Infrastructure for Inference and Forecasting in Probabilistic Models*. R package version 0.1-0/r1498, URL `https://R-Forge.R-project.org/projects/topmodels/`.

Affiliation:

Nikos I. Bosse
London School of Hygiene & Tropical Medicine (LSHTM)
Centre for Mathematical Modelling of Infectious Diseases
London School of Hygiene & Tropical Medicine
Keppel Street
London WC1E 7HT
E-mail: `nikos.bosse@lshtm.ac.uk`
URL: `https://lshtm.ac.uk`

Hugo Gruson
LSHTM
Centre for Mathematical Modelling of Infectious Diseases
London School of Hygiene & Tropical Medicine
Keppel Street
London WC1E 7HT
E-mail: `hugo.gruson@lshtm.ac.uk`

Anne Cori
Imperial College London
MRC Centre for Global Infectious Disease Analysis, School of Public Health
Imperial College London
Norfolk Place
London W2 1PG
E-mail: `a.cori@imperial.ac.uk`

Edwin van Leeuwen
UK Health Security Agency, LSHTM
Statistics, Modelling and Economics Department
UK Health Security Agency
London NW9 5EQ
E-mail: `Edwin.VanLeeuwen@phe.gov.uk`

Sebastian Funk
LSHTM
Centre for Mathematical Modelling of Infectious Diseases
London School of Hygiene & Tropical Medicine
Keppel Street
London WC1E 7HT
E-mail: sebastian.funk@lshtm.ac.uk

Sam Abbott
LSHTM
Centre for Mathematical Modelling of Infectious Diseases
London School of Hygiene & Tropical Medicine
Keppel Street
London WC1E 7HT
E-mail: sam.abbott@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