

Package ‘hubEvals’

June 29, 2026

Title Tools for Scoring and Evaluating Hubverse Model Outputs

Version 0.3.0

Description Provides tools for scoring and evaluating 'hubverse' model outputs against observed data, wrapping scoring workflows from the 'scoringutils' package and bridging hubverse model output formats to 'scoringutils' forecast classes.

License MIT + file LICENSE

Encoding UTF-8

Imports cli, dplyr, hubUtils (>= 1.2.0), purrr, rlang, scoringutils (>= 2.2.0), tibble

RoxygenNote 7.3.3

URL <https://hubverse-org.github.io/hubEvals/>,
<https://github.com/hubverse-org/hubEvals>

BugReports <https://github.com/hubverse-org/hubEvals/issues>

Additional_repositories <https://hubverse-org.r-universe.dev>

Depends R (>= 4.1.0)

Config/Needs/website hubverse-org/hubStyle

Suggests hubExamples, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Anna Krystalli [aut, cre] (ORCID:

<https://orcid.org/0000-0002-2378-4915>),

Nicholas Reich [aut] (ORCID: <https://orcid.org/0000-0003-3503-9899>),

Evan Ray [aut],

Nikos Bosse [aut] (ORCID: <https://orcid.org/0000-0002-7750-5280>),

Kimberlyn Roosa [aut],

Zhian Kamvar [ctb] (ORCID: <https://orcid.org/0000-0003-1458-7108>),

Li Shandross [ctb] (ORCID: <https://orcid.org/0009-0008-1348-1954>),

Becky Sweger [ctb],

Lucie Contamin [ctb],

Consortium of Infectious Disease Modeling Hubs [cph]

Maintainer Anna Krystalli <annakrystalli@googlemail.com>

Repository CRAN

Date/Publication 2026-06-29 13:40:02 UTC

Contents

score_model_out	2
transform_pmf_model_out	7
transform_point_model_out	8
transform_quantile_model_out	9
transform_sample_model_out	10

Index	12
--------------	-----------

score_model_out	<i>Score model output predictions</i>
-----------------	---------------------------------------

Description

Scores model outputs with a single output_type against observed data.

Usage

```
score_model_out(
  model_out_tbl,
  oracle_output,
  metrics = NULL,
  relative_metrics = NULL,
  baseline = NULL,
  summarize = TRUE,
  by = "model_id",
  output_type_id_order = NULL,
  compound_taskid_set = NULL,
  transform = NULL,
  transform_append = FALSE,
  transform_label = NULL,
  ...
)
```

Arguments

model_out_tbl	Model output tibble with predictions
oracle_output	Predictions that would have been generated by an oracle model that knew the observed target data values in advance
metrics	Character vector of scoring metrics to compute. If NULL (the default), appropriate metrics are chosen automatically. See details for more.

relative_metrics	Character vector of scoring metrics for which to compute relative skill scores. The <code>relative_metrics</code> should be a subset of <code>metrics</code> and should only include proper scores. Interval coverage metrics (e.g., <code>"interval_coverage_90"</code>) and <code>"bias"</code> are not supported as relative metrics: interval coverage targets a nominal level rather than a "lower is better" direction, and bias is a signed quantity for which a geometric-mean pairwise ratio has no clean interpretation. If NULL (the default), no relative metrics will be computed. Relative metrics are only computed if <code>summarize = TRUE</code> , and require that <code>"model_id"</code> is included in <code>by</code> . If only one model is present in the data, relative-skill columns are filled with 1 (a model has trivial skill 1 relative to itself) rather than erroring.
baseline	String with the name of a model to use as a baseline for relative skill scores. If a baseline is given, then a scaled relative skill with respect to the baseline will be returned. By default (NULL), relative skill will not be scaled with respect to a baseline model.
summarize	Boolean indicator of whether summaries of forecast scores should be computed. Defaults to TRUE.
by	Character vector naming columns to summarize by. For example, specifying <code>by = "model_id"</code> (the default) will compute average scores for each model.
output_type_id_order	For ordinal variables in pmf format, this is a vector of levels for pmf forecasts, in increasing order of the levels. The order of the values for the <code>output_type_id</code> can be found by referencing the hub's <code>tasks.json</code> configuration file. For all output types other than pmf, this is ignored.
compound_taskid_set	When NULL (the default), sample forecasts are scored marginally (each modeling task scored independently). When a character vector of task ID column names is provided, it sets the compound grouping: the specified columns stay constant within each sample draw, while the remaining task ID dimensions vary within a draw and are scored jointly (using the energy score). Only applicable when <code>output_type == "sample"</code> . The value of <code>compound_taskid_set</code> can be found by referencing the hub's <code>tasks.json</code> configuration file.
transform	A function to apply as a scale transformation to both predictions and observations before scoring. Common choices include <code>log_shift</code> (recommended for log transformation as it handles zeros via an offset parameter), <code>sqrt</code> , or <code>log1p</code> . Avoid using <code>log</code> directly if data may contain zeros. If NULL (the default), no transformation is applied. Only supported for quantile, mean, and median output types.
transform_append	Logical. If FALSE (the default), scores are computed only on the transformed scale. If TRUE, scores are computed on both original and transformed scales, with a scale column distinguishing them. Ignored if <code>transform = NULL</code> .
transform_label	A character string label for the transformation (e.g., <code>"log"</code>). If NULL (the default), the label is auto-generated from the function name (e.g., <code>"log_shift"</code> for <code>scoringutils::log_shift</code>). Required when using an anonymous transform

function. Ignored if transform = NULL. Note: the label only appears in output when transform_append = TRUE, where it distinguishes transformed rows (labeled with this value) from original rows (labeled "natural") in the scale column.

... Additional arguments passed to the transform function. For example, allows use of the offset and base arguments of `scoringutils::log_shift()`. Ignored if transform = NULL.

Details

See the hubverse documentation for the expected format of the [oracle output data](#).

Default metrics are provided by the `scoringutils` package. You can select metrics by passing in a character vector of metric names to the `metrics` argument.

The following metrics can be selected (all are used by default) for the different `output_type`s:

Quantile forecasts: (`output_type == "quantile"`)

- wis
- overprediction
- underprediction
- dispersion
- bias
- ae_median
- "interval_coverage_XX": interval coverage at the "XX" level. For example, "interval_coverage_95" is the 95% interval coverage rate, which would be calculated based on quantiles at the probability levels 0.025 and 0.975.

See `scoringutils::get_metrics.forecast_quantile` for details.

Nominal forecasts: (`output_type == "pmf"` and `output_type_id_order` is NULL)

- log_score

See `scoringutils::get_metrics.forecast_nominal` for details.

Ordinal forecasts: (`output_type == "pmf"` and `output_type_id_order` is a vector)

- log_score
- rps

See `scoringutils::get_metrics.forecast_ordinal` for details.

Median forecasts: (`output_type == "median"`)

- ae_point: absolute error of the point forecast (recommended for the median, see Gneiting (2011))

See `scoringutils::get_metrics.forecast_point` for details.

Mean forecasts: (`output_type == "mean"`)

- se_point: squared error of the point forecast (recommended for the mean, see Gneiting (2011))

Sample forecasts (marginal): (output_type == "sample", compound_taskid_set = NULL)

- bias
- dss
- crps
- overprediction
- underprediction
- dispersion
- log_score
- mad
- ae_median
- se_mean

Note: log_score uses kernel density estimation, which may not be appropriate for integer-valued forecasts. scoringutils will warn when this is detected.

See [scoringutils::get_metrics.forecast_sample](#) for details.

Sample forecasts (compound): (output_type == "sample", compound_taskid_set provided)

- energy_score
- variogram_score

See [scoringutils::get_metrics.forecast_multivariate_sample](#) for details. The output includes a .mv_group_id column assigned by scoringutils to identify the multivariate groups used for scoring (equivalent to the compound_idx concept in the [hubverse sample output type documentation](#)). Correct scoring depends on providing the right compound_taskid_set from the hub's tasks.json configuration. If the specified grouping does not match the actual dependence structure of the submitted samples (e.g., because some models submitted coarser samples than configured), .mv_group_id may not correspond to the original sample draws as indicated by their output_type_id values.

See [scoringutils::add_relative_skill](#) for details on relative skill scores.

Value

A tibble of scores, inheriting from scoringutils' scores class so that downstream scoringutils helpers (e.g. [scoringutils::get_metrics\(\)](#)) continue to work. The tibble has a metrics attribute holding the names of the scoring rules that were applied.

References

Gneiting, Tilmann. 2011. "Making and Evaluating Point Forecasts." Journal of the American Statistical Association 106 (494): 746–62. doi:10.1198/jasa.2011.r10138.

Examples

```

# compute WIS and interval coverage rates at 80% and 90% levels based on
# quantile forecasts, summarized by the mean score for each model
quantile_scores <- score_model_out(
  model_out_tbl = hubExamples::forecast_outputs |>
    dplyr::filter(.data[["output_type"]] == "quantile"),
  oracle_output = hubExamples::forecast_oracle_output,
  metrics = c("wis", "interval_coverage_80", "interval_coverage_90"),
  relative_metrics = "wis",
  by = "model_id"
)
quantile_scores

# compute log scores based on pmf predictions for categorical targets,
# summarized by the mean score for each combination of model and location.
# Note: if the model_out_tbl had forecasts for multiple targets using a
# pmf output_type with different bins, it would be necessary to score the
# predictions for those targets separately.
pmf_scores <- score_model_out(
  model_out_tbl = hubExamples::forecast_outputs |>
    dplyr::filter(.data[["output_type"]] == "pmf"),
  oracle_output = hubExamples::forecast_oracle_output,
  metrics = c("log_score", "rps"),
  by = c("model_id", "location", "horizon"),
  output_type_id_order = c("low", "moderate", "high", "very high")
)
head(pmf_scores)

# Score sample forecasts marginally (each modeling task scored independently).
# Note: this data has compound structure (samples span horizons), but marginal
# scoring is still valid -- it evaluates each horizon independently.
sample_scores <- score_model_out(
  model_out_tbl = hubExamples::forecast_outputs |>
    dplyr::filter(.data[["output_type"]] == "sample"),
  oracle_output = hubExamples::forecast_oracle_output,
  metrics = "crps",
  by = "model_id"
)
sample_scores

# Score compound sample forecasts jointly using the energy score.
# compound_taskid_set specifies which task IDs stay constant within
# a sample group -- here, each sample draw spans all horizons for a
# given reference_date and location (i.e., a trajectory over time).
compound_scores <- score_model_out(
  model_out_tbl = hubExamples::forecast_outputs |>
    dplyr::filter(.data[["output_type"]] == "sample"),
  oracle_output = hubExamples::forecast_oracle_output,
  compound_taskid_set = c("reference_date", "location"),
  by = "model_id"
)
compound_scores

```

`transform_pmf_model_out`*Transform pmf model output into a forecast object*

Description

Transform pmf model output into a forecast object

Usage

```
transform_pmf_model_out(  
  model_out_tbl,  
  oracle_output,  
  output_type_id_order = NULL  
)
```

Arguments

`model_out_tbl` Model output tibble with predictions

`oracle_output` Predictions that would have been generated by an oracle model that knew the observed target data values in advance

`output_type_id_order`
For ordinal variables in pmf format, this is a vector of levels for pmf forecasts, in increasing order of the levels. The order of the values for the `output_type_id` can be found by referencing the hub's `tasks.json` configuration file. For all output types other than pmf, this is ignored.

Value

A `forecast_ordinal` object (when `output_type_id_order` is provided) or a `forecast_nominal` object (when `output_type_id_order` is `NULL`).

Examples

```
# Nominal pmf forecast (no output_type_id_order provided)  
pmf_outputs <- hubExamples::forecast_outputs |>  
  dplyr::filter(.data[["output_type"]] == "pmf")  
  
nominal_forecast <- transform_pmf_model_out(  
  model_out_tbl = pmf_outputs,  
  oracle_output = hubExamples::forecast_oracle_output  
)  
nominal_forecast  
  
# Ordinal pmf forecast (output_type_id_order provided)  
ordinal_forecast <- transform_pmf_model_out(  
  model_out_tbl = pmf_outputs,  
  oracle_output = hubExamples::forecast_oracle_output,  
  output_type_id_order = hubExamples::forecast_oracle_output_levels  
)  
ordinal_forecast
```

```

model_out_tbl = pmf_outputs,
oracle_output = hubExamples::forecast_oracle_output,
output_type_id_order = c("low", "moderate", "high", "very high")
)
ordinal_forecast

```

```
transform_point_model_out
```

Transform either mean or median model output into a point forecast object:

Description

Transform either mean or median model output into a point forecast object:

Usage

```
transform_point_model_out(model_out_tbl, oracle_output, output_type)
```

Arguments

model_out_tbl	Model output tibble with predictions
oracle_output	Predictions that would have been generated by an oracle model that knew the observed target data values in advance
output_type	Forecast output type: "mean" or "median"

Details

This function transforms a model output tibble in the Hubverse format (with either "mean" or "median" output type) to a scoringutils "point" forecast object

Value

forecast_point

Examples

```

# Median point forecast
median_forecast <- hubExamples::forecast_outputs |>
  dplyr::filter(.data[["output_type"]] == "median") |>
  transform_point_model_out(
    oracle_output = hubExamples::forecast_oracle_output,
    output_type = "median"
  )
median_forecast

# Mean point forecast

```

```
mean_forecast <- hubExamples::forecast_outputs |>
  dplyr::filter(.data[["output_type"]] == "mean") |>
  transform_point_model_out(
    oracle_output = hubExamples::forecast_oracle_output,
    output_type = "mean"
  )
mean_forecast
```

transform_quantile_model_out

Transform quantile model output into a forecast object

Description

Transform quantile model output into a forecast object

Usage

```
transform_quantile_model_out(model_out_tbl, oracle_output)
```

Arguments

`model_out_tbl` Model output tibble with predictions

`oracle_output` Predictions that would have been generated by an oracle model that knew the observed target data values in advance

Value

forecast_quantile

Examples

```
quantile_forecast <- hubExamples::forecast_outputs |>
  dplyr::filter(.data[["output_type"]] == "quantile") |>
  transform_quantile_model_out(
    oracle_output = hubExamples::forecast_oracle_output
  )
quantile_forecast
```

`transform_sample_model_out`*Transform sample model output into a forecast object*

Description

Transform sample model output into a forecast object

Usage

```
transform_sample_model_out(  
  model_out_tbl,  
  oracle_output,  
  compound_taskid_set = NULL  
)
```

Arguments

`model_out_tbl` Model output tibble with predictions

`oracle_output` Predictions that would have been generated by an oracle model that knew the observed target data values in advance

`compound_taskid_set`

Character vector of task ID column names that stay constant within each sample draw (i.e., define the compound modeling task grouping). When NULL (the default), each modeling task is scored independently (marginal scoring). When provided, sample draws are treated as joint predictions across the task ID dimensions **not** in `compound_taskid_set`, and multivariate scoring metrics are used.

Value

A `forecast_sample` object (when `compound_taskid_set` is NULL) or a `forecast_multivariate_sample` object (when `compound_taskid_set` is provided).

Examples

```
# Marginal sample forecast: each modeling task scored independently  
sample_forecast <- hubExamples::forecast_outputs |>  
  dplyr::filter(.data[["output_type"]] == "sample") |>  
  transform_sample_model_out(  
    oracle_output = hubExamples::forecast_oracle_output  
  )  
sample_forecast  
  
# Compound sample forecast: jointly score across non-compound task IDs  
compound_forecast <- hubExamples::forecast_outputs |>  
  dplyr::filter(.data[["output_type"]] == "sample") |>  
  transform_sample_model_out(  
    oracle_output = hubExamples::forecast_oracle_output  
  )  
compound_forecast
```

```
    oracle_output = hubExamples::forecast_oracle_output,  
    compound_taskid_set = c("reference_date", "location")  
  )  
compound_forecast
```

Index

log_shift, 3

score_model_out, 2

scoringutils::add_relative_skill, 5

scoringutils::get_metrics(), 5

scoringutils::get_metrics.forecast_multivariate_sample,
5

scoringutils::get_metrics.forecast_nominal,
4

scoringutils::get_metrics.forecast_ordinal,
4

scoringutils::get_metrics.forecast_point,
4

scoringutils::get_metrics.forecast_quantile,
4

scoringutils::get_metrics.forecast_sample,
5

scoringutils::log_shift(), 4

transform_pmf_model_out, 7

transform_point_model_out, 8

transform_quantile_model_out, 9

transform_sample_model_out, 10