Package 'AMRvalid'

November 25, 2026

Title Population-Aware Evaluation of Antimicrobial Resistance Predictions

Version 1.0.0

Description Evaluates antimicrobial resistance phenotype predictions made by external models and rule-based tools against phenotypic reference results, accounting for the population structure of bacterial isolates. Provides a documented data contract and input validation; classification and probability metrics and antimicrobial susceptibility testing agreement metrics (categorical agreement, very major, major and minor errors, essential agreement) per species and antimicrobial; checks for overlap between training and test isolates; performance by genomic distance to the training data; cluster bootstrap intervals and paired model comparisons; calibration; prevalence-adjusted predictive values; and reports aligned with the TRIPOD+AI checklist (Collins et al. (2024) <doi:10.1136/bmj-2023-078378>). Organism and antimicrobial identifiers use the classes of the 'AMR' package. Models are not trained inside the package.

License MIT + file LICENSE

URL https://github.com/omarfaruqe/AMRvalid

BugReports https://github.com/omarfaruqe/AMRvalid/issues

Depends R (>= 4.1.0)

Imports AMR (>= 3.0.0), stats, utils

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Md Omar Faruqe [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1226-7226>)

Maintainer Md Omar Faruqe <faruqe@ru.ac.bd>

Repository CRAN

Date/Publication 2026-11-25 12:20:18 UTC

Contents

amr_ast_agreement 2
amr_bootstrap 4
amr_calibration 6
amr_contract 7
amr_contract_schema 8
amr_data_summary 8
amr_distance_eval 9
amr_evaluate 11
amr_example_data 12
amr_import_predictions 14
amr_metrics 15
amr_nearest_train_distance 17
amr_paired_compare 18
amr_ppv_npv 19
amr_prop_ci 20
amr_provenance 21
amr_report 22
amr_simulate 23
amr_validate 25
amr_validate_splits 25
plot.amr_bootstrap 26
plot.amr_calibration 27
plot.amr_distance_eval 28
plot.amr_eval 29
plot.amr_paired_compare 29

Index **31**

`amr_ast_agreement` *Antimicrobial susceptibility testing agreement metrics*

Description

Computes the agreement metrics used to compare a susceptibility test (or a genome-based prediction) with a reference method.

Usage

```
amr_ast_agreement(
  x,
  threshold = NULL,
  vme_denominator = c("class", "all"),
  ci = c("wilson", "clopper-pearson"),
  conf = 0.95,
  weight_by = NULL,
  split = "test"
)
```

Arguments

x	An amr_data object.
threshold	Probability at or above which a score is called R. Required for classification metrics when predictions are scores; there is deliberately no default of 0.5. Choose it on training or validation data, never on the test set.
vme_denominator	Denominator for very major and major error rates: "class" (default) divides very major errors by reference-resistant isolates and major errors by reference-susceptible isolates; "all" divides both by all evaluable isolates. Guidance documents differ, so always report which one you used.
ci	Interval method for proportions, passed to amr_prop_ci().
conf	Confidence level.
weight_by	Optional column for cluster-weighted rates; see amr_metrics().
split	Which partition to evaluate; see amr_metrics().

Details

Category metrics compare predicted and reference categories (S, SDD, I, R); NI and missing values are excluded:

- **Categorical agreement (CA)**: predicted category equals reference.
- **Very major error (VME)**: predicted S, reference R.
- **Major error (ME)**: predicted R, reference S.
- **Minor error (mE)**: one of the two is I or SDD and the other is S or R. The denominator is all evaluable isolates.

For scores, the category is R when the probability is at or above threshold, otherwise S.

MIC metrics need predicted_mic and truth_mic:

- **Essential agreement (EA)**: predicted MIC within one doubling dilution of the reference MIC. Off-scale reference values are handled conservatively: a reference of <=a agrees with predictions <= 2a, and >a agrees with predictions >= a. Predicted off-scale values are compared at their bound.
- **Bias**: mean of log2(predicted / reference) over on-scale pairs; positive values mean predictions are too high.

Acceptance thresholds (for example from ISO 20776-2 or regulatory guidance) are not applied here; compare the interval bounds against the criteria that apply to your setting.

Value

A data frame with columns `metric`, `estimate`, `lower`, `upper`, `numerator` and `denominator`.

Examples

```
preds <- data.frame(
  isolate_id = sprintf("iso%02d", 1:6),
  species = "E. coli",
  drug = "ciprofloxacin",
  truth = c("R", "R", "I", "S", "S", "S"),
  prediction = c("R", "S", "R", "S", "R", "S")
)
amr_ast_agreement(amr_import_predictions(preds))
```

---

amr_bootstrap *Group-aware bootstrap intervals*

Description

Resamples whole groups (isolates, lineages or genomic clusters) with replacement and recomputes metrics in every replicate. Resampling rows as if isolates were independent understates uncertainty when related isolates share outcomes; resampling clusters respects that dependence.

Usage

```
amr_bootstrap(
  x,
  resample_by = "isolate_id",
  B = 2000,
  threshold = NULL,
  intermediate = c("exclude", "susceptible", "resistant"),
  conf = 0.95,
  seed = NULL,
  split = "test",
  min_groups = 20,
  group = NULL
)
```

Arguments

`x`	An `amr_data` object.
`resample_by`	Column defining the resampling unit, e.g. `"isolate_id"` (default), `"lineage"` or `"genomic_cluster"`. Rows with a missing group label are treated as their own single-isolate groups.

B	Number of bootstrap replicates.
threshold	Probability at or above which a score is called R. Required for classification metrics when predictions are scores; there is deliberately no default of 0.5. Choose it on training or validation data, never on the test set.
intermediate	How to treat I and SDD in binary metrics, for both reference and prediction: "exclude" (default) drops those rows, "susceptible" counts them as S and "resistant" as R. NI results are always excluded.
conf	Confidence level.
seed	Optional random seed for reproducibility. The global random state is restored afterwards.
split	Which partition to evaluate when a split column exists. Defaults to "test", so training predictions are not reported as performance by accident. Use NULL to evaluate all rows.
min_groups	Strata with fewer independent groups than this are flagged: cluster bootstrap intervals are unreliable with few clusters.
group	Deprecated; use resample_by.

Details

Metrics are computed on S/R categories after applying the intermediate policy, so the very major error rate equals 1 - sensitivity and the major error rate equals 1 - specificity. Minor errors are not bootstrapped. Intervals are percentile intervals; n_valid counts the replicates in which a metric was defined (for example, sensitivity is undefined in a replicate that drew no resistant isolates). Treat intervals with many invalid replicates with caution.

When every replicate gives the same value (typically a sensitivity of 1 or a very major error rate of 0), the percentile interval has zero width and cannot express uncertainty. Such rows are marked zero_width; use the Wilson or Clopper-Pearson intervals from amr_evaluate() for them instead.

Value

An object of class amr_bootstrap: a data frame with the stratum columns, metric, estimate, lower, upper, n_groups, B, n_valid, few_groups and zero_width.

Examples

```
set.seed(2)
preds <- data.frame(
  isolate_id = sprintf("iso%02d", 1:30),
  species = "K. pneumoniae",
  drug = "meropenem",
  truth = rep(c("R", "S", "S"), 10),
  probability = runif(30),
  lineage = rep(sprintf("ST%d", 1:10), each = 3)
)
amr_bootstrap(amr_import_predictions(preds), resample_by = "lineage", B = 200,
              threshold = 0.5, seed = 1)
```

`amr_calibration` *Calibration of predicted probabilities*

Description

Assesses whether predicted probabilities of resistance match observed frequencies. Discrimination (AUC) and calibration are different properties: a model can rank isolates well while its probabilities are systematically too high or too low, which matters whenever the probability itself is used, for example to set a threshold or to report a risk.

Usage

```
amr_calibration(
  x,
  bins = 10,
  binning = c("equal_width", "equal_count"),
  by = NULL,
  intermediate = c("exclude", "susceptible", "resistant"),
  conf = 0.95,
  split = "test",
  min_events = 10
)
```

Arguments

`x`	An `amr_data` object.
`bins`	Number of bins for the binned reliability curve and the expected calibration error.
`binning`	`"equal_width"` (default; bins of width 1 / `bins` on the probability scale) or `"equal_count"` (quantile bins with similar numbers of isolates).
`by`	Optional column name to evaluate calibration separately for each value, in addition to the stratum (e.g. `"lineage"` or a distance band column you have added).
`intermediate`	How to treat I and SDD in binary metrics, for both reference and prediction: `"exclude"` (default) drops those rows, `"susceptible"` counts them as S and `"resistant"` as R. NI results are always excluded.
`conf`	Confidence level.
`split`	Which partition to evaluate when a `split` column exists. Defaults to `"test"`, so training predictions are not reported as performance by accident. Use `NULL` to evaluate all rows.
`min_events`	Strata with fewer resistant or fewer susceptible isolates than this are flagged as too sparse for stable estimates.

Details

Reported per stratum (model, dataset, species, antimicrobial):

- **mean predicted** and **observed** proportion resistant, and their ratio **O/E** (observed / expected; 1 is ideal);
- **calibration intercept** (calibration-in-the-large): the intercept of a logistic model of the outcome with the logit of the predicted probability as an offset; 0 is ideal, positive means predictions are too low on average;
- **calibration slope**: the coefficient of a logistic regression of the outcome on the logit of the predicted probability; 1 is ideal, below 1 means predictions are too extreme (over-confident);
- **expected calibration error (ECE)**: the count-weighted mean absolute difference between mean predicted and observed proportion across bins, and the **maximum calibration error (MCE)**. Both depend on the binning and are secondary to the slope and intercept.

Probabilities are clipped to [1e-6, 1 - 1e-6] before taking logits. Intervals for the intercept and slope are Wald intervals assuming independent isolates.

The `curve` element holds a loess-smoothed reliability curve (drawn when there are at least 20 isolates and 10 distinct predicted values) and the `bins` element the binned points with Wilson intervals; `plot()` draws both.

Value

An object of class `amr_calibration`: a list with `summary` (one row per stratum), `bins` and `curve`.

References

Van Calster B, McLernon DJ, van Smeden M, Wynants L, Steyerberg EW (2019). Calibration: the Achilles heel of predictive analytics. *BMC Medicine* 17:230.

Examples

```
sim <- amr_simulate(n_lineages = 20, isolates_per_lineage = 10,
                    external_lineages = 20, return_distances = FALSE, seed = 1)
x <- amr_import_predictions(sim$predictions)
cal <- amr_calibration(x[x$dataset_id == "external", ])
cal
plot(cal)
```

---

amr_contract *The AMRvalid data contract*

Description

Returns the field specification that every prediction table must follow after import. One row is one prediction for one isolate, one antimicrobial, one model and one evaluation context (dataset, split and fold).

Usage

```
amr_contract()
```

Value

A data frame with columns `field`, `requirement`, `type` and `description`.

Examples

```
amr_contract()
```

---

amr_contract_schema *Path to the JSON Schema of the data contract*

Description

The schema lets workflows in other languages (for example Python) export prediction tables that AMRvalid can import without changes.

Usage

```
amr_contract_schema()
```

Value

Path to `amrvalid-contract.json` inside the installed package.

Examples

```
amr_contract_schema()
```

---

amr_data_summary *Summarise an AMRvalid prediction table*

Description

Counts predictions and reference phenotypes per model, dataset, species and antimicrobial.

Usage

```
amr_data_summary(x)
```

Arguments

`x`	An `amr_data` object.

Value

A data frame with one row per stratum.

Examples

```
preds <- data.frame(
  isolate_id = c("a", "b", "c", "d"),
  species = "K. pneumoniae",
  drug = c("meropenem", "meropenem", "ciprofloxacin", "ciprofloxacin"),
  truth = c("R", "S", "I", "R"),
  prediction = c("R", "S", "R", "R")
)
amr_data_summary(amr_import_predictions(preds))
```

---

amr_distance_eval *Evaluate performance as a function of distance to the training set*

Description

Assigns each test isolate the distance to its nearest training isolate, groups test isolates into distance bands and evaluates each band per species-antimicrobial stratum. A trend model then summarises how fast performance decays as isolates become less similar to anything the model was trained on.

Usage

```
amr_distance_eval(
  x,
  distances,
  train_manifest = NULL,
  bands = 3,
  threshold = NULL,
  intermediate = c("exclude", "susceptible", "resistant"),
  ci = c("wilson", "clopper-pearson"),
  conf = 0.95,
  vme_denominator = c("class", "all"),
  target_prevalence = NULL,
  weight_by = NULL
)
```

Arguments

x	An amr_data object. If it has a split column, rows with split == "test" are the evaluation set and rows with split == "train" are training isolates (unless train_manifest is given). Without a split column every row is treated as test.
distances	Pairwise distances: a matrix or long table, see amr_nearest_train_distance(). Alternatively a data frame with isolate_id and nearest_distance (and optionally model_id) if nearest-training distances were computed upstream.

train_manifest	Optional data frame listing training isolates, with an isolate_id column and optionally model_id (if absent, the manifest applies to every model) and the grouping, date and country columns to compare.
bands	Either a single number of quantile-based bands (default 3) or a numeric vector of band boundaries, e.g. c(0, 10, 100, Inf) for SNP distances.
threshold	Probability at or above which a score is called R. Required for classification metrics when predictions are scores; there is deliberately no default of 0.5. Choose it on training or validation data, never on the test set.
intermediate	How to treat I and SDD in binary metrics, for both reference and prediction: "exclude" (default) drops those rows, "susceptible" counts them as S and "resistant" as R. NI results are always excluded.
ci	Interval method for proportions, passed to amr_prop_ci().
conf	Confidence level.
vme_denominator, target_prevalence, weight_by	Passed to amr_evaluate() for the metrics in each band; weight_by also weights the trend model.

Details

The trend summary depends on what the predictions allow:

- with categories (binary, S/I/R, or scores with a threshold): a logistic regression of correct classification on distance, reported as the odds ratio per standard deviation of distance;
- with scores and no threshold: a linear regression of the squared error (the per-isolate Brier score) on distance, reported as the change per standard deviation.

With weight_by, band metrics are cluster-weighted (see amr_metrics()) and the trend model weights each cluster equally, so its standard error reflects the number of clusters rather than isolates. Calibration rows are not weighted.

For score predictions, each band also reports the calibration intercept and slope (see amr_calibration()), showing whether a model becomes over-confident on isolates unlike its training data.

An odds ratio below 1 (or a positive Brier slope) means performance worsens with distance. Intervals are Wald intervals that treat isolates as independent; use them as a description, not a test.

Value

An object of class amr_distance_eval: a list with nearest (per-isolate distances and bands), by_band (metrics per band, in the format of amr_evaluate()) and trend (one row per stratum).

Examples

```
set.seed(1)
ids <- sprintf("iso%02d", 1:40)
preds <- data.frame(
  isolate_id = ids,
  species = "E. coli",
  drug = "ciprofloxacin",
  truth = rep(c("R", "S"), 20),
```

```
  probability = runif(40),
  split = rep(c("train", "test"), each = 20)
)
dist <- as.matrix(stats::dist(seq_along(ids)))
dimnames(dist) <- list(ids, ids)
amr_distance_eval(amr_import_predictions(preds), dist, threshold = 0.5)
```

`amr_evaluate` *Evaluate predictions per stratum*

Description

Splits an `amr_data` object by model, dataset, species and antimicrobial (and optionally further columns) and computes `amr_metrics()` and, where possible, `amr_ast_agreement()` in each stratum. There is no pooled headline score.

Usage

```
amr_evaluate(
  x,
  threshold = NULL,
  intermediate = c("exclude", "susceptible", "resistant"),
  vme_denominator = c("class", "all"),
  ci = c("wilson", "clopper-pearson"),
  conf = 0.95,
  split = "test",
  validate = TRUE,
  target_prevalence = NULL,
  weight_by = NULL,
  by = NULL
)
```

Arguments

`x`	An `amr_data` object.
`threshold`	Probability at or above which a score is called R. Required for classification metrics when predictions are scores; there is deliberately no default of 0.5. Choose it on training or validation data, never on the test set.
`intermediate`	How to treat I and SDD in binary metrics, for both reference and prediction: `"exclude"` (default) drops those rows, `"susceptible"` counts them as S and `"resistant"` as R. NI results are always excluded.
`vme_denominator`	Denominator for very major and major error rates: `"class"` (default) divides very major errors by reference-resistant isolates and major errors by reference-susceptible isolates; `"all"` divides both by all evaluable isolates. Guidance documents differ, so always report which one you used.
`ci`	Interval method for proportions, passed to `amr_prop_ci()`.

conf	Confidence level.
split	Which partition to evaluate when a split column exists. Defaults to "test", so training predictions are not reported as performance by accident. Use NULL to evaluate all rows.
validate	If TRUE (default), run amr_validate() first and stop if it reports errors.
target_prevalence, weight_by	Passed to amr_metrics() (and weight_by to amr_ast_agreement()): PPV and NPV at a target prevalence, and cluster-weighted metrics.
by	Optional column name(s) to evaluate subgroups within each stratum, e.g. "country" or a period column you have added. Missing values form their own subgroup, labelled "(missing)". Factor columns keep their level order in the output. Subgroup results are descriptive: small subgroups give wide intervals, and differences between subgroups are not evidence of a cause.

Value

An object of class amr_eval: a long data frame with the stratum columns (and any by columns), species_name, drug_name, family ("ml" or "ast") and the metric columns.

Examples

```
preds <- data.frame(
  isolate_id = sprintf("iso%02d", 1:8),
  species = "K. pneumoniae",
  drug = rep(c("meropenem", "ciprofloxacin"), each = 4),
  truth = c("R", "S", "S", "R", "R", "R", "S", "S"),
  probability = c(0.9, 0.2, 0.4, 0.7, 0.8, 0.3, 0.1, 0.6)
)
x <- amr_import_predictions(preds)
amr_evaluate(x, threshold = 0.5)

# subgroups, e.g. by country of isolation
x$country <- c("CA", "CA", "US", "US", "CA", "US", "CA", "US")
amr_evaluate(x, threshold = 0.5, by = "country")
```

---

amr_example_data *Example data: rule-based predictions for Klebsiella pneumoniae*

Description

Real, public data for trying AMRvalid: laboratory susceptibility results for meropenem and ciprofloxacin in *Klebsiella pneumoniae* isolates from NCBI Pathogen Detection, with predictions from two transparent rule-based predictors per antimicrobial applied to the AMRFinderPlus genotypes.

Usage

```
amr_example_data()
```

Details

Source: NCBI Pathogen Detection, *Klebsiella* analysis PDG000000012.2568 (metadata and SNP cluster files, downloaded 2 November 2026). Isolates were kept if NCBI labelled them *K. pneumoniae* and they had a meropenem or ciprofloxacin phenotype. Breakpoint standards are those of the submitting laboratories and are not recorded consistently, so the data are suitable for demonstrating methods, not for drawing clinical conclusions.

Rules applied to the AMRFinderPlus calls (calls flagged MISTRANSLATION ignored):

- **carbapenemase genes** (meropenem): any KPC, NDM, VIM, IMP or OXA-48-like gene;
- **carbapenemase or porin + ESBL/AmpC** (meropenem): the above, or an OmpK35/OmpK36 alteration together with a CTX-M, DHA, CMY, ACT or FOX gene;
- **QRDR mutations** (ciprofloxacin): gyrA S83/D87 or parC S80/E84 substitutions;
- **QRDR mutations or PMQR genes** (ciprofloxacin): the above, or a qnr, aac(6')-Ib-cr or qepA gene.

These are illustrative textbook rules, not recommendations. The script that builds the data, `data-raw/ncbi_kpneumoniae.R`, is in the package source repository.

Value

A data frame ready for `amr_import_predictions()`, with one row per isolate, antimicrobial and predictor:

isolate_id NCBI BioSample accession.

species Always "Klebsiella pneumoniae".

drug "meropenem" or "ciprofloxacin".

truth Reference category (S, I or R) as submitted to BioSample.

prediction Predicted category (S or R) from the rule.

model_id The rule (see Details).

dataset_id Source and NCBI analysis version.

genomic_cluster NCBI SNP cluster (PDS accession); empty when the isolate is not in a cluster.

bioproject BioProject accession (the submitting study).

collection_year, collection_date Year, and full date when available.

country Country of isolation.

amrfinder_version AMRFinderPlus version used by NCBI.

Source

NCBI Pathogen Detection, https://www.ncbi.nlm.nih.gov/pathogens/. Data produced by the U.S. National Center for Biotechnology Information are in the public domain.

Examples

```
d <- amr_example_data()
head(d)
x <- amr_import_predictions(d)
x
```

`amr_import_predictions` *Import predictions into the AMRvalid data contract*

Description

Renames columns, coerces identifiers to AMR package classes, converts predictions to a common representation and detects the prediction type. The function never drops rows; problems are reported by `amr_validate()`.

Usage

```
amr_import_predictions(
  data,
  mapping = NULL,
  prediction_type = NULL,
  source = "custom",
  strict = TRUE
)
```

Arguments

`data`	A data frame with one row per prediction.
`mapping`	Optional named character vector mapping contract field names to column names in `data`, e.g. `c(isolate_id = "Sample", truth_sir = "phenotype")`.
`prediction_type`	One of `"binary"`, `"score"`, `"sir"` or `"mic"`. If `NULL` (default) it is detected from the columns present.
`source`	Short label for where the predictions came from (recorded in the `source` attribute).
`strict`	If `TRUE` (default), species and antimicrobial names must match an AMR package name, code, abbreviation or synonym exactly (ignoring case and punctuation). Approximate matches are set to `NA`, which `amr_validate()` reports as errors. The AMR package matches names fuzzily, so without this check a typo can silently become a different antimicrobial. If `FALSE`, approximate matches are kept and reported as warnings.

Details

A column called `truth` is treated as `truth_sir`. Predicted classes may be given as S/I/R text, `0`/`1` or `FALSE`/`TRUE` (where 1 or `TRUE` means resistant). Missing `model_id` and `dataset_id` columns are filled with `"model_1"` and `"dataset_1"`.

The `coercion` attribute of the result lists every distinct input name, the code it was matched to and whether the match was exact.

Detection order for `prediction_type`: `predicted_mic` gives `"mic"`; otherwise `probability` gives `"score"`; otherwise a `prediction` column containing I or SDD gives `"sir"`; otherwise `"binary"`.

Value

A data frame of class amr_data.

Examples

```
preds <- data.frame(
  isolate_id = c("a", "b", "c"),
  species = "K. pneumoniae",
  drug = "meropenem",
  truth = c("R", "S", "S"),
  probability = c(0.9, 0.2, 0.6)
)
amr_import_predictions(preds)
```

---

amr_metrics *Classification and probability metrics for one set of predictions*

Description

Computes machine-learning performance metrics for the rows of an amr_data object, treating resistance (R) as the positive class. Use amr_evaluate() to compute them per species-antimicrobial stratum; pooling strata is rarely meaningful.

Usage

```
amr_metrics(
  x,
  threshold = NULL,
  intermediate = c("exclude", "susceptible", "resistant"),
  ci = c("wilson", "clopper-pearson"),
  conf = 0.95,
  target_prevalence = NULL,
  weight_by = NULL,
  split = "test"
)
```

Arguments

x	An amr_data object.
threshold	Probability at or above which a score is called R. Required for classification metrics when predictions are scores; there is deliberately no default of 0.5. Choose it on training or validation data, never on the test set.
intermediate	How to treat I and SDD in binary metrics, for both reference and prediction: "exclude" (default) drops those rows, "susceptible" counts them as S and "resistant" as R. NI results are always excluded.
ci	Interval method for proportions, passed to amr_prop_ci().

`conf` Confidence level.

`target_prevalence`
Optional proportion resistant in the setting where the predictor will be used. Adds `ppv_at_target_prevalence` and `npv_at_target_prevalence`, computed from sensitivity and specificity, because PPV and NPV depend on prevalence and public collections often over-represent resistant isolates.

`weight_by` Optional column (e.g. `"genomic_cluster"` or `"lineage"`) for cluster-weighted metrics: every group contributes a total weight of 1, shared equally by its isolates, so a large outbreak clone counts as much as a single sporadic isolate. Rows with a missing label are their own group.

`split` Which partition to evaluate when a `split` column exists. Defaults to `"test"`, so training predictions are never counted as performance by accident. Use `NULL` to evaluate all rows.

Details

ROC-AUC is computed with the Mann-Whitney statistic (ties count one half) and its interval with the DeLong variance. Row-level intervals assume isolates are independent, which is optimistic when isolates are clonally related.

Prevalence-adjusted values use PPV = sens x p / (sens x p + (1 - spec) x (1 - p)) and NPV = spec x (1 - p) / (spec x (1 - p) + (1 - sens) x p). Both increase with sensitivity and specificity, so their bounds combine the lower (and upper) bounds of the two; these bounds are conservative.

With `weight_by`, numerators and denominators are sums of weights, and proportion intervals use the Kish effective sample size (sum of weights squared / sum of squared weights). The AUC becomes a weighted Mann-Whitney statistic without a closed-form interval.

Value

A data frame with one row per metric and columns `metric`, `estimate`, `lower`, `upper`, `numerator` and `denominator`. Metrics without a closed-form interval (balanced accuracy, F1, MCC, Brier score, log loss) have `NA` bounds; use `amr_bootstrap()` for group-aware intervals.

References

DeLong ER, DeLong DM, Clarke-Pearson DL (1988). Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics* 44:837-845.

Examples

```
preds <- data.frame(
  isolate_id = sprintf("iso%02d", 1:8),
  species = "K. pneumoniae",
  drug = "meropenem",
  truth = c("R", "R", "R", "S", "S", "S", "S", "R"),
  probability = c(0.91, 0.75, 0.30, 0.10, 0.20, 0.65, 0.05, 0.88),
  lineage = c("ST258", "ST258", "ST258", "ST11", "ST15", "ST15", "ST307", "ST147")
)
x <- amr_import_predictions(preds)
```

```
amr_metrics(x, threshold = 0.5)
# PPV and NPV where 10% of isolates are resistant
amr_metrics(x, threshold = 0.5, target_prevalence = 0.1)
# each lineage counts once
amr_metrics(x, threshold = 0.5, weight_by = "lineage")
```

`amr_nearest_train_distance`

Nearest-training distance for each test isolate

Description

For every test isolate, finds the genomically closest training isolate using distances computed upstream (for example Mash, SKA, core-genome SNP or cgMLST allele distances). AMRvalid does not compute distances.

Usage

```
amr_nearest_train_distance(test_ids, train_ids, distances)
```

Arguments

`test_ids`	Character vector of test isolate identifiers.
`train_ids`	Character vector of training isolate identifiers.
`distances`	Either a square numeric matrix with isolate identifiers as row and column names, or a data frame with columns `isolate_1`, `isolate_2` and `distance` (pairs may appear in either order).

Value

A data frame with `isolate_id`, `nearest_train_id` and `distance`. Test isolates with no distance to any training isolate get `NA`. A test isolate that is itself in the training set gets distance 0.

Examples

```
m <- matrix(c(0, 2, 9,
              2, 0, 7,
              9, 7, 0), 3, dimnames = list(c("a", "b", "c"), c("a", "b", "c")))
amr_nearest_train_distance(c("b", "c"), "a", m)
```

`amr_paired_compare` *Paired comparison of two models on the same isolates*

Description

Restricts both models to the isolates they share in each stratum and bootstraps the difference in each metric, resampling the same groups for both models so the pairing is kept in every replicate.

Usage

```
amr_paired_compare(
  x,
  models = NULL,
  resample_by = "isolate_id",
  B = 2000,
  threshold = NULL,
  intermediate = c("exclude", "susceptible", "resistant"),
  conf = 0.95,
  seed = NULL,
  split = "test",
  min_groups = 20,
  group = NULL
)
```

Arguments

`x`	An `amr_data` object.
`models`	Character vector of two `model_id` values. Differences are reported as `models[2] - models[1]`. If `NULL`, the data must contain exactly two models.
`resample_by`	Column defining the resampling unit, e.g. `"isolate_id"` (default), `"lineage"` or `"genomic_cluster"`. Rows with a missing group label are treated as their own single-isolate groups.
`B`	Number of bootstrap replicates.
`threshold`	Probability threshold for scores: a single number, or a named numeric vector with one threshold per model.
`intermediate`	How to treat I and SDD in binary metrics, for both reference and prediction: `"exclude"` (default) drops those rows, `"susceptible"` counts them as S and `"resistant"` as R. NI results are always excluded.
`conf`	Confidence level.
`seed`	Optional random seed for reproducibility. The global random state is restored afterwards.
`split`	Which partition to evaluate when a `split` column exists. Defaults to `"test"`, so training predictions are not reported as performance by accident. Use `NULL` to evaluate all rows.

min_groups	Strata with fewer independent groups than this are flagged: cluster bootstrap intervals are unreliable with few clusters.
group	Deprecated; use resample_by.

Details

Besides the bootstrap intervals, an exact McNemar test compares the discordant pairs of correct and incorrect classifications. It assumes independent isolates, so prefer the group-aware intervals when isolates are clustered.

Value

An object of class amr_paired_compare: a data frame with dataset_id, species, drug, metric, the estimate for each model (model_a, model_b), difference, lower, upper, n_matched, n_groups, n_valid and mcnemar_p (on the accuracy row).

Examples

```
set.seed(3)
ids <- sprintf("iso%02d", 1:40)
truth <- rep(c("R", "S"), 20)
preds <- rbind(
  data.frame(isolate_id = ids, truth = truth, model_id = "old",
             probability = ifelse(truth == "R", 0.6, 0.4) + rnorm(40, 0, 0.2)),
  data.frame(isolate_id = ids, truth = truth, model_id = "new",
             probability = ifelse(truth == "R", 0.8, 0.2) + rnorm(40, 0, 0.1))
)
preds$probability <- pmin(pmax(preds$probability, 0), 1)
preds$species <- "E. coli"
preds$drug <- "ciprofloxacin"
amr_paired_compare(amr_import_predictions(preds), models = c("old", "new"),
                   threshold = 0.5, B = 200, seed = 1)
```

---

amr_ppv_npv *PPV and NPV at a given prevalence*

Description

Converts sensitivity and specificity into the positive and negative predictive values expected where the proportion of resistant isolates is prevalence.

Usage

```
amr_ppv_npv(sensitivity, specificity, prevalence)
```

Arguments

sensitivity, specificity	Numeric vectors of values in [0, 1].
prevalence	Proportion resistant in the target setting.

Value

A data frame with `prevalence`, `ppv` and `npv`.

Examples

```
# a test with 95% sensitivity and 90% specificity
amr_ppv_npv(0.95, 0.90, prevalence = c(0.01, 0.1, 0.5))
```

`amr_prop_ci` *Confidence intervals for a proportion*

Description

Wilson score and Clopper-Pearson (exact) intervals. Both behave well when the denominator is small or the proportion is near 0 or 1, as is common for very major error rates where few isolates are resistant.

Usage

```
amr_prop_ci(x, n, method = c("wilson", "clopper-pearson"), conf = 0.95)
```

Arguments

`x`	Number of events (numerator).
`n`	Number of trials (denominator).
`method`	`"wilson"` (default) or `"clopper-pearson"`.
`conf`	Confidence level.

Value

A data frame with `estimate`, `lower` and `upper`. Rows with `n == 0` give `NA`.

References

Wilson EB (1927). Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association* 22:209-212.

Clopper CJ, Pearson ES (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika* 26:404-413.

Examples

```
# 1 very major error among 12 resistant isolates
amr_prop_ci(1, 12)
amr_prop_ci(1, 12, method = "clopper-pearson")
```

amr_provenance *Record the provenance of an evaluation*

Description

Collects what was evaluated and how, and labels every item by where it came from: **checked** (derived by AMRvalid from the data or results), **supplied** (declared by you in `model_info` or `dataset_info`) or **not assessable** (needed for reporting but neither available nor declared). Each item is mapped to the TRIPOD+AI checklist item it informs.

Usage

```
amr_provenance(x, ..., model_info = list(), dataset_info = list())
```

Arguments

`x`	An `amr_data` object.
`...`	Result objects to include: any of `amr_validation`, `amr_split_check`, `amr_eval`, `amr_bootstrap`, `amr_paired_compare`, `amr_calibration` and `amr_distance_eval`.
`model_info`	Named list describing the model(s), e.g. `list(description = "XGBoost on AMRFinderPlus gene presence", training_data = "BV-BRC isolates collected 2015-2020", threshold_selection = "Youden index on the validation split", prediction_method = "predict() on a saved model object", class_imbalance = "none", fairness = "...")`.
`dataset_info`	Named list describing the evaluation data, e.g. `list(source = "NCBI Pathogen Detection AST browser", setting = "...", eligibility = "...", data_availability = "...", code_availability = "...")`.

Value

An object of class `amr_provenance`: a data frame with columns `section`, `item`, `value`, `status` and `tripod_item`.

References

Collins GS, Moons KGM, Dhiman P, et al. (2024). TRIPOD+AI statement: updated guidance for reporting clinical prediction models that use regression or machine learning methods. *BMJ* 385:e078378.

See Also

`amr_report()`

Examples

```
sim <- amr_simulate(n_lineages = 20, isolates_per_lineage = 10,
                    external_lineages = 0, return_distances = FALSE, seed = 1)
x <- amr_import_predictions(sim$predictions)
e <- amr_evaluate(x, threshold = 0.5)
amr_provenance(x, e, model_info = list(threshold_selection = "fixed at 0.5"))
```

---

amr_report *Write an evaluation report*

Description

Writes a self-contained Markdown (or HTML) report: data summary, validation checks, split integrity, performance per stratum with intervals, calibration, distance-to-training results, uncertainty, automatically detected limitations, and a TRIPOD+AI checklist table showing which items the analysis covers.

Usage

```
amr_report(
  x,
  ...,
  model_info = list(),
  dataset_info = list(),
  file = NULL,
  format = c("markdown", "html"),
  title = "AMRvalid evaluation report"
)
```

Arguments

x	An amr_data object.
...	Result objects to include: any of amr_validation, amr_split_check, amr_eval, amr_bootstrap, amr_paired_compare, amr_calibration and amr_distance_eval.
model_info	Named list describing the model(s), e.g. list(description = "XGBoost on AMRFinderPlus gene presence", training_data = "BV-BRC isolates collected 2015-2020", threshold_selection = "Youden index on the validation split", prediction_method = "predict() on a saved model object", class_imbalance = "none", fairness = "...").
dataset_info	Named list describing the evaluation data, e.g. list(source = "NCBI Pathogen Detection AST browser", setting = "...", eligibility = "...", data_availability = "...", code_availability = "...").
file	Output path. If NULL, the Markdown text is returned instead of written.
format	"markdown" (default) or "html". HTML needs the rmarkdown package and pandoc.
title	Report title.

Value

The path to the report (invisibly), or the Markdown text when `file` is `NULL`.

Examples

```
sim <- amr_simulate(n_lineages = 20, isolates_per_lineage = 10,
                    external_lineages = 0, return_distances = FALSE, seed = 1)
x <- amr_import_predictions(sim$predictions)
e <- amr_evaluate(x, threshold = 0.5)
report <- amr_report(x, e, file = file.path(tempdir(), "amrvalid-report.md"))
cat(readLines(report, n = 20), sep = "\n")
```

---

amr_simulate *Simulate AMR predictions with known population structure*

Description

Generates isolates in lineages, a resistance phenotype driven by a causal marker plus lineage-inherited background, genomic distances, and predictions from two fixed predictors evaluated under a random and a lineage-held-out split. It exists to teach and test evaluation methods: because the generating process is known, it shows how much each validation design overstates performance on new lineages.

Usage

```
amr_simulate(
  n_lineages = 40,
  isolates_per_lineage = 25,
  external_lineages = 40,
  marker_effect = 3,
  lineage_sd = 1.5,
  intercept = -1.5,
  within_sd = 5,
  test_fraction = 0.2,
  k = 5,
  return_distances = TRUE,
  seed = NULL
)
```

Arguments

`n_lineages`	Number of lineages in the development dataset.
`isolates_per_lineage`	Isolates per lineage.
`external_lineages`	Number of fresh lineages in an external dataset (`dataset_id = "external"`), never seen in training. Large values give a close approximation of true performance on new lineages.

marker_effect	Log-odds effect of the causal marker.
lineage_sd	Standard deviation of the lineage effect on the log-odds of resistance (resistance inherited clonally but not explained by the marker).
intercept	Baseline log-odds of resistance.
within_sd	Spread of isolates around their lineage centre, in distance units (lineage centres are spread over 0 to 1000).
test_fraction	Fraction of isolates (random split) or lineages (lineage-held-out split) placed in the test set.
k	Number of neighbours for the nearest-neighbour predictor.
return_distances	If TRUE, include the full pairwise distance matrix (memory grows with the square of the number of isolates).
seed	Optional random seed; the global random state is restored.

Details

Each lineage has a centre in a two-dimensional space; isolate distances are Euclidean distances between isolates, standing in for genomic distances. Marker frequency varies by lineage (Beta(0.5, 0.5)), so the marker is partly confounded with lineage, as resistance genes often are.

The two predictors are fixed rules, not trained machine-learning models:

- knn: the proportion of resistant isolates among the k nearest training isolates. It memorises lineages.
- marker: the proportion resistant among training isolates with the same marker status (with a +1/+2 smoothing). It learns the mechanism.

Each predictor is applied under two designs, giving model identifiers knn/random, knn/lineage, marker/random and marker/lineage.

Value

A list with predictions (a data frame ready for amr_import_predictions(); test isolates only), train_manifest (training isolates per model), isolates (all isolates with lineage, marker, phenotype and coordinates), distances (a matrix or NULL) and parameters.

Examples

```
sim <- amr_simulate(n_lineages = 20, isolates_per_lineage = 10,
                    external_lineages = 10, seed = 1)
x <- amr_import_predictions(sim$predictions)
e <- amr_evaluate(x, threshold = 0.5, validate = FALSE)
e[e$metric == "accuracy", c("model_id", "dataset_id", "estimate")]
```

amr_validate *Validate an AMRvalid prediction table*

Description

Runs structural and scientific checks before evaluation. Blocking problems are reported as errors; problems that need attention but do not prevent analysis are warnings; notes that affect interpretation are info.

Usage

```
amr_validate(x, min_n = 10, min_resistant = 5)
```

Arguments

x	An amr_data object from amr_import_predictions().
min_n	Strata with fewer evaluable isolates than this trigger a warning.
min_resistant	Strata with fewer reference-resistant isolates than this trigger a warning, because very major error rates are then unstable.

Value

An object of class amr_validation: a list with ok (TRUE if there are no errors) and issues (a data frame with columns level, check, message and n_rows).

Examples

```
preds <- data.frame(
  isolate_id = c("a", "a", "b"),
  species = "E. coli",
  drug = "ciprofloxacin",
  truth = c("R", "R", "S"),
  probability = c(0.9, 0.8, 1.4)
)
amr_validate(amr_import_predictions(preds))
```

amr_validate_splits *Check the integrity of a train/test split*

Description

Compares the isolates a model was trained on with the isolates it is evaluated on, and reports which validation designs the split is consistent with. AMRvalid cannot see upstream modelling steps, so a clean result shows only that no visible overlap exists in the supplied metadata.

Usage

```
amr_validate_splits(
  x,
  train_manifest = NULL,
  group = NULL,
  claimed_design = NULL
)
```

Arguments

x	An amr_data object. If it has a split column, rows with split == "test" are the evaluation set and rows with split == "train" are training isolates (unless train_manifest is given). Without a split column every row is treated as test.
train_manifest	Optional data frame listing training isolates, with an isolate_id column and optionally model_id (if absent, the manifest applies to every model) and the grouping, date and country columns to compare.
group	Grouping columns to check for overlap. Defaults to whichever of lineage, genomic_cluster and sequence_type are available.
claimed_design	Optional design the evaluation claims to follow: "isolate", "group" (held-out groups of the first group column), "temporal" or "geographic". A split that contradicts the claim makes ok FALSE.

Value

An object of class amr_split_check: a list with ok, summary (one row per model and check) and consistent_designs (one row per model).

Examples

```
preds <- data.frame(
  isolate_id = c("a", "b", "c", "d", "e", "f"),
  species = "K. pneumoniae",
  drug = "meropenem",
  truth = c("R", "S", "R", "S", "R", "S"),
  probability = c(0.9, 0.1, 0.8, 0.3, 0.7, 0.2),
  split = c("train", "train", "train", "test", "test", "test"),
  lineage = c("ST258", "ST11", "ST258", "ST15", "ST258", "ST307")
)
amr_validate_splits(amr_import_predictions(preds), claimed_design = "group")
```

---

plot.amr_bootstrap *Plot bootstrap intervals*

Description

Dot-and-interval plots of bootstrap estimates, one panel per metric.

Usage

```
## S3 method for class 'amr_bootstrap'
plot(x, metrics = NULL, ...)
```

Arguments

x	An amr_bootstrap object from amr_bootstrap().
metrics	Metrics to draw; by default sensitivity and specificity when available, otherwise ROC-AUC.
...	Unused.

Value

x, invisibly.

Examples

```
x <- amr_import_predictions(amr_example_data())
b <- amr_bootstrap(x, resample_by = "genomic_cluster", B = 100, seed = 1)
plot(b)
```

---

plot.amr_calibration *Plot a reliability diagram*

Description

Draws one panel per stratum (model, dataset, species and antimicrobial, plus the by column if used) with the binned observed proportions, their Wilson intervals and the loess-smoothed calibration curve.

Usage

```
## S3 method for class 'amr_calibration'
plot(x, max_panels = 12, ...)
```

Arguments

x	An amr_calibration object.
max_panels	Maximum number of panels to draw.
...	Unused.

Value

x, invisibly.

Examples

```
sim <- amr_simulate(n_lineages = 20, isolates_per_lineage = 10,
                    external_lineages = 20, return_distances = FALSE, seed = 1)
x <- amr_import_predictions(sim$predictions)
plot(amr_calibration(x[x$dataset_id == "external" & x$model_id == "marker/lineage", ]))
```

`plot.amr_distance_eval`

Plot performance by distance to the training set

Description

Draws one metric across the distance bands, with intervals: one line per stratum, coloured by model, with line type distinguishing datasets.

Usage

```
## S3 method for class 'amr_distance_eval'
plot(x, metric = NULL, ...)
```

Arguments

`x`	An `amr_distance_eval` object from `amr_distance_eval()`.
`metric`	Metric to draw. Defaults to ROC-AUC for scores without a threshold and accuracy otherwise.
`...`	Unused.

Value

`x`, invisibly.

Examples

```
sim <- amr_simulate(n_lineages = 20, isolates_per_lineage = 10,
                    external_lineages = 20, seed = 1)
x <- amr_import_predictions(sim$predictions)
d <- amr_distance_eval(x[x$model_id == "knn/random", ], sim$distances,
                       train_manifest = sim$train_manifest)
plot(d)
```

plot.amr_eval *Plot evaluation results*

Description

Dot-and-interval plots of selected metrics, one panel per metric and one row per stratum. Rows are labelled with the antimicrobial (and model, dataset and by values where there are several); colour repeats the model.

Usage

```
## S3 method for class 'amr_eval'
plot(x, metrics = NULL, ...)
```

Arguments

x	An amr_eval object from amr_evaluate().
metrics	Metrics to draw. By default sensitivity and specificity when available, otherwise ROC-AUC (threshold-free scores) or essential agreement (MIC predictions).
...	Unused.

Value

x, invisibly.

Examples

```
x <- amr_import_predictions(amr_example_data())
plot(amr_evaluate(x), metrics = c("very_major_error", "major_error"))
```

plot.amr_paired_compare *Plot a paired model comparison*

Description

Differences between the two models (second minus first) with bootstrap intervals and a reference line at zero.

Usage

```
## S3 method for class 'amr_paired_compare'
plot(x, metrics = NULL, ...)
```

Arguments

x	An amr_paired_compare object from amr_paired_compare().
metrics	Metrics to draw; by default every metric in the results.
...	Unused.

Value

x, invisibly.

Examples

```
x <- amr_import_predictions(amr_example_data())
mero <- x[x$drug == "MEM", ]
p <- amr_paired_compare(mero, models = c("carbapenemase genes",
                        "carbapenemase or porin + ESBL/AmpC"), B = 100, seed = 1)
plot(p)
```

Index

amr_ast_agreement, 2
amr_ast_agreement(), *11, 12*
amr_bootstrap, 4
amr_bootstrap(), *16*, *27*
amr_calibration, 6
amr_calibration(), *10*
amr_contract, 7
amr_contract_schema, 8
amr_data_summary, 8
amr_distance_eval, 9
amr_distance_eval(), *28*
amr_evaluate, 11
amr_evaluate(), *5*, *10*, *15*, *29*
amr_example_data, 12
amr_import_predictions, 14
amr_import_predictions(), *13*, *24, 25*
amr_metrics, 15
amr_metrics(), *3*, *10–12*
amr_nearest_train_distance, 17
amr_nearest_train_distance(), *9*
amr_paired_compare, 18
amr_paired_compare(), *30*
amr_ppv_npv, 19
amr_prop_ci, 20
amr_prop_ci(), *3*, *10, 11*, *15*
amr_provenance, 21
amr_report, 22
amr_report(), *21*
amr_simulate, 23
amr_validate, 25
amr_validate(), *12*, *14*
amr_validate_splits, 25

plot(), *7*
plot.amr_bootstrap, 26
plot.amr_calibration, 27
plot.amr_distance_eval, 28
plot.amr_eval, 29
plot.amr_paired_compare, 29