

Package: anumaan (via r-universe)

October 8, 2026

Type Package

Title Antimicrobial Resistance Surveillance Preprocessing and Burden Estimation

Version 0.1.0.9036

Description anumaan provides a preprocessing pipeline for antimicrobial resistance (AMR) surveillance data, exploratory data analysis (EDA) tools for reviewing cohorts prior to modelling, and burden estimation. Supports polymicrobial weighting, MDR/XDR classification, resistance profile estimation, and burden estimation calculations.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports dplyr (>= 1.0.0), tidyr (>= 1.0.0), stringr (>= 1.4.0), stringdist (>= 0.9.0), lubridate (>= 1.7.0), magrittr (>= 2.0.0), readr (>= 2.0.0), readxl (>= 1.3.0), tibble (>= 3.0.0), rlang (>= 0.4.0), ggplot2 (>= 3.4.0), scales (>= 1.2.0), fitdistrplus (>= 1.1.0), RColorBrewer (>= 1.1.0), ggridges (>= 0.5.0), tidytext (>= 0.3.0)

SystemRequirements Python (>= 3.8) with the 'alethia' package for ICD-10 semantic embedding (optional; only needed when `prep_map_diagnosis_to_icd()` is called with `method = ``python_embedding```). CmdStan (>= 2.33) for Pathway 2 Bayesian estimation (optional; install via `cmdstanr::install_cmdstan()` after installing the 'cmdstanr' package).

Suggests bayesplot (>= 1.10.0), cmdstanr (>= 0.7.0), testthat (>= 3.0.0), knitr, rmarkdown, pkgdown, ggpubr (>= 0.4.0), reticulate (>= 1.28), Matrix, osqp, quadprog, posterior (>= 1.0.0), pbivnorm, ComplexHeatmap, circlize

Additional_repositories <https://stan-dev.r-universe.dev>

VignetteBuilder knitr

URL <https://saketlab.github.io/anumaan/>,
<https://github.com/saketlab/anumaan>

BugReports <https://github.com/saketlab/anumaan/issues>

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev python3 libx11-dev

Repository <https://saketlab.r-universe.dev>

Date/Publication 2026-10-08 05:22:27 UTC

RemoteUrl <https://github.com/saketlab/anumaan>

RemoteRef HEAD

RemoteSha f498030b1265b310ba4c862e38b988daaa87d090

Contents

aggregate_profiles_for_daly	6
amr_config	8
assess_gibbs_profile_stability	10
bootstrap_profiles_convex	11
build_constraint_matrix	13
check_profile_constraints	14
class_display_label	16
class_pair_label	17
class_short_label	17
compare_distribution_aic	18
compute_event_profile_probabilities	18
compute_hospital_daly	20
compute_marginal_resistance	21
compute_marginals_from_data	23
compute_pairwise_coresistance	24
compute_pairwise_from_data	26
compute_posterior_predictive_status	27
compute_prior_predictive_fingerprint	28
compute_prior_predictive_status	29
compute_probit_ppc_statistics	30
compute_processing_statistics	32
compute_profile_validation_status	32
compute_resistance_profiles	34
daly_add_rr_mappings	37
daly_assign_rr_to_profiles	37
daly_calc_case_fatality	39
daly_calc_cr_l	40
daly_calc_deaths_by_cause	42
daly_calc_deaths_by_syndrome	43
daly_calc_incidence_from_cfr	46

daly_calc_infection_fraction	47
daly_calc_paf_los	48
daly_calc_paf_mortality	49
daly_calc_pathogen_fraction_fatal	51
daly_calc_resistance_prevalence_fatal	52
daly_calc_syndrome_fraction	53
daly_calc_yld_associated	54
daly_calc_yld_attributable	56
daly_calc_yll_associated	58
daly_calc_yll_attributable	61
daly_compute_patient_los	63
daly_count_incident_cases	65
daly_derive_hai_cai_for_mortality	66
daly_filter_profiles_to_rr_classes	67
daly_fit_los_rr	68
daly_fit_los_rr_distribution	70
daly_fit_mortality_rr	72
daly_get_top_pathogens	75
daly_load_rr_reference	76
default_column_mappings	76
detect_preprocessing_capabilities	77
eda_theme	77
enumerate_binary_profiles	78
estimate_profiles_convex	78
estimate_resistance_profiles	80
export_report	82
fit_bayesian_multivariate_probit	83
fit_distributions	86
generate_preprocessing_report	87
get_age_bins	88
get_los_by_resistance	88
hospital_display_label	89
infer_patient_syndrome_wide	90
mask_and_validate_ast	91
plot_abx_complex_heatmap	93
plot_abx_heatmap	95
plot_abx_susceptibility	97
plot_age_ridge	98
plot_burden_by_hospital	100
plot_burden_by_organism	101
plot_death_discharged	103
plot_hai_cai_by_facility	105
plot_infection_type_by_location	106
plot_location_by_facility	108
plot_los_by_agebin	109
plot_los_distributions	111
plot_los_ridge	112
plot_mono_poly_by_facility	113

plot_omega_convergence_heatmap	115
plot_omega_correlation_heatmap	115
plot_omega_degeneracy_diagnostic	116
plot_omega_summary_table	117
plot_outcome_by_agebin	118
plot_outcome_by_organism	119
plot_outcome_by_year	121
plot_outcome_distribution	123
plot_patients_by_hospital	124
plot_probit_beta_family_diagnostics	125
plot_probit_diagnostics	126
plot_probit_fixed_effect_diagnostics	126
plot_probit_posterior_predictive_checks	127
plot_probit_worst_offender_diagnostics	128
plot_probit_worst_parameters	129
plot_resistance_by_agebin	129
plot_resistance_by_organism	131
plot_resistance_by_sample	132
plot_syndrome_distribution	134
plot_top_organisms	135
plot_yld_heatmap	136
plot_yll_heatmap	137
prep_apply_column_map	139
prep_assert_standard_names	139
prep_assign_age_bins	140
prep_assign_facility_type	141
prep_assign_organism_group	142
prep_assign_patient_syndrome	143
prep_attrition_flow	144
prep_build_column_map	145
prep_build_fatal_cohort	146
prep_build_nonfatal_cohort	146
prep_check_columns	147
prep_check_keys	148
prep_check_organism_ast_consistency	148
prep_classify_antibiotic_class	149
prep_classify_aware	150
prep_classify_mdr_xdr	150
prep_classify_mortality	151
prep_classify_readmission	152
prep_clean_ast_values	153
prep_clean_optional_columns	153
prep_coerce_dates	154
prep_collapse_class_level	154
prep_compute_poly_weights	155
prep_create_event_ids	156
prep_create_resistance_profile	158
prep_create_wide_ast_matrix	159

prep_decode_antibiotic_code	160
prep_deduplicate_ast	161
prep_deduplicate_events	162
prep_derive_dob_from_components	163
prep_derive_hai_cai	164
prep_derive_icu_flag	165
prep_derive_los_from_dates	165
prep_detect_schema_drift	166
prep_diagnosis_text	167
prep_extract_genus	168
prep_extract_species	168
prep_fill_age	169
prep_filter_analysis_ready	169
prep_filter_fungal	170
prep_filter_minimally_usable	171
prep_flag_contaminants	172
prep_flag_hai_inferred	173
prep_flag_invalid_ast	173
prep_flag_organism_unmatched	174
prep_flag_polymicrobial	174
prep_flag_readmission	175
prep_get_contaminant_list	176
prep_harmonize_ast	177
prep_infer_department	178
prep_inventory_columns	178
prep_is_contaminant	179
prep_log_source	179
prep_map_diagnosis_to_icd	180
prep_map_icd_to_syndrome	181
prep_missingness_report	182
prep_parse_date_column	183
prep_pivot_ast_wide_to_long	183
prep_recode_intermediate_ast	185
prep_reconcile_hai_observed_inferred	185
prep_report_capabilities	186
prep_split_poly_episode	187
prep_standardize_antibiotics	187
prep_standardize_column_names	188
prep_standardize_final_outcome	189
prep_standardize_infection_type	190
prep_standardize_organisms	190
prep_standardize_outcome	191
prep_standardize_sex	191
prep_standardize_specimens	192
prep_validate_analysis_ready	192
prep_validate_date_logic	193
prep_validate_table	194
prepare_los_data	195

prepare_random_effects	196
preprocess_for_profiles	197
print.amr_config	199
print.amr_preprocessing_report	199
print.amr_result	200
re_contribution	200
round_to_sum	201
run_preprocess	201
safe_fit	203
select_resistance_class	203
shorten_drug_class	204
simulate_probit_mixed_predictive	205
simulate_probit_posterior_predictive	206
simulate_probit_prior_predictive	208
summarise_distribution	210
summarize_column_mapping	210
summarize_data_quality	211
summarize_fit_correlation_matrix	211
summarize_raw_data	212
summarize_transformations	213
summary.amr_result	213
validate_aggregate_inputs	214
validate_complete_profile_calibration	215
validate_compute_backend	216
validate_config	217
validate_data_quality	217
validate_marginal_calibration	218
validate_pairwise_calibration	219
validate_profile_inputs	220
validate_required_fields	222
Index	223

aggregate_profiles_for_daly
<i>Aggregate Posterior Profile Draws into R_ALL / R_KNOWN_OUTCOME / R_NF Summaries</i>

Description

Summarises draw-level R_ALL, R_KNOWN_OUTCOME, and R_NF values from compute_event_profile_probabilities into posterior mean and credible interval per hospital x pathogen x profile combination, and derives explicit eligible_for_YLL/eligible_for_YLD flags (with reasons) rather than asserting universal usability.

Usage

```

aggregate_profiles_for_daly(
  profile_output,
  hospital_col = "hospital",
  pathogen_col = "pathogen",
  estimand = "observed_stewardship_event_mix",
  ci_level = 0.95,
  min_n_events = 10L,
  min_n_draws = 100L,
  sampler_acceptable = TRUE
)

```

Arguments

profile_output	List returned by <code>compute_event_profile_probabilities()</code> .
hospital_col	Character. Hospital column in <code>aggregate_draws</code> . Must match the upper RE column used during fitting. Default "hospital".
pathogen_col	Character. Pathogen column. Default "pathogen".
estimand	Character. Estimand label to attach to output. Default "observed_stewardship_event_mix".
ci_level	Numeric. Credible interval coverage. Default 0.95.
min_n_events	Integer. Minimum events in the relevant cohort (known-outcome for YLL, non-fatal for YLD) for eligibility. Default 10L.
min_n_draws	Integer. Minimum valid posterior draws in the relevant summary for eligibility. Default 100L.
sampler_acceptable	Logical, scalar or one value per row of <code>profile_output\$aggregate_draws</code> groups. Whether the fit's sampler diagnostics are acceptable (e.g. <code>fitted_model\$diagnostics\$convergence</code> from <code>fit_bayesian_multivariate_probit()</code>); FALSE makes every row ineligible for YLL/YLD regardless of panel/count support. Default TRUE (i.e. does not gate on sampler status unless the caller supplies it – <code>estimate_resistance_profiles()</code> supplies it automatically).

Details

Cohort definitions: `R_ALL` is the mean over truly all events in the hospital-pathogen-panel cohort (no outcome filter). `R_KNOWN_OUTCOME` restricts to events with a known patient outcome (this is what earlier versions of this function called `R_ALL` – renamed because it is not actually all events). `R_NF` restricts to the non-fatal subset of the known-outcome cohort. All three are computed per posterior draw and then summarised here; `n_draws_all/n_draws_known_outcome/n_draws_nf` count only the non-NA draws feeding each summary (a cohort that is empty for a given hospital-pathogen pair yields NA draws, which must not be silently counted as valid support).

Eligibility: `eligible_for_profile_inference` is a hard boolean, TRUE only for rows generated by a conditional (observed-plus-imputed) method – "conditional_analytic_identity" or "conditional_gibbs_correlated"; see `compute_event_profile_probabilities()`. FALSE for any other `profile_generation_method` value, including the legacy "unconditional_simulation_correlated_not_d" tag from an older `profile_output` – callers must not rely on the descriptive `profile_generation_method`

string alone. `eligible_for_YLL/eligible_for_YLD` additionally require: `sampler_acceptable` (passed in by the caller, e.g. from `fitted_model$diagnostics$converged_structural`); the relevant event cohort has at least `min_n_events` events; and at least `min_n_draws` valid posterior draws contributed to the relevant summary. `exclusion_reason_YLL/exclusion_reason_YLD` record why, distinguishing a cohort that is completely empty ("`no_known_outcome_events`" / "`no_nonfatal_events`") from one that is merely small ("`too_few_known_outcome_events`" / "`too_few_nonfatal_events`").

Value

Tibble with one row per hospital x pathogen x profile: `R_ALL`, `R_KNOWN_OUTCOME`, and `R_NF` posterior mean and credible interval, panel composition (`n_profile_classes`, `panel_eligibility_method`, `classes_excluded`, `classes_excluded_reason`), event and draw counts, and profile/YLL/YLD eligibility flags with reasons.

amr_config

Create AMR Preprocessing Configuration

Description

Creates a configuration object that controls the behavior of the preprocessing pipeline. Users can override default settings for their specific use case.

Usage

```
amr_config(
  column_mappings = NULL,
  fuzzy_match = TRUE,
  strict_validation = FALSE,
  date_columns = NULL,
  hai_cutoff = 2,
  infer_department = TRUE,
  event_gap_days = 14,
  mortality_window = 14,
  age_bins = "GBD_standard",
  contaminant_method = "auto",
  mdr_definition = "CDC",
  intermediate_as_resistant = TRUE,
  verbose = TRUE
)
```

Arguments

`column_mappings` Named list of column name mappings. Default uses `default_column_mappings`.

`fuzzy_match` Logical. Enable fuzzy matching for column names. Default TRUE.

strict_validation	Logical. If TRUE, stops execution if required fields are missing. If FALSE, issues warnings. Default FALSE.
date_columns	Character vector of date column names to parse. Default NULL auto-detects date-like columns by name pattern.
hai_cutoff	Numeric. Number of days after admission to classify as Hospital-Acquired Infection (HAI). Default 2.
infer_department	Logical. Attempt to infer hospital department from other variables if missing. Default TRUE.
event_gap_days	Numeric. Minimum days between events to create new event_id. Default 14.
mortality_window	Numeric. Days after culture to classify death as infection-related. Default 14.
age_bins	Character. Age binning strategy: "GBD_standard", "pediatric", or "geriatric". Can also be a custom vector. Default "GBD_standard".
contaminant_method	Character. Method for contaminant classification: "auto" (cascade through available methods), "device_based", "heuristic", "provided". Default "auto".
mdr_definition	Character or numeric. MDR/XDR definition (used for both): "CDC", "WHO", or numeric threshold for number of resistant classes. Default "CDC".
intermediate_as_resistant	Logical. Treat Intermediate (I) as Resistant (R) except for special cases like Colistin. Default TRUE.
verbose	Logical. Print progress messages during processing. Default TRUE.

Value

An amr_config object (list with class)

Examples

```
## Not run:
# Use defaults
config <- amr_config()

# Customize for specific hospital
config <- amr_config(
  hai_cutoff = 3,
  mdr_definition = 5,
  strict_validation = FALSE
)

## End(Not run)
```

assess_gibbs_profile_stability

Assess Numerical Stability of Correlated Profile Completion

Description

Repeats the existing conditional truncated-MVN Gibbs profile completion for fixed posterior draws and a fixed, reproducibly selected subset of incomplete events. It does not refit Stan or alter the fitted model. The comparison therefore isolates finite Gibbs-chain length. The conditional completion algorithm is the correct profile-completion algorithm; this diagnostic checks whether its finite Monte Carlo run length is adequate for a particular fit, especially when posterior residual correlations are high.

Usage

```
assess_gibbs_profile_stability(
  fitted_model,
  schedules = list(baseline = list(burnin = 30L, kept = 50L), medium = list(burnin = 60L,
    kept = 100L), long = list(burnin = 100L, kept = 250L)),
  n_posterior_draws = 200L,
  max_events = 250L,
  seed = 123L,
  tolerance = 0.01,
  event_selection = c("stratified", "all")
)
```

Arguments

<code>fitted_model</code>	Object returned by <code>[fit_bayesian_multivariate_probit()]</code> .
<code>schedules</code>	Named list of 'list(burnin=, kept=)' schedules. It must contain 'baseline', 'medium', and 'long'.
<code>n_posterior_draws</code>	Number of fixed posterior draws to use.
<code>max_events</code>	Maximum number of incomplete events to assess.
<code>seed</code>	Seed used once to select draws/events and then reset before each schedule. Thus schedules share posterior draws, events, and RNG streams.
<code>tolerance</code>	Maximum aggregate difference for status "pass"; twice this value is the "warning" upper bound.
<code>event_selection</code>	Currently "stratified" or "all".

Details

The default 0.01/0.02 thresholds are numerical-stability conventions for this diagnostic, not universal statistical thresholds.

Value

A structured list with summary, event/aggregate comparisons, selected events, schedules, posterior draw indices, and a separate stability status.

bootstrap_profiles_convex

Bootstrap Uncertainty Intervals for Resistance Profile Probabilities

Description

Quantifies uncertainty in resistance-profile probability estimates by repeatedly resampling isolate counts from a binomial distribution implied by the observed `n_tested` and `n_resistant` values, refitting the convex optimisation QP for each replicate, and returning percentile confidence intervals across replicates.

Usage

```
bootstrap_profiles_convex(
  marginals,
  coresistance_output = NULL,
  B = 500L,
  seed = 123L,
  alpha = 0.05,
  n_cores = 1L,
  exclude_near_zero = TRUE,
  top_n_classes = NULL,
  sigma_sq = 1,
  ridge = 1e-08,
  pathogen_col = "organism_name",
  class_col = "antibiotic_class",
  n_tested_col = "n_tested",
  n_resistant_col = "n_resistant",
  org_group_col = "org_group"
)
```

Arguments

<code>marginals</code>	Data frame. Marginal resistance rates with at minimum columns <code>pathogen_col</code> , <code>class_col</code> , <code>n_tested_col</code> , <code>n_resistant_col</code> . This is the <code>\$marginal</code> element from <code>compute_marginal_resistance()</code> , or a flat tibble with the same structure from <code>compute_marginals_from_data()</code> .
<code>coresistance_output</code>	Named list or NULL. Output of <code>compute_pairwise_coresistance()</code> (one entry per pathogen with <code>\$T_matrix</code> , <code>\$R_matrix</code> , <code>\$prevalence</code>). When supplied, pairwise co-resistance is resampled per replicate. When NULL, independence fallback $P(A \cap B) = P(A)P(B)$ is used in every replicate. Default NULL.

B	Integer. Number of bootstrap replicates. Default 500.
seed	Integer. Random seed for reproducibility. Default 123.
alpha	Numeric. Two-sided interval width. Default 0.05 (i.e., 95% intervals).
n_cores	Integer. Parallel cores via <code>parallel::mclapply</code> . Default 1L.
exclude_near_zero	Logical. Passed to <code>compute_resistance_profiles()</code> . Default TRUE.
top_n_classes	Integer or NULL. Cap on number of classes per pathogen. Default NULL.
sigma_sq	Numeric. QP constraint variance. Default 1.
ridge	Numeric. QP Hessian ridge term. Default 1e-8.
pathogen_col	Character. Pathogen column in marginals. Default "organism_name".
class_col	Character. Class column in marginals. Default "antibiotic_class".
n_tested_col	Character. Tested-count column. Default "n_tested".
n_resistant_col	Character. Resistant-count column. Default "n_resistant".
org_group_col	Character. Organism-group column (passed through to QP engine). Default "org_group".

Details

Resampling mechanism: For each bootstrap replicate b and each (pathogen, class) cell:

$$n_{\text{resistant}}^{(b)} \sim \text{Binomial}(n_{\text{tested}}, \hat{r}_{kd})$$

The bootstrap marginal is then $\hat{r}_{kd}^{(b)} = n_{\text{resistant}}^{(b)} / n_{\text{tested}}$. Pairwise co-resistance rates (if supplied) are resampled analogously. The QP is re-solved on each bootstrap marginal using the existing `compute_resistance_profiles()` engine.

Performance: For $n \leq 12$ classes the QP solves in milliseconds; $B = 500$ replicates will complete in seconds. For $n \geq 14$ classes use `n_cores > 1` or reduce B .

Value

A named list, one entry per pathogen. Each entry is a tibble with columns:

`profile` Profile label (e.g. "RSS").
`probability_mean` Mean profile probability across B replicates.
`probability_median` Median across replicates.
`lower` Lower percentile ($\alpha / 2$).
`upper` Upper percentile ($1 - \alpha / 2$).
`n_replicates_converged` Number of replicates for which the QP converged (non-uniform solution).
`convergence_rate` Proportion of replicates that converged.

The point-estimate profiles tibble (from a single run on the original marginals) is stored as the attribute "point_estimate".

Examples

```
## Not run:
marg <- compute_marginal_resistance(amr_clean)
co_res <- compute_pairwise_coresistance(marg)

boot <- bootstrap_profiles_convex(
  marginals = marg$marginal,
  coresistance_output = co_res,
  B = 500,
  seed = 42
)

# 95% intervals for K. pneumoniae
boot[["Klebsiella pneumoniae"]]

# Point estimate
attr(boot[["Klebsiella pneumoniae"]], "point_estimate")

## End(Not run)
```

build_constraint_matrix

Build QP Constraint Matrix and Target Vector

Description

Constructs the constraint matrix $\mathbf{M}$ and target vector $\mathbf{v}$ used in the simplex-constrained weighted least-squares QP:

Usage

```
build_constraint_matrix(profiles_enum, r_marg, co_mat = NULL)
```

Arguments

profiles_enum	Tibble from <code>enumerate_binary_profiles()</code> . The <code>profile_delta</code> column is ignored; only the binary class columns are used.
r_marg	Named numeric vector. Marginal resistance rates, one per class. Names must match column names in <code>profiles_enum</code> .
co_mat	Numeric matrix or NULL. Square symmetric pairwise co-resistance prevalence matrix with row/column names matching <code>names(r_marg)</code> . NA cells trigger independence fallback. Default NULL.

Details

$$\min_p \|Mp - v\|_W^2 \quad \text{s.t.} \quad p \geq 0, \sum p = 1$$

Marginal rows: $M_{d,\delta} = 1$ iff class d is resistant in profile δ . Pairwise rows: $M_{d_1 d_2, \delta} = 1$ iff both classes are resistant. When a pairwise value is unavailable (too few co-tested isolates), the product of marginals (independence assumption) is substituted. Pairwise values that exceed $\min(P(A), P(B))$ are capped.

Value

Named list:

M Numeric matrix $(n + n(n - 1)/2) \times 2^n$.

v Numeric vector of constraint targets.

constraint_names Character vector labelling each row of **M**.

fallback_pairs Character vector of class pairs that used the independence fallback.

capped_pairs Named numeric vector of pairs whose pairwise value was capped, showing original and capped values.

check_profile_constraints

Formally Check Resistance Profile Probability Constraints

Description

Validates that the profile probability distributions returned by `compute_resistance_profiles()` satisfy all mathematical requirements: non-negativity, sum-to-one, and reconstruction of the input marginal and pairwise resistance rates within a specified tolerance.

Usage

```
check_profile_constraints(
  profiles_output,
  marginals = NULL,
  pairwise = NULL,
  tolerance = 1e-06,
  pathogen_col = "pathogen",
  class_col = "antibiotic_class",
  rate_col = "marginal_resistance",
  class1_col = "antibiotic_class_1",
  class2_col = "antibiotic_class_2",
  pairwise_rate_col = "pairwise_resistance_prevalence"
)
```

Arguments

profiles_output	Named list returned by compute_resistance_profiles(). Each element must contain \$profiles (data frame with probability and binary class columns) and optionally \$constraint_residuals.
marginals	Data frame or NULL. Original marginal resistance rates used as QP constraints. When provided, reconstructed marginals are compared against these values. Must have columns pathogen_col, class_col, and rate_col. Default NULL.
pairwise	Data frame or NULL. Original pairwise co-resistance rates. When provided, reconstructed pairwise values are also checked. Must have columns pathogen_col, class1_col, class2_col, and pairwise_rate_col. Default NULL.
tolerance	Numeric. Maximum absolute residual considered a pass. Default 1e-6.
pathogen_col	Character. Pathogen column in marginals / pairwise. Default "pathogen".
class_col	Character. Class column in marginals. Default "antibiotic_class".
rate_col	Character. Marginal rate column in marginals. Default "marginal_resistance".
class1_col	Character. First class column in pairwise. Default "antibiotic_class_1".
class2_col	Character. Second class column in pairwise. Default "antibiotic_class_2".
pairwise_rate_col	Character. Pairwise rate column. Default "pairwise_resistance_prevalence".

Details

The function uses two complementary sources of evidence:

1. **Constraint residuals** already stored in profiles_output[[pathogen]]\$constraint_residuals – these are $M\hat{p} - v$ computed at solve time and are available without re-supplying the original rates.
2. **Direct reconstruction** from the binary class-indicator columns in the profiles data frame when marginals and/or pairwise are supplied – independently verifies the residuals and catches any post-solve modifications to the probability vector.

Value

A tibble with one row per constraint per pathogen:

pathogen	Pathogen name.
constraint_type	One of "nonneg", "sum_to_one", "marginal", "pairwise".
constraint_name	Specific label (e.g. "marg_3GC", "pair_3GC_Carbapenems").
target	The constraint target value (NA for nonneg).
reconstructed	The value implied by $\hat{p}$.
abs_residual	Absolute difference between target and reconstructed.
pass	Logical. TRUE if abs_residual < tolerance or the structural check passed.
source	Either "stored_residuals" (from solve-time values) or "recomputed" (from direct reconstruction).

Also prints a per-pathogen summary of passed and failed checks.

Examples

```
## Not run:
marg <- compute_marginal_resistance(amr_clean)
co_res <- compute_pairwise_coresistance(marg)
rp <- compute_resistance_profiles(marg, co_res)

# Using stored constraint residuals only
checks <- check_profile_constraints(rp)

# Full re-verification with original rates
checks <- check_profile_constraints(
  rp,
  marginals = marg$marginal,
  pairwise = NULL,
  tolerance = 1e-5
)

# Inspect failures
checks[!checks$pass, ]

## End(Not run)
```

class_display_label	<i>Human-readable antimicrobial class label</i>
---------------------	---

Description

Looks up inst/extdata/class_display_labels.csv; falls back to replacing underscores with spaces for any class not in the table (e.g. a pathogen-specific class not yet added there), so this never errors on an unmapped class – it just degrades to a readable-if-generic label.

Usage

```
class_display_label(x)
```

Arguments

x Character vector of raw class names (e.g. "Third_generation_cephalosporins").

Value

Character vector of display labels, same length as x.

class_pair_label	<i>Human-readable class-pair label</i>
------------------	--

Description

Human-readable class-pair label

Usage

```
class_pair_label(class_1, class_2, short = TRUE)
```

Arguments

class_1, class_2	Character vectors of raw class names.
short	Logical; use class_short_label() instead of class_display_label(). Default TRUE (pairwise facet strips get crowded fast with full names).

Value

Character vector, e.g. "AMG x CARB".

class_short_label	<i>Short antimicrobial class label (for dense facet strips)</i>
-------------------	---

Description

Same lookup/fallback behaviour as class_display_label(), but returns the abbreviated form (e.g. "3GC"). Falls back to the first 4 characters of the underscore-cleaned name, upper-cased, for any class not in the reference table.

Usage

```
class_short_label(x)
```

Arguments

x	Character vector of raw class names.
---	--------------------------------------

Value

Character vector of short labels, same length as x.

```
compare_distribution_aic
```

Compare Distribution Fits by AIC

Description

Fits Weibull, Lognormal, and Gamma distributions to a numeric vector and returns their AIC values for comparison.

Usage

```
compare_distribution_aic(x, fits = NULL)
```

Arguments

x Numeric vector of positive values (e.g., LOS in days).
fits Optional. Pre-computed fits from `fit_distributions()`.

Value

Data frame with columns `Weibull_AIC`, `Lognormal_AIC`, `Gamma_AIC`.

Examples

```
compare_distribution_aic(rlnorm(200, meanlog = 2, sdlog = 0.5))
```

```
compute_event_profile_probabilities
```

Compute Observed-Plus-Imputed Resistance Profile Probabilities

Description

For each event, retains every observed (tested) AST cell exactly and computes a probability distribution only over the genuinely untested (NA) antibiotic-class cells in that event's profile panel. Any enumerated profile inconsistent with the event's observed cells receives probability 0 – this function never overwrites an observed R/S result.

Usage

```
compute_event_profile_probabilities(  
  fitted_model,  
  n_posterior_draws_for_profiles = 2000L,  
  outcome_col = NULL,  
  nonfatal_values = c("Discharged", "Survived", "Alive", "discharged", "survived",  
    "alive"),
```

```

    seed = 123L,
    n_gibbs_burnin = 10L,
    n_gibbs_kept = 20L,
    posterior_draw_indices = NULL,
    event_indices = NULL
)

```

Arguments

fitted_model List returned by `fit_bayesian_multivariate_probit()`.

n_posterior_draws_for_profiles Integer. Number of posterior draws averaged over for both the event-level profile probabilities and the draw-level aggregates. Subsampled without replacement when total draws exceed this value. Default 2000L.

outcome_col Character or NULL. Patient outcome column in `fitted_model$event_metadata`. When NULL, all events are treated as having a known outcome and `R_NF` is not computed separately.

nonfatal_values Character vector. Outcome values for the non-fatal cohort (`R_NF`). Default covers common discharge/survival labels.

seed Integer. Random seed. Default 123L.

n_gibbs_burnin, n_gibbs_kept Integer. Only used when `fitted_model$residual_structure == "correlated"` – see `.gibbs_conditional_profile_probs()`. Defaults 10L/20L.

posterior_draw_indices Optional unique indices into the fitted posterior draws. When supplied, these replace random draw subsampling.

event_indices Optional canonical event indices to include. When supplied, only observed events with these indices are processed.

Details

Identity residual structure (`fitted_model$residual_structure == "identity"`): classes are conditionally independent given the linear predictor, so the missing-cell probabilities are computed analytically as $P(Y_{ed} = 1 \mid \theta) = \Phi(\mu_{ed})$ for each missing class d , and the profile probability over the missing dimensions is the exact product of per-class Bernoulli probabilities – no latent-variable simulation is used, so there is no added latent-profile-simulation noise. Rows carry `profile_generation_method = "conditional_analytic_identity"`.

Correlated residual structure: the missing latent dimensions are sampled conditional on the observed resistance SIGNS of the tested classes via a Gibbs sampler on the truncated multivariate normal – see `.gibbs_conditional_profile_probs()`. This replaces the earlier unconditional latent- Z simulation, which resampled tested cells along with untested ones and was therefore not observed-plus-imputed. Rows carry `profile_generation_method = "conditional_gibbs_correlated"` and ARE eligible for downstream DALY use (subject to the same panel-support and sampler-acceptability gates as identity-residual profiles – see `aggregate_profiles_for_daly()`). Because Gibbs is only run for `n_gibbs_burnin + n_gibbs_kept` iterations per draw rather than to

full convergence, correlated-residual profiles carry additional Monte Carlo error beyond identity-residual profiles at the same `n_posterior_draws_for_profiles`; increase `n_gibbs_kept` (and/or `n_gibbs_burnin`) for lower-noise estimates at higher compute cost.

Both residual structures produce, per event per posterior draw, a profile probability vector over the panel's 2^{D_p} enumerated profiles that (a) sums to exactly 1 and (b) assigns exactly 0 to any profile inconsistent with that event's observed cells – enforced by construction in both cases (an exact product for identity; Gibbs never visiting an inconsistent pattern for correlated), and checked with a hard `stop()` for the identity path since it has no other source of Monte Carlo noise to blur a genuine bug.

Class panels: the antibiotic-class panel enumerated for each hospital x pathogen pair is drawn from the approved eligibility rules computed at fit time (`fitted_model$eligibility_report`: marginal `n_tested/n_resistant/n_susceptible` thresholds, plus pairwise co-testing sufficiency for correlated-residual fits) – not from "tested at least once". See `.resolve_profile_class_panel()`.

Estimand: The posterior distribution over the observed event case-mix, conditional on each event's own observed AST results. This is labelled "observed_stewardship_event_mix".

Value

Named list: `event_profiles` (event-level posterior mean observed-plus-imputed profile probabilities, with `n_classes_observed/n_classes_missing` per event) and `aggregate_draws` (per-draw `R_ALL` [truly all events in the hp pair], `R_KNOWN_OUTCOME` [known-outcome cohort], and `R_NF` [nonfatal cohort], used by `aggregate_profiles_for_daly()` for credible intervals). Both tibbles contain all 2^D profiles per hospital-pathogen pair and carry `profile_generation_method` and `panel_reason` columns.

`compute_hospital_daly` *Compute Hospital-Level DALY Breakdown*

Description

Distributes pooled YLL/YLD totals to individual hospitals proportionally by deaths and discharges, and computes per-1 000 rates.

Usage

```
compute_hospital_daly(
  hospital_counts,
  total_deaths,
  total_discharged,
  yll_base,
  yll_associated,
  yll_attributable,
  yld_base,
  yld_associated,
  yld_attributable
)
```

Arguments

hospital_counts	Data frame with columns: center_name, deaths_h, discharged_h, cases_h.
total_deaths	Numeric. Pooled total deaths across all hospitals.
total_discharged	Numeric. Pooled total discharges.
yll_base	Numeric. Total baseline YLL.
yll_associated	Numeric. Total YLL associated with resistance.
yll_attributable	Numeric. Total YLL attributable to resistance.
yld_base	Numeric. Total baseline YLD.
yld_associated	Numeric. Total YLD associated.
yld_attributable	Numeric. Total YLD attributable.

Value

Data frame with hospital-level YLL, YLD, DALY (absolute and per 1 000 cases) columns.

```
compute_marginal_resistance
```

Compute Marginal Resistance per Pathogen and Antibiotic Class

Description

Collapses drug-level susceptibility data to antibiotic-class level per isolate ($R_{e,k,c} = 1$ if resistant to **any** drug in class c), then computes marginal resistance for every pathogen x class combination found in the data.

Usage

```
compute_marginal_resistance(
  data,
  pathogen_col = "organism_name",
  org_group_col = "org_group",
  isolate_col = "isolate_id",
  antibiotic_class_col = "antibiotic_class",
  antibiotic_value_col = "antibiotic_value",
  zero_threshold = 0,
  min_n_tested = 30,
  facility_col = NULL,
  facility_name = NULL,
  outcome_col = NULL,
  outcome_value = NULL
)
```

Arguments

data	Data frame. Pre-processed AMR data at isolate x antibiotic level (one row per isolate-antibiotic combination). Results must already be binary ("S" / "R"); no reclassification is applied.
pathogen_col	Character. Column with pathogen names. Default "organism_name".
org_group_col	Character. Column with organism group labels. Default "org_group".
isolate_col	Character. Column uniquely identifying each isolate. Default "isolate_id".
antibiotic_class_col	Character. Column with the antibiotic class for each drug. Default "antibiotic_class".
antibiotic_value_col	Character. Column with susceptibility result ("S" or "R"). Default "antibiotic_value".
zero_threshold	Numeric. Classes with marginal_resistance <= zero_threshold are listed in \$near_zero. Default 0.
min_n_tested	Integer or NULL. Minimum number of isolates that must have been tested for a pathogen-class combination to be retained. Combinations with n_tested < min_n_tested are dropped from \$marginal and from \$class_long, so the exclusion propagates automatically into compute_pairwise_coresistance() and compute_resistance_profiles(). Set to NULL or 0 to disable the filter. Default 30.
facility_col	Character or NULL. Name of the column identifying the facility/site. When provided together with facility_name, data are filtered to the specified facility before any computation. Both facility_col and facility_name must be supplied together or both left NULL. When provided, facility_col is also retained in \$class_long and \$marginal so that downstream steps can apply the same filter. Default NULL.
facility_name	Character or NULL. The facility value to retain (matched via == against facility_col). Default NULL.
outcome_col	Character or NULL. Name of the column containing patient outcomes (e.g. "final_outcome"). When provided together with outcome_value, data are filtered to isolates with the specified outcome before any computation. Both outcome_col and outcome_value must be supplied together or both left NULL. When provided, outcome_col is retained in \$class_long and \$marginal so that downstream steps can apply the same filter. Default NULL.
outcome_value	Character or NULL. The outcome value to retain (e.g. "discharged", "dead"; matched via == against outcome_col). Default NULL.

Details

Classes whose marginal resistance is at or below zero_threshold are listed in \$near_zero as a flag for downstream use – they are **not** removed here.

Value

Named list:

marginal Data frame with columns: pathogen_col, org_group_col, antibiotic_class_col, n_tested, n_resistant, marginal_resistance. Sorted descending by marginal_resistance within each pathogen.

near_zero Subset of marginal where marginal_resistance <= zero_threshold. These classes are candidates for exclusion in downstream profiling.

class_long Collapsed isolate x pathogen x class data frame (columns: isolate_col, pathogen_col, org_group_col, antibiotic_class_col, class_result). Pass this directly to compute_pairwise_coresistance()

Examples

```
## Not run:
marg <- compute_marginal_resistance(
  data           = amr_clean,
  pathogen_col   = "organism_name",
  org_group_col  = "org_group",
  isolate_col    = "isolate_id",
  antibiotic_class_col = "antibiotic_class",
  antibiotic_value_col = "antibiotic_value"
)

marg$marginal # full marginal resistance table
marg$near_zero # classes flagged as near-zero

## End(Not run)
```

compute_marginals_from_data

Compute Marginal Resistance Rates from Preprocessed Wide Data

Description

Takes the wide-format tibble produced by preprocess_for_profiles() and computes marginal resistance rates per pathogen x antibiotic class, optionally stratified by geography, year, and/or patient outcome. The counting unit is always isolate_id: a patient with multiple isolates contributes one count per isolate.

Usage

```
compute_marginals_from_data(
  data_wide,
  col_map,
  panel_map,
  stratify_by = NULL,
  outcome_col = NULL,
  min_n_tested = 30L,
  external_marginals = NULL,
  ext_col_map = list(pathogen_col = "pathogen", class_col = "antibiotic_class",
    geography_col = "geography", year_col = "year", rate_col = "resistance_prevalence")
)
```

Arguments

data_wide	Tibble. Output of preprocess_for_profiles()\$data_wide. One row per isolate; antibiotic-class columns contain "S", "R", or NA (not tested).
col_map	Named list. col_map_resolved from preprocess_for_profiles().
panel_map	Named list. Same panel used in preprocess_for_profiles().
stratify_by	Character vector or NULL. Dimensions to stratify over. Valid: "geography", "year". Default NULL.
outcome_col	Character or NULL. Column name for patient outcome. When supplied, separate marginals are computed for each outcome value (e.g. Died vs Discharged). Default NULL.
min_n_tested	Integer. Minimum tested-isolate count per (stratum x pathogen x class) cell. Cells below this threshold are dropped with a logged reason. Default 30L.
external_marginals	Data frame or NULL. Pre-modelled marginal resistance rates (e.g. GBD ST-GPR estimates). When provided, locally computed rates are replaced where a match exists on pathogen x class [x geography x year]. Must have columns matching ext_col_map. Default NULL.
ext_col_map	Named list. Column names in external_marginals. Default list(pathogen_col="pathogen", class_col="antibiotic_class", geography_col="geography", year_col="year", rate_col="resistance_prevalence").

Details

Optionally, externally modelled marginals (e.g. from GBD ST-GPR) can be supplied to override the locally computed values while retaining the pairwise correlation structure from the local data.

Value

Tibble with columns: pathogen, antibiotic_class, [stratum columns], n_tested, n_resistant, marginal_resistance, marginal_source ("computed" or "external").

compute_pairwise_coresistance

Compute Pairwise Co-resistance Matrices per Pathogen

Description

For every pathogen and every pair of antibiotic classes (c_i, c_j) that were both tested:

$$T_{k,i,j} = \sum_e \mathbf{1}(c_i \text{ tested} \wedge c_j \text{ tested})$$

$$R_{k,i,j} = \sum_e \mathbf{1}(R_{e,k,c_i} = 1 \wedge R_{e,k,c_j} = 1)$$

$$\text{Prev}_{k,i,j} = R_{k,i,j} / T_{k,i,j}$$

Usage

```
compute_pairwise_coresistance(
  marginal_output,
  pathogen_col = "organism_name",
  org_group_col = "org_group",
  isolate_col = "isolate_id",
  antibiotic_class_col = "antibiotic_class",
  min_co_tested = 10,
  facility_col = NULL,
  facility_name = NULL,
  outcome_col = NULL,
  outcome_value = NULL
)
```

Arguments

marginal_output	The list returned by compute_marginal_resistance(). The \$class_long element is used as input.
pathogen_col	Character. Must match the column name used in Step 1. Default "organism_name".
org_group_col	Character. Default "org_group".
isolate_col	Character. Default "isolate_id".
antibiotic_class_col	Character. Default "antibiotic_class".
min_co_tested	Integer. Pairwise cells with fewer than min_co_tested co-tested isolates are set to NA in the prevalence matrix. Default 10.
facility_col	Character or NULL. Column identifying the facility/site. When provided together with facility_name, filters class_long to the specified facility before building matrices. For this to work, compute_marginal_resistance() must have been called with the same facility_col argument so that the column is present in \$class_long. Both must be supplied together or both left NULL. Default NULL.
facility_name	Character or NULL. Facility value to retain. Default NULL.
outcome_col	Character or NULL. Column containing patient outcomes. When provided together with outcome_value, filters class_long to isolates with the specified outcome before building matrices. compute_marginal_resistance() must have been called with the same outcome_col so that the column is present in \$class_long. Both must be supplied together or both left NULL. Default NULL.
outcome_value	Character or NULL. Outcome value to retain (e.g. "discharged", "dead"). Default NULL.

Details

Matrices are computed for **all** tested classes – no filtering by marginal resistance or GBD core list is applied here.

Value

Named list. One entry per pathogen (keyed by pathogen name), each containing:

`prevalence` Square symmetric matrix of pairwise co-resistance rates. Diagonal and cells with `n < min_co_tested` are NA.

`T_matrix` Integer matrix of co-tested isolate counts.

`R_matrix` Integer matrix of co-resistant isolate counts.

`classes` Character vector of class names (row/column order).

Examples

```
## Not run:
marg <- compute_marginal_resistance(amr_clean, ...)
co_res <- compute_pairwise_coresistance(marg)

# Prevalence matrix for K. pneumoniae
co_res[["Klebsiella pneumoniae"]]$prevalence

# Co-tested counts
co_res[["Klebsiella pneumoniae"]]$T_matrix

## End(Not run)
```

```
compute_pairwise_from_data
```

Compute Pairwise Co-resistance Using Pearson Back-calculation

Description

For each antibiotic-class pair (A, B) and each pathogen [x stratum], this function:

1. Computes the Pearson correlation ρ_{AB} between binary resistance indicators across co-tested isolates.
2. Back-calculates pairwise prevalence using the GBD formula:

$$P(A \cap B) = P(A)P(B) + \rho_{AB}\sqrt{P(A)(1 - P(A)) \cdot P(B)(1 - P(B))}$$

where $P(A)$ and $P(B)$ come from marginals (which may include externally overridden values).

3. Caps the result to $\min(P(A), P(B))$ and floors it to 0. Cells below the floor use the independence product as fallback.

Usage

```
compute_pairwise_from_data(
  data_wide,
  marginals,
  col_map,
  panel_map,
  stratify_by = NULL,
  outcome_col = NULL,
  min_co_tested = 10L
)
```

Arguments

<code>data_wide</code>	Tibble. Output of <code>preprocess_for_profiles()</code> \$ <code>data_wide</code> .
<code>marginals</code>	Tibble. Output of <code>compute_marginals_from_data()</code> .
<code>col_map</code>	Named list. <code>col_map_resolved</code> .
<code>panel_map</code>	Named list. Same panel as used in preprocessing.
<code>stratify_by</code>	Character vector or NULL. Must match what was used in <code>compute_marginals_from_data()</code> . Default NULL.
<code>outcome_col</code>	Character or NULL. Default NULL.
<code>min_co_tested</code>	Integer. Pairs with fewer co-tested isolates are reported with <code>method = "independence_fallback"</code> . Default 10L.

Value

Tibble with one row per pathogen x class-pair [x stratum]:

- `pathogen, antibiotic_class_1, antibiotic_class_2` Keys.
- `n_co_tested` Isolates tested for both classes.
- `rho` Pearson correlation between binary resistance indicators.
- `pairwise_prevalence` Back-calculated $P(A \cap B)$, capped and floored.
- `method` "pearson_back_calc" or "independence_fallback".

```
compute_posterior_predictive_status
```

Classify overall posterior predictive fit from a discrepancy-statistics table

Description

Does NOT classify from a single tail probability. Groups supported statistics into families (marginal, pairwise, profile [complete-profile and resistant-count statistics – both address joint multidrug-resistance burden], hospital_heterogeneity, cluster [admission and patient clustering]) and flags a family when either (a) any statistic in it is SEVERE (`ppc_tail_probability` < `tail_severe` or > `1 - tail_severe`), or (b) the FRACTION of that family's supported statistics that are extreme exceeds `max_fraction_core_extreme`. Thresholds are configurable, not universal hard-coded truths, and are applied as discrepancy FLAGS, not frequentist rejection tests.

Usage

```
compute_posterior_predictive_status(ppc_statistics, thresholds = list())
```

Arguments

ppc_statistics Tibble returned by [compute_probit_ppc_statistics](#).

thresholds Named list overriding any of the defaults: `tail_warning = 0.025`, `tail_severe = 0.005`, `max_fraction_core_extreme = 0.20`. These are INITIAL AMR-PROJECT DEFAULTS, not universal statistical truths – validated only against this package’s own synthetic recovery scenarios (`test-probit-predictive-synthetic-recovery.R`). Expect to recalibrate them once real multi-pathogen experiment results are available; override via this argument rather than editing the defaults in place, so the thresholds actually used for any given run remain visible in `$thresholds_used`.

Details

Statistics with essentially ZERO posterior-predictive variance (`replicated_sd` at or near 0 – e.g. a discrete statistic that lands on the same value in every replicated state, such as `max(resistant_count)` pinned at its ceiling for a small class panel) are excluded from extreme/severe classification (though still returned in `ppc_statistics` with their true, unmodified `ppc_tail_probability`): `mean(T_rep >= T_obs)` is exactly 1 whenever every replicate exactly EQUALS the observed value, which is a well-known degeneracy of posterior-predictive tail probabilities for near-constant discrete statistics, not evidence of misfit – a discrepancy statistic that cannot vary under the posterior predictive distribution carries no discriminating information about model fit.

Value

List with status (one of "pass", "warning_marginal_ppc", "warning_pairwise_ppc", "warning_profile_ppc", "warning_hospital_heterogeneity_ppc", "warning_cluster_ppc", "fail_major_ppc_misfit", "insufficient_ppc_support"), reasons (character vector of every triggered family-level flag), `thresholds_used`, and `family_status` (per-family `n/n_extreme/n_severe/fraction_extreme`).

```
compute_prior_predictive_fingerprint
```

Fingerprint the prior-generative configuration of a fitted model

Description

Two experiments that share the same prior-generative structure (same residual structure, same beta/tau/LKJ priors, same declared random-effect block structure, same fixed-effect design-column definitions, same D) can safely REUSE a prior predictive check’s results rather than rerunning [simulate_probit_prior_predictive](#) for every experiment – prior predictive checking depends only on the generative structure, never on the fitted posterior. If ANY component changes (a prior, a scaling, the declared block structure), the fingerprint changes and any cached result must be invalidated.

Usage

```
compute_prior_predictive_fingerprint(fitted_model)
```

Arguments

`fitted_model` A fitted model object as returned by [fit_bayesian_multivariate_probit](#).

Value

List with fingerprint (a single character hash via `rlang::hash()`, already an **anumaan** dependency – no new package dependency introduced) and components (the exact named list that was hashed, for audit/debugging).

```
compute_prior_predictive_status
```

Classify plausibility of the prior predictive distribution

Description

Unlike posterior predictive checking, the goal here is NOT to make the simulated distribution match the observed data closely – it is to flag obviously implausible/extreme implications of the CHOSEN priors, before conditioning on any outcomes. Computes, per prior state and averaged across states: the fraction of event-class probabilities $\Phi(\mu_{ed})$ below 0.001 or above 0.999 (near-deterministic implied resistance), the fraction of generated events that are degenerate (all classes resistant or all susceptible), and the spread of per-"hospital" (the model's primary declared random-effect grouping) mean implied probability – a very large spread implies an implausibly extreme facility-to-facility prior.

Usage

```
compute_prior_predictive_status(prior_draws, thresholds = list())
```

Arguments

`prior_draws` An "amr_prior_predictive_draws" object from [simulate_probit_prior_predictive](#).

`thresholds` Named list overriding any of the defaults: `max_fraction_extreme_probability = 0.10`, `severe_fraction_extreme_probability = 0.50`, `max_fraction_degenerate_profiles = 0.50`, `severe_fraction_degenerate_profiles = 0.90`, `max_hospital_spread_sd = 0.35`. All thresholds are project decisions, not universal truths – override freely and document why.

Value

List with status (one of "pass", "warning_extreme_prior_predictions", "warning_excessive_hospital_heterogeneity", "warning_degenerate_profiles", "fail_implausible_prior_predictive", "insufficient_prior_check"), reasons, thresholds_used, and summary (the averaged plausibility fractions).

compute_probit_ppc_statistics

Compute AMR-specific posterior predictive discrepancy statistics

Description

Drives `ppc_draws$generate_state(s)` once per posterior state (see [simulate_probit_posterior_predictive](#)), streaming each state's complete and observation-masked replicate through every requested discrepancy statistic and accumulating only the compact per-state scalar contributions – never an $S \times N_{\text{events}} \times D$ array. Statistic families (Stan User's Guide-style posterior predictive checks, adapted to AMR resistance data):

marginal Per-class resistance rate, pooled and per eligible hospital x pathogen x class cell (reusing `fitted_model$eligibility_report$marginal`).

resistant_count Number of resistant classes per event, among whatever was tested for that event, summarised (mean/variance/median/ p90/max); plus, for fully-observed eligible panels, the full count distribution.

pairwise RR/RS/SR/SS co-resistance proportions for every sufficiently co-tested class pair (reusing `fitted_model$eligibility_report$pairwise`).

complete_profile All-susceptible/all-resistant/most-common-profile frequency, distinct-profile count, Shannon entropy, Simpson concentration, and the top-3 most frequent OBSERVED profiles (`profile_top_observed_frequency` – a purely data-dominant descriptive summary, deliberately NOT a claim of clinical importance; see the file header of `probit_posterior_predictive.R` for the distinction and the deferred `clinical_priority_profiles` extension point), restricted to adequately-supported fully-observed hospital x pathogen panels ([enumerate_binary_profiles](#)-style class panel via `.resolve_profile_class_panel()`).

hospital_heterogeneity SD/IQR/MAD/range of per-hospital resistance proportions among hospitals meeting `min_hospital_support`, per class.

admission_clustering, patient_clustering Within-/between-group same-class agreement and Hamming distance, only when the relevant role is declared eligible (see `random_effect_roles` and `random_effect_eligibility_audit` below); skipped cleanly (`support_status != "supported"`), never as a failure, when unsupported.

Usage

```
compute_probit_ppc_statistics(
  fitted_model_or_ppc_draws,
  n_states = 500L,
  seed = 123L,
  preserve_observation_mask = TRUE,
  statistics = c("marginal", "resistant_count", "pairwise", "complete_profile",
    "hospital_heterogeneity", "admission_clustering", "patient_clustering"),
  random_effect_roles = NULL,
  random_effect_eligibility_audit = NULL,
  min_hospital_support = 30L,
```

```

    min_complete_profile_events = 30L,
    ci_level = 0.95
  )

```

Arguments

- fitted_model_or_ppc_draws**
 Either a fitted model object (as for [simulate_probit_posterior_predictive](#)) or an "amr_ppc_draws" object already returned by that function – the latter avoids re-deriving posterior draw setup when statistics are recomputed for the same simulated replicates.
- n_states, seed, preserve_observation_mask**
 See [simulate_probit_posterior_predictive](#). Ignored when fitted_model_or_ppc_draws is already an "amr_ppc_draws" object (its own settings are used).
- statistics**
 Character vector of statistic families to compute. Default: all of "marginal", "resistant_count", "pairwise", "complete_profile", "hospital_heterogeneity", "admission_clustering", "patient_clustering".
- random_effect_roles**
 Optional named character vector/list mapping the name of a GROUPING COLUMN present in fitted_model\$event_metadata (the NAME of each entry) to one of "hospital", "admission", "patient" (the VALUE – a closed set of ROLE labels, never an assumption about what any particular experiment's columns are actually named), e.g. c(admission_id = "admission"). Deliberately decoupled from whether that column is declared as a random-effect block in this particular fit – e.g. tagging a column that is NOT modelled as a random intercept with role "admission" is exactly how a fit lacking an admission random effect gets checked for whether it fails to reproduce observed within-admission clustering (see Stan User's Guide- style synthetic recovery tests in test-probit-predictive-synthetic-recovery.R). NULL (default) means no admission/patient-role clustering statistics can run (reported with support_status = "role_not_mapped").
- random_effect_eligibility_audit**
 Optional data frame with columns block (role: "hospital"/"admission"/"patient") and status ("eligible_primary"/"eligible_sensitivity_only"/"not_eligible") – e.g. the pathogen-specific random-effect eligibility audit produced by the analysis layer. When NULL, a generic structural fallback check is used instead (see [simulate_probit_posterior_predictive](#)'s file header for the general never-hardcode-eligibility-in-the-package philosophy this mirrors).
- min_hospital_support**
 Integer; minimum tested events for a hospital to enter the hospital-heterogeneity comparison. Default 30.
- min_complete_profile_events**
 Integer; minimum fully-observed events for a hospital x pathogen panel to enter complete-profile / full resistant-count-distribution statistics. Default 30.
- ci_level**
 Numeric; determines the replicated_q025/ replicated_q975 quantile levels ((1-ci_level)/2 and its complement). Default 0.95.

Value

A tibble with columns `statistic_name`, `stratum`, `observed_value`, `replicated_mean`, `replicated_sd`, `replicated_q025`, `replicated_q50`, `replicated_q975`, `ppc_tail_probability`, `ppc_two_sided`, `n_replications`, `support_status`.

References

Stan Development Team. "Posterior and Prior Predictive Checks." Stan User's Guide. <https://mc-stan.org/docs/stan-users-guide/posterior-predictive-checks.html>

`compute_processing_statistics`

Compute Processing Statistics

Description

Calculates performance and processing statistics.

Usage

```
compute_processing_statistics(raw_data, processed_data, preprocessing_log)
```

Arguments

`raw_data` Original data
`processed_data` Processed data
`preprocessing_log`
 Full log

Value

List with statistics

`compute_profile_validation_status`

Summarise Validation Checks into a Profile-Validation Status

Description

Combines the outputs of `validate_marginal_calibration()`, `validate_pairwise_calibration()`, `validate_complete_profile_calibration()`, and `mask_and_validate_ast()` into a single `profile_validation_status` field. **This is deliberately separate from** `diagnostic_status` (see `fit_bayesian_multivariate_probit()`): `diagnostic_status` describes HMC sampler/model-fitting health (R-hat, ESS, divergences, E-BFMI, tree-depth); `profile_validation_status` describes predictive agreement with the observed data and data support for that agreement. A model can converge cleanly (`diagnostic_status` = "pass") while its profile predictions are poorly calibrated, or vice versa, and the two should never be merged into one flag.

Usage

```
compute_profile_validation_status(
  marginal_tbl = NULL,
  pairwise_tbl = NULL,
  complete_profile_tbl = NULL,
  masked_summary = NULL,
  thresholds = list()
)
```

Arguments

marginal_tbl Tibble from `validate_marginal_calibration()`, or NULL if not run.

pairwise_tbl Tibble from `validate_pairwise_calibration()`, or NULL if not run.

complete_profile_tbl Tibble from `validate_complete_profile_calibration()`, or NULL if not run.

masked_summary Tibble (`$summary` element) from `mask_and_validate_ast()`, or NULL if not run.

thresholds Named list overriding any of: `max_mean_abs_error_marginal` (default 0.10), `min_marginal_coverage` (default 0.80), `max_mean_abs_error_pairwise` (default 0.10), `min_pairwise_coverage` (default 0.80), `min_complete_profile_panels_evaluated` (default 1L), `max_masked_brier` (default 0.25), `min_masked_auroc` (default 0.6).

Details

All thresholds are explicit parameters with documented defaults – none are silently hard-coded elsewhere. Pass thresholds to override any subset; unspecified entries keep their default.

In-sample masked validation can never produce a bare "pass" on its own. When `masked_summary$validation_mode == "in_sample_no_refit"` (see `mask_and_validate_ast()`), the masked cells were already seen during fitting, so predicting them well is not genuine validation. If every threshold is otherwise cleared, the status becomes `"in_sample_validation_only"` rather than `"pass"` – a real `"pass"` requires either `validation_mode == "refit_masked"` (a genuine holdout) or no masked-AST check at all (the other observed-versus-model checks being used purely descriptively).

Value

Named list: `status` (one of `"pass"`, `"in_sample_validation_only"`, `"warning_marginal_miscalibration"`, `"warning_pairwise_miscalibration"`, `"warning_sparse_complete_profiles"`, `"fail_masked_ast_validation"`, or `"not_evaluated"` when no inputs were supplied), `reasons` (character vector of every triggered condition – including `"masked_validation_in_sample_only"` as an informational reason even when a more severe status wins – the most severe status is returned but all are listed), and `thresholds_used` (the resolved threshold list, for audit trail).

compute_resistance_profiles

Compute Resistance Profile Probabilities per Pathogen

Description

For each pathogen k with n_k antibiotic classes, enumerates all 2^{n_k} binary resistance profiles $\delta \in \{0, 1\}^{n_k}$ and estimates their probabilities by solving a simplex-constrained weighted least-squares Quadratic Programme (GBD equation 7.5.1.3):

Usage

```
compute_resistance_profiles(
  marginal_output,
  coresistance_output,
  pathogens = NULL,
  top_n_pathogens = NULL,
  exclude_near_zero = TRUE,
  top_n_classes = NULL,
  sigma_sq = 1,
  ridge = 1e-08,
  pathogen_col = "organism_name",
  antibiotic_class_col = "antibiotic_class",
  facility_col = NULL,
  facility_name = NULL,
  outcome_col = NULL,
  outcome_value = NULL,
  n_cores = 1L
)
```

Arguments

`marginal_output`

List returned by `compute_marginal_resistance()`.

`coresistance_output`

List returned by `compute_pairwise_coresistance()`.

`pathogens`

Character vector. Pathogen(s) to process. NULL (default) processes every pathogen in `marginal_output`.

`top_n_pathogens`

Integer or NULL (default). When set, only the top `top_n_pathogens` pathogens ranked by total isolates tested (sum of `n_tested` across all antibiotic classes, descending) are processed. Applied after the `pathogens` argument filter. Useful for focusing on the most data-rich pathogens – e.g. `top_n_pathogens = 5` runs profiles for only the 5 most-tested pathogens.

`exclude_near_zero`

Logical. If TRUE (default), antibiotic classes that appear in `marginal_output$near_zero` for a given pathogen are excluded from profile enumeration.

top_n_classes	Integer or NULL (default). When set, only the top top_n_classes antibiotic classes ranked by n_tested (descending) are kept per pathogen before profile enumeration. Useful for capping the combinatorial explosion (2^n profiles) for pathogens tested against many drug classes – e.g. top_n_classes = 5 gives at most 32 profiles. Applied after exclude_near_zero.
sigma_sq	Positive numeric. Assumed variance for each constraint (uniform). Default 1.
ridge	Positive numeric. Ridge term added to the QP Hessian for numerical stability. Default 1e-8.
pathogen_col	Character. Column name for pathogens. Must match the column used in Steps 1-2. Default "organism_name".
antibiotic_class_col	Character. Column name for antibiotic classes. Default "antibiotic_class".
facility_col	Character or NULL. Column identifying the facility/site. When provided together with facility_name, filters marginal_output\$marginal and marginal_output\$near_zero to the specified facility before profile enumeration. For this to work, compute_marginal_resistance() must have been called with the same facility_col argument. Both must be supplied together or both left NULL. Default NULL.
facility_name	Character or NULL. Facility value to retain. Default NULL.
outcome_col	Character or NULL. Column containing patient outcomes. When provided together with outcome_value, filters marginal_output\$marginal and marginal_output\$near_zero to the specified outcome before profile enumeration. compute_marginal_resistance() must have been called with the same outcome_col argument. Both must be supplied together or both left NULL. Default NULL.
outcome_value	Character or NULL. Outcome value to retain (e.g. "discharged", "dead"). Default NULL.
n_cores	Integer. Number of CPU cores for parallel computation. Default 1L (sequential).

Details

$$\hat{p} = \arg \min_{p \in \Delta_{2^n}} \sum_{i=1}^m \frac{(m_i^\top p - v_i)^2}{\sigma_i^2}$$

where $m = n(n+1)/2$ data-derived linear constraints encode **n marginal** resistance rates and **n(n-1)/2 pairwise** co-resistance rates, and Δ is the standard probability simplex.

Constraint rows in M:

Marginal (rows 1 to n) Row d : $M_{d,\delta} = 1$ iff $\delta_d = 1$ (i.e., class d is resistant in profile δ).

Constraint: $\sum_{\delta} M_{d,\delta} p_{\delta} = \hat{r}_{kd}$.

Pairwise (rows n+1 to m) Row (d_1, d_2) : $M_{d_1 d_2, \delta} = 1$ iff $\delta_{d_1} = 1 \wedge \delta_{d_2} = 1$. Constraint:

$\sum_{\delta} M_{d_1 d_2, \delta} p_{\delta} = \hat{r}_{k, d_1 d_2}$. When a pairwise estimate is unavailable (too few co-tested isolates), the product of marginals (independence assumption) is used as fallback.

The QP is solved via `quadprog::solve.QP`. A small ridge term (ridge) is added to the Hessian to guarantee strict positive-definiteness. On solver failure the pathogen gets a uniform distribution over all profiles.

Performance Notes: This function has been optimized for speed with vectorized profile generation and label creation (10-100x faster than previous versions). However, computational complexity is still exponential in the number of classes:

- **n <= 14:** Fast (seconds to minutes)
- **n = 15:** Moderate (minutes)
- **n >= 16:** Slow and memory-intensive (use top_n_classes)

For large datasets with many pathogens, consider using top_n_classes to limit each pathogen to its most-tested classes (e.g., top_n_classes = 12).

Value

Named list, one entry per pathogen, each a list with:

profiles Data frame: profile (character label, e.g. "RSR"), probability ($\hat{p}_\delta$), and one binary (0/1) integer column per antibiotic class indicating whether that class is resistant (1) or susceptible (0) in each profile.

classes Character vector of antibiotic class names used (alphabetical; bit 0 = classes[1]).

n_classes Integer. Number of classes used.

constraint_residuals Named numeric vector of $m_i^\top \hat{p} - v_i$ for each constraint. Small absolute values indicate good constraint satisfaction.

Examples

```
## Not run:
marg <- compute_marginal_resistance(amr_clean)
co_res <- compute_pairwise_coresistance(marg)
rp <- compute_resistance_profiles(marg, co_res)

# All profiles and probabilities for K. pneumoniae
rp[["Klebsiella pneumoniae"]]$profiles

# Constraint residuals (quality check)
rp[["Klebsiella pneumoniae"]]$constraint_residuals

# Single pathogen
rp_kp <- compute_resistance_profiles(
  marg, co_res,
  pathogens = "Klebsiella pneumoniae"
)

## End(Not run)
```

daly_add_rr_mappings *Add RR Pathogen and Drug Class Mappings*

Description

Maps organism names to GBD RR pathogen categories and antibiotic class names to RR drug categories in a single pass. Either mapping is skipped silently when the corresponding column is absent from data.

Usage

```
daly_add_rr_mappings(
  data,
  organism_col = "organism_normalized",
  class_col = "antibiotic_class"
)
```

Arguments

data	Data frame with organism and/or antibiotic class columns.
organism_col	Character. Column with organism names to map to rr_pathogen. Set to NULL to skip. Default "organism_normalized".
class_col	Character. Column with antibiotic class names to map to rr_drug. Set to NULL to skip. Default "antibiotic_class".

Details

Pathogen mapping uses the organism taxonomy (get_organism_taxonomy()); drug class mapping uses inst/extdata/WHO_aware_class.csv.

Value

Data frame with rr_pathogen and/or rr_drug columns appended.

daly_assign_rr_to_profiles

Assign Per-Class LOS RR to Resistance Profiles (Max Rule)

Description

For each resistance profile delta (from compute_resistance_profiles()), determines the profile-level RR_kd_LOS using the GBD max rule: $RR_kd_LOS = \max_{c \in C_R(d)} RR_kc_LOS$ [if $C_R(d)$ non-empty] = 1 [if $d = \text{all-susceptible}$] where $C_R(d) = \{c : d_c = 1\}$. The CI reported for each profile is that of its dominant (max-RR) class.

Usage

```
daly_assign_rr_to_profiles(
  profiles_output,
  rr_table,
  pathogen_col = "pathogen",
  class_col = "antibiotic_class",
  rr_col = "RR_LOS",
  fallback_rr = 1,
  los_r_col = NULL,
  los_s_col = NULL,
  los_unit = c("years", "days")
)
```

Arguments

profiles_output	Named list from compute_resistance_profiles().
rr_table	Data frame from daly_fit_los_rr() or daly_fit_los_rr_distribution(). Must have columns pathogen_col, class_col, rr_col, and optionally CI_lower / CI_upper.
pathogen_col	Character. Default "pathogen".
class_col	Character. Default "antibiotic_class".
rr_col	Character. Default "RR_LOS".
fallback_rr	Numeric. RR for resistant classes with no match. Default 1 (no attributable effect).
los_r_col	Character or NULL. Column in rr_table with absolute mean LOS for resistant patients (e.g. "mean_LOS_R"). When supplied together with los_s_col, LOSR_years / LOSS_years are added to the output. Default NULL (no absolute LOS propagated; behaviour identical to before this parameter existed).
los_s_col	Character or NULL. Column in rr_table with absolute mean LOS for susceptible patients (e.g. "mean_LOS_S"). Required when los_r_col is supplied.
los_unit	Character. Unit of los_r_col / los_s_col in rr_table: "years" (used as-is) or "days" (divided by 365, matching the day-to-year conversion already used in daly_fit_los_rr_distribution()). Default "years". Ignored when los_r_col is NULL.

Details

If rr_table was fit with a syndrome_name filter, its RR values are syndrome-specific but get applied here to all profiles of a pathogen – i.e. it assumes syndrome-invariant LOS prolongation. Refit with syndrome_name = NULL for a RR pooled across syndromes.

Optionally, when los_r_col and los_s_col are supplied, also propagates the **absolute** mean LOS for the profile's dominant class (resistant and susceptible) as LOSR_years / LOSS_years. This does not change the RR assignment above; it is an additive lookup against the dominant class already selected by the max rule, using whichever absolute-LOS columns are present in rr_table (e.g. mean_LOS_R / mean_LOS_S from daly_fit_los_rr_distribution(), or mean_los_resistant / mean_los_susceptible from daly_fit_los_rr()). LOSR_years/LOSS_years are set to NA for the all-susceptible profile, which has no dominant class.

Value

Named list (one entry per pathogen): original profiles data frame augmented with RR_LOS_profile, dominant_class, (if available) CI_lower_profile / CI_upper_profile, and (when los_r_col / los_s_col are supplied) LOSR_years / LOSS_years.

daly_calc_case_fatality
<i>Calculate case fatality ratio by syndrome and pathogen (CFR_{Lk})</i>

Description

Computes the case fatality ratio (CFR) for each pathogen (k) within a specified infectious syndrome (L) using facility-level microbiology data.

Usage

```
daly_calc_case_fatality(  
  data,  
  syndrome_col,  
  syndrome_name,  
  specimen_col,  
  specimen_name,  
  polymicrobial_col,  
  patient_col,  
  pathogen_col,  
  outcome_col,  
  death_value = "Died",  
  glass_ref = NULL,  
  facility_col = NULL,  
  facility_name = NULL,  
  pathogen_name = NULL  
)
```

Arguments

data	Data frame of facility-level microbiology records.
syndrome_col	Character. Column containing infectious syndrome labels (L).
syndrome_name	Character. Syndrome to analyse.
specimen_col	Character. Column containing specimen type.
specimen_name	Character. Specimen to restrict to (e.g., "Blood").
polymicrobial_col	Character. Column flagging polymicrobial patients (1 = polymicrobial, 0 = monomicrobial).
patient_col	Character. Unique patient identifier column.

pathogen_col	Character. Pathogen (organism) column (k).
outcome_col	Character. Final patient outcome column.
death_value	Character. Value indicating death. Default "Died".
glass_ref	Character vector of valid pathogen names, or a data frame with columns specimen and pathogen. Applied to polymicrobial patients only. NULL skips GLASS filtering.
facility_col	Character or NULL. Facility identifier column.
facility_name	Character or NULL. Restricts to a single facility.
pathogen_name	Character vector or NULL. Filter to specific pathogen(s).

Details

The unit of analysis is the **patient**. Each patient contributes total weight 1, distributed equally across their valid pathogens via a fractional weight $1/m_r$ (where m_r is the number of distinct valid pathogens for patient r). This ensures polymicrobial patients are not double-counted.

For **polymicrobial patients** (polymicrobial_col == 1), only pathogens listed in the GLASS reference (glass_ref) for the given specimen type are retained before weighting. Monomicrobial patients (polymicrobial_col == 0) are never filtered.

When facility_col is supplied and facility_name is NULL, both per-facility and pooled CFR are returned. The pooled CFR sums weighted deaths and totals across facilities before dividing.

Value

A list:

cfr_table Pooled CFR: pathogen_col, weighted_deaths, weighted_total, CFR_LK.

facility_level Per-facility CFR (only when facility_col is supplied and facility_name is NULL).

daly_calc_cr_l	<i>Calculate the CFR adjustment factor (CR_L)</i>
----------------	---

Description

Computes CR_L, the factor that adjusts a hospital-derived CFR to account for infection cases managed outside the inpatient setting. The adjustment type for each syndrome is looked up from the adjustment_ref table (loaded from 'inst/extdata/adjustment_for_CFR').

Usage

```
daly_calc_cr_l(
  data,
  syndrome_col,
  syndrome_name,
  patient_col,
  visit_type_col,
```

```

    inpatient_value = "Inpatient",
    outpatient_value = "Outpatient",
    adjustment_ref,
    facility_col = NULL,
    facility_name = NULL
  )

```

Arguments

<code>data</code>	Data frame of facility-level records.
<code>syndrome_col</code>	Character. Column containing infectious syndrome labels.
<code>syndrome_name</code>	Character. Syndrome to compute CR_L for.
<code>patient_col</code>	Character. Unique patient identifier column.
<code>visit_type_col</code>	Character. Column indicating visit type per record (inpatient / outpatient).
<code>inpatient_value</code>	Character. Value in <code>visit_type_col</code> that denotes an inpatient visit. Default "Inpatient".
<code>outpatient_value</code>	Character. Value in <code>visit_type_col</code> that denotes an outpatient visit. Default "Outpatient".
<code>adjustment_ref</code>	Data frame with columns <code>infectious_syndrome</code> and <code>adjustment_factor_on_CFR</code> . Load from 'inst/extdata/adjustment_for_CFR'.
<code>facility_col</code>	Character or NULL. Facility identifier column. When provided (and <code>facility_name</code> is NULL), CR_L is returned per facility.
<code>facility_name</code>	Character or NULL. If provided, restricts to that facility only.

Details

Three adjustment types are supported:

None CR_L = 1. The hospital CFR applies directly (e.g., BSI, Meningitis, hospital-acquired infections).

Inpatient ratio CR_L = (patients with ≥ 1 inpatient visit) / (all patients). Used when community cases are captured partly in outpatient data (e.g., community-acquired LRI, UTI).

Outpatient to inpatient ratio CR_L = (patients with ≥ 1 outpatient AND ≥ 1 inpatient visit) / (patients with ≥ 1 outpatient visit). Used for syndromes where OP-to-IP transition captures disease severity (e.g., STI, Skin, Eye, Oral, Bone/joint infections).

Value

Data frame with columns `syndrome` (= `syndrome_name`), `adjustment_type`, `CR_L`, and (when `facility_col` is supplied) `facility_col`. Additional columns (`n_inpatient`, `n_total`, etc.) give the raw counts used.

daly_calc_deaths_by_cause

Calculate Deaths by Underlying Cause (D_J)

Description

Computes D_J, the number of deaths for each underlying cause J from population-level vital registration or mortality data. Optionally stratified by grouping variables such as age group, sex, year, or location.

Usage

```
daly_calc_deaths_by_cause(
  pop_data,
  cause_col = "cause_of_death",
  deaths_col = NULL,
  groupby_cols = NULL
)
```

Arguments

pop_data	Data frame. Population-level vital registration or mortality data. Required – this function does not accept facility-level data.
cause_col	Character. Column containing the underlying cause of death (cause J), e.g. an ICD-10 code or cause name. Default "cause_of_death".
deaths_col	Character. Column with pre-aggregated death counts. Set to NULL if each row represents one individual death record. Default NULL.
groupby_cols	Character vector. Additional stratification columns (e.g., c("Age_bin", "gender", "year")). Default NULL.

Details

This function requires population-level data. If only facility-level data are available, use `daly_calc_deaths_by_syndrome()` with `facility_data` instead, which directly counts deaths by syndrome.

Value

Data frame with columns: `cause_col`, any `groupby_cols`, `D_J` (death count), `D_J_method` ("population"), `D_J_confidence` ("high").

References

Antimicrobial Resistance Collaborators. Global burden of bacterial antimicrobial resistance in 2019. *Lancet*. 2022.

Examples

```
## Not run:
# One row per death record
d_j <- daly_calc_deaths_by_cause(
  pop_data = vital_reg,
  cause_col = "icd10_cause"
)

# Pre-aggregated counts, stratified by age and sex
d_j <- daly_calc_deaths_by_cause(
  pop_data      = vital_reg,
  cause_col     = "icd10_cause",
  deaths_col    = "n_deaths",
  groupby_cols  = c("Age_bin", "gender")
)

## End(Not run)
```

daly_calc_deaths_by_syndrome

Calculate Deaths by Infectious Syndrome (D_L)

Description

Computes D_L, the number of deaths due to each infectious syndrome L.

Usage

```
daly_calc_deaths_by_syndrome(
  d_j = NULL,
  s_j = NULL,
  m_lj = NULL,
  pop_data = NULL,
  facility_data = NULL,
  cause_col = "cause_of_death",
  syndrome_col = "syndrome",
  syndrome = NULL,
  deaths_col = NULL,
  infection_flag_col = "is_infection_death",
  outcome_col = "final_outcome",
  death_value = "Died",
  patient_col = "patient_id",
  facility_col = NULL,
  facility_name = NULL,
  groupby_cols = NULL
)
```

Arguments

d_j	Data frame. Output of daly_calc_deaths_by_cause(). Supply together with s_j and m_lj for Mode 1. Default NULL.
s_j	Data frame. Output of daly_calc_infection_fraction(). Default NULL.
m_lj	Data frame. Output of daly_calc_syndrome_fraction(). Default NULL.
pop_data	Data frame. Raw population-level data for Mode 2. Default NULL.
facility_data	Data frame. Facility-level data for Mode 3 fallback. Default NULL.
cause_col	Character. Underlying cause column used in Modes 1 and 2. Default "cause_of_death".
syndrome_col	Character. Column containing syndrome labels in both population and facility data. Default "syndrome".
syndrome	Character. Optional. Name of a specific syndrome to filter results to (e.g., "Bloodstream infection"). NULL returns all syndromes. Default NULL.
deaths_col	Character. Pre-aggregated deaths column in pop_data (Mode 2 only). NULL means each row is one death record. Default NULL.
infection_flag_col	Character. Binary infection flag column in pop_data (Modes 1 and 2). Default "is_infection_death".
outcome_col	Character. Outcome column in facility_data (Mode 3). Default "final_outcome".
death_value	Character. Value in outcome_col that represents death. Accepts any string, e.g., "Died" or "Death". Default "Died".
patient_col	Character. Unique patient identifier column in facility_data. Deaths are counted as distinct patients, not rows. Default "patient_id".
facility_col	Character. Column containing facility names in facility_data. Required when facility_name is specified. Default NULL.
facility_name	Character. Name of a specific facility to restrict the analysis to. NULL uses all facilities combined. Default NULL.
groupby_cols	Character vector. Additional stratification columns (e.g., c("Age_bin", "gender")). Default NULL.

Details

Three operating modes, evaluated in priority order:

1. **Pre-computed components** (HIGH confidence): pass the outputs of daly_calc_deaths_by_cause(), daly_calc_infection_fraction(), and daly_calc_syndrome_fraction() via d_j, s_j, and m_lj. Computes: $D_L = \text{sum}_J(D_J * S_J * M_{LJ})$.
2. **Raw population data** (HIGH confidence): pass pop_data and all three components are computed internally before combining. Computes: $D_L = \text{sum}_J(D_J * S_J * M_{LJ})$.
3. **Facility fallback** (LOW confidence): when no population inputs are available, counts the number of **unique patients** who died for each syndrome directly from facility_data.

Use syndrome to restrict results to a single syndrome of interest. Use facility_col and facility_name to restrict the facility fallback to a specific site.

Value

Data frame with columns: syndrome_col, any groupby_cols, D_L (deaths by syndrome), D_L_method, D_L_confidence.

References

Antimicrobial Resistance Collaborators. Global burden of bacterial antimicrobial resistance in 2019. Lancet. 2022.

Examples

```
## Not run:
# Mode 1: pass pre-computed components, all syndromes
d_j <- daly_calc_deaths_by_cause(pop_data = vr, cause_col = "icd10")
s_j <- daly_calc_infection_fraction(pop_data = vr, cause_col = "icd10")
m_lj <- daly_calc_syndrome_fraction(
  pop_data = vr, cause_col = "icd10", syndrome_col = "syndrome"
)
d_l <- daly_calc_deaths_by_syndrome(
  d_j = d_j, s_j = s_j, m_lj = m_lj,
  cause_col = "icd10", syndrome_col = "syndrome"
)

# Mode 2: raw population data, filter to one syndrome
d_l <- daly_calc_deaths_by_syndrome(
  pop_data      = vital_reg,
  cause_col     = "icd10_cause",
  syndrome_col  = "infectious_syndrome",
  syndrome     = "Bloodstream infection"
)

# Mode 3: facility fallback, one facility, all syndromes
d_l <- daly_calc_deaths_by_syndrome(
  facility_data = amr_data,
  syndrome_col  = "specimen_normalized",
  outcome_col   = "final_outcome",
  death_value   = "Died",
  patient_col   = "patient_id",
  facility_col  = "location",
  facility_name = "Mumbai"
)

# Mode 3: facility fallback, all facilities, one syndrome, stratified by age
d_l <- daly_calc_deaths_by_syndrome(
  facility_data = amr_data,
  syndrome_col  = "specimen_normalized",
  syndrome     = "Blood",
  death_value   = "Died",
  groupby_cols  = c("Age_bin")
)

## End(Not run)
```

daly_calc_incidence_from_cfr

Calculate syndrome incidence from deaths, CFR, and CR_L (formula-based)

Description

Estimates the number of incident cases of syndrome L using:

Usage

```
daly_calc_incidence_from_cfr(
  deaths_L,
  cfr_lk_tbl,
  P_Lk_prime_tbl,
  CR_L = 1,
  pathogen_col = "pathogen",
  cfr_col = "CFR_LK",
  plk_col = "P_Lk_prime",
  facility_col = NULL,
  deaths_col = "deaths"
)
```

Arguments

deaths_L	Numeric scalar (pooled mode) or data frame with a facility_col column and a deaths_col column (facility-level mode).
cfr_lk_tbl	Data frame with at minimum columns pathogen_col and cfr_col. Typically the cfr_table element from calculate_cfr_lk().
P_Lk_prime_tbl	Data frame with pathogen_col and plk_col. Use the P_Lk_prime (pooled) element from calculate_P_Lk_prime_BSI() or calculate_P_Lk_prime().
CR_L	Numeric scalar. CFR adjustment factor from daly_calc_cr_l(). Default 1 (no adjustment).
pathogen_col	Character. Pathogen column in both tables. Default "pathogen".
cfr_col	Character. CFR column in cfr_lk_tbl. Default "CFR_LK".
plk_col	Character. P'LK column in P_Lk_prime_tbl. Default "P_Lk_prime".
facility_col	Character or NULL. Facility identifier. When provided, cfr_lk_tbl and P_Lk_prime_tbl must each contain facility_col, and deaths_L must be a data frame with facility_col + deaths_col.
deaths_col	Character. Column in deaths_L data frame containing death counts. Default "deaths". Ignored when deaths_L is a scalar.

Details

$$I_L = \frac{D_L}{\text{CFR}_L \times \text{CR}_L}$$

where the syndrome-level CFR is the pathogen-weighted average:

$$\text{CFR}_L = \sum_k P'_{Lk} \times \text{CFR}_{Lk}$$

Use this when you have population- or facility-level death counts and want to back-calculate incidence. For a direct patient count from facility data, use `count_incident_cases()` instead.

Value

Data frame with columns `deaths`, `CFR_L`, `CR_L`, `I_L` (incident cases), and `facility_col` when applicable.

`daly_calc_infection_fraction`

Calculate Infection Fraction of Deaths by Cause (S_J)

Description

Computes `S_J`, the fraction of deaths for underlying cause `J` that were related to infection. Values range from 0 (no infection involvement) to 1 (all deaths for cause `J` involved infection).

Usage

```
daly_calc_infection_fraction(
  pop_data,
  cause_col = "cause_of_death",
  infection_flag_col = "is_infection_death",
  groupby_cols = NULL
)
```

Arguments

<code>pop_data</code>	Data frame. Population-level mortality data. Required.
<code>cause_col</code>	Character. Underlying cause of death column (cause <code>J</code>). Default <code>"cause_of_death"</code> .
<code>infection_flag_col</code>	Character. Binary column (TRUE/FALSE or 1/0) in <code>pop_data</code> indicating whether the death involved infection. Default <code>"is_infection_death"</code> .
<code>groupby_cols</code>	Character vector. Stratification columns present in <code>pop_data</code> . Default <code>NULL</code> .

Details

This function requires population-level data where each death record carries a binary flag indicating whether infection was involved. If only facility data are available, use `daly_calc_deaths_by_syndrome()` with `facility_data` instead.

Value

Data frame with columns: `cause_col`, any `groupby_cols`, `D_J` (total deaths), `infection_deaths` (infection-related death count), `S_J` (infection fraction 0-1), `S_J_method`, `S_J_confidence`.

References

Antimicrobial Resistance Collaborators. Global burden of bacterial antimicrobial resistance in 2019. *Lancet*. 2022.

Examples

```
## Not run:
s_j <- daly_calc_infection_fraction(
  pop_data      = vital_reg,
  cause_col     = "icd10_cause",
  infection_flag_col = "is_infectious",
  groupby_cols  = c("Age_bin", "gender")
)

## End(Not run)
```

daly_calc_paf_los	<i>Compute PAF for length of stay per resistance profile</i>
-------------------	--

Description

Computes the population attributable fraction (PAF) for length of stay from a named list of profile data frames, each containing resistance-profile probabilities and profile-level relative risk of LOS.

Usage

```
daly_calc_paf_los(
  profiles_with_rr,
  probability_col = "probability",
  rr_profile_col = "RR_LOS_profile",
  profile_col = "profile"
)
```

Arguments

- `profiles_with_rr` Named list returned by `assign_rr_to_profiles()` or `filter_profiles_to_rr_classes()`. Each entry is a data frame of resistance profiles for one pathogen.
- `probability_col` Character. Column name for profile probability (must sum to 1 within each pathogen). Default "probability".
- `rr_profile_col` Character. Column name for profile-level LOS relative risk. Default "RR_LOS_profile".
- `profile_col` Character. Column name for the profile identifier. Default "profile".

Details

For each pathogen the formula is: $PAF = \sum_d R'_{kd} * (RR_{kd} - 1) / (1 + \sum_d R'_{kd} * (RR_{kd} - 1))$

where R'_{kd} is the probability of resistance profile d and RR_{kd} is the corresponding relative LOS multiplier.

Value

Named list (one entry per pathogen) with columns `profile`, `probability`, `rr_profile_col`, `numerator`, `PAF_LOS`, and `denominator` added to the input profile data frame.

`daly_calc_paf_mortality`

Compute Mortality Population Attributable Fraction per Resistance Profile

Description

Computes `PAF_kd_mortality` for each pathogen k and resistance profile δ using the GBD multi-exposure Levin formula, substituting the mortality odds ratio (`OR_death`) from `fit_mortality_rr_logistic()` in place of the LOS relative risk used by `compute_paf_los()`:

Usage

```
daly_calc_paf_mortality(
  profiles_with_rr,
  probability_col = "probability",
  rr_profile_col = "RR_LOS_profile",
  profile_col = "profile"
)
```

Arguments

- `profiles_with_rr` Named list from `assign_rr_to_profiles()` called with `rr_col = "OR_death"`. Each element is a profile data frame for one pathogen.
- `probability_col` Character. Profile probability column name. Default "probability".
- `rr_profile_col` Character. Profile-level OR column as produced by `assign_rr_to_profiles()`. Default "RR_LOS_profile".
- `profile_col` Character. Profile label column. Default "profile".

Details

$$\text{PAF}_{kd,\text{mort}} = \frac{R'_{K\delta}(\text{OR}_{K\delta} - 1)}{1 + \sum_{\delta} R'_{K\delta}(\text{OR}_{K\delta} - 1)}$$

The all-susceptible profile carries $\text{OR} = 1$ and contributes 0. The denominator equals $E[\text{OR}_k] = \sum_{\delta} R'_{K\delta} \cdot \text{OR}_{K\delta}$, numerically identical to the denominator produced by `compute_paf_los()`.

Overall mortality PAF for pathogen k :

$$\text{PAF}_{k,\text{mort}} = \sum_{\delta} \text{PAF}_{kd,\text{mort}} = \frac{\sum_{\delta} R'_{K\delta}(\text{OR}_{K\delta} - 1)}{1 + \sum_{\delta} R'_{K\delta}(\text{OR}_{K\delta} - 1)}$$

Usage pipeline:

```
# 1. Fit mortality OR per class
mort_or <- fit_mortality_rr_logistic(data, ...)

# 2. Assign OR to profiles via max rule (rr_col = "OR_death")
profiles_with_or <- assign_rr_to_profiles(
  profiles_output,
  rr_table = mort_or,
  rr_col   = "OR_death"
)

# 3. Compute per-profile and overall mortality PAF
paf_mort <- daly_calc_paf_mortality(profiles_with_or)
```

Value

Named list (one entry per pathogen) containing:

- `per_profile`: profile data frame augmented with `numerator_mort` ($= R'_{K\delta}(\text{OR}_{K\delta} - 1)$), `PAF_mortality` ($= \text{numerator} / \text{denominator}$), and `denominator_mort` ($= 1 + \sum_{\delta} \text{numerator}$).
- `PAF_k_mort`: overall mortality PAF for pathogen k .
- `denominator_mort`: shared denominator $E[\text{OR}_k]$.

daly_calc_pathogen_fraction_fatal

Calculate fatal pathogen distribution ($P_{[Lk]}$)

Description

Computes the fatal pathogen distribution for a given infectious syndrome (L) using facility-level microbiology data. This quantity represents the fractional contribution of each pathogen (k) to **fatal** infection cases, and is used in YLL / mortality burden estimation.

Usage

```
daly_calc_pathogen_fraction_fatal(
  data,
  syndrome_col,
  syndrome_name,
  specimen_col = NULL,
  specimen_name = NULL,
  polymicrobial_col,
  patient_col,
  pathogen_col,
  outcome_col,
  death_value = "Death",
  glass_ref = NULL,
  facility_col = NULL,
  facility_name = NULL,
  pathogen_name = NULL
)
```

Arguments

data	Data frame of facility-level microbiology records.
syndrome_col	Character. Column containing infectious syndrome labels (L).
syndrome_name	Character. Syndrome to analyse.
specimen_col	Character or NULL. Column containing specimen type. If NULL, no specimen filter is applied.
specimen_name	Character or NULL. Specimen to restrict to (e.g., "Blood"). Required when specimen_col is provided.
polymicrobial_col	Character. Column flagging polymicrobial patients (1 = polymicrobial, 0 = monomicrobial).
patient_col	Character. Unique patient identifier column.
pathogen_col	Character. Pathogen (organism) column (k).
outcome_col	Character. Final patient outcome column.

death_value	Character. Value indicating a fatal outcome. Default "Death".
glass_ref	Character vector of valid pathogen names, or a data frame with columns specimen and pathogen. Applied to polymicrobial patients only. When specimen_name is NULL and glass_ref is a data frame, all pathogens in the reference are used regardless of specimen. NULL skips GLASS filtering.
facility_col	Character or NULL. Facility identifier column. When provided without facility_name, returns both facility-level and pooled P_LK.
facility_name	Character or NULL. Restricts to a single facility.
pathogen_name	Character vector or NULL. Filter to specific pathogen(s).

Details

The unit of analysis is the **patient**. Each patient contributes total weight 1, distributed equally across their valid pathogens. For a patient with m_r valid pathogens, each pathogen receives weight $1/m_r$.

For **polymicrobial patients** (polymicrobial_col == 1), only pathogens listed in the GLASS reference (glass_ref) for the given specimen type are retained before weighting. Monomicrobial patients (polymicrobial_col == 0) are never filtered.

The pooled formula across facilities is:

$$P_{LK}^{\text{pooled}} = \frac{\sum_f N_{f,L,K}^F}{\sum_f N_{f,L}^F}$$

Value

A list:

P_Lk_fatal Pooled P_LK data frame: pathogen_col, N_F_LK, N_F_L, P_Lk_fatal.

facility_level Per-facility P_LK (only when facility_col is supplied and facility_name is NULL).

daly_calc_resistance_prevalence_fatal

Calculate Fatal Resistance Prevalence (R_k)

Description

Computes the profile-weighted expected proportion of deaths attributable to resistance (R_k) and the expected odds ratio of death (E[OR_death]) for each pathogen.

Usage

```
daly_calc_resistance_prevalence_fatal(
  profiles_with_rr,
  probability_col = "probability",
  rr_profile_col = "RR_LOS_profile"
)
```

Arguments

`profiles_with_rr` Named list returned by `daly_assign_rr_to_profiles()` or `daly_filter_profiles_to_rr_classes()`

`probability_col` Character. Profile probability column. Default "probability".

`rr_profile_col` Character. Profile-level OR column. Default "RR_LOS_profile".

Value

Named list (one entry per pathogen) each containing `per_profile` data frame, `R_k`, and `E_OR_k`.

`daly_calc_syndrome_fraction`
Calculate Infectious Syndrome Fraction (M_{LJ})

Description

Computes M_{LJ} , the fraction of infection-related deaths for underlying cause J that are attributed to infectious syndrome L. This distributes infection deaths across clinical syndrome categories (e.g., bloodstream infection, pneumonia, urinary tract infection).

Usage

```
daly_calc_syndrome_fraction(
  pop_data,
  cause_col = "cause_of_death",
  syndrome_col = "syndrome",
  infection_flag_col = "is_infection_death",
  groupby_cols = NULL
)
```

Arguments

`pop_data` Data frame. Population-level mortality data with cause and syndrome columns. Required.

`cause_col` Character. Underlying cause of death column (cause J). Default "cause_of_death".

`syndrome_col` Character. Infectious syndrome column (syndrome L), e.g., "infectious_syndrome". Default "syndrome".

`infection_flag_col` Character. Binary column (TRUE/FALSE or 1/0) indicating infection involvement in `pop_data`. Default "is_infection_death".

`groupby_cols` Character vector. Additional stratification columns. Default NULL.

Details

This function requires population-level data. If only facility data are available, use `daly_calc_deaths_by_syndrome()` with `facility_data` instead.

Value

Data frame with columns: cause_col, syndrome_col, any groupby_cols, infection_deaths_LJ (deaths for syndrome L given cause J), infection_deaths_J (total infection deaths for cause J), M_LJ (syndrome fraction 0-1), M_LJ_method, M_LJ_confidence.

References

Antimicrobial Resistance Collaborators. Global burden of bacterial antimicrobial resistance in 2019. Lancet. 2022.

Examples

```
## Not run:
m_lj <- daly_calc_syndrome_fraction(
  pop_data      = vital_reg,
  cause_col     = "icd10_cause",
  syndrome_col  = "infectious_syndrome",
  infection_flag_col = "is_infectious",
  groupby_cols  = c("Age_bin")
)

## End(Not run)
```

daly_calc_yld_associated

Compute YLD Associated with Resistance (Direct, Profile-Specific)

Description

Primary, direct implementation of the profile-specific associated-YLD equation:

Usage

```
daly_calc_yld_associated(
  profiles_with_los,
  P_Lk_prime_tbl,
  yld_ref = NULL,
  DW_sepsis = NULL,
  pathogen_col = "pathogen",
  plk_col = "P_Lk_prime",
  incidence_col = "N_NF_L",
  facility_col = NULL,
  facility_name = NULL,
  facility_state_map = NULL,
  state_col = "state",
  state_name = NULL,
  probability_col = "probability",
  dominant_class_col = "dominant_class",
```

```

    losr_col = "LOSR_years"
)

```

Arguments

profiles_with_los	Named list (one entry per pathogen) from <code>daly_assign_rr_to_profiles(..., los_r_col = ..., los_s_col = ...)</code> . Each entry must have <code>probability_col</code> , <code>dominant_class_col</code> , and <code>losr_col</code> columns.
P_Lk_prime_tbl	Data frame: the P_Lk_prime (pooled) or facility_level element from <code>daly_calc_pathogen_fraction</code> . Must contain <code>pathogen_col</code> , <code>plk_col</code> , and <code>incidence_col</code> .
yld_ref	Data frame with columns <code>location_name</code> and <code>DW_sepsis</code> . Ignored when <code>DW_sepsis</code> is supplied.
DW_sepsis	Numeric scalar or NULL. Disability weight used directly for every row when supplied (recommended – see "Disability weight caveat").
pathogen_col	Character. Default "pathogen".
plk_col	Character. P' Lk column in P_Lk_prime_tbl. Default "P_Lk_prime".
incidence_col	Character. I_L column in P_Lk_prime_tbl. Default "N_NF_L" (the column already produced by <code>daly_calc_pathogen_fraction_nonfatal()</code>).
facility_col	Character or NULL. Facility identifier column in P_Lk_prime_tbl. Default NULL.
facility_name	Character or NULL. Restrict to a single facility. Default NULL.
facility_state_map	Data frame with <code>facility_col</code> and <code>state_col</code> , mapping each facility to a state. Used only when <code>DW_sepsis</code> is not supplied and <code>facility_col</code> is present.
state_col	Character. Default "state".
state_name	Character or NULL. State for the pooled/no-facility DW lookup. NULL uses the "India" row. Ignored when <code>DW_sepsis</code> is supplied.
probability_col	Character. R' k_delta column in profiles_with_los. Default "probability".
dominant_class_col	Character. d*(delta) column. Default "dominant_class".
losr_col	Character. LOSR_k,d*(delta) column (years). Default "LOSR_years".

Details

$$YLD_assoc_k = I_L * P_Lk * DW_L * \sum_{\{delta \text{ in } D_k\}} [R_k_delta * LOSR_k, d*(delta)]$$

where D_k is the set of RESISTANT resistance profiles for pathogen k (the all-susceptible profile has no dominant class and is excluded – see "Details"). $d*(delta)$ is the dominant drug class of profile $delta$ (GBD max rule, assigned by `daly_assign_rr_to_profiles()`).

This does **not** require computing a baseline/pooled YLD first: unlike the pre-existing multiplicative pathway (a pooled baseline YLD multiplied by an associated-burden fraction), this is the direct equation, evaluated per pathogen (and per facility, if `P_Lk_prime_tbl` is facility-level) from its own inputs.

Component sourcing:

- **I_L** and **P'_Lk**: read directly from `P_Lk_prime_tbl` – the `P_Lk_prime` (pooled) or `facility_level` element already returned by `daly_calc_pathogen_fraction_nonfatal()`, which computes both the non-fatal incidence count and the pathogen fraction from the same observed cohort. Incidence is never back-calculated from deaths, CFR, or LOS.
- **DW_L**: resolved exactly as in the package's pre-existing YLD weight mechanism – either the `DW_sepsis` scalar, or a lookup in `yld_ref` (loaded from `inst/extdata/Proxy_YLD_per_case.csv`) by state/facility. See "Disability weight caveat" below.
- **R'_k_delta**, **d*(delta)**, and **LOSR**: read from `profiles_with_los`, the list returned by `daly_assign_rr_to_profile` when called with `los_r_col/los_s_col` set.

Why D_k excludes the all-susceptible profile: this matches the pre-existing convention elsewhere in this package that "associated" burden is a partition over resistant profiles only (the removed `daly_calc_fraction_associated_yld()`'s `Fraction_k` summed over resistant profiles only; `daly_calc_paf_los()`'s all-susceptible term already contributes exactly zero by construction, since `RR=1` there). `YLD_associated` therefore represents the burden occurring specifically among pathogen-k patients with a resistant profile, not the total burden across all pathogen-k patients.

Disability weight caveat: `Proxy_YLD_per_case.csv`'s `DW_sepsis` column is empirically a duration-inclusive "YLD per case" proxy (`proxy_yld_per_case == yld_days_per_case / 365`), not a pure disability weight. Because this function already multiplies by profile-specific `LOSR_years` separately, sourcing `DW_L` from `yld_ref` risks double-counting duration; a warning is emitted when `DW_sepsis` is not supplied directly. Prefer supplying a pure disability weight via the `DW_sepsis` scalar for this pathway.

Value

`P_Lk_prime_tbl` augmented with `I_L`, `DW_L`, `profile_term` ($= \sum \text{delta } R'_k \text{delta} * \text{LOSR}_k \text{delta}$), and `YLD_associated`.

`daly_calc_yld_attributable`

Compute YLD Attributable to Resistance (Direct, Profile-Specific)

Description

Primary, direct implementation of the profile-specific attributable-YLD equation:

Usage

```
daly_calc_yld_attributable(
  profiles_with_los,
  P_Lk_prime_tbl,
  yld_ref = NULL,
  DW_sepsis = NULL,
  pathogen_col = "pathogen",
  plk_col = "P_Lk_prime",
```

```

    incidence_col = "N_NF_L",
    facility_col = NULL,
    facility_name = NULL,
    facility_state_map = NULL,
    state_col = "state",
    state_name = NULL,
    probability_col = "probability",
    dominant_class_col = "dominant_class",
    losr_col = "LOSR_years",
    loss_col = "LOSS_years"
)

```

Arguments

profiles_with_los	Named list (one entry per pathogen) from daly_assign_rr_to_profiles(..., los_r_col = ..., los_s_col = ...). Each entry must have probability_col, dominant_class_col, and losr_col columns.
P_Lk_prime_tbl	Data frame: the P_Lk_prime (pooled) or facility_level element from daly_calc_pathogen_fraction. Must contain pathogen_col, plk_col, and incidence_col.
yld_ref	Data frame with columns location_name and DW_sepsis. Ignored when DW_sepsis is supplied.
DW_sepsis	Numeric scalar or NULL. Disability weight used directly for every row when supplied (recommended – see "Disability weight caveat").
pathogen_col	Character. Default "pathogen".
plk_col	Character. P'_Lk column in P_Lk_prime_tbl. Default "P_Lk_prime".
incidence_col	Character. I_L column in P_Lk_prime_tbl. Default "N_NF_L" (the column already produced by daly_calc_pathogen_fraction_nonfatal()).
facility_col	Character or NULL. Facility identifier column in P_Lk_prime_tbl. Default NULL.
facility_name	Character or NULL. Restrict to a single facility. Default NULL.
facility_state_map	Data frame with facility_col and state_col, mapping each facility to a state. Used only when DW_sepsis is not supplied and facility_col is present.
state_col	Character. Default "state".
state_name	Character or NULL. State for the pooled/no-facility DW lookup. NULL uses the "India" row. Ignored when DW_sepsis is supplied.
probability_col	Character. R'_k_delta column in profiles_with_los. Default "probability".
dominant_class_col	Character. d*(delta) column. Default "dominant_class".
losr_col	Character. LOSR_k,d*(delta) column (years). Default "LOSR_years".
loss_col	Character. LOSS_k,d*(delta) column (years) in profiles_with_los. Default "LOSS_years".

Details

$$YLD_attrib_k = I_L * P_Lk * DW_L * \sum_{\{delta \text{ in } D_k\}} [R_k_delta * (LOSR_k,d*(delta) - LOSS_k,d*(delta))]$$

The excess (LOSR - LOSS) term makes this a genuine counterfactual: how much MORE disability burden exists because these infections were resistant, versus if they had been susceptible to the same (dominant) class. D_k excludes the all-susceptible profile from the sum entirely (rather than including a zero LOSR - LOSS term for it) – see `daly_calc_yld_associated()` for why.

Obtains I_L , P_Lk , DW_L , R_k_delta , $d*(delta)$, LOSR, and LOSS the same way as `daly_calc_yld_associated()` – see that function’s documentation for full component-by-component sourcing and the disability-weight caveat. Does **not** require a baseline YLD or a PAF_LOS computed first; unlike the pre-existing multiplicative pathway (`daly_calc_paf_los()` multiplied against a pooled baseline YLD), this is the direct equation. `daly_calc_paf_los()` remains available standalone for diagnostics / comparison against this direct calculation.

Value

$P_Lk_prime_tbl$ augmented with I_L , DW_L , $profile_term (= \sum_delta R_k_delta * (LOSR_k_delta - LOSS_k_delta))$, and $YLD_attributable$.

daly_calc_yll_associated

Compute YLL Associated with AMR (Patient-Level, Facility-Direct)

Description

Computes years of life lost (YLL) associated with AMR directly from patient-level facility records, without requiring population-level P_LK (syndrome fractions) or R_kd (resistance profile scalars).

Usage

```
daly_calc_yll_associated(
  data,
  outcome_col,
  death_value = "Death",
  pathogen_col,
  patient_col,
  age_bin_col,
  sex_col,
  facility_col = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL,
  date_culture_col = NULL,
  specimen_col = NULL,
  poly_weight_method = "monomicrobial_proportion",
  min_mono_per_facility = 30L,
  gap_days = 14,
```

```

le_path = system.file("extdata", "life_expectancy_all.xlsx", package = "anumaan"),
male_value = "Male",
female_value = "Female",
age_bin_map = c(` $<1$ ` = "0-1"),
stratify_by = NULL
)

```

Arguments

data	Data frame of patient-level facility records.
outcome_col	Character. Final outcome column.
death_value	Character. Value(s) indicating a fatal outcome. Default "Death". Pass a vector to match multiple labels (e.g. c("Death", "Died")).
pathogen_col	Character. Pathogen / organism column (k).
patient_col	Character. Unique patient identifier column.
age_bin_col	Character. Column containing GBD-standard age bin labels (e.g. "0-1", "1-5", ..., "85+"). Use age_bin_map to recode non-standard labels.
sex_col	Character. Column containing patient sex.
facility_col	Character or NULL. Facility identifier column. When supplied, polymicrobial weights are computed per facility and results include a by_facility breakdown. Default NULL.
syndrome_col	Character or NULL. Syndrome column. When NULL all syndromes are pooled; when supplied the by_syndrome and by_syndrome_pathogen outputs are populated.
syndrome_name	Character or NULL. If supplied, data are filtered to this syndrome before computation. NULL = all.
date_culture_col	Character or NULL. Culture date column used for polymicrobial episode detection. When NULL polymicrobial flagging is skipped and all weights default to 1.
specimen_col	Character or NULL. Specimen type column used alongside date_culture_col for polymicrobial detection.
poly_weight_method	Character. Method for computing polymicrobial death weights. Default "monomicrobial_proportion".
min_mono_per_facility	Integer. Minimum monomicrobial records per facility to use facility-specific weights; smaller facilities fall back to global weights. Default 30L.
gap_days	Integer. Window (days) used for polymicrobial episode detection. Default 14.
le_path	Character. Path to the India life expectancy xlsx file. Defaults to the bundled inst/extdata copy.
male_value	Character. Value in sex_col for males. Default "Male".
female_value	Character. Value in sex_col for females. Default "Female". All other values use the combined LE.

age_bin_map	Named character vector remapping non-standard age bin labels to LE-table labels. Default <code>c("<1" = "0-1")</code> .
stratify_by	Character vector or NULL. Additional columns to aggregate results by. Default NULL.

Details

For every fatal patient with pathogen k (and optionally syndrome L) the individual YLL contribution is:

$$YLL_{r,k} = LE(\text{age_bin}_r, \text{sex}_r) \times w_{r,k}$$

where $w_{r,k}$ is the polymicrobial death weight for patient r and pathogen k ($= 1$ for monomicrobial, $0-1$ for polymicrobial episodes). Total YLL associated:

$$YLL_{\text{associated}} = \sum_{r,k} YLL_{r,k}$$

Polymicrobial weights are computed via `prep_flag_polymicrobial() + prep_compute_poly_weights()`. When `facility_col` is provided the weights are derived per facility (reflecting local organism distributions), with automatic fallback to globally-pooled proportions for facilities whose monomicrobial reference pool is smaller than `min_mono_per_facility`. If `date_culture_col` is NULL polymicrobial flagging is skipped and all weights default to 1.

Value

A named list:

`total` Scalar: total YLL associated across all pathogens and facilities.

`per_pathogen` Data frame: YLL summed per pathogen k , pooled across facilities. Columns: `pathogen_col`, `n_patients`, `YLL_associated_k`.

`by_age_sex` Data frame: YLL by `age_bin_col` x `sex`.

`by_pathogen_age_sex` Data frame: YLL by pathogen x `age_bin_col` x `sex`.

`by_facility` Data frame (only when `facility_col` is supplied): per-facility YLL, one row per facility x pathogen.

`by_syndrome` Data frame (only when `syndrome_col` is supplied and `syndrome_name` is NULL): YLL by syndrome.

`by_syndrome_pathogen` Data frame (only when `syndrome_col` supplied): YLL by syndrome x pathogen.

`stratified` Data frame (only when `stratify_by` is supplied): YLL aggregated by the requested columns.

`patient_data` The death-cohort data frame used for computation, with `polymicrobial_weight`, `life_expectancy`, and `yll_contribution` columns attached.

Examples

```
## Not run:
yll <- daly_calc_yll_associated(
  data          = cohort_df,
  outcome_col   = "final_outcome",
  death_value   = "Death",
  pathogen_col  = "organism_name",
  patient_col   = "PatientInformation_id",
  age_bin_col   = "Age_bin",
  sex_col       = "gender",
  facility_col  = "center_name",
  syndrome_col  = "infectious_syndrome",
  date_culture_col = "culture_date",
  specimen_col  = "sample_type",
  stratify_by   = c("location", "infectious_syndrome")
)
yll$total
yll$per_pathogen
yll$by_age_sex
yll$stratified

## End(Not run)
```

daly_calc_yll_attributable

Compute YLL Attributable to AMR

Description

Takes the per-patient YLL data produced by `daly_calc_yll_associated()` (which already contains `life_expectancy`, `death_weight`, and `yll_contribution`) and multiplies by the mortality PAF from `daly_calc_paf_mortality()` to produce AMR-attributable YLL.

Usage

```
daly_calc_yll_attributable(
  yll_patient_data,
  paf_mort,
  pathogen_col,
  patient_col,
  age_bin_col,
  sex_col = ".sex_norm",
  resistance_profile_col = NULL,
  profile_col = "profile",
  facility_col = NULL,
  syndrome_col = NULL,
  stratify_by = NULL
)
```

Arguments

yll_patient_data	Data frame: the patient_data element from daly_calc_yll_associated(), with life_expectancy, death_weight, and yll_contribution columns.
paf_mort	Named list returned by daly_calc_paf_mortality().
pathogen_col	Character. Pathogen column.
patient_col	Character. Patient identifier column.
age_bin_col	Character. Age bin column.
sex_col	Character. Normalised sex column. Default ".sex_norm".
resistance_profile_col	Character or NULL. Column holding the resistance profile identifier per patient row. When supplied, per-profile PAF is applied instead of a scalar PAF_k. Default NULL.
profile_col	Character. Profile identifier column in the paf_mort per-profile data frames. Default "profile".
facility_col	Character or NULL. Facility identifier column. When supplied results include a by_facility breakdown. Default NULL.
syndrome_col	Character or NULL. Syndrome column.
stratify_by	Character vector or NULL. Additional columns to aggregate results by. Default NULL.

Details**Two PAF modes:**

PAF_k scalar mode (default) When resistance_profile_col is NULL, the overall mortality PAF per pathogen k (PAF_k_mort) from paf_mort is used as a scalar multiplier.

Per-profile mode When resistance_profile_col is supplied, each patient row is matched to its resistance profile delta and the profile-specific PAF_mortality is applied. Patients whose profile does not appear in paf_mort receive NA and a warning is issued.

$$YLL_{i,k}^{\text{attr}} = \text{yll_contribution}_{i,k} \times \text{PAF}_{k(\delta)}$$

Value

A list:

total Scalar: total AMR-attributable YLL.

per_pathogen One row per pathogen: pathogen_col, n_patients, YLL_associated_k, PAF_k_mort, YLL_attributable_k.

by_age_sex YLL attributable stratified by age_bin x sex.

by_pathogen_age_sex YLL attributable by pathogen x age_bin x sex.

by_facility Per-facility breakdown (when facility_col supplied).

by_syndrome Per-syndrome breakdown (when syndrome_col supplied).

by_syndrome_pathogen Per-syndrome x pathogen (when syndrome_col supplied).

stratified User-defined stratification (when stratify_by supplied).

patient_data Patient-level data augmented with PAF_kd and YLL_attributable_contribution.

Examples

```
## Not run:
yll_assoc <- daly_calc_yll_associated(data = cohort, ...)
mort_or <- fit_mortality_rr_logistic(data = rr_data, ...)
profiles_or <- assign_rr_to_profiles(profiles_out,
  rr_table = mort_or,
  rr_col = "OR_death"
)
paf_mort <- daly_calc_paf_mortality(profiles_or)

yll_attr <- daly_calc_yll_attributable(
  yll_patient_data = yll_assoc$patient_data,
  paf_mort          = paf_mort,
  pathogen_col      = "organism_name",
  patient_col       = "PatientInformation_id",
  age_bin_col       = "Age_bin",
  sex_col           = ".sex_norm",
  facility_col      = "center_name",
  syndrome_col      = "infectious_syndrome",
  stratify_by       = c("location", "infectious_syndrome")
)

## End(Not run)
```

daly_compute_patient_los

Compute Patient-Level Post-Infection LOS

Description

Computes LOS with infection-type-specific clock start:

- **CAI:** LOS = date_discharge - date_admission
- **HAI:** LOS = date_discharge - date_culture
- **Unknown / Not known:** LOS = date_discharge - date_culture (if culture date present), else date_discharge - date_admission. If neither reference date is available the patient is excluded.

Only discharged patients are retained. A checkpoint reports how many are missing a discharge date; those patients fall back to los_col when provided, otherwise they are excluded. Rows with LOS <= 0 or LOS > max_los are dropped. Returns one row per patient.

Usage

```
daly_compute_patient_los(
  data,
  patient_id_col = "PatientInformation_id",
  facility_col = "center_name",
  facility_name = NULL,
  organism_col = "organism_name",
  syndrome_col = "syndrome",
  syndrome_name = NULL,
  date_admission_col = "date_of_admission",
  date_discharge_col = "final_outcome_date",
  date_culture_col = "date_of_first_positive_culture",
  final_outcome_col = "final_outcome",
  final_outcome_value = "Discharged",
  infection_type_derived_col = "infection_type_derived",
  los_col = NULL,
  max_los = 200
)
```

Arguments

data	Data frame with an infection_type_derived_col column already populated ("HAI" / "CAI" / other).
patient_id_col	Character. Default "PatientInformation_id".
facility_col	Character. Default "center_name".
facility_name	Character or NULL. If provided, filters data to the specified facility before computing LOS. Default NULL.
organism_col	Character. Column containing organism/pathogen names. Used for episode grouping: a new episode is only created when the culture date gap exceeds 14 days AND the organism differs from the episode's first organism. Default "organism_name".
syndrome_col	Character. Syndrome column name. Only used when syndrome_name is not NULL. Default "syndrome".
syndrome_name	Character or NULL. If provided, only patients with this syndrome are retained before LOS computation. Default NULL.
date_admission_col	Character. Default "date_of_admission".
date_discharge_col	Character. Default "final_outcome_date".
date_culture_col	Character. Default "date_of_first_positive_culture".
final_outcome_col	Character. Default "final_outcome".
final_outcome_value	Character. Default "Discharged".

infection_type_derived_col	Character. Column holding each patient's HAI/CAI classification (e.g. copied from infection_type by the caller). Default "infection_type_derived".
los_col	Character or NULL. Optional name of a pre-computed LOS column (e.g. recorded directly in the data). Used only for patients whose discharge date is missing. Values that are NA, zero, or negative are treated as invalid and those patients are excluded. Default NULL.
max_los	Numeric. Upper cap on LOS in days; patients exceeding this are excluded. Default 200.

Value

Data frame: one row per patient with patient_id_col, facility_col, infection_type_derived_col, LOS_days.

daly_count_incident_cases

Count incident cases by syndrome from facility data

Description

Counts the number of unique patients per infectious syndrome directly from facility-level data. This is the direct-count approach to incidence – no CFR, no CR_L adjustment, no pathogen weighting. Use this when you want raw facility-reported case counts rather than the formula-derived estimate from calculate_incidence_L().

Usage

```
daly_count_incident_cases(
  data,
  syndrome_col,
  syndrome_name,
  patient_col,
  facility_col = NULL,
  facility_name = NULL,
  pathogen_col = NULL,
  pathogen_name = NULL
)
```

Arguments

data	Data frame of facility-level records.
syndrome_col	Character. Column containing infectious syndrome labels.
syndrome_name	Character. Syndrome to count cases for (e.g., "Bloodstream infections").
patient_col	Character. Unique patient identifier column. Cases are counted as distinct patients, not rows.

facility_col	Character or NULL. Facility identifier column. When provided without facility_name, counts are broken down per facility. Default NULL.
facility_name	Character or NULL. If provided, restricts the count to that facility only. Default NULL.
pathogen_col	Character or NULL. Pathogen identifier column. Required when pathogen_name is specified. Default NULL.
pathogen_name	Character or NULL. If provided, restricts the count to the specified pathogen(s). Default NULL.

Details

Results are returned facility-wise when facility_col is supplied and facility_name is NULL. When facility_name is specified, only that facility is returned. When no facility information is provided, a single pooled count across all records is returned.

Value

Data frame with columns: syndrome_col, n_cases (unique patient count), and facility_col if supplied.

daly_derive_hai_cai_for_mortality

Classify HAI/CAI Infection Type for the Mortality Cohort

Description

Derives the healthcare-associated infection (HAI) vs community-acquired infection (CAI) classification for each patient record in the mortality cohort, using admission-to-culture date differences. Also performs a data-quality check for deceased patients missing an outcome date.

Usage

```
daly_derive_hai_cai_for_mortality(
  data,
  infection_type_col = "type_of_infection",
  date_admission_col = "date_of_admission",
  date_culture_col = "date_of_first_positive_culture",
  final_outcome_col = "final_outcome",
  final_outcome_date_col = "final_outcome_date",
  death_value = "Death",
  hai_threshold_hours = 48,
  patient_id_col = "PatientInformation_id"
)
```

Arguments

`data` Data frame of patient-level records.

`infection_type_col` Character. Column containing the raw infection type label. Default "type_of_infection".

`date_admission_col` Character. Admission date column. Default "date_of_admission".

`date_culture_col` Character. First positive culture date column. Default "date_of_first_positive_culture".

`final_outcome_col` Character. Final outcome column. Default "final_outcome".

`final_outcome_date_col` Character. Final outcome date column. Default "final_outcome_date".

`death_value` Character. Value in `final_outcome_col` indicating a fatal outcome. Default "Death".

`hai_threshold_hours` Numeric. Hours after admission at which a positive culture is classified as HAI. Default 48.

`patient_id_col` Character. Patient identifier column. Default "PatientInformation_id".

Value

The input data frame with a derived `infection_type` column (values "HAI" or "CAI").

`daly_filter_profiles_to_rr_classes`

Filter Resistance Profiles to Classes with RR Estimates

Description

Drops profiles whose resistant classes have no RR estimate in `rr_table` and re-normalises the remaining probabilities. The all-susceptible reference profile is always kept.

Usage

```
daly_filter_profiles_to_rr_classes(
  profiles_with_rr,
  rr_table,
  pathogen_col = "pathogen",
  class_col = "antibiotic_class",
  probability_col = "probability",
  fallback_rr = 1
)
```

Arguments

- profiles_with_rr Named list returned by daly_assign_rr_to_profiles().
- rr_table Data frame of RR estimates.
- pathogen_col Character. Default "pathogen".
- class_col Character. Default "antibiotic_class".
- probability_col Character. Default "probability".
- fallback_rr Numeric. Default 1.

Value

Named list (one entry per pathogen) with filtered and re-normalised profiles.

daly_fit_los_rr	<i>Fit relative LOS using Gamma GLM with log link</i>
-----------------	---

Description

Estimates class-specific relative length of stay (RR_LOS) for each pathogen using a Gamma generalized linear model with log link:

Usage

```
daly_fit_los_rr(  
  data,  
  patient_id_col = "PatientInformation_id",  
  facility_col = "center_name",  
  organism_col = "organism_name",  
  syndrome_col = "syndrome",  
  infection_type_col = "infection_type",  
  antibiotic_class_col = "antibiotic_class",  
  antibiotic_name_col = "antibiotic_name",  
  antibiotic_value_col = "antibiotic_value",  
  date_admission_col = "date_of_admission",  
  date_discharge_col = "final_outcome_date",  
  date_culture_col = "date_of_first_positive_culture",  
  final_outcome_col = "final_outcome",  
  final_outcome_value = "Discharged",  
  age_col = "Age",  
  sex_col = "Gender",  
  unit_type_col = NULL,  
  comorbidity_col = NULL,  
  syndrome_name = NULL,  
  organism_name = NULL,
```

```

    hai_threshold_hours = 48,
    max_los = 365,
    min_n = 20L,
    min_resistant = 5L,
    min_susceptible = 5L
  )

```

Arguments

<code>data</code>	Data frame.
<code>patient_id_col</code>	Character. Patient identifier column.
<code>facility_col</code>	Character or NULL. Facility / centre column. When NULL, no hospital effect is included in the model and a single pooled RR is returned (hospital column will be NA). When provided, the model is fit separately per hospital and the result contains one row per pathogen-class-hospital combination (hospital-specific RR).
<code>organism_col</code>	Character. Pathogen column.
<code>syndrome_col</code>	Character. Syndrome column.
<code>infection_type_col</code>	Character. Column already holding the final HAI/CAI classification (e.g. <code>prep_derive_hai_cai()</code> 's output). Used as-is – not re-derived from admission/culture dates. Default "infection_type".
<code>antibiotic_class_col</code>	Character. Antibiotic class column.
<code>antibiotic_name_col</code>	Character. Antibiotic name column.
<code>antibiotic_value_col</code>	Character. Antibiotic susceptibility column.
<code>date_admission_col</code>	Character. Admission date column.
<code>date_discharge_col</code>	Character. Discharge / final outcome date column.
<code>date_culture_col</code>	Character. First positive culture date column.
<code>final_outcome_col</code>	Character. Final outcome column.
<code>final_outcome_value</code>	Character. Value indicating discharged patients.
<code>age_col</code>	Character. Age column.
<code>sex_col</code>	Character. Sex column.
<code>unit_type_col</code>	Character or NULL. ICU / ward location column.
<code>comorbidity_col</code>	Character or NULL. Comorbidity column.
<code>syndrome_name</code>	Character or NULL. Restrict to one syndrome.
<code>organism_name</code>	Character vector or NULL. Restrict to these pathogens.

hai_threshold_hours	Numeric. Unused – retained only for backward API compatibility. HAI/CAI is read directly from infection_type_col rather than re-derived from a date gap.
max_los	Numeric. Maximum retained LOS in days.
min_n	Integer. Minimum patients required in a class model.
min_resistant	Integer. Minimum resistant patients required.
min_susceptible	Integer. Minimum susceptible patients required.

Details

$E(\text{LOS}_i) = \exp(\text{beta0} + \text{beta1} * \text{Resistance_ci} + \text{beta2} * \text{HAI_i} + \text{beta3} * \text{Age_i} + \text{beta4} * \text{Sex_i} + \text{beta5} * \text{ICU_i} + \text{beta6} * \text{Comorbidity_i} + \text{centre fixed effects})$

RR_LOS for a resistance class is $\exp(\text{beta1})$.

Value

Data frame with pathogen, antibiotic_class, hospital (NA when facility_col is NULL), RR_LOS, CI_lower, CI_upper, beta_log_rr, mean_los_resistant, mean_los_susceptible, n_patients, n_resistant, n_susceptible, model_family, link_function, and syndrome_scope.

daly_fit_los_rr_distribution

Estimate Per-Class LOS Relative Risk via Parametric Distribution Fitting

Description

For each pathogen x antibiotic class, splits patients into resistant (R) and susceptible (S) groups, fits a parametric distribution (default: gamma) to each group's LOS, and returns: $\text{RR_LOS} = \text{mean_LOS(R)} / \text{mean_LOS(S)}$

Usage

```
daly_fit_los_rr_distribution(
  data,
  patient_id_col = "PatientInformation_id",
  facility_col = "center_name",
  organism_col = "organism_name",
  syndrome_col = "syndrome",
  infection_type_col = "infection_type",
  antibiotic_class_col = "antibiotic_class",
  antibiotic_name_col = "antibiotic_name",
  antibiotic_value_col = "antibiotic_value",
  date_admission_col = "date_of_admission",
  date_discharge_col = "final_outcome_date",
```

```

date_culture_col = "date_of_first_positive_culture",
final_outcome_col = "final_outcome",
final_outcome_value = "Discharged",
syndrome_name = NULL,
organism_name = NULL,
facility_name = NULL,
hai_threshold_hours = 48,
distributions = "gamma",
max_los = 365,
min_n = 10L
)

```

Arguments

data	Data frame of isolate/patient rows (long format).
patient_id_col	Character. Patient identifier column.
facility_col	Character. Facility/centre column.
organism_col	Character. Organism name column.
syndrome_col	Character. Syndrome column (used only when syndrome_name is not NULL).
infection_type_col	Character. Column already holding the final HAI/CAI classification (e.g. prep_derive_hai_cai()'s output). Used as-is – not re-derived from admission/culture dates. Default "infection_type".
antibiotic_class_col	Character. Antibiotic class column.
antibiotic_name_col	Character. Antibiotic name column.
antibiotic_value_col	Character. Antibiotic value column (S/I/R).
date_admission_col	Character. Date of admission column.
date_discharge_col	Character. Date of discharge column.
date_culture_col	Character. Date of culture column.
final_outcome_col	Character. Final outcome column.
final_outcome_value	Character. Value indicating discharge.
syndrome_name	Character or NULL. Restrict to this syndrome.
organism_name	Character vector or NULL. Restrict to these pathogen(s); otherwise all pathogens in the filtered data.
facility_name	Character or NULL. If provided, filters data to the specified facility before fitting. Default NULL.

hai_threshold_hours	Numeric. Unused – retained only for backward API compatibility. HAI/CAI is read directly from infection_type_col rather than re-derived from a date gap.
distributions	Character vector. Candidate distributions to fit. Default "gamma".
max_los	Numeric. Maximum plausible LOS in days. Default 200.
min_n	Integer. Minimum patients required in both R and S groups. Default 10L.

Details

This is an alternative to `daly_fit_los_rr()` that avoids regression and models the full LOS distribution per group. The output flat data frame is fully compatible with `assign_rr_to_profiles()` and `filter_profiles_to_rr_classes()`.

Value

A data frame with one row per pathogen x class:

pathogen Pathogen name.

antibiotic_class Antibiotic class name.

RR_LOS Fitted mean ratio: $\text{mean_LOS(R)} / \text{mean_LOS(S)}$.

mean_LOS_R Fitted mean LOS for resistant patients.

mean_LOS_S Fitted mean LOS for susceptible patients.

best_dist_R Distribution fitted to the R group.

best_dist_S Distribution fitted to the S group.

n_R Number of resistant patients.

n_S Number of susceptible patients.

syndrome_scope Syndrome filter applied, or "all".

See Also

[daly_fit_los_rr](#), [daly_assign_rr_to_profiles](#)

`daly_fit_mortality_rr` *Fit mortality model and derive adjusted relative risk of death*

Description

Fits a pathogen-class specific mixed-effects logistic regression model:

Usage

```
daly_fit_mortality_rr(
  data,
  patient_id_col = "PatientInformation_id",
  facility_col = "center_name",
  organism_col = "organism_name",
  syndrome_col = "syndrome",
  infection_type_col = "type_of_infection",
  antibiotic_class_col = "antibiotic_class",
  antibiotic_name_col = "antibiotic_name",
  antibiotic_value_col = "antibiotic_value",
  unit_type_col = "unit_type",
  date_admission_col = "date_of_admission",
  date_culture_col = "date_of_first_positive_culture",
  final_outcome_col = "final_outcome",
  final_outcome_date_col = "final_outcome_date",
  age_col = "Age",
  sex_col = "Gender",
  comorbidity_col = NULL,
  death_value = "Death",
  syndrome_name = NULL,
  organism_name = NULL,
  hai_threshold_hours = 48,
  icu_values = c("ICU", "Intensive Care", "Critical Care", "PICU", "NICU"),
  phi_threshold = 0.7,
  min_n = 20L,
  min_deaths = 10L
)
```

Arguments

<code>data</code>	Data frame.
<code>patient_id_col</code>	Character. Patient identifier column.
<code>facility_col</code>	Character or NULL. Facility / centre column. When NULL, no hospital random effect is used and a single pooled RR is returned using standard logistic regression (hospital column will be NA). When provided, a separate logistic model is fit per hospital and the result contains one row per pathogen-class-hospital combination (hospital-specific RR).
<code>organism_col</code>	Character. Pathogen column.
<code>syndrome_col</code>	Character. Syndrome column.
<code>infection_type_col</code>	Character. Raw infection-type column.
<code>antibiotic_class_col</code>	Character. Antibiotic class column.
<code>antibiotic_name_col</code>	Character. Antibiotic name column.

antibiotic_value_col	Character. Antibiotic susceptibility column.
unit_type_col	Character or NULL. ICU / ward location column.
date_admission_col	Character. Admission date column.
date_culture_col	Character. First positive culture date column.
final_outcome_col	Character. Final outcome column.
final_outcome_date_col	Character. Final outcome date column.
age_col	Character. Age column.
sex_col	Character. Sex column.
comorbidity_col	Character or NULL. Comorbidity column.
death_value	Character. Value indicating death.
syndrome_name	Character or NULL. Restrict to one syndrome.
organism_name	Character vector or NULL. Restrict to these pathogens.
hai_threshold_hours	Numeric. HAI derivation threshold.
icu_values	Character vector. Values treated as ICU.
phi_threshold	Numeric. HAI / ICU collinearity warning threshold.
min_n	Integer. Minimum patients per fitted model.
min_deaths	Integer. Minimum deaths per fitted model.

Details

$\text{Death}_i \sim \text{Bernoulli}(\pi_i) \text{ logit}(\pi_i) = \beta_0 + \beta_{\text{kd}} * \text{Resistant}_{\text{id}} + \beta_{\text{age}} * \text{Age}_i + \beta_{\text{sex}} * \text{Sex}_i + \beta_{\text{hai}} * \text{HAI}_i + \beta_{\text{icu}} * \text{ICU}_i + \beta_{\text{comorb}} * \text{Comorbidity}_i + \beta_{\text{syn}} * \text{Syndrome}_i + u_{\text{facility}}$

where u_{facility} is a random intercept when more than one facility is present.

The function returns both: - $\text{OR}_{\text{death}} = \exp(\beta_{\text{kd}})$ - $\text{RR}_{\text{death}} = \text{mean}(p_{\text{resistant_cf}}) / \text{mean}(p_{\text{susceptible_cf}})$

RR_{death} is derived from model-based predicted probabilities under resistant and susceptible counterfactual scenarios, as required by the burden methodology.

Value

Data frame with hospital (NA when `facility_col` is NULL), adjusted OR and adjusted RR of death.

daly_get_top_pathogens

Identify top N pathogens by occurrence

Description

Ranks pathogens by the number of records (rows) in the dataset, optionally filtered to a specific syndrome, specimen, facility, or outcome. Returns the top N pathogens overall or broken down per facility.

Usage

```
daly_get_top_pathogens(
  data,
  pathogen_col,
  n = 5L,
  syndrome_col = NULL,
  syndrome_name = NULL,
  specimen_col = NULL,
  specimen_name = NULL,
  outcome_col = NULL,
  outcome_name = NULL,
  facility_col = NULL,
  facility_name = NULL
)
```

Arguments

data	Data frame of facility-level records.
pathogen_col	Character. Pathogen column.
n	Integer. Number of top pathogens to return. Default 5.
syndrome_col	Character or NULL. Filter to this syndrome column.
syndrome_name	Character or NULL. Syndrome value to filter to.
specimen_col	Character or NULL. Filter to this specimen column.
specimen_name	Character or NULL. Specimen value to filter to.
outcome_col	Character or NULL. Filter to this outcome column.
outcome_name	Character or NULL. Outcome value to filter to (e.g., "Died" or "Discharged").
facility_col	Character or NULL. Facility identifier column. When provided without facility_name, returns top N per facility.
facility_name	Character or NULL. If provided, restricts to that facility only before ranking.

Details

Use this to decide which pathogens to focus on before calling `calculate_P_Lk_prime_BSI()`, `calculate_cfr_lk()`, or `calculate_YLD()`.

Value

Data frame with columns: pathogen_col, n_records, rank (1 = most common), and facility_col if supplied.

daly_load_rr_reference

Load RR (Relative Risk) Reference Data

Description

Loads the rr_list_gbd.xlsx file containing pathogen-drug combinations with relative risk values for burden estimation.

Usage

```
daly_load_rr_reference()
```

Value

Data frame with Pathogen, Drug, and RR columns

Examples

```
## Not run:
rr_data <- daly_load_rr_reference()
head(rr_data)

## End(Not run)
```

default_column_mappings

Default column name mappings for fuzzy matching

Description

Default column name mappings for fuzzy matching

Usage

```
default_column_mappings
```

Format

A named list with elements mapping standard names to character vectors of acceptable aliases.

`detect_preprocessing_capabilities`*Detect Preprocessing Capabilities*

Description

Inspects a data frame (after column mapping) and returns a named logical vector indicating which preprocessing steps can be executed given the columns that are present.

Usage

```
detect_preprocessing_capabilities(data)
```

Arguments

`data` Data frame after column mapping.

Details

This is used by `run_preprocess()` to decide which layers to run and which to skip gracefully.

Value

Named logical vector. TRUE means the step can run; FALSE means required inputs are absent and the step will be skipped with a warning.

`eda_theme`*EDA ggplot2 Theme*

Description

Consistent theme used across all EDA plot functions in this file. Uses `ggpubr::theme_pubr()` when `ggpubr` is installed and falls back to `ggplot2::theme_bw()` otherwise.

Usage

```
eda_theme(base_size = 14, legend_position = "top")
```

Arguments

`base_size` Numeric. Base font size. Default 14.

`legend_position` Character. Legend position ("top", "bottom", "left", "right", "none"). Default "top".

Value

A `ggplot2` theme object.

```
enumerate_binary_profiles
```

Enumerate All Binary Resistance Profiles for a Set of Antibiotic Classes

Description

Generates all 2^n binary resistance profiles for n antibiotic classes using vectorised bit manipulation. Each row is one profile; each class column contains 1 (resistant) or 0 (susceptible).

Usage

```
enumerate_binary_profiles(classes)
```

Arguments

classes	Character vector. Ordered antibiotic class names. The order determines which bit position each class occupies and must be consistent across all calls within a single analysis.
---------	---

Value

Tibble with 2^n rows and $n + 1$ columns: profile_delta (character label, e.g. "RSS") and one integer column per class (0 / 1).

```
estimate_profiles_convex
```

Estimate Resistance Profile Probabilities via Convex Optimisation

Description

The public-facing solver for Pathway 1. Accepts pre-computed marginal and pairwise resistance rates (from `compute_marginals_from_data()` / `compute_pairwise_from_data()`, or from `validate_aggregate_inputs` for GBD-style pre-computed inputs), enumerates all 2^n binary resistance profiles, builds the constraint matrix, and solves the simplex-constrained weighted least-squares QP:

Usage

```
estimate_profiles_convex(
  marginals,
  pairwise = NULL,
  panel_map = NULL,
  exclude_near_zero = TRUE,
  zero_threshold = 0,
  top_n_classes = NULL,
  lambda = 1e-08,
```

```

sigma_sq = 1,
solver = c("osqp", "quadprog"),
n_cores = 1L,
pathogen_col = "pathogen",
class_col = "antibiotic_class",
rate_col = "marginal_resistance",
n_tested_col = "n_tested"
)

```

Arguments

marginals	Tibble. Output of <code>compute_marginals_from_data()</code> or a validated aggregate table. Must have at minimum <code>pathogen_col</code> , <code>class_col</code> , and <code>rate_col</code> .
pairwise	Tibble or NULL. Output of <code>compute_pairwise_from_data()</code> . When NULL, independence $P(A \cap B) = P(A)P(B)$ is assumed for all class pairs. Default NULL.
panel_map	Named list or NULL. If supplied, classes for each pathogen are restricted to the panel. When NULL, all classes present in <code>marginals</code> are used. Default NULL.
exclude_near_zero	Logical. Drop classes with <code>marginal_resistance</code> $\leq$ <code>zero_threshold</code> . Default TRUE.
zero_threshold	Numeric. Classes at or below this value are considered near-zero. Default 0.
top_n_classes	Integer or NULL. Cap classes per pathogen to top N by <code>n_tested</code> . Limits 2^n explosion. Default NULL.
lambda	Numeric. Ridge regularisation added to QP Hessian diagonal. Default $1e-8$.
sigma_sq	Numeric. Uniform constraint variance. Default 1.
solver	Character. "osqp" (default, preferred) or "quadprog" (fallback).
n_cores	Integer. Parallel cores. Default 1L.
pathogen_col	Character. Default "pathogen".
class_col	Character. Default "antibiotic_class".
rate_col	Character. Default "marginal_resistance".
n_tested_col	Character. Default "n_tested".

Details

$$\hat{p} = \arg \min_{p \in \Delta} \|Mp - v\|^2 + \lambda \|p\|^2$$

Stratification columns (geography, year, outcome) present in `marginals` are detected automatically and carried through to the output.

Value

Tibble with one row per (stratum x) pathogen x profile:

Stratum columns Any geography / year / outcome columns from marginals.

pathogen Pathogen name.

profile_set_type "aggregate_convex".

profile_class_set Ordered class names joined by "|".

profile_delta Binary profile label, e.g. "RSS".

profile_probability Estimated probability $\hat{p}_\delta$.

estimator "convex".

convergence_flag Logical; FALSE if QP failed and uniform distribution was returned.

identifiability_flag Logical; TRUE when the constraint matrix is rank-deficient (system is underdetermined).

max_abs_residual Maximum absolute constraint residual.

notes Semicolon-separated notes: capped pairs, fallback pairs.

Examples

```
## Not run:
marg <- compute_marginals_from_data(result$data_wide, result$col_map_resolved, panel_map)
pw <- compute_pairwise_from_data(result$data_wide, marg, result$col_map_resolved, panel_map)
out <- estimate_profiles_convex(marg, pw, panel_map)

## End(Not run)
```

```
estimate_resistance_profiles
```

Estimate Resistance Profiles: Pathway 1 (Convex) or Pathway 2 (Bayesian)

Description

Top-level dispatcher. Runs either the convex optimisation pathway (Pathway 1, aggregate surveillance data) or the Bayesian hierarchical multivariate probit pathway (Pathway 2, facility-level AST data).

Usage

```
estimate_resistance_profiles(
  data,
  method = c("convex", "bayesian"),
  pairwise = NULL,
  panel_map = NULL,
  class_cols = NULL,
```

```

fixed_effects = NULL,
random_effects = list(),
profile_group_col = NULL,
pathogen = NULL,
pathogen_col = "pathogen",
eligible_pairs = NULL,
outcome_col = NULL,
nonfatal_values = c("Discharged", "Survived", "Alive", "discharged", "survived",
  "alive"),
panel_eligibility = list(),
residual_structure = c("identity", "correlated"),
estimand = "observed_stewardship_event_mix",
prior_config = list(),
sampler_config = list(),
n_posterior_draws_for_profiles = 2000L
)

```

Arguments

<code>data</code>	For Pathway 1: marginals tibble from <code>compute_marginals_from_data()</code> . For Pathway 2: wide-format event-level tibble (one row per event; class columns 0/1/NA).
<code>method</code>	Character. "convex" or "bayesian".
<code>pairwise</code>	Tibble or NULL. Pathway 1 only.
<code>panel_map</code>	Named list or NULL. Pathway 1 only.
<code>class_cols</code>	Character vector. Pathway 2 only. Required.
<code>fixed_effects</code>	Character vector. Pathway 2 only. Required.
<code>random_effects</code>	Character vector or named list of random-intercept blocks. Pathway 2 only. Use <code>list()</code> for a fixed-effects-only model.
<code>profile_group_col</code>	Character scalar or NULL. Pathway 2 only. Grouping column for profile aggregation and validation. Required when <code>random_effects</code> is empty.
<code>pathogen</code>	Character or NULL. Pathway 2 only. When supplied, filters data to a single pathogen before fitting.
<code>pathogen_col</code>	Character. Default "pathogen".
<code>eligible_pairs</code>	Tibble or NULL. Pathway 2 only.
<code>outcome_col</code>	Character or NULL. Pathway 2 only. Patient outcome column for <code>R_ALL</code> / <code>R_NF</code> split.
<code>nonfatal_values</code>	Character vector. Outcome values for <code>R_NF</code> cohort.
<code>panel_eligibility</code>	Named list. Eligibility thresholds; passed to <code>fit_bayesian_multivariate_probit()</code> .
<code>residual_structure</code>	Character. Pathway 2. "identity" (default) or "correlated". See <code>fit_bayesian_multivariate_probit()</code> .
<code>estimand</code>	Character. Estimand label. Default "observed_stewardship_event_mix".

`prior_config` Named list. Pathway 2 only. Priors.

`sampler_config` Named list. Pathway 2 only. Structured sampler settings: `chains` (4), `iter_warmup` (1000), `iter_sampling` (1000), `seed` (123), `adapt_delta`, `max_treedepth`, `parallel_chains`.

`n_posterior_draws_for_profiles`
 Integer. Pathway 2. Number of posterior draws used for MVN simulation of profiles. Default 2000L.

Details

For Pathway 2, pass pathogen to fit one pathogen at a time – the recommended workflow. Run this function once per pathogen and collect the results in the analysis repository.

Value

Named list: `profiles`, `eligibility`, `diagnostics`, `fitted_models`, `config_used`.

<code>export_report</code>	<i>Export Report to File</i>
----------------------------	------------------------------

Description

Saves preprocessing report to HTML, PDF, TXT, or RDS format.

Usage

```
export_report(report, file)
```

Arguments

<code>report</code>	An <code>amr_preprocessing_report</code> object
<code>file</code>	Path to output file

Value

Invisibly returns file path

fit_bayesian_multivariate_probit

Fit Bayesian Hierarchical Multivariate Probit Model for Resistance Profiles

Description

Accepts wide-format event-level AST data (one row per organism-event, one column per antibiotic class with 0 / 1 / NA values) and fits a hierarchical Bayesian multivariate probit model via **cmdstanr**.

Usage

```
fit_bayesian_multivariate_probit(
  event_class_data,
  class_cols,
  fixed_effects,
  random_effects = list(),
  profile_group_col = NULL,
  pathogen = NULL,
  pathogen_col = "pathogen",
  event_id_col = "event_id",
  eligible_pairs = NULL,
  outcome_col = NULL,
  reserve_drug_cols = NULL,
  panel_eligibility = list(),
  residual_structure = c("identity", "correlated"),
  estimand = "observed_stewardship_event_mix",
  prior_config = list(),
  sampler_config = list(),
  compute = list(),
  show_messages = TRUE,
  save_full_latent_diagnostics = FALSE,
  ...
)
```

Arguments

event_class_data	Data frame. One row per organism-event. Antibiotic class columns hold 0 (susceptible), 1 (resistant), or NA (not tested). Metadata columns hold covariates and grouping variables.
class_cols	Character vector. Names of the antibiotic class columns. Required.
fixed_effects	Character vector. Event-level covariate column names. Required.
random_effects	Character vector or named list of random-intercept blocks. Use <code>list()</code> for no random-effect blocks. When non-empty, the legacy character-vector form names grouping columns; the list form uses <code>name</code> , <code>group_col</code> , and optional <code>terms = "intercept"</code> .

profile_group_col	Character scalar or NULL. Column used for downstream profile aggregation, eligibility summaries, and validation. It is independent of the random-effect specification. Defaults to the first random-effect grouping column when random effects exist; it is required for fixed-effects-only fits.
pathogen	Character or NULL. When supplied, filters event_class_data to rows where pathogen_col equals this value before fitting. Recommended: fit one pathogen at a time.
pathogen_col	Character. Column identifying the pathogen. Default "pathogen".
event_id_col	Character. Column in event_class_data holding unique event identifiers. Default "event_id".
eligible_pairs	Tibble or NULL. Hospital x pathogen pairs to include. NULL uses all pairs present in the data.
outcome_col	Character or NULL. Patient outcome column. Only used downstream to split R_ALL and R_NF cohorts – does not enter the probit likelihood. Default NULL.
reserve_drug_cols	Character vector or NULL. Class columns to exclude from the main model.
panel_eligibility	Named list. Eligibility thresholds per hospital x class cell: min_tested (default 30), min_resistant (default 5), min_susceptible (default 5). Cells not meeting thresholds are reported but fitting still proceeds.
residual_structure	Character. Controls the residual covariance structure. "identity" (default): classes are conditionally independent given fixed and random effects – residual covariance = I_D. "correlated": estimates a full residual correlation matrix Ω via LKJCholesky prior. Use "correlated" only when panel co-testing overlap is adequate (check \$eligibility_report\$pairwise); otherwise Ω is driven mainly by the LKJ prior and adds identifiability risk.
estimand	Character. Identifies the target quantity. Only "observed_stewardship_event_mix" is currently supported.
prior_config	Named list. Any subset of beta_sd (default 1.5), tau_sd (default 1.0), legacy lkj_eta (default 2.0), or separate lkj_eta_residual and lkj_eta_random_effect. Legacy lkj_eta resolves to both values for exact backward compatibility.
sampler_config	Named list. Sampler settings forwarded to cmdstanr::sample(): chains (4), iter_warmup (1000), iter_sampling (1000), adapt_delta (NULL, uses Stan default), max_treedepth (NULL), seed (123), parallel_chains (NULL), max_param_count (NULL – set to a positive integer to stop if approximate parameter count exceeds the threshold), save_warmup (NULL, i.e. cmdstanr's own default of FALSE – set TRUE to retain warmup-phase draws in the raw CmdStan CSV for later adaptation-trajectory forensics; note this recovers per-iteration diagnostics and parameter draws during warmup, NOT the full evolving HMC mass matrix at every adaptation window – only the final adapted metric is ever written to the CSV, regardless of this setting), metric (NULL, i.e. cmdstanr's default "diag_e" – set "dense_e" to adapt a full mass matrix instead of a diagonal one; substantially more expensive to adapt at high parameter counts, e.g. models with a large z_free block), init (NULL, i.e. CmdStan's default Uniform(-2,2) unconstrained init – set to a value accepted by cmdstanr::sample()'s own

	init argument, e.g. a list of per-chain initial-value lists, to run an initialization-sensitivity check). Any additional entries are forwarded via . . .
compute	Named list controlling the Stan execution backend. Supported fields are backend ("cpu" default or "opencl"), opencl_platform_id, opencl_device_id, and allow_cpu_fallback (FALSE by default). OpenCL changes the compilation/sampling backend only; it does not change the statistical model, priors, diagnostics, or downstream estimands.
show_messages	Logical. Print sampling progress. Default TRUE.
save_full_latent_diagnostics	Logical. When FALSE (default), diagnostics_detail\$all_parameters (per-parameter Rhat/ESS including the $N_{\text{events}} \times D$ latent z_{free} nuisance parameters – tens of thousands of rows for realistic data) is returned with 0 rows to keep the fitted object small; the full-scope summary scalars (max_rhat_full, min_ess_bulk_full, min_ess_tail_full, n_params_full, pct_rhat_full_above_1_01, pct_ess_bulk_full_below_100) are always populated in diagnostics regardless. Set TRUE to retain the full per-parameter table for debugging.
. . .	Additional arguments forwarded to <code>cmdstanr::sample()</code> .

Details

Model (per pathogen):

$$Y_{ed} = \mathbf{1}(Z_{ed} > 0)$$

$$Z_{ed} = \mathbf{x}_e^\top \beta_d + \text{hospital_effect}_{d,h(e)} [+ \text{patient_effect}_{d,p(e)}] [+ \text{admission_effect}_{d,a(e)}] + (L_\Omega \eta_e)_d$$

$$\eta_e \sim N(0, I_D), \quad L_\Omega \sim \text{LKJCholesky}(\eta)$$

The per-event latent noise η_e makes L_Ω (and therefore $\Omega = L_\Omega L_\Omega^\top$) identifiable from the observed binary outcomes. Hospital (and lower-level) effects use a $\text{diag}(\tau) L_{\text{corr}} z_{\text{raw}}$ parameterisation so the D -dim random effect vectors are correlated across antibiotic classes.

Single-pathogen design: pass pathogen to restrict the fit. Run once per pathogen and orchestrate in the analysis repository.

Random effects (`random_effects`): an optional character vector or list of grouping blocks defining the clustering hierarchy. Use `list()` for a fixed-effects-only model. When blocks are present, the first element is the upper-most level; subsequent elements are nested within it. Nested levels receive globally unique composite keys built internally. Any hierarchical grouping variable can occupy any slot – the labels hospital/patient/admission are semantic conventions, not constraints. Isolate- or sample-event-level effects can be passed as the second or third element, but note the returned object uses generic level names (`upper_re_col`, `middle_re_col`, `lower_re_col`).

Missing-AST handling: Observed AST cells impose sign constraints on the latent variables; untested cells impose no sign constraint and are represented as unconstrained latent values. The model assumes testing is conditionally ignorable given included covariates and random effects. It does NOT correct for selective testing bias. See `residual_structure` for control over residual covariance complexity.

Fixed-effect missingness: the function warns but does NOT silently impute covariates. Imputation is the caller's responsibility; columns with any remaining NA values after the call will cause Stan to fail.

Value

Named list with elements: draws, diagnostics, diagnostics_detail, fit, data_long, index_maps, X_event, event_re_idx, class_cols, event_metadata, n_re_levels, upper_re_col, middle_re_col, lower_re_col, patient_key_col, admission_key_col, pathogen_col, pathogen_fitted, residual_structure, estimand, prior_config_used, sampler_config_used, compute_config_used, and eligibility_report.

diagnostics is a one-row monitored summary. The main diagnostic fields max_rhat, min_ess_bulk, and min_ess_tail are computed over the monitored parameters retained in draws. These exclude the latent z_free nuisance parameters used for probit data augmentation. converged_structural/converged_full incorporate tree-depth saturation as well as R-hat/ESS/divergences/E-BFMI (a run that repeatedly saturates max_treedepth is not reported as converged). diagnostic_status is a canonical multi-level status – one of "pass", "warning_rhat", "warning_low_bulk_ess", "warning_low_tail_ess", "warning_treedepth", "fail_rhat", "fail_energy", or "fail_divergent" (most severe first) – intended as the single authoritative status field for callers, replacing ad hoc re-derivations downstream.

Full-fit diagnostic fields, including max_rhat_all, min_ess_bulk_all, and min_ess_tail_all, are reported separately for visibility and include all Stan parameters, including z_free.

diagnostics_detail contains monitored_parameters, all_parameters (per-parameter Rhat/ESS with parameter_group), grouped (per-parameter_group Rhat/ESS quantiles and percentage-over-threshold, not only the single worst parameter), and chains (per-chain n_sampling, n_divergent, n_treedepth_sat, ebfmi, mean_accept_stat, mean_stepsize, mean_treedepth, max_treedepth) – the canonical diagnostic tables; callers should not recompute these independently from fit\$fit\$sampler_diagnostics()

fit_distributions	<i>Fit Multiple Distributions</i>
-------------------	-----------------------------------

Description

Fits Weibull, Lognormal, and Gamma distributions to a numeric vector, returning all fit objects. Failed fits are NULL.

Usage

```
fit_distributions(x)
```

Arguments

x Numeric vector of positive values (e.g., LOS in days).

Value

Named list with elements weibull, lnorm, gamma, each a fitdist object or NULL.

Examples

```
fits <- fit_distributions(rlnorm(200, 2, 0.5))
fits$lnorm$estimate
```

```
generate_preprocessing_report
```

Generate Preprocessing Report

Description

Creates a comprehensive report documenting all transformations applied during preprocessing, including column mappings, data quality metrics, and transformation summaries.

Usage

```
generate_preprocessing_report(  
  raw_data,  
  processed_data,  
  preprocessing_log,  
  config,  
  output_file = NULL,  
  include_plots = TRUE  
)
```

Arguments

<code>raw_data</code>	Original input data frame (before preprocessing)
<code>processed_data</code>	Final output data frame (after preprocessing)
<code>preprocessing_log</code>	List containing logs from each preprocessing step
<code>config</code>	AMR configuration object used
<code>output_file</code>	Optional. Path to save report. If NULL, returns list. Supports .html, .pdf, .txt, .rds formats.
<code>include_plots</code>	Logical. Include data quality plots. Default TRUE.

Value

If `output_file` is NULL, returns a list with report components. Otherwise, saves report to file and returns file path.

Examples

```
## Not run:  
# Generate report  
report <- generate_preprocessing_report(  
  raw_data = my_raw_data,  
  processed_data = result$data,  
  preprocessing_log = result$log,  
  config = result$config,  
  output_file = "preprocessing_report.html"  
)
```

```
## End(Not run)
```

get_age_bins	<i>Get Age Bin Labels</i>
--------------	---------------------------

Description

Returns predefined age bin label sets for use with prep_assign_age_bins().

Usage

```
get_age_bins(type = "GBD_standard")
```

Arguments

type	Character. One of "GBD_standard" (5-year bins), "pediatric", "geriatric", or "neonatal". The "neonatal" preset uses fractional-year labels corresponding to: <0.02 (0-7 days), 0.02-0.08 (7-28 days), 0.08-0.25 (28-90 days), 0.25-1 (3 months-1 year), then standard pediatric bands through adulthood.
------	--

Value

Character vector of age bin labels.

get_los_by_resistance	<i>Extract LOS Vectors by Resistance Status</i>
-----------------------	---

Description

For a given organism (and optionally center), computes per-patient resistance status and returns separate LOS vectors for resistant and susceptible exposures.

Usage

```
get_los_by_resistance(  
  abx_data,  
  los_data,  
  organism,  
  center = NULL,  
  patient_id_col = "PatientInformation_id",  
  organism_col = "organism_clean",  
  center_col = "center_name",  
  antibiotic_col = "antibiotic_name",  
  sus_col = "sus"  
)
```

Arguments

abx_data	Data frame of antibiotic susceptibility results with columns for patient ID, organism, antibiotic name, and susceptibility value (already cleaned to "R"/"S").
los_data	Data frame from prepare_los_data().
organism	Character. Organism name (lowercase, trimmed) to filter on.
center	Optional character. Center name to filter on. Default NULL (all centers).
patient_id_col	Character. Patient ID column. Default "PatientInformation_id".
organism_col	Character. Organism column. Default "organism_clean".
center_col	Character. Center column. Default "center_name".
antibiotic_col	Character. Antibiotic name column. Default "antibiotic_name".
sus_col	Character. Susceptibility column (values "R"/"S"). Default "sus".

Value

List with elements R (LOS vector for resistant exposures) and S (LOS vector for susceptible exposures).

hospital_display_label

Human-readable hospital/site display label

Description

Replaces underscores with spaces only – does not alter the underlying ‘center_name’/hospital identifier used anywhere else (joins, grouping, file paths all keep the raw value; this is a plotting-display transform only).

Usage

```
hospital_display_label(x)
```

Arguments

x	Character vector of raw hospital/site names (e.g. "AIIMS_trauma_center").
---	---

Value

Character vector, e.g. "AIIMS trauma center".

infer_patient_syndrome_wide

Assign Syndrome from Wide-Format Syndrome Flags

Description

Converts a wide-format data frame (one column per syndrome, values indicating presence) to long format and assigns the highest-priority syndrome per patient using the infectious syndrome hierarchy.

Usage

```
infer_patient_syndrome_wide(
  data,
  patient_col = "patient_id",
  syndrome_cols = NULL,
  positive_values = c(1, "1", TRUE, "TRUE", "True", "true", "Yes", "YES", "yes"),
  collapse_unspecified_respiratory = TRUE,
  keep_only_burden_syndromes = FALSE
)
```

Arguments

data	Data frame in wide format where syndrome columns contain presence indicators (1/TRUE/"Yes" etc.).
patient_col	Character. Patient ID column. Default "patient_id".
syndrome_cols	Character vector. Columns to treat as syndrome flags. If NULL, all columns except patient_col are used. Default NULL.
positive_values	Character/logical/numeric vector. Values treated as "syndrome present". Default covers common representations of TRUE/1/Yes.
collapse_unspecified_respiratory	Logical. Passed to prep_assign_patient_syndrome(). Default TRUE.
keep_only_burden_syndromes	Logical. Passed to prep_assign_patient_syndrome(). Default FALSE.

Details

This is a convenience wrapper around prep_assign_patient_syndrome(). All syndrome-selection logic and parameters (hierarchy, burden filter, respiratory collapsing) are handled there.

Value

Data frame with one row per patient and the selected syndrome.

mask_and_validate_ast *Masked-AST Holdout Validation*

Description

Masks a reproducible, stratified subset of genuinely **observed** AST cells, predicts those cells' resistance probability from a fitted model, and scores the predictions against their true values.

Usage

```
mask_and_validate_ast(
  fitted_model,
  fixed_effects = NULL,
  event_id_col = "event_id",
  outcome_col = NULL,
  fraction_to_mask = 0.05,
  max_mask_per_cell = NULL,
  seed = 123L,
  refit = FALSE,
  n_posterior_draws_for_validation = 2000L,
  min_tested_after_mask = 30L,
  min_resistant_after_mask = 5L,
  min_susceptible_after_mask = 5L,
  panel_eligibility = NULL,
  prior_config = NULL,
  sampler_config = NULL,
  n_gibbs_burnin = 10L,
  n_gibbs_kept = 20L
)
```

Arguments

- | | |
|-------------------|---|
| fitted_model | List returned by fit_bayesian_multivariate_probit(). |
| fixed_effects | Character vector. Required when refit = TRUE (not recoverable from fitted_model due to dummy-coding of categorical covariates); must match the original fitting call. |
| event_id_col | Character. Column in fitted_model\$event_metadata holding the original event id; only used when refit = TRUE. Default "event_id". |
| outcome_col | Character or NULL. Only used when refit = TRUE. |
| fraction_to_mask | Numeric in (0, 1]. Target fraction of observed cells to mask per hospital x pathogen x class cell, before the eligibility floor and max_mask_per_cell caps are applied. Default 0.05. |
| max_mask_per_cell | Integer or NULL. Optional additional cap on cells masked per hospital x pathogen x class cell. |

seed Integer. Random seed for mask selection (and posterior draw subsampling). Default 123L.
refit Logical. See modes above. Default FALSE.
n_posterior_draws_for_validation Integer. Default 2000L.
min_tested_after_mask, **min_resistant_after_mask,**
min_susceptible_after_mask Integer. Panel-eligibility floors enforced during masking. Default 30L, 5L, 5L.
panel_eligibility Named list or NULL. When NULL, a masked refit inherits `fitted_model$panel_eligibility_used`; a non-NULL value explicitly overrides those resolved thresholds.
prior_config, sampler_config Named list or NULL. Forwarded when `refit = TRUE`; default to the original fit's `prior_config_used/sampler_config_used`.
n_gibbs_burnin, n_gibbs_kept Integer. Gibbs burn-in/kept-iteration counts forwarded to `.gibbs_conditional_profile_probs()` for correlated-residual fits' masked-cell prediction. Ignored for identity-residual fits. Default 10L/20L.

Details

Two validation modes:

refit = FALSE (default) Predicts masked cells using the *already-fitted* `fitted_model`. Because μ_{ed} never depends on class d 's own observed value (only on event-level covariates and random-effect memberships), this is honestly an **in-sample conditional-prediction diagnostic** (`validation_mode = "in_sample_no_refit"`), not a genuine held-out test – the masked cells still contributed to estimating beta/hospital_effect etc. It is useful for spotting miscalibrated subgroups quickly, but is optimistic relative to a real holdout.

refit = TRUE (preferred, more expensive) Sets the selected cells to NA in a copy of `fitted_model$event_metadata`, re-runs `fit_bayesian_multivariate_probit()` on the masked data, and predicts the held-out cells from that refit (`validation_mode = "refit_masked"`). This is a genuine holdout and is the recommended final procedure once computationally affordable.

Masking procedure: reproducible via `seed`; stratified by hospital x pathogen x class x observed R/S state; for every hospital x pathogen x class cell, at most enough cells are masked to keep `n_tested/n_resistant/n_susceptible` at or above `min_tested_after_mask/min_resistant_after_mask/min_susceptible_after_mask` (panel eligibility is not destroyed); only observed (non-NA) cells are ever candidates; and no event is ever left with zero observed classes across *all* `class_cols` (which would otherwise silently drop that event from a refit).

Prediction: for `fitted_model$residual_structure == "identity"`, $\Phi(\mu_{ed})$ is the exact model-implied probability for a masked cell regardless of what else is known about that event (classes are conditionally independent given μ). For "correlated", using $\Phi(\mu_{ed})$ alone would ignore every OTHER still-observed class for that event – wrong whenever classes are correlated. Instead each masked event's full `class_cols` panel is rebuilt with the masked cell(s) set to NA and every other class at its true observed value, and `.gibbs_conditional_profile_probs()` is used to get $P(\text{masked class} = R \mid \text{that event's other observed classes}, \theta)$ – the same conditional-imputation machinery used for DALY profile generation, not a marginal prediction.

Value

Named list: predictions (one row per masked cell: hospital, pathogen, event id, class, true value, predicted probability, log score, Brier score, predicted class at 0.5, calibration group, validation mode), summary (one-row aggregate: counts, mean log/Brier score, accuracy, AUROC \[only when both classes are present in the masked set\], masking configuration), and refit_model (the refit fitted_model, or NULL when refit = FALSE).

plot_abx_complex_heatmap

Plot Antibiotic Resistance ComplexHeatmap

Description

Produces a clustered ComplexHeatmap::Heatmap() of the proportion of patients Resistant, with **centres on rows** and **antibiotics on columns**. Unlike [plot_abx_heatmap](#) (a ggplot2 tile heatmap with a fixed row/column order), this version hierarchically clusters both rows and columns by default, which groups centres with similar resistance profiles and antibiotics with similar resistance patterns together.

Usage

```
plot_abx_complex_heatmap(
  data,
  patient_col = "PatientInformation_id",
  antibiotic_col = "antibiotic_name",
  value_col = "antibiotic_value",
  organism_col = "organism_name",
  center_col = "center_name",
  low_colour = "#1a9850",
  mid_colour = "#fee08b",
  high_colour = "#d73027",
  midpoint = 0.5,
  cluster_rows = TRUE,
  cluster_columns = TRUE,
  legend_name = "Resistance",
  row_title = "Centres",
  column_title = "Antibiotics",
  base_size = 14,
  column_names_max_height = grid::unit(10, "cm"),
  save_path = NULL,
  width = 15,
  height = 8,
  dpi = 300
)
```

Arguments

<code>data</code>	Data frame. Long-format AMR dataset.
<code>patient_col</code>	Character. Patient ID column. Default "PatientInformation_id".
<code>antibiotic_col</code>	Character. Antibiotic name column. Default "antibiotic_name".
<code>value_col</code>	Character. Susceptibility result column (R/S values). Default "antibiotic_value".
<code>organism_col</code>	Character. Organism name column. Default "organism_name".
<code>center_col</code>	Character. Centre/facility column. Default "center_name".
<code>low_colour</code>	Character. Colour for proportion = 0 (fully susceptible). Default "#1a9850" (green).
<code>mid_colour</code>	Character. Colour for the midpoint. Default "#fee08b" (yellow).
<code>high_colour</code>	Character. Colour for proportion = 1 (fully resistant). Default "#d73027" (red).
<code>midpoint</code>	Numeric. Midpoint of the colour gradient (0-1). Default 0.5.
<code>cluster_rows</code>	Logical. Hierarchically cluster centres. Default TRUE.
<code>cluster_columns</code>	Logical. Hierarchically cluster antibiotics. Default TRUE.
<code>legend_name</code>	Character. Heatmap legend title. Default "Resistance".
<code>row_title</code>	Character. Row-axis title. Default "Centres".
<code>column_title</code>	Character. Column-axis title. Default "Antibiotics".
<code>base_size</code>	Numeric. Base font size for titles/labels. Default 14.
<code>column_names_max_height</code>	<code>grid::unit</code> . Vertical space reserved for the (45-degree-rotated) column-name labels below the heatmap body. Increase this (and/or height) if long labels – e.g. antibiotic class names – are being clipped. Default <code>grid::unit(10, "cm")</code> .
<code>save_path</code>	Character or NULL. If supplied, the heatmap is drawn to a PNG at this path (via <code>grDevices::png()</code> + <code>ComplexHeatmap::draw()</code>) in addition to being returned. Default NULL (not saved).
<code>width</code>	Numeric. PNG width in inches. Used only when <code>save_path</code> is supplied. Default 15.
<code>height</code>	Numeric. PNG height in inches. Used only when <code>save_path</code> is supplied. Default 8.
<code>dpi</code>	Numeric. PNG resolution. Used only when <code>save_path</code> is supplied. Default 300.

Details

Only R and S results are used. The same **worst-phenotype rule** as `plot_abx_heatmap()` is applied: per patient x organism x antibiotic, any single R marks the episode as R. Centre/antibiotic combinations with no tests are shown as 0 (fully susceptible) in the underlying matrix – there is no "no data" grey cell in this version, since `Heatmap()` requires a complete numeric matrix.

Requires the Bioconductor package `ComplexHeatmap` and the CRAN package `circlize`, neither of which is a hard dependency of `anumaan`. Install with: `if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager"); BiocManager::install("ComplexHeatmap")` and `install.packages("circlize")`.

Value

A ComplexHeatmap::Heatmap object (draw with ComplexHeatmap::draw(ht) if not saving to file).

plot_abx_heatmap

Plot Antibiotic Resistance Heatmap

Description

Produces a ggplot2 tile heatmap where:

- **x-axis** – antibiotics (ordered by total patients tested, most tested on the left)
- **y-axis** – centres
- **fill colour** – proportion of patients who were Resistant for that antibiotic at that centre (0 = fully susceptible, 1 = fully resistant)

Usage

```
plot_abx_heatmap(
  data,
  n = NULL,
  mode = c("all", "single"),
  center = NULL,
  patient_col = "PatientInformation_id",
  antibiotic_col = "antibiotic_name",
  value_col = "antibiotic_value",
  organism_col = "organism_name",
  center_col = "center_name",
  show_values = TRUE,
  midpoint = 0.5,
  low_colour = "#1a9850",
  mid_colour = "#fee08b",
  high_colour = "#d73027",
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

data	Data frame. Long-format AMR dataset.
n	Integer or NULL. If an integer, only the top <i>n</i> antibiotics by total patients tested are shown. If NULL, all antibiotics are shown. Default NULL.
mode	Character. "all" or "single". Default "all".

center	Character. Required when mode = "single". Exact centre name.
patient_col	Character. Patient ID column. Default "PatientInformation_id".
antibiotic_col	Character. Antibiotic name column. Default "antibiotic_name".
value_col	Character. Susceptibility result column (R/S values). Default "antibiotic_value".
organism_col	Character. Organism name column. Default "organism_name".
center_col	Character. Centre/facility column. Default "center_name".
show_values	Logical. Print proportion values inside tiles. Default TRUE.
midpoint	Numeric. Midpoint of the colour gradient (0-1). Default 0.5.
low_colour	Character. Colour for proportion = 0 (fully susceptible). Default "#1a9850" (green).
mid_colour	Character. Colour for the midpoint. Default "#fee08b" (yellow).
high_colour	Character. Colour for proportion = 1 (fully resistant). Default "#d73027" (red).
base_size	Numeric. Base font size. Default 14.
title	Character. Custom plot title. Auto-generated if NULL. Default NULL.
syndrome_col	Character or NULL. Column name containing syndrome/infection category labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
syndrome_name	Character or NULL. The syndrome value to retain (e.g. "BSI", "VAP"). Requires syndrome_col to be set. If NULL (default), all syndromes are included.

Details

Only R and S results are used. The same **worst-phenotype rule** as plot_abx_susceptibility() is applied: per patient x organism x antibiotic, any single R marks the episode as R.

Cells with no data (antibiotic not tested at that centre) are shown in light grey.

Supports two display modes:

- **"all"** (default) – all centres on the y-axis.
- **"single"** – filter to one centre before plotting (produces a single-row heatmap useful for per-centre reports).

Value

A ggplot object.

plot_abx_susceptibility

Plot Antibiotic Susceptibility Pattern (Stacked R/S Bars)

Description

Produces a horizontal stacked bar chart showing the number of admissions tested as Resistant (R) or Susceptible (S) for the top n antibiotics. Percentage labels are shown inside each bar segment.

Usage

```
plot_abx_susceptibility(
  data,
  n = 5,
  mode = c("faceted", "overall", "single"),
  center = NULL,
  patient_col = "PatientInformation_id",
  antibiotic_col = "antibiotic_name",
  value_col = "antibiotic_value",
  organism_col = "organism_name",
  center_col = "center_name",
  colours = c(R = "#D73027", S = "#1A9850"),
  ncol = 2,
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

data	Data frame. Long-format AMR dataset.
n	Integer. Number of top antibiotics to show (ranked by total patients tested). Default 5.
mode	Character. One of "faceted", "overall", or "single". Default "faceted".
center	Character. Required when mode = "single". Exact name of the centre to plot.
patient_col	Character. Patient ID column. Default "PatientInformation_id".
antibiotic_col	Character. Antibiotic name column. Default "antibiotic_name".
value_col	Character. Susceptibility result column containing "R" and "S" values. Default "antibiotic_value".
organism_col	Character. Organism name column (used in worst-phenotype deduplication). Default "organism_name".
center_col	Character. Centre/facility column. Default "center_name".

colours	Named character vector. Fill colours for R and S. Default <code>c("R" = "#D73027", "S" = "#1A9850")</code> .
ncol	Integer. Facet columns (mode = "faceted" only). Default 2.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom plot title. Auto-generated if NULL. Default NULL.
syndrome_col	Character or NULL. Column name containing syndrome/infection category labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
syndrome_name	Character or NULL. The syndrome value to retain (e.g. "BSI", "VAP"). Requires syndrome_col to be set. If NULL (default), all syndromes are included.

Details

Worst-phenotype rule: when a patient has multiple records for the same organism-antibiotic combination, any single R result marks the whole episode as R. This prevents double-counting repeat cultures from the same infection episode.

Counting unit: bars show unique patients (not isolate rows), so a patient tested against the same antibiotic twice is counted once.

Supports three display modes:

- **"faceted"** – top *n* antibiotics per centre independently, one panel per centre.
- **"overall"** – all centres pooled; top *n* antibiotics globally.
- **"single"** – one specific centre; pass the centre name via center.

Value

A ggplot object.

plot_age_ridge	<i>Ridge / Density Plot of Patient Age</i>
----------------	--

Description

Visualises the distribution of patient age on the x-axis using either a stacked ridge plot (mode = "all") or a single-centre density curve (mode = "single").

Usage

```
plot_age_ridge(
  data,
  mode = c("all", "single"),
  center = NULL,
  patient_col = "PatientInformation_id",
  age_col = "age_years",
  center_col = "center_name",
```

```

    min_age = 0,
    max_age = 120,
    colours = NULL,
    scale = 1.2,
    alpha = 0.7,
    base_size = 14,
    title = NULL,
    syndrome_col = NULL,
    syndrome_name = NULL
  )

```

Arguments

data	A data frame.
mode	Character. One of "all" (ridge plot, all centres stacked on y-axis ordered by median age ascending) or "single" (density curve for one centre specified by center).
center	Character. Centre name required when mode = "single".
patient_col	Character. Column of patient identifiers.
age_col	Character. Column of patient age (numeric, in years). Default "age_years".
center_col	Character. Column of centre identifiers.
min_age	Numeric. Minimum age to include. Default 0.
max_age	Numeric. Maximum age to include. Default 120.
colours	Named character vector of colours for centres. If NULL (default), the RColorBrewer "Set2" palette is used.
scale	Numeric. scale argument passed to <code>ggridges::geom_density_ridges()</code> . Controls overlap between ridges. Default 1.2.
alpha	Numeric. Fill transparency (0-1). Default 0.7.
base_size	Numeric. Base font size. Default 14.
title	Character or NULL. Plot title. Auto-generated if NULL.
syndrome_col	Character or NULL. Column of syndrome labels for pre-filtering. Default NULL.
syndrome_name	Character or NULL. Syndrome value to retain. Requires syndrome_col. Default NULL.

Details

One row per unique patient x centre is retained before plotting. Rows with age outside [min_age, max_age] are dropped.

Value

A ggplot object.

plot_burden_by_hospital

Plot YLL or YLD Associated vs Attributable per Hospital

Description

Produces a horizontal grouped bar chart comparing the associated and attributable burden (YLL or YLD) per 1 000 patients across hospitals. Bars are sorted by the associated value (largest at top by default). Each bar is labelled with its rounded value.

Usage

```
plot_burden_by_hospital(
  data,
  metric = c("YLL", "YLD"),
  facility_col = "center_name",
  center_col = NULL,
  assoc_col = NULL,
  attr_col = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL,
  sort_by = c("associated", "attributable", "none"),
  label_digits = 1,
  colours = NULL,
  base_size = 14,
  title = NULL,
  subtitle = NULL
)
```

Arguments

data	Data frame. One row per hospital (or one row per hospital x syndrome). Must contain the columns named by facility_col, assoc_col, and attr_col.
metric	Character. "YLL" or "YLD". Controls which default column names and axis labels are used. Ignored when assoc_col and attr_col are supplied explicitly. Default "YLL".
facility_col	Character. Column containing hospital / facility names. Default "center_name".
center_col	Deprecated alias for facility_col.
assoc_col	Character or NULL. Column for the associated per-1 000 value. If NULL (default), auto-set to "YLL_assoc_per_1000" or "YLD_assoc_per_1000" based on metric.
attr_col	Character or NULL. Column for the attributable per-1 000 value. If NULL (default), auto-set to "YLL_attr_per_1000" or "YLD_attr_per_1000" based on metric.

syndrome_col	Character or NULL. Column name containing syndrome labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
syndrome_name	Character or NULL. The syndrome value to retain (e.g. "Bloodstream infection"). Requires syndrome_col to be set. If NULL (default), all rows are used.
sort_by	Character. "associated" (default), "attributable", or "none". Controls hospital ordering on the y-axis.
label_digits	Integer. Decimal places in bar value labels. Default 1.
colours	Named character vector with exactly two elements: "associated" and "attributable". If NULL (default), a colour pair is chosen automatically based on metric (red tones for YLL, blue tones for YLD).
base_size	Numeric. Base font size. Default 14.
title	Character. Custom plot title. Auto-generated if NULL.
subtitle	Character. Custom subtitle. Auto-generated if NULL.

Details

The input data should be a hospital-level summary data frame (e.g. the output of `compute_hospital_daly()`) that already contains the per-1 000 columns. If the summary was computed across multiple syndromes, use `syndrome_col` and `syndrome_name` to restrict the plot to one syndrome before pivoting.

Value

A ggplot object.

plot_burden_by_organism

Plot YLL or YLD Associated vs Attributable by Organism

Description

Produces a horizontal grouped bar chart showing the top `n` organisms ranked by their associated burden (YLL or YLD), with a paired attributable bar alongside each. Organisms are ranked by the associated value so the highest-burden pathogen appears at the top.

Usage

```
plot_burden_by_organism(
  data,
  metric = c("YLL", "YLD"),
  n = 10,
  organism_col = NULL,
  assoc_col = NULL,
  attr_col = NULL,
  n_admissions = NULL,
```

```

syndrome_col = NULL,
syndrome_name = NULL,
label_digits = 1,
colours = NULL,
base_size = 14,
title = NULL,
subtitle = NULL
)

```

Arguments

<code>data</code>	Data frame. One row per organism. Must contain the columns named by <code>organism_col</code> , <code>assoc_col</code> , and <code>attr_col</code> .
<code>metric</code>	Character. "YLL" or "YLD". Controls default column names, colours, and axis labels. Ignored when <code>assoc_col</code> and <code>attr_col</code> are supplied explicitly. Default "YLL".
<code>n</code>	Integer. Number of top organisms to display. Default 10.
<code>organism_col</code>	Character or NULL. Column containing organism / pathogen names. If NULL (default), auto-set to "organism_name" for YLL (matching <code>daly_calc_yll_*</code> examples) and "pathogen" for YLD (matching <code>daly_calc_yld_*</code> defaults).
<code>assoc_col</code>	Character or NULL. Column for the associated burden value. If NULL (default), auto-set to "YLL_associated_k" (YLL) or "YLD_associated" (YLD) – matching the output column names of <code>compute_yll_associated()</code> and <code>compute_yld_associated()</code> respectively.
<code>attr_col</code>	Character or NULL. Column for the attributable burden value. If NULL (default), auto-set to "YLL_attributable_k" (YLL) or "YLD_attributable" (YLD) – matching the output column names of <code>compute_yll_attributable()</code> and <code>compute_yld_attributable()</code> respectively.
<code>n_admissions</code>	Numeric or NULL. When supplied, the associated and attributable values are divided by <code>n_admissions</code> and multiplied by 1 000 before plotting. Default NULL (no normalisation; values are plotted in thousands as returned by the YLL/YLD functions).
<code>syndrome_col</code>	Character or NULL. Column name containing syndrome labels. If NULL (default), no filtering is applied.
<code>syndrome_name</code>	Character or NULL. Syndrome value to retain. Requires <code>syndrome_col</code> . Default NULL.
<code>label_digits</code>	Integer. Decimal places in bar value labels. Default 1.
<code>colours</code>	Named character vector with elements "associated" and "attributable". If NULL (default), auto-chosen based on <code>metric</code> .
<code>base_size</code>	Numeric. Base font size. Default 14.
<code>title</code>	Character. Custom plot title. Auto-generated if NULL.
<code>subtitle</code>	Character. Custom subtitle. Auto-generated if NULL.

Details

The input data should be a pathogen-level summary data frame with one row per organism, already containing both the associated and attributable burden columns. For YLD this is typically the output of `compute_yld_associated()` left-joined with `compute_yld_attributable()`. For YLL it is the `by_pathogen` slot of `compute_yll_associated()` left-joined with the equivalent slot from `compute_yll_attributable()`.

Value

A ggplot object.

`plot_death_discharged` *Plot Death vs Discharged Counts for Top Pathogens*

Description

Produces a side-by-side (dodged) bar chart comparing Death and Discharged patient counts for the top `n` organisms. LAMA and Referred are excluded – only the two definitive outcomes are shown.

Usage

```
plot_death_discharged(
  data,
  n = 5,
  mode = c("faceted", "overall", "single"),
  center = NULL,
  resistance_filter = NULL,
  patient_col = "PatientInformation_id",
  organism_col = "organism_name",
  outcome_col = "final_outcome",
  center_col = "center_name",
  antibiotic_col = "antibiotic_name",
  value_col = "antibiotic_value",
  death_label = "Death",
  discharged_label = "Discharged",
  colours = NULL,
  bar_width = 0.65,
  ncol = 2,
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

<code>data</code>	Data frame. Long-format AMR dataset.
<code>n</code>	Integer. Number of top organisms to show. Default 5.
<code>mode</code>	Character. One of "faceted", "overall", or "single". Default "faceted".
<code>center</code>	Character. Required when mode = "single". Exact centre name.
<code>resistance_filter</code>	Character or NULL. When "R", only patients who had at least one Resistant result for the organism are included (worst-phenotype rule: any R = R). When "S", only patients with exclusively Susceptible results are included. NULL (default) includes all patients regardless of resistance status.
<code>patient_col</code>	Character. Patient ID column. Default "PatientInformation_id".
<code>organism_col</code>	Character. Organism name column. Default "organism_name".
<code>outcome_col</code>	Character. Final outcome column. Default "final_outcome".
<code>center_col</code>	Character. Centre/facility column. Default "center_name".
<code>antibiotic_col</code>	Character. Antibiotic name column. Only used when resistance_filter is not NULL. Default "antibiotic_name".
<code>value_col</code>	Character. Susceptibility result column (R/S). Only used when resistance_filter is not NULL. Default "antibiotic_value".
<code>death_label</code>	Character. Exact string used for death in the outcome column. Default "Death".
<code>discharged_label</code>	Character. Exact string used for discharged. Default "Discharged".
<code>colours</code>	Named character vector with keys matching death_label and discharged_label. Default c("Death" = "#E74C3C", "Discharged" = "#2ECC71").
<code>bar_width</code>	Numeric. Width of each individual bar (0-1). Default 0.65.
<code>ncol</code>	Integer. Facet columns (mode = "faceted" only). Default 2.
<code>base_size</code>	Numeric. Base font size. Default 14.
<code>title</code>	Character. Custom title. Auto-generated if NULL. Default NULL.
<code>syndrome_col</code>	Character or NULL. Column name containing syndrome/infection category labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
<code>syndrome_name</code>	Character or NULL. The syndrome value to retain (e.g. "BSI", "VAP"). Requires syndrome_col. If NULL (default), all syndromes are included.

Details

Each bar is labelled with the count and the percentage that outcome represents out of all Death + Discharged patients for that organism (e.g. 45 (32%) on a Death bar means 45 patients died, which is 32% of the organism's known outcomes). This makes the mortality rate immediately readable without a separate calculation.

Deduplication: uses `distinct(centre, patient, organism, outcome)` – one record per patient per organism per outcome – consistent with the original EDA script.

Top N organisms are ranked by total unique patients with a known outcome (Death or Discharged). For mode = "faceted" the ranking is done independently per centre; for "overall" it is done globally.

Supports three display modes:

- **"faceted"** (default) – one panel per centre.
- **"overall"** – all centres pooled into one chart.
- **"single"** – one specific centre.

Value

A ggplot object.

```
plot_hai_cai_by_facility
```

Plot HAI vs CAI Infection Distribution by Facility

Description

Derives Healthcare-Associated Infection (HAI) vs Community-Associated Infection (CAI) status from dates, then produces a stacked bar chart showing unique patient counts per centre. Each bar carries count and percentage labels inside each segment, and an n= total label above.

Usage

```
plot_hai_cai_by_facility(
  data,
  patient_col = "PatientInformation_id",
  center_col = "center_name",
  admission_col = "admission_date",
  culture_col = "culture_date",
  hai_threshold_hours = 48,
  hai_label = "HAI",
  cai_label = "CAI",
  colours = NULL,
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

data	Data frame. Long-format AMR dataset.
patient_col	Character. Patient ID column. Default "PatientInformation_id".
center_col	Character. Centre/facility column. Default "center_name".

admission_col	Character. Admission date column. Default "admission_date".
culture_col	Character. Culture/specimen date column. Default "culture_date".
hai_threshold_hours	Numeric. Hours from admission to first culture at or above which a case is classified as HAI. Default 48.
hai_label	Character. Label for HAI bars. Default "HAI".
cai_label	Character. Label for CAI bars. Default "CAI".
colours	Named character vector. Fill colours keyed by hai_label and cai_label. Default red/green.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom title. Auto-generated if NULL.
syndrome_col	Character or NULL. Column name containing syndrome/infection category labels. NULL = no filter.
syndrome_name	Character or NULL. Syndrome value to retain. Requires syndrome_col.

Details

Classification rule: for each patient x centre, the earliest culture date is used. If the time from admission to that culture is $\geq$ hai_threshold_hours (default 48 h), the patient is classified as HAI; otherwise CAI. The existing type_of_infection column in the dataset is *ignored* – classification is derived entirely from dates.

Deduplication: one classification per patient per centre (earliest culture date).

Value

A ggplot object.

plot_infection_type_by_location
<i>Plot Infection Type (HAI/CAI) by Location (ICU/Ward), Faceted by Hospital</i>

Description

100 types (e.g. HAI vs CAI) among patients in each location (e.g. ICU vs Ward). Each bar is normalised *within* its location, so the two bars in a panel directly compare $P(\text{infection type} \mid \text{location})$ – i.e. whether infection type is associated with location, per hospital.

Usage

```
plot_infection_type_by_location(
  data,
  patient_col = "PatientInformation_id",
  center_col = "center_name",
  location_col = "location",
  infection_col = "infection_type",
  location_levels = c("ICU", "Ward"),
  infection_levels = c("HAI", "CAI"),
  style = c("bar", "heatmap"),
  colours = NULL,
  ncol = 5,
  base_size = 12,
  show_counts = TRUE,
  title = NULL
)
```

Arguments

<code>data</code>	Data frame. Long-format AMR / cleaned stewardship data.
<code>patient_col</code>	Character. Patient ID column. Default "PatientInformation_id".
<code>center_col</code>	Character. Centre/facility column. Default "center_name".
<code>location_col</code>	Character. Location column. Default "location".
<code>infection_col</code>	Character. Infection-type column. Default "infection_type".
<code>location_levels</code>	Character vector. Location values to keep (become the x-axis bars). Default c("ICU", "Ward").
<code>infection_levels</code>	Character vector. Infection-type values to keep (become the stacked fills). Default c("HAI", "CAI").
<code>style</code>	Character. "bar" (default) draws a 100 bar per hospital (x = location, fill = infection type, normalised within location). "heatmap" draws a tile grid per hospital (x = location, y = infection type, fill = within-hospital joint proportion).
<code>colours</code>	Named character vector of fill colours keyed by infection_levels. Auto-generated if NULL.
<code>ncol</code>	Integer. Number of facet columns. Default 5.
<code>base_size</code>	Numeric. Base font size. Default 12.
<code>show_counts</code>	Logical. Print unique-patient counts inside each segment. Default TRUE.
<code>title</code>	Character. Custom title. Auto-generated if NULL.

Details

Deduplication: unique patients per centre x location x infection type. Only the requested location_levels and infection_levels are kept (other/blank/compound values are dropped).

Value

A ggplot object (one panel per centre).

plot_location_by_facility
<i>Plot Patient Distribution by Location Type Across Facilities</i>

Description

Produces a dodged bar chart showing the number of unique patients in each location type (ICU, Ward, Other) per facility. Raw location strings are normalised automatically via case-insensitive pattern matching, so centre-specific ICU names (e.g. "Respiratory ICU (RICU)") all map to "ICU".

Usage

```
plot_location_by_facility(  
  data,  
  patient_col = "PatientInformation_id",  
  location_col = "location",  
  center_col = "center_name",  
  icu_pattern = "icu",  
  ward_pattern = "ward",  
  icu_label = "ICU",  
  ward_label = "Ward",  
  other_label = "Other",  
  colours = NULL,  
  bar_width = 0.68,  
  base_size = 14,  
  title = NULL,  
  syndrome_col = NULL,  
  syndrome_name = NULL  
)
```

Arguments

data	Data frame. Long-format AMR dataset.
patient_col	Character. Patient ID column. Default "PatientInformation_id".
location_col	Character. Location/unit column. Default "location".
center_col	Character. Centre/facility column. Default "center_name".
icu_pattern	Character. Case-insensitive regex pattern that identifies ICU values. Default "icu".
ward_pattern	Character. Case-insensitive regex pattern that identifies Ward values. Default "ward".
icu_label	Character. Display label for ICU bars. Default "ICU".

ward_label	Character. Display label for Ward bars. Default "Ward".
other_label	Character. Display label for all other locations. Default "Other".
colours	Named character vector. Fill colours keyed by icu_label, ward_label, and other_label. Default pink/green/grey.
bar_width	Numeric. Width of each individual bar. Default 0.68.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom title. Auto-generated if NULL.
syndrome_col	Character or NULL. Column name containing syndrome/infection category labels. NULL = no filter.
syndrome_name	Character or NULL. Syndrome value to retain. Requires syndrome_col.

Details

Normalisation: a location value is classified as:

- **ICU** – if the lower-cased string contains icu_pattern (default "icu")
- **Ward** – else if it contains ward_pattern (default "ward")
- **Other** – everything else

Deduplication: distinct(patient, centre, location type) – each unique patient-centre-location combination is counted once.

Value

A ggplot object.

plot_los_by_agebin	<i>Boxplot of Length of Stay by Age Group</i>
--------------------	---

Description

Displays the distribution of LOS (y-axis, log10 scale) across patient age groups (x-axis) using boxplots. Supports "faceted" (one panel per centre), "overall" (all centres pooled), and "single" (one centre) modes.

Usage

```
plot_los_by_agebin(
  data,
  mode = c("faceted", "overall", "single"),
  center = NULL,
  patient_col = "PatientInformation_id",
  agebin_col = "Age_bin",
  age_col = "age_years",
  age_breaks = NULL,
```

```

age_labels = NULL,
center_col = "center_name",
admission_col = "admission_date",
discharge_col = "final_outcome_date",
min_los = 1,
max_los = 365,
age_levels = NULL,
fill_colour = "#2C7FB8",
fill_alpha = 0.7,
outlier_alpha = 0.25,
box_width = 0.6,
ncol = 3,
base_size = 14,
title = NULL,
subtitle = NULL,
syndrome_col = NULL,
syndrome_name = NULL
)

```

Arguments

<code>data</code>	A data frame.
<code>mode</code>	Character. One of "faceted", "overall", or "single".
<code>center</code>	Character. Centre name required when mode = "single".
<code>patient_col</code>	Character. Column of patient identifiers.
<code>agebin_col</code>	Character. Pre-existing age group column. Used when age_breaks is NULL. Default "Age_bin".
<code>age_col</code>	Character. Continuous age column used when age_breaks is provided. Default "age_years".
<code>age_breaks</code>	Numeric vector or NULL. Breakpoints for deriving custom bins from age_col (e.g. c(0, 5, 15, 30, 50, 70, Inf)). NULL = use agebin_col directly.
<code>age_labels</code>	Character vector or NULL. Labels for each bin created by age_breaks. Must have length length(age_breaks) - 1. NULL = auto-generate from breaks.
<code>center_col</code>	Character. Column of centre identifiers.
<code>admission_col</code>	Character. Admission date column. Default "admission_date".
<code>discharge_col</code>	Character. Discharge / final outcome date column. Default "final_outcome_date".
<code>min_los</code>	Numeric. Minimum LOS (days) to include. Default 1.
<code>max_los</code>	Numeric. Maximum LOS (days) to include. Default 365.
<code>age_levels</code>	Character vector or NULL. Custom bin order when using agebin_col. NULL = auto-sort.
<code>fill_colour</code>	Character. Box fill colour. Default "#2C7FB8".
<code>fill_alpha</code>	Numeric. Box fill transparency (0-1). Default 0.7.
<code>outlier_alpha</code>	Numeric. Outlier point transparency (0-1). Default 0.25.
<code>box_width</code>	Numeric. Width of each box. Default 0.6.

ncol	Integer. Number of facet columns (mode = "faceted" only). Default 3.
base_size	Numeric. Base font size. Default 14.
title	Character or NULL. Plot title. Auto-generated if NULL.
subtitle	Character or NULL. Plot subtitle. NULL = none.
syndrome_col	Character or NULL. Syndrome column for pre-filtering. Default NULL.
syndrome_name	Character or NULL. Syndrome value to retain. Requires syndrome_col. Default NULL.

Details

LOS is computed as `difftime(discharge_col, admission_col, units = "days")`. One row per unique patient x centre x admission x discharge is retained. Rows with `LOS < min_los` or `LOS > max_los` are dropped.

Two ways to supply age groups:

1. **Pre-existing column** (default) – set `agebin_col` to an existing categorical column (e.g. "Age_bin"). Bins are sorted automatically (<1, 1-5, ..., 85+); override order with `age_levels`.
2. **Custom breaks from continuous age** – set `age_breaks` to a numeric vector (e.g. `c(0, 5, 15, 30, 50, 70, Inf)`) and `age_col` to the continuous age column (default "age_years"). Bins are derived via `cut()` and labelled automatically. `agebin_col` is ignored when `age_breaks` is provided.

Value

A ggplot object.

plot_los_distributions

Plot LOS Distribution with Fitted Overlays

Description

Creates a histogram of LOS values with Weibull, Lognormal, and Gamma density curves overlaid.

Usage

```
plot_los_distributions(los_vec, title, bins = 35, fits = NULL)
```

Arguments

los_vec	Numeric vector of LOS values (days).
title	Character. Plot title.
bins	Integer. Number of histogram bins. Default 35.
fits	Optional. Pre-computed fits from <code>fit_distributions()</code> .

Value

A ggplot object.

Examples

```
plot_los_distributions(rlnorm(200, 2, 0.5), "Example LOS Distribution")
```

plot_los_ridge	<i>Ridge / Density Plot of Length of Stay (LOS)</i>
----------------	---

Description

Visualises the distribution of length of stay (LOS) on a log10 x-axis using either a stacked ridge plot (mode = "all") or a single-centre density curve (mode = "single").

Usage

```
plot_los_ridge(  
  data,  
  mode = c("all", "single"),  
  center = NULL,  
  patient_col = "patient_id",  
  center_col = "center_name",  
  admission_col = "admission_date",  
  discharge_col = "final_outcome_date",  
  filter_outcome = NULL,  
  outcome_col = "final_outcome",  
  min_los = 1,  
  max_los = 365,  
  colours = NULL,  
  scale = 1.2,  
  alpha = 0.7,  
  base_size = 14,  
  title = NULL,  
  syndrome_col = NULL,  
  syndrome_name = NULL  
)
```

Arguments

data	A data frame.
mode	Character. One of "all" (ridge plot, all centres stacked on y-axis, ordered by median LOS ascending) or "single" (density curve for one centre specified by center).
center	Character. Centre name required when mode = "single".
patient_col	Character. Column of patient identifiers.

center_col	Character. Column of centre identifiers.
admission_col	Character. Column containing admission date/datetime. Default "admission_date".
discharge_col	Character. Column containing discharge / final outcome date. Default "final_outcome_date".
filter_outcome	Character or NULL. If not NULL, only rows whose outcome column (outcome_col) equals this value are used (e.g. "Discharged"). Set NULL to include all outcomes. Default NULL.
outcome_col	Character. Column of patient outcomes, used only when filter_outcome is not NULL. Default "final_outcome".
min_los	Numeric. Minimum LOS (days) to include. Rows below this are removed (prevents $\log(0)$ issues). Default 1.
max_los	Numeric. Maximum LOS (days) to include. Default 365.
colours	Named character vector of colours for centres. If NULL (default), the RColorBrewer "Set2" palette is used.
scale	Numeric. scale argument passed to <code>ggribes::geom_density_ridges()</code> . Controls overlap between ridges. Default 1.2.
alpha	Numeric. Fill transparency (0-1). Default 0.7.
base_size	Numeric. Base font size. Default 14.
title	Character or NULL. Plot title. Auto-generated if NULL.
syndrome_col	Character or NULL. Column of syndrome labels for pre-filtering. Default NULL.
syndrome_name	Character or NULL. Syndrome value to retain. Requires syndrome_col. Default NULL.

Details

LOS is computed as `difftime(discharge_col, admission_col, units = "days")`. One row per unique patient x centre x admission_date x discharge_date is retained before plotting. Rows with $LOS < min_los$ or $LOS > max_los$ are dropped.

Value

A ggplot object.

plot_mono_poly_by_facility

Plot Mono vs Polymicrobial Infections by Facility

Description

Produces a dodged bar chart comparing the number of unique patients with Monomicrobial vs Polymicrobial infections at each centre. Hospitals appear on the x-axis; the two bars per hospital are coloured by infection type. Count labels are printed above each bar.

Usage

```
plot_mono_poly_by_facility(
  data,
  patient_col = "PatientInformation_id",
  poly_col = "is_polymicrobial",
  center_col = "center_name",
  mono_label = "Monomicrobial",
  poly_label = "Polymicrobial",
  colours = NULL,
  bar_width = 0.68,
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

data	Data frame. Long-format AMR dataset.
patient_col	Character. Patient ID column. Default "PatientInformation_id".
poly_col	Character. Column with binary infection type (0 = mono, 1 = poly). Default "is_polymicrobial".
center_col	Character. Centre/facility column. Default "center_name".
mono_label	Character. Label for monomicrobial bars. Default "Monomicrobial".
poly_label	Character. Label for polymicrobial bars. Default "Polymicrobial".
colours	Named character vector. Fill colours keyed by mono_label and poly_label. Default blue/red.
bar_width	Numeric. Width of each individual bar. Default 0.68.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom title. Auto-generated if NULL.
syndrome_col	Character or NULL. Column name containing syndrome/infection category labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
syndrome_name	Character or NULL. The syndrome value to retain (e.g. "BSI", "VAP"). Requires syndrome_col to be set. If NULL (default), all syndromes are included.

Details

Deduplication: distinct(patient, centre, infection type) – each unique patient-centre-infection type combination is counted once. A patient who has both mono and polymicrobial episodes at the same centre contributes to both bars, matching the original EDA script.

Infection type: the poly_col column is expected to contain binary values (0 = Monomicrobial, 1 = Polymicrobial). The labels shown in the legend and plot are controlled by mono_label and poly_label.

Value

A ggplot object.

plot_omega_convergence_heatmap

Sampling-diagnostic (Rhat) heatmap for Omega

Description

Companion to plot_omega_correlation_heatmap() – shows Rhat for each $\Omega[i, j]$ entry, so a reader cannot look at an attractive correlation heatmap without also seeing whether those specific parameters actually converged. A uniformly high correlation heatmap alongside a uniformly poor Rhat heatmap here is the signature of a near-degenerate, weakly-identified correlation matrix, not a settled finding.

Usage

```
plot_omega_convergence_heatmap(corr_summary, class_cols, title_base = "")
```

Arguments

corr_summary	Tibble from summarize_fit_correlation_matrix() (with class_1, class_2, correlation_median, rhat columns), or NULL.
class_cols	Character vector of all class names, in the fit's canonical order (for consistent row/column ordering).
title_base	Character. Prefixed to the plot title.

Value

A ggplot object, or NULL if corr_summary is NULL/empty or has no rhat column (e.g. computed by an older summarize_fit_correlation_matrix() call).

plot_omega_correlation_heatmap

Posterior-median latent cross-class correlation heatmap (Omega)

Description

Rows/columns are antimicrobial classes; each cell is the posterior median of the corresponding $\Omega[i, j]$ (or $R_block[r, i, j]$) entry. Only meaningful for correlated-residual fits – pass NULL (e.g. an identity-residual fit's summarize_fit_correlation_matrix() result) to get NULL back rather than an empty/misleading plot.

Usage

```
plot_omega_correlation_heatmap(
  corr_summary,
  class_cols,
  title_base = "",
  rhat_flag_threshold = 1.01
)
```

Arguments

corr_summary	Tibble from summarize_fit_correlation_matrix() (with class_1, class_2, correlation_median, rhat columns), or NULL.
class_cols	Character vector of all class names, in the fit's canonical order (for consistent row/column ordering).
title_base	Character. Prefixed to the plot title.
rhat_flag_threshold	Numeric. Cells with rhat above this are marked with an asterisk, cross-referencing the companion convergence heatmap rather than repeating full diagnostics on this plot.

Value

A ggplot object, or NULL if corr_summary is NULL/empty.

plot_omega_degeneracy_diagnostic

Near-degeneracy diagnostic for the posterior Omega correlation matrix

Description

A multivariate-probit correlated-residual fit can push Omega toward a near-singular boundary (very high pairwise correlations across the board) without that being visible from any single $\Omega_{[i,j]}$ entry. For each posterior draw this computes the smallest eigenvalue of the full $D \times D$ Omega matrix – a value approaching 0 means that specific draw's correlation matrix is nearly singular – and reports the distribution across draws, plus the condition number (largest / smallest eigenvalue) as a supplementary summary.

Usage

```
plot_omega_degeneracy_diagnostic(
  fit,
  class_cols,
  title_base = "",
  degenerate_threshold = 0.05
)
```

Arguments

<code>fit</code>	List returned by <code>fit_bayesian_multivariate_probit()</code> .
<code>class_cols</code>	Character vector of all class names, in the fit's canonical order.
<code>title_base</code>	Character. Prefixed to the plot title.
<code>degenerate_threshold</code>	Numeric. Draws with smallest eigenvalue below this are counted as "near-degenerate" in the subtitle. Default 0.05.

Value

A ggplot object, or NULL if this is not a correlated-residual fit, or Omega draws are unavailable.

<code>plot_omega_summary_table</code>	<i>Full per-pair Omega summary table (rho, 95% CrI, Rhat, ESS bulk/tail)</i>
---------------------------------------	--

Description

The two heatmaps show the correlation estimate and its Rhat as separate pictures; this page puts every number for every class pair – median rho, credible interval, Rhat, bulk and tail ESS – in one table, for a reader who wants the exact figures rather than reading them off a colour scale.

Usage

```
plot_omega_summary_table(corr_summary, class_cols, title_base = "")
```

Arguments

<code>corr_summary</code>	Tibble from <code>summarize_fit_correlation_matrix()</code> (with <code>class_1</code> , <code>class_2</code> , <code>correlation_median</code> , <code>rhat</code> columns), or NULL.
<code>class_cols</code>	Character vector of all class names, in the fit's canonical order (for consistent row/column ordering).
<code>title_base</code>	Character. Prefixed to the plot title.

Value

A ggplot object (table page), or NULL if `corr_summary` is NULL/empty.

plot_outcome_by_agebin

Plot Final Outcome Proportions by Age Bin

Description

Produces a horizontal 100% stacked bar chart showing how final outcomes (Death, Discharged, LAMA, Referred) are distributed across age groups. Each bar represents one age bin and sums to 100%. An n= label above each bar shows the total patients in that age group.

Usage

```
plot_outcome_by_agebin(
  data,
  mode = c("faceted", "overall", "single"),
  center = NULL,
  patient_col = "PatientInformation_id",
  agebin_col = "Age_bin",
  outcome_col = "final_outcome",
  center_col = "center_name",
  merge_referred = TRUE,
  age_levels = NULL,
  palette = c(Death = "#E74C3C", Died = "#E74C3C", Discharged = "#2ECC71", LAMA =
    "#95A5A6", Referred = "#3498DB", `Transferred to other hospital` = "#3498DB"),
  ncol = 2,
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

data	Data frame. Long-format AMR dataset.
mode	Character. One of "faceted", "overall", or "single". Default "faceted".
center	Character. Required when mode = "single". Exact centre name.
patient_col	Character. Patient ID column. Default "PatientInformation_id".
agebin_col	Character. Age bin column. Default "Age_bin".
outcome_col	Character. Final outcome column. Default "final_outcome".
center_col	Character. Centre/facility column. Default "center_name".
merge_referred	Logical. Recode "Transferred to other hospital" to "Referred". Default TRUE.
age_levels	Character vector or NULL. Custom ordering of age bin labels. If NULL (default), bins are sorted automatically (<1 first, numeric ranges in order, open-ended bins last).

palette	Named character vector. Outcome -> fill colour mapping. Unmatched outcomes receive grey. Default uses red/green/grey/blue preset.
ncol	Integer. Facet columns (mode = "faceted" only). Default 2.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom title. Auto-generated if NULL. Default NULL.
syndrome_col	Character or NULL. Column name containing syndrome/infection category labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
syndrome_name	Character or NULL. The syndrome value to retain (e.g. "BSI", "VAP"). Requires syndrome_col to be set. If NULL (default), all syndromes are included.

Details

Deduplication: one record per patient per centre – first recorded outcome and age bin per patient per centre. Each patient contributes exactly one bar segment per centre.

Age bin ordering: bins are ordered numerically (<1, 1-5, 5-10, ..., 85+) automatically. Supply age_levels to override with a custom order.

Supports three display modes:

- **"faceted"** (default) – one panel per centre.
- **"overall"** – all centres pooled; one chart.
- **"single"** – one specific centre.

Value

A ggplot object.

plot_outcome_by_organism

Plot Final Outcome Proportions for Resistant or Susceptible Patients

Description

Produces a 100% horizontal stacked bar chart showing the distribution of final outcomes (Death, Discharged, LAMA, Referred) for the top n organisms, split by whether the patient had a Resistant or Susceptible result.

Usage

```
plot_outcome_by_organism(
  data,
  n = 5,
  resistance_filter = c("R", "S"),
  mode = c("faceted", "overall", "single"),
  center = NULL,
```

```

patient_col = "PatientInformation_id",
organism_col = "organism_name",
antibiotic_col = "antibiotic_name",
value_col = "antibiotic_value",
outcome_col = "final_outcome",
center_col = "center_name",
merge_referred = TRUE,
palette = c(Death = "#E74C3C", Died = "#E74C3C", Discharged = "#2ECC71", LAMA =
"#95A5A6", Referred = "#3498DB", `Transferred to other hospital` = "#3498DB"),
ncol = 2,
base_size = 14,
title = NULL,
syndrome_col = NULL,
syndrome_name = NULL
)

```

Arguments

<code>data</code>	Data frame. Long-format AMR dataset.
<code>n</code>	Integer. Number of top organisms to show. Default 5.
<code>resistance_filter</code>	Character. "R" or "S". Which patient group to display. Default "R".
<code>mode</code>	Character. One of "faceted", "overall", or "single". Default "faceted".
<code>center</code>	Character. Required when mode = "single". Exact centre name.
<code>patient_col</code>	Character. Patient ID column. Default "PatientInformation_id".
<code>organism_col</code>	Character. Organism name column. Default "organism_name".
<code>antibiotic_col</code>	Character. Antibiotic name column. Default "antibiotic_name".
<code>value_col</code>	Character. Susceptibility result column (R/S). Default "antibiotic_value".
<code>outcome_col</code>	Character. Final outcome column. Default "final_outcome".
<code>center_col</code>	Character. Centre/facility column. Default "center_name".
<code>merge_referred</code>	Logical. Recode "Transferred to other hospital" to "Referred". Default TRUE.
<code>palette</code>	Named character vector mapping outcome values to colours. Unmatched outcomes get grey automatically.
<code>ncol</code>	Integer. Facet columns (mode = "faceted" only). Default 2.
<code>base_size</code>	Numeric. Base font size. Default 14.
<code>title</code>	Character. Custom title. Auto-generated if NULL. Default NULL.
<code>syndrome_col</code>	Character or NULL. Column name containing syndrome/infection category labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
<code>syndrome_name</code>	Character or NULL. The syndrome value to retain (e.g. "BSI", "VAP"). Requires syndrome_col. If NULL (default), all syndromes are included.

Details

Each bar represents one organism. The fill shows outcome proportions. A $n=$ label is printed above each bar showing the total number of unique patients in that organism x resistance group.

Resistance grouping: `resistance_filter = "R"` shows patients who had *at least one* R result for the given organism. `"S"` shows patients with only S results. Because these groups overlap (a patient resistant to one antibiotic may be susceptible to another), calling the function twice – once for "R" and once for "S" – gives the most complete picture.

Top N organisms are ranked by total unique patients per centre (or globally for `mode = "overall"`), calculated before the R/S split so the same organisms appear in both R and S plots.

Supports three display modes:

- **"faceted"** (default) – one panel per centre.
- **"overall"** – all centres pooled.
- **"single"** – one specific centre.

Value

A ggplot object.

plot_outcome_by_year *Plot Distribution of Final Outcomes by Year*

Description

Produces a stacked bar chart where the x-axis shows the year derived from `date_col`, the y-axis shows the number of unique patients, and bars are filled by final outcome category. Each bar segment is labelled with count and percentage; a total $n=$ label sits above each bar.

Usage

```
plot_outcome_by_year(
  data,
  mode = c("faceted", "overall", "single"),
  center = NULL,
  patient_col = "PatientInformation_id",
  outcome_col = "final_outcome",
  date_col = "final_outcome_date",
  center_col = "center_name",
  merge_referred = TRUE,
  palette = c(Death = "#E74C3C", Died = "#E74C3C", Discharged = "#2ECC71", LAMA =
    "#95A5A6", Referred = "#3498DB", `Transferred to other hospital` = "#3498DB"),
  ncol = 2,
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

<code>data</code>	Data frame. Long-format AMR dataset.
<code>mode</code>	Character. One of "faceted", "overall", or "single". Default "faceted".
<code>center</code>	Character. Required when mode = "single". Exact centre name.
<code>patient_col</code>	Character. Patient ID column. Default "PatientInformation_id".
<code>outcome_col</code>	Character. Final outcome column. Default "final_outcome".
<code>date_col</code>	Character. Date column used to extract the year (must be a Date or POSIXct column). Default "final_outcome_date".
<code>center_col</code>	Character. Centre/facility column. Default "center_name".
<code>merge_referred</code>	Logical. Recode "Transferred to other hospital" to "Referred". Default TRUE.
<code>palette</code>	Named character vector mapping outcome values to colours. Unmatched outcomes get a default ggplot colour.
<code>ncol</code>	Integer. Facet columns (mode = "faceted" only). Default 2.
<code>base_size</code>	Numeric. Base font size. Default 14.
<code>title</code>	Character. Custom plot title. Auto-generated if NULL. Default NULL.
<code>syndrome_col</code>	Character or NULL. Syndrome filter column. Default NULL.
<code>syndrome_name</code>	Character or NULL. Syndrome value to retain. Requires syndrome_col. Default NULL.

Details

One outcome is taken per patient per centre (the first recorded value) to avoid double-counting long-format rows.

Supports three display modes:

- **"faceted"** (default) – one panel per centre.
- **"overall"** – all centres pooled into a single chart.
- **"single"** – one specific centre; pass its name via center.

Value

A ggplot object.

plot_outcome_distribution

Plot Distribution of Final Outcomes

Description

Produces a bar chart of final outcome counts per centre, with each bar labelled with the count and percentage of that centre's (or overall) unique patients. Bars are coloured by outcome category.

Usage

```
plot_outcome_distribution(
  data,
  mode = c("faceted", "overall", "single"),
  center = NULL,
  patient_col = "PatientInformation_id",
  outcome_col = "final_outcome",
  center_col = "center_name",
  merge_referred = TRUE,
  palette = c(Death = "#E74C3C", Died = "#E74C3C", Discharged = "#2ECC71", LAMA =
    "#95A5A6", Referred = "#3498DB", `Transferred to other hospital` = "#3498DB"),
  ncol = 2,
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

data	Data frame. Long-format AMR dataset.
mode	Character. One of "faceted", "overall", or "single". Default "faceted".
center	Character. Required when mode = "single". Exact centre name.
patient_col	Character. Patient ID column. Default "PatientInformation_id".
outcome_col	Character. Final outcome column. Default "final_outcome".
center_col	Character. Centre/facility column. Default "center_name".
merge_referred	Logical. Recode "Transferred to other hospital" to "Referred". Default TRUE.
palette	Named character vector mapping outcome values to colours. Any outcome not in the vector gets a default ggplot colour. Default uses a preset palette for common outcome categories.
ncol	Integer. Facet columns (mode = "faceted" only). Default 2.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom plot title. Auto-generated if NULL. Default NULL.

syndrome_col	Character or NULL. Column name containing syndrome/infection category labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
syndrome_name	Character or NULL. The syndrome value to retain (e.g. "BSI", "VAP"). Requires syndrome_col to be set. If NULL (default), all syndromes are included.

Details

Deduplication: a patient may appear multiple times in a long-format dataset (one row per isolate or antibiotic). This function takes the *first* recorded outcome per patient per centre so each patient is counted once.

Referred merging: by default, "Transferred to other hospital" is recoded to "Referred" since they represent the same clinical event across different centres. Set merge_referred = FALSE to keep them separate.

Supports three display modes:

- **"faceted"** (default) – one panel per centre.
- **"overall"** – all centres pooled into one chart.
- **"single"** – one specific centre; pass the name via center.

Value

A ggplot object.

plot_patients_by_hospital

Plot Unique Patient Count by Hospital

Description

Produces a column chart showing the number of unique patients per hospital. Bars are sorted in descending order (highest count on the left) and each bar is labelled with its count.

Usage

```
plot_patients_by_hospital(
  data,
  patient_col = "PatientInformation_id",
  center_col = "center_name",
  colour = "#4393C3",
  bar_width = 0.68,
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

data	Data frame. Must contain the columns named by patient_col and center_col.
patient_col	Character. Column containing patient identifiers. Default "PatientInformation_id".
center_col	Character. Column containing hospital / centre names. Default "center_name".
colour	Character. Fill colour for the bars. Default "#4393C3".
bar_width	Numeric. Width of bars (0-1). Default 0.68.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom plot title. Auto-generated if NULL.
syndrome_col	Character or NULL. Syndrome filter column. Default NULL.
syndrome_name	Character or NULL. Syndrome value to retain. Requires syndrome_col. Default NULL.

Value

A ggplot object.

plot_probit_beta_family_diagnostics

Fixed-effect convergence broken out by source of variation

Description

Splits the fixed-effect ("beta") parameter group by which covariate family each coefficient belongs to (Intercept / Hospital / Age / Sex / Specimen / Ward-ICU / Other, by covariate-name pattern), so a reviewer can see directly whether e.g. hospital contrasts specifically are destabilising the fit, rather than only a single aggregate Rhat/ESS for all of "beta".

Usage

```
plot_probit_beta_family_diagnostics(fit, title_base = "")
```

Arguments

fit	List returned by fit_bayesian_multivariate_probit().
title_base	Character. Prefixed to plot titles.

Value

Named list with table (a formatted summary page) and bar (a percent-Rhat-above-1.01-by-family bar chart), or NULL if fixed-effect parameter diagnostics are unavailable.

```
plot_probit_diagnostics
```

Generate diagnostic plots for a fitted Bayesian multivariate probit model

Description

Generate diagnostic plots for a fitted Bayesian multivariate probit model

Usage

```
plot_probit_diagnostics(
  fit_obj,
  output_dir,
  experiment_id,
  pathogen,
  max_params = 30L
)
```

Arguments

<code>fit_obj</code>	Named list returned by <code>fit_bayesian_multivariate_probit()</code> .
<code>output_dir</code>	Directory where the PDF will be saved.
<code>experiment_id</code>	Character. Experiment identifier (used in filename and titles).
<code>pathogen</code>	Character. Pathogen name (used in plot titles).
<code>max_params</code>	Integer. Maximum number of parameters shown in trace/rank plots. Defaults to 30. Includes <code>lp__</code> , all <code>tau_*</code> , and controlled subsets of <code>beta</code> , random-effect terms, random-effect correlations, and residual-correlation terms when available.

Value

Invisibly returns the path to the saved PDF, or NULL if skipped.

```
plot_probit_fixed_effect_diagnostics
```

Fixed-effect coefficient diagnostics, split by contrast type and faceted by antimicrobial class

Description

Returns a named list of up to two ggplot objects: `other` (every non-hospital-dummy covariate – intercept, age, gender, etc.) and `hospital` (hospital contrasts only, if the fit used hospital as a fixed effect rather than a random effect). Both facet by antimicrobial class rather than putting every class's coefficients on one axis.

Usage

```
plot_probit_fixed_effect_diagnostics(fit, title_base = "", ci_level = 0.95)
```

Arguments

fit	List returned by <code>fit_bayesian_multivariate_probit()</code> .
title_base	Character. Prefixed to plot titles.
ci_level	Numeric. Credible interval level. Default 0.95.

Value

Named list of ggplot objects (other, hospital), either of which may be absent if that group has no covariates in this fit. NULL if `fit$draws` or `fit$X_event` is missing.

```
plot_probit_posterior_predictive_checks
```

Plot posterior predictive checks for a fitted probit model

Description

Compact multi-page PDF: observed-vs-replicated marginal resistance, number-resistant-per-event distribution, observed-vs-replicated pairwise RR/RS/SR/SS, complete-profile concentration/entropy, hospital heterogeneity, admission/patient clustering, and (when `ppc_replicates` with materialised replicate arrays is supplied) a small-multiples page comparing the observed data against `n_small_multiples` randomly selected posterior replications. A good posterior predictive plot makes it visually apparent whether the observed data look typical among the replicated datasets.

Usage

```
plot_probit_posterior_predictive_checks(
  ppc_statistics,
  ppc_replicates = NULL,
  output_pdf_path,
  title_base,
  n_small_multiples = 10L,
  small_multiple_seed = 123L
)
```

Arguments

ppc_statistics	Tibble returned by compute_probit_ppc_statistics .
ppc_replicates	Optional "amr_ppc_draws" object returned by simulate_probit_posterior_predictive with <code>return_replicates = TRUE</code> – enables the small-multiples page. NULL (default) skips that page.
output_pdf_path	Path to the PDF to write.

title_base Character; prefixed to every page's title (e.g. "<experiment_id>\n<pathogen>").
n_small_multiples Integer; number of replicate states to show on the small-multiples page. Default 10.
small_multiple_seed Integer seed for selecting which replicate states appear on the small-multiples page. Default 123.

Value

Invisibly returns the path to the saved PDF, or NULL if skipped.

plot_probit_worst_offender_diagnostics
Trace and rank plots for the worst-converging structural parameters

Description

Auto-selects the union of the top `n_worst` parameters by Rhat and the top `n_worst` by lowest bulk ESS (from `fit$diagnostics_detail$monitored_parameters`), and shows their chain trace and rank-overlay plots with human-readable facet labels (covariate x class, or correlation pair) instead of raw Stan parameter names – enough to tell slow-mixing, chain separation, a sticky region, and multimodality apart, which a single Rhat number cannot.

Usage

```
plot_probit_worst_offender_diagnostics(
  fit,
  draws,
  title_base = "",
  n_worst = 6L
)
```

Arguments

fit List returned by `fit_bayesian_multivariate_probit()`.
draws A `posterior::draws_array` (typically `fit$draws`).
title_base Character. Prefixed to plot titles.
n_worst Integer. How many parameters to select by each criterion (Rhat, bulk ESS) before deduplicating. Default 6.

Value

Named list with trace and rank ggplot objects, or NULL if no structural parameter diagnostics are available.

plot_probit_worst_parameters

Human-readable worst-converging structural parameters

Description

Replaces raw Stan parameter names (e.g. "beta[12,2]") with the substantive covariate/class (or correlation pair) they refer to, and shows only parameters that actually exceed the Rhat threshold – not an arbitrary fixed top-N – alongside their bulk and tail ESS: a high Rhat with healthy ESS is a different diagnostic situation than a high Rhat with very low ESS, and showing only one of the two obscures that distinction.

Usage

```
plot_probit_worst_parameters(
  fit,
  title_base = "",
  rhat_threshold = 1.01,
  max_rows = 40L
)
```

Arguments

fit	List returned by fit_bayesian_multivariate_probit().
title_base	Character. Prefixed to plot titles.
rhat_threshold	Numeric. Only parameters with Rhat above this are shown. Default 1.01.
max_rows	Integer cap on rows shown, in case very many parameters fail. Default 40.

Value

A ggplot object (table page), or NULL if no structural parameter exceeds the threshold, or diagnostics are unavailable.

plot_resistance_by_agebin

Resistance proportion by age group

Description

100 vs Susceptible patients within each age bin. A patient is classified as Resistant if they have ANY (organism × antibiotic) result of "R"; Susceptible only when every result is "S" (worst-phenotype rule applied at patient × organism × antibiotic level, then collapsed to patient level).

Usage

```
plot_resistance_by_agebin(
  data,
  mode = c("faceted", "overall", "single"),
  center = NULL,
  patient_col = "patient_id",
  organism_col = "organism_normalized",
  antibiotic_col = "antibiotic_name",
  value_col = "result",
  agebin_col = "Age_bin",
  center_col = "center_name",
  colours = NULL,
  base_size = 11,
  ncol = 3,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

<code>data</code>	Data frame (one row per patient × organism × antibiotic result).
<code>mode</code>	One of "faceted", "overall", or "single".
<code>center</code>	Centre name used when mode = "single".
<code>patient_col</code>	Column for patient ID.
<code>organism_col</code>	Column for organism name.
<code>antibiotic_col</code>	Column for antibiotic name.
<code>value_col</code>	Column for resistance result ("R" / "S").
<code>agebin_col</code>	Column for age bin label.
<code>center_col</code>	Column for centre name.
<code>colours</code>	Named character vector with "R" and "S" entries.
<code>base_size</code>	Base font size for <code>eda_theme()</code> .
<code>ncol</code>	Facet columns (faceted mode only).
<code>title</code>	Optional title override.
<code>syndrome_col</code>	Optional column used to pre-filter by syndrome.
<code>syndrome_name</code>	Optional value of <code>syndrome_col</code> to keep.

plot_resistance_by_organism

Resistance proportion for top N organisms

Description

100 vs Susceptible patients for the top n organisms (ranked by total distinct patient count). A patient x organism pair is classified as Resistant if any antibiotic result for that pair is "R" (worst-phenotype rule at patient x organism x antibiotic level).

Usage

```
plot_resistance_by_organism(
  data,
  n = 5,
  mode = c("faceted", "overall", "single"),
  center = NULL,
  patient_col = "patient_id",
  organism_col = "organism_normalized",
  antibiotic_col = "antibiotic_name",
  value_col = "result",
  center_col = "center_name",
  colours = NULL,
  base_size = 11,
  ncol = 3,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

data	Data frame (one row per patient x organism x antibiotic result).
n	Number of top organisms to display.
mode	One of "faceted", "overall", or "single".
center	Centre name used when mode = "single".
patient_col	Column for patient ID.
organism_col	Column for organism name.
antibiotic_col	Column for antibiotic name.
value_col	Column for resistance result ("R" / "S").
center_col	Column for centre name.
colours	Named character vector with "R" and "S" entries.
base_size	Base font size for eda_theme().
ncol	Facet columns (faceted mode only).

title	Optional title override.
syndrome_col	Optional column used to pre-filter by syndrome.
syndrome_name	Optional value of syndrome_col to keep.

plot_resistance_by_sample

Plot Distribution of Antibiotic Resistance Across Sample Types

Description

Produces a horizontal 100% stacked bar chart showing the proportion of Resistant (R) and Susceptible (S) deduplicated AST results for the top n sample types. Each bar sums to 100% and is labelled with the R and S percentages.

Usage

```
plot_resistance_by_sample(  
  data,  
  n = 10,  
  mode = c("faceted", "overall", "single"),  
  center = NULL,  
  patient_col = "PatientInformation_id",  
  sample_col = "sample_type",  
  organism_col = "organism_name",  
  antibiotic_col = "antibiotic_name",  
  value_col = "antibiotic_value",  
  center_col = "center_name",  
  colours = c(R = "#D73027", S = "#1A9850"),  
  ncol = 2,  
  base_size = 14,  
  title = NULL,  
  syndrome_col = NULL,  
  syndrome_name = NULL  
)
```

Arguments

data	Data frame. Long-format AMR dataset.
n	Integer. Number of top sample types to show (ranked by deduplicated test count). Default 10.
mode	Character. One of "faceted", "overall", or "single". Default "faceted".
center	Character. Required when mode = "single". Exact centre name.
patient_col	Character. Patient ID column. Used for deduplication. Default "PatientInformation_id".
sample_col	Character. Sample/specimen type column. Default "sample_type".
organism_col	Character. Organism name column. Used for deduplication. Default "organism_name".

antibiotic_col	Character. Antibiotic name column. Default "antibiotic_name".
value_col	Character. Susceptibility result column (R/S values). Default "antibiotic_value".
center_col	Character. Centre/facility column. Default "center_name".
colours	Named character vector for R and S bars. Default c("R" = "#D73027", "S" = "#1A9850").
ncol	Integer. Facet columns (mode = "faceted" only). Default 2.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom title. Auto-generated if NULL. Default NULL.
syndrome_col	Character or NULL. Column name containing syndrome/infection category labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
syndrome_name	Character or NULL. The syndrome value to retain (e.g. "BSI", "VAP"). Requires syndrome_col to be set. If NULL (default), all syndromes are included.

Details

Deduplication (worst-phenotype): before counting, rows are collapsed by patient x centre x sample type x organism x antibiotic. If the same antibiotic appears more than once for a patient-organism-sample combination (data entry error or repeat test), any single R result marks the episode as R. This prevents duplicate rows from inflating counts.

Counting unit: deduplicated AST test results – one record per patient x organism x antibiotic x sample type combination.

Top N ranking: sample types are ranked by total deduplicated test count, so the most-tested specimen types appear first.

Default sample column: "sample_type" – the raw column. Switch to "specimen_normalized" for standardised values.

Supports three display modes:

- **"faceted"** (default) – top *n* sample types per centre independently, one panel per centre.
- **"overall"** – all centres pooled; top *n* globally.
- **"single"** – one specific centre.

Value

A ggplot object.

```
plot_syndrome_distribution
```

Plot Syndrome Distribution

Description

Produces a column chart showing the number of unique patients per infectious syndrome. Supports three modes: pooled across all centres ("overall"), faceted by centre ("faceted"), or a single specified centre ("single").

Usage

```
plot_syndrome_distribution(
  data,
  mode = c("overall", "faceted", "single"),
  center = NULL,
  patient_col = "PatientInformation_id",
  syndrome_col = "infectious_syndrome",
  center_col = "center_name",
  colour = "#4393C3",
  ncol = 2,
  base_size = 14,
  title = NULL
)
```

Arguments

data	Data frame. Must contain the columns named by patient_col and syndrome_col. In "faceted" and "single" modes, center_col is also required.
mode	Character. One of "overall" (default), "faceted", or "single".
center	Character or NULL. Centre name to display when mode = "single". Ignored otherwise.
patient_col	Character. Column containing patient identifiers. Default "PatientInformation_id".
syndrome_col	Character. Column containing infectious syndrome labels. Default "infectious_syndrome".
center_col	Character. Column containing hospital / centre names. Default "center_name".
colour	Character. Fill colour for the bars. Default "#4393C3".
ncol	Integer. Number of columns in faceted layout. Default 2.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom plot title. Auto-generated if NULL.

Value

A ggplot object.

plot_top_organisms *Plot Top Pathogens by Unique Patients*

Description

Produces a horizontal bar chart of the top n organisms ranked by unique patient count. Supports three display modes:

- **"faceted"** – one panel per centre, top n organisms within each centre independently. Facet strips show patient and organism counts for that centre.
- **"overall"** – all centres pooled; single chart of the top n organisms across the entire dataset.
- **"single"** – one specific centre; pass the centre name via center.

Usage

```
plot_top_organisms(
  data,
  n = 5,
  mode = c("faceted", "overall", "single"),
  center = NULL,
  patient_col = "PatientInformation_id",
  organism_col = "organism_name",
  center_col = "center_name",
  fill_colour = "steelblue",
  ncol = 2,
  base_size = 14,
  title = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

Arguments

data	Data frame. Long-format AMR dataset (one row per isolate/antibiotic record).
n	Integer. Number of top organisms to show. Default 5.
mode	Character. One of "faceted", "overall", or "single". Default "faceted".
center	Character. Required when mode = "single". Exact name of the centre to plot (must match a value in center_col).
patient_col	Character. Column name for patient identifiers. Default "PatientInformation_id".
organism_col	Character. Column name for organism names. Default "organism_name".
center_col	Character. Column name for centre/facility names. Default "center_name".
fill_colour	Character. Bar fill colour (any R colour string or hex). Default "steelblue".
ncol	Integer. Number of columns in the facet grid (mode = "faceted" only). Default 2.

base_size	Numeric. Base font size passed to eda_theme(). Default 14.
title	Character. Custom plot title. If NULL a title is generated automatically from mode and n. Default NULL.
syndrome_col	Character or NULL. Column name containing syndrome/infection category labels (e.g. "infectious_syndrome"). If NULL (default), no syndrome filtering is applied.
syndrome_name	Character or NULL. The syndrome value to retain (e.g. "BSI", "VAP"). Requires syndrome_col to be set. If NULL (default), all syndromes are included.

Details

Labels on each bar show absolute count and percentage of that centre’s (or overall) unique patient total.

Value

A ggplot object. Print it or pass to ggsave().

plot_yld_heatmap	<i>Heatmap of YLD per 1 000 Admissions by Organism Group</i>
------------------	--

Description

Produces a two-column tile heatmap where the x-axis shows YLD Associated and YLD Attributable, the y-axis shows organism groups (or individual pathogens), and each cell’s fill reflects the YLD burden per 1 000 admissions.

Usage

```
plot_yld_heatmap(  
  data,  
  n_admissions = NULL,  
  pathogen_col = "pathogen",  
  assoc_col = "YLD_associated",  
  attr_col = "YLD_attributable",  
  group_col = NULL,  
  syndrome_col = NULL,  
  syndrome_name = NULL,  
  show_values = TRUE,  
  value_digits = 2,  
  palette = "Blues",  
  coord_fixed = TRUE,  
  base_size = 14,  
  title = NULL,  
  subtitle = NULL  
)
```

Arguments

data	Data frame. Must contain the columns named by pathogen_col, assoc_col, and attr_col.
n_admissions	Numeric or NULL. Total admissions used as the denominator for per-1 000 normalisation. If NULL (default), values are plotted as supplied (assumed pre-normalised).
pathogen_col	Character. Column containing pathogen names. Default "pathogen".
assoc_col	Character. Column containing YLD associated values. Default "YLD_associated".
attr_col	Character. Column containing YLD attributable values. Default "YLD_attributable".
group_col	Character or NULL. Column containing organism group labels (e.g. "org_group"). If NULL (default) or not present in data, pathogen names are used on the y-axis.
syndrome_col	Character or NULL. Syndrome filter column. Default NULL.
syndrome_name	Character or NULL. Syndrome value to retain. Requires syndrome_col. Default NULL.
show_values	Logical. Print rounded values inside each cell. Default TRUE.
value_digits	Integer. Decimal places for cell labels. Default 2.
palette	Character. RColorBrewer palette for the fill scale. Default "Blues".
coord_fixed	Logical. Whether to apply ggplot2::coord_fixed() to produce square tiles. Default TRUE.
base_size	Numeric. Base font size. Default 14.
title	Character. Custom plot title. Auto-generated if NULL.
subtitle	Character. Custom subtitle. Auto-generated if NULL.

Details

The input data should contain one row per pathogen with YLD_associated and YLD_attributable columns – typically the output of daly_calc_yld_attributable() (which augments the data frame with both columns). If an organism-group column has been pre-joined (e.g. org_group), supply its name via group_col to aggregate cells to the group level.

Value

A ggplot object.

plot_yll_heatmap	<i>Heatmap of YLL per 1 000 Admissions by Resistance Class and Pathogen Group</i>
------------------	---

Description

Produces a tile heatmap where the x-axis shows resistance classes (dominant AMR profiles), the y-axis shows pathogen groups (or individual pathogens), and each cell's fill reflects the associated or attributable YLL burden per 1 000 admissions.

Usage

```
plot_yll_heatmap(
  data,
  type = c("associated", "attributable"),
  value_col,
  n_admissions = NULL,
  pathogen_col = "pathogen",
  class_col = "profile",
  group_col = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL,
  show_values = TRUE,
  value_digits = 2,
  palette = "Spectral",
  base_size = 14,
  title = NULL,
  subtitle = NULL
)
```

Arguments

<code>data</code>	Data frame. Must contain the columns named by <code>pathogen_col</code> , <code>class_col</code> , and <code>value_col</code> .
<code>type</code>	Character. "associated" (default) or "attributable". Used only for axis and title labels.
<code>value_col</code>	Character. Required. Column containing the YLL value to plot (e.g. "YLL_class" after distributing per-pathogen YLL across resistance classes, or "YLL_attributable_k" if your data already has one row per pathogen x class).
<code>n_admissions</code>	Numeric or NULL. Total admissions used as the denominator for per-1 000 normalisation. If NULL (default), values are plotted as supplied (assumed pre-normalised).
<code>pathogen_col</code>	Character. Column containing pathogen names. Default "pathogen".
<code>class_col</code>	Character. Column containing the resistance class / dominant profile. Default "profile".
<code>group_col</code>	Character or NULL. Column containing organism group labels (e.g. "org_group"). If NULL (default) or not present in data, pathogen names are used on the y-axis.
<code>syndrome_col</code>	Character or NULL. Syndrome filter column. Default NULL.
<code>syndrome_name</code>	Character or NULL. Syndrome value to retain. Requires <code>syndrome_col</code> . Default NULL.
<code>show_values</code>	Logical. Print rounded values inside each cell. Default TRUE.
<code>value_digits</code>	Integer. Decimal places for cell labels. Default 2.
<code>palette</code>	Character. RColorBrewer palette for the fill scale. Default "Spectral".
<code>base_size</code>	Numeric. Base font size. Default 14.
<code>title</code>	Character. Custom plot title. Auto-generated if NULL.
<code>subtitle</code>	Character. Custom subtitle. Auto-generated if NULL.

Details

The input data should be the `$by_class` slot (for associated) or the `$by_profile` slot (for attributable) returned by `daly_calc_yll_associated()` / `daly_calc_yll_attributable()`. Both slots contain columns `pathogen`, `profile`, and `YLL_Kdelta`.

If an organism-group column has been pre-joined onto data (e.g. `org_group`), supply its name via `group_col` and the function will aggregate to the group x class level. When `group_col` is `NULL` or absent, individual pathogen names are used on the y-axis.

Value

A ggplot object.

```
prep_apply_column_map  Apply Column Map to Rename Columns
```

Description

Renames data frame columns using a `standard_name = raw_name` map. Warns on unmapped columns; does not drop them.

Usage

```
prep_apply_column_map(data, column_map)
```

Arguments

<code>data</code>	Data frame.
<code>column_map</code>	Named character vector from <code>prep_build_column_map()</code> .

Value

Data frame with standard column names where mapping existed.

```
prep_assert_standard_names
      Assert Standard Names Are Present
```

Description

Confirms that a column mapping succeeded for critical standard columns. Stops on failure when `strict = TRUE`.

Usage

```
prep_assert_standard_names(data, required_standard_names, strict = TRUE)
```

Arguments

<code>data</code>	Data frame after column mapping.
<code>required_standard_names</code>	Character vector of standard column names that must be present.
<code>strict</code>	Logical. Stop on failure. Default TRUE.

Value

Invisibly returns data. Stops or warns on missing columns.

```
prep_assign_age_bins  Assign Age Bins
```

Description

Categorizes age into bins for stratification using GBD standard or custom break points. Supports ages stored as a single column in years, months, or days, as well as compound datasets where years, months, and days are in separate columns (e.g. neonatal records).

Usage

```
prep_assign_age_bins(
  data,
  age_col = "Age",
  bins = "GBD_standard",
  age_unit = "years",
  age_months_col = NULL,
  age_days_col = NULL,
  negative_age_strategy = c("fallback", "na"),
  fallback_years_col = "year",
  fallback_months_col = "months",
  fallback_days_col = "age_days",
  fallback_dob_col = "dob",
  fallback_admission_col = "admission_date"
)
```

Arguments

<code>data</code>	Data frame with an age column.
<code>age_col</code>	Character. Primary age column. Default "Age".
<code>bins</code>	Character or character vector. Preset name ("GBD_standard", "pediatric", "geriatric", "neonatal") or a custom vector of bin labels. Default "GBD_standard".
<code>age_unit</code>	Character. Unit of <code>age_col</code> : "years" (default), "months", or "days". The value is converted to decimal years before binning.

age_months_col	Character or NULL. Optional column holding the months component of age (0-11). Added to age_col after unit conversion.
age_days_col	Character or NULL. Optional column holding the days component of age (0-30). Added to age_col after unit conversion.
negative_age_strategy	Character. How to handle negative ages in age_col. "fallback" (default) tries alternate columns / DOB logic and falls back to 0 years when unresolved; "na" sets negative ages to NA.
fallback_years_col	Character or NULL. Optional years column used when age_col is negative. Default "year".
fallback_months_col	Character or NULL. Optional months column used when age_col is negative. Default "months".
fallback_days_col	Character or NULL. Optional days column used when age_col is negative. Default "age_days".
fallback_dob_col	Character or NULL. Optional DOB column used with fallback_admission_col to derive age when age_col is negative. Default "dob".
fallback_admission_col	Character or NULL. Optional admission/reference date column used with fallback_dob_col. Default "admission_date".

Details

When age_months_col or age_days_col are supplied they are treated as the fractional remainder on top of age_col (e.g. age_years = 0, age_months = 3, age_days = 5 -> 0.27 years). Missing values in the component columns are treated as zero so that a partially-recorded age is still binnable.

Value

Data frame with resolved numeric age in Age_years, Age_bin factor column, and Age_resolved (decimal years rounded to one decimal place, derived from whatever year / month / day components were available — e.g. 45 months with no year value yields Age_resolved = 3.8) added.

```
prep_assign_facility_type
```

Assign a facility type / sector column from a user-supplied mapping

Description

Adds a column that labels each row's facility with a category defined by the caller (for example "Government" vs "Private", "Urban" vs "Rural", or a region). The function is deliberately generic: it contains no hardcoded facility names, so any dataset can be classified by passing an appropriate mapping.

Usage

```
prep_assign_facility_type(
  data,
  facility_col = "center_name",
  mapping,
  new_col = "facility_type",
  default = NA_character_,
  case_insensitive = TRUE
)
```

Arguments

<code>data</code>	A data frame.
<code>facility_col</code>	Name of the column holding facility names. Default "center_name".
<code>mapping</code>	A named character vector where names are facility values and values are the type/sector to assign, e.g. <code>c(AFMC = "Government", SGRH = "Private")</code> .
<code>new_col</code>	Name of the column to create. Default "facility_type".
<code>default</code>	Value assigned to facilities not found in mapping. Default <code>NA_character_</code> .
<code>case_insensitive</code>	Logical. If TRUE (default), matching ignores case and surrounding whitespace, so "AFMC", "afmc" and " AFMC " all match.

Value

data with new_col added.

Examples

```
df <- data.frame(center_name = c("AFMC", "SGRH", "Unknown"))
m <- c(AFMC = "Government", SGRH = "Private")
prep_assign_facility_type(df, mapping = m)
```

```
prep_assign_organism_group
```

Assign Organism Group

Description

Assign Organism Group

Usage

```
prep_assign_organism_group(data, organism_col = "organism_normalized")
```

Arguments

`data` Data frame.

`organism_col` Character. Normalized organism column. Default "organism_normalized".

Value

Data frame with `org_group` column added.

```
prep_assign_patient_syndrome
```

Assign One Syndrome Per Patient or Event

Description

Selects the highest-priority syndrome per patient using the infectious syndrome hierarchy. Matching is case-insensitive (uses normalized names).

Usage

```
prep_assign_patient_syndrome(
  data,
  patient_col = "patient_id",
  syndrome_col = "syndrome",
  score_col = "icd_score",
  hierarchy_ref = NULL,
  hierarchy_syndrome_col = "infectious_syndrome",
  hierarchy_rank_col = "rank",
  keep_all_candidates = TRUE,
  collapse_unspecified_respiratory = FALSE,
  keep_only_burden_syndromes = FALSE
)
```

Arguments

`data` Data frame with one row per syndrome candidate.

`patient_col` Character. Patient or event ID column. Default "patient_id".

`syndrome_col` Character. Syndrome column. Default "syndrome".

`score_col` Character or NULL. ICD match score column for secondary priority. Default "icd_score".

`hierarchy_ref` Data frame or NULL. Syndrome hierarchy (from `load_syndrome_hierarchy()`). If NULL, loads automatically.

`hierarchy_syndrome_col` Character. Syndrome column in hierarchy. Default "infectious_syndrome".

`hierarchy_rank_col` Character. Rank column in hierarchy. Default "rank".

keep_all_candidates

Logical. TRUE -> return all rows with syndrome_selected flag; FALSE -> return one row per patient_col (selected only). Default TRUE.

collapse_unspecified_respiratory

Logical. Map "Other unspecified respiratory site infections" -> "Lower respiratory infections" before selection. Default FALSE.

keep_only_burden_syndromes

Logical. Retain only syndromes where contributes_to_amr_burden == TRUE. Default FALSE.

Details

Priority order per patient:

1. Hierarchy rank (lower = higher priority)
2. ICD match score (higher = higher priority), when score_col present
3. Alphabetical tie-break

This function absorbs the logic of the former infer_patient_syndrome_long(): use collapse_unspecified_respiratory == TRUE and keep_only_burden_syndromes = TRUE to replicate that behaviour.

Value

Data frame. When keep_all_candidates = FALSE, includes contributes_to_amr_burden column from the hierarchy.

prep_attrition_flow	<i>Track Attrition Through the Pipeline</i>
---------------------	---

Description

Records a stage entry in an attrition flow table. Call once per pipeline stage to build up a cumulative record of how many rows and patients are retained after each step.

Usage

```
prep_attrition_flow(
  flow,
  data,
  stage_name,
  reason = "",
  patient_col = "patient_id",
  event_col = "event_id"
)
```

Arguments

flow	Data frame of previous attrition stages (or NULL to start).
data	Current data frame at this stage.
stage_name	Character. Short label for this stage.
reason	Character. Description of what changed at this stage.
patient_col	Character. Patient ID column. Default "patient_id".
event_col	Character. Event ID column (optional). Default "event_id".

Details

Usage pattern:

```
flow <- NULL
flow <- prep_attrition_flow(flow, data_raw, "raw_intake", "All raw records")
flow <- prep_attrition_flow(flow, data_dated, "dates_parsed", "After date coercion")
flow <- prep_attrition_flow(flow, data_ready, "analysis_ready", "After all filters")
print(flow)
```

Value

Updated attrition flow data frame.

prep_build_column_map *Build and Validate a Column Map Against a Dataset*

Description

Validates a user-supplied column map against the actual columns present in a data frame. Reports which mappings are present, which are absent, and which data columns are not covered.

Usage

```
prep_build_column_map(data, column_map = NULL, custom_map = NULL)
```

Arguments

data	Data frame with raw column names.
column_map	Named character vector: standard_name = raw_name. Pass a dataset-specific map from your analysis layer. If NULL, generic alias detection is attempted.
custom_map	Named character vector. Additional raw -> standard overrides applied on top of column_map. Optional.

Details

Dataset-specific maps (ICMR stewardship, surveillance, AIIMS, etc.) should be defined in the analysis layer and passed in via column_map. If no map is supplied, falls back to auto-detection using default_column_mappings aliases.

Value

Named character vector with only mappings whose raw names exist in data.

prep_build_fatal_cohort

Build Fatal Cohort

Description

Extracts records for patients who died, for use in YLL calculation.

Usage

```
prep_build_fatal_cohort(  
  data,  
  outcome_col = "final_outcome",  
  died_value = "Died"  
)
```

Arguments

data	Analysis-ready data frame.
outcome_col	Character. Final outcome column. Default "final_outcome".
died_value	Character. Value indicating death. Default "Died".

Value

Data frame with only fatal episodes.

prep_build_nonfatal_cohort

Build Non-Fatal Cohort

Description

Extracts records for patients who survived, for use in YLD calculation.

Usage

```
prep_build_nonfatal_cohort(  
  data,  
  outcome_col = "final_outcome",  
  survived_value = "Survived"  
)
```

Arguments

data Analysis-ready data frame.
 outcome_col Character. Final outcome column. Default "final_outcome".
 survived_value Character. Value indicating survival. Default "Survived".

Value

Data frame with only non-fatal episodes.

prep_check_columns	<i>Check required columns exist and report types</i>
--------------------	--

Description

Runs before any table is merged. Reports missing columns, type mismatches, and blank/duplicate column names. Stops on hard failures; warns on soft ones.

Usage

```

prep_check_columns(
  data,
  required = character(),
  expected_types = character(),
  table_label = "table",
  stop_on_missing = TRUE
)

```

Arguments

data Data frame to inspect.
 required Character vector of column names that must be present.
 expected_types Named character vector mapping column names to expected R classes (e.g. `c(PatientInformation_id = "character")`). Only checked when the column exists.
 table_label Character. Label used in messages (e.g. "patient_KGMU").
 stop_on_missing Logical. Stop if required columns are absent (default TRUE).

Value

Invisibly returns a tibble summarising every checked column.

```
prep_check_keys          Check join key quality
```

Description

Reports missing, blank, duplicate, and placeholder keys in a column before a merge. Run this on both sides of every planned join.

Usage

```
prep_check_keys(
  data,
  key_col,
  table_label = "table",
  warn_missing_pct = 5,
  admission_col = "date_of_admission",
  culture_col = "date_of_culture"
)
```

Arguments

<code>data</code>	Data frame.
<code>key_col</code>	Character. Name of the key column.
<code>table_label</code>	Character. Label used in messages.
<code>warn_missing_pct</code>	Numeric. Warn when proportion of missing keys exceeds this threshold (0-100). Default 5.
<code>admission_col</code>	Character. Admission date column used for the optional paired missingness check. Default "date_of_admission".
<code>culture_col</code>	Character. Culture date column used for the optional paired missingness check. Default "date_of_culture".

Value

Invisibly returns a one-row summary tibble.

```
prep_check_organism_ast_consistency
          Check Organism-AST Consistency
```

Description

Flags impossible/implausible AST results where a drug is inappropriate for the organism type (e.g., a Gram-positive organism tested against a Gram-negative-only drug).

Usage

```
prep_check_organism_ast_consistency(
  data,
  organism_col = "organism_normalized",
  antibiotic_col = "antibiotic_normalized"
)
```

Arguments

`data` Data frame with organism and antibiotic columns.

`organism_col` Character. Normalized organism column. Default "organism_normalized".

`antibiotic_col` Character. Normalized antibiotic column. Default "antibiotic_normalized".

Details

Uses inst/extdata/Organism_AST_drugs.csv which maps organism groups to expected antibiotic panels.

Value

Data frame with `is_ast_inconsistent` logical column added.

```
prep_classify_antibiotic_class
```

Classify Antibiotic to WHO Class

Description

Maps antibiotic names to WHO antibiotic classes.

Usage

```
prep_classify_antibiotic_class(
  data,
  antibiotic_col = "antibiotic_normalized",
  who_table = NULL
)
```

Arguments

`data` Data frame.

`antibiotic_col` Character. Normalized antibiotic column. Default "antibiotic_normalized".

`who_table` Data frame. WHO classification table. If NULL uses built-in.

Value

Data frame with `antibiotic_class` column added.

prep_classify_aware	<i>Classify AWaRe Category</i>
---------------------	--------------------------------

Description

Assigns WHO AWaRe (Access, Watch, Reserve) categories to antibiotics.

Usage

```
prep_classify_aware(  
  data,  
  antibiotic_col = "antibiotic_normalized",  
  who_table = NULL  
)
```

Arguments

data	Data frame.
antibiotic_col	Character. Antibiotic column. Default "antibiotic_normalized".
who_table	Data frame. WHO AWaRe table. If NULL, uses built-in.

Value

Data frame with aware_category column added.

prep_classify_mdr_xdr	<i>Classify MDR and XDR</i>
-----------------------	-----------------------------

Description

Classifies isolates as MDR (Multidrug Resistant) and XDR (Extensively Drug Resistant) in a single pass using Magiorakos 2012 criteria.

Usage

```
prep_classify_mdr_xdr(  
  data,  
  definition = "Magiorakos",  
  organism_group_col = "org_group"  
)
```

Arguments

`data` Data frame with class-level resistance data.

`definition` Character. Classification criteria. Default "Magiorakos".

`organism_group_col` Character. Organism group column for pathogen-specific thresholds. Default "org_group".

Details

MDR: resistant to at least one agent in three or more antimicrobial categories. XDR: susceptible to at most two antimicrobial categories.

Requires `class_result_event` column (run `prep_collapse_class_level()` first, then rename `class_resistance` to `class_result_event`).

Value

Data frame with `mdr`, `mdr_confidence`, `mdr_method`, `n_resistant_categories`, `resistant_categories`, `xdr`, `xdr_confidence`, and `xdr_method` columns added.

References

Magiorakos AP et al. Clin Microbiol Infect. 2012;18(3):268-281.

```
prep_classify_mortality
```

Classify Infection-Related Mortality

Description

Determines if death was related to infection using date window logic. Implements proxy logic when dates are missing.

Usage

```
prep_classify_mortality(
  data,
  outcome_col = "final_outcome",
  event_date_col = "date_of_culture",
  outcome_date_col = "date_of_final_outcome",
  window = 14
)
```

Arguments

data	Data frame
outcome_col	Character. Outcome column. Default "final_outcome".
event_date_col	Character. Event/culture date. Default "date_of_culture".
outcome_date_col	Character. Outcome date. Default "date_of_final_outcome".
window	Numeric. Days after event to classify death as infection-related. Default 14.

Value

Data frame with mortality_infection, mortality_method, mortality_confidence columns

```
prep_classify_readmission
```

Classify Readmission Type

Description

Applies readmission classification rules to a data frame that already has a readmission_class column, re-standardizing the values.

Usage

```
prep_classify_readmission(data, readmission_col = "readmission_class")
```

Arguments

data	Data frame.
readmission_col	Character. Readmission column to standardize. Default "readmission_class".

Details

Useful when data arrives with non-standard labels that need mapping to the controlled vocabulary used by prep_flag_readmission().

Value

Data frame with standardized readmission classification.

`prep_clean_ast_values` *Clean Antibiotic Susceptibility Values*

Description

Extracts clean S/I/R values from messy antibiotic result columns. Handles common patterns like "S (HIGH LEVEL)", "R Escherichia coli", etc.

Usage

```
prep_clean_ast_values(data, value_col = "antibiotic_value", strict = FALSE)
```

Arguments

<code>data</code>	Data frame with antibiotic susceptibility data.
<code>value_col</code>	Character. Column containing susceptibility values. Default "antibiotic_value".
<code>strict</code>	Logical. If TRUE, only accept S/I/R values. If FALSE, attempt to parse from messy strings. Default FALSE.

Value

Data frame with cleaned antibiotic_value column.

`prep_clean_optional_columns`
Clean Optional Columns

Description

Cleans and standardizes all optional columns in a single pass. Handles value normalization, whitespace trimming, and consistency checks.

Usage

```
prep_clean_optional_columns(data, optional_cols = NULL)
```

Arguments

<code>data</code>	Data frame.
<code>optional_cols</code>	Character vector. Optional column names to groom. If NULL, grooms all known optional columns. Default NULL.

Value

Data frame with groomed optional columns.

```
prep_coerce_dates      Detect and convert all date-like columns in a table
```

Description

Auto-detects columns with date/time-like names (or uses a supplied list), then calls `prep_parse_date_column()` on each.

Usage

```
prep_coerce_dates(data, cols = NULL, table_label = "table")
```

Arguments

<code>data</code>	Data frame.
<code>cols</code>	Character vector of column names to convert. When <code>NULL</code> , columns whose names match a date/time pattern are detected automatically.
<code>table_label</code>	Character. Label used in messages.

Value

Data frame with date columns converted to Date.

```
prep_collapse_class_level
      Collapse to Class Level
```

Description

Aggregates resistance at antibiotic class level instead of individual drugs. Uses "any R in class -> class R" logic.

Usage

```
prep_collapse_class_level(
  data,
  event_col = "event_id",
  organism_col = "organism_normalized",
  class_col = "antibiotic_class",
  susceptibility_col = "antibiotic_value",
  extra_cols = NULL
)
```

Arguments

<code>data</code>	Data frame with antibiotic class information
<code>event_col</code>	Character. Event ID column. Default "event_id".
<code>organism_col</code>	Character. Organism column. Default "organism_normalized".
<code>class_col</code>	Character. Antibiotic class column. Default "antibiotic_class".
<code>susceptibility_col</code>	Character. Susceptibility column. Default "antibiotic_value".
<code>extra_cols</code>	Character vector or NULL. Additional columns to carry through the aggregation. Default NULL.

Value

Aggregated data frame (one row per event-organism-class)

Examples

```
## Not run:
class_level <- prep_collapse_class_level(data)

## End(Not run)
```

```
prep_compute_poly_weights
```

Compute Polymicrobial Weights

Description

Calculates proportional weights for polymicrobial infections. Monomicrobial patients always receive weight = 1.0. For polymicrobial patients, weight is computed per organism within each episode using one of three methods.

Usage

```
prep_compute_poly_weights(
  data,
  episode_col = "episode_id",
  organism_col = "organism_normalized",
  polymicrobial_col = "is_polymicrobial",
  method = "monomicrobial_proportion",
  weight_map = NULL,
  facility_col = NULL,
  facility_name = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)
```

```
compute_polymicrobial_weight(  
  data,  
  episode_col = "episode_id",  
  organism_col = "organism_normalized",  
  polymicrobial_col = "is_polymicrobial",  
  method = "monomicrobial_proportion",  
  weight_map = NULL,  
  facility_col = NULL,  
  facility_name = NULL,  
  syndrome_col = NULL,  
  syndrome_name = NULL  
)
```

Arguments

data	Data frame with episode_id, is_polymicrobial (0/1), and organism columns (output of prep_flag_polymicrobial()).
episode_col	Character. Episode ID column. Default "episode_id".
organism_col	Character. Organism column. Default "organism_normalized".
polymicrobial_col	Character. Polymicrobial flag column (0/1). Default "is_polymicrobial".
method	Character. One of "monomicrobial_proportion" (default), "equal", or "manual".
weight_map	Named numeric vector. Custom organism weights when method = "manual".
facility_col	Character or NULL. Scope monomicrobial reference pool per facility.
facility_name	Character or NULL. Filter to this facility.
syndrome_col	Character or NULL. Scope reference pool per syndrome.
syndrome_name	Character or NULL. Filter to this syndrome.

Value

Data frame with poly_weight, weight_method, and weight_confidence columns.

prep_create_event_ids	Create Event IDs from Patient-Level Data
-----------------------	--

Description

Converts patient-level isolate data to event-level data by grouping repeat cultures from the same infection episode.

Usage

```

prep_create_event_ids(
  data,
  patient_col = "patient_id",
  date_col = "date_of_culture",
  organism_col = "organism_normalized",
  specimen_col = "specimen_type",
  antibiotic_col = "antibiotic_name",
  value_col = "antibiotic_value",
  gap_days = 14,
  culture_col = "id_organisminfo",
  admission_col = "date_of_admission",
  keep_event_culture = c("all", "closest_to_admission")
)

```

Arguments

<code>data</code>	Data frame (long format – one row per antibiotic test).
<code>patient_col</code>	Patient ID column. Default "patient_id".
<code>date_col</code>	Culture date column. Default "date_of_culture".
<code>organism_col</code>	Organism column. Default "organism_normalized".
<code>specimen_col</code>	Specimen type column. Default "specimen_type".
<code>antibiotic_col</code>	Antibiotic name column. Default "antibiotic_name".
<code>value_col</code>	Susceptibility result column (S/I/R). Default "antibiotic_value".
<code>gap_days</code>	Days threshold: gap > gap_days triggers a new event. Default 14.
<code>culture_col</code>	Culture/isolate identifier column. When present, same-day cultures from the same site and organism are evaluated separately before duplicate culture removal. Default "id_organisminfo".
<code>admission_col</code>	Admission date column. Used when keep_event_culture is "closest_to_admission". Default "date_of_admission".
<code>keep_event_culture</code>	Character. "all" keeps all culture rows after assigning event IDs. "closest_to_admission" keeps all antibiotic rows from only the culture date closest to admission within each event/organism/site. Default "all".

Details

Event classification rules (in order of precedence):

| Scenario | Result | |-----|-----| | Different body sites (any date) | Separate events | | Same site, same day, same organism, same ABG | One event | | Same site, same day, same organism, diff ABG | Separate events | | Same site, same organism, within gap_days, same ABG | Same event | | Same site, same organism, within gap_days, ABG changed | New event | | Same site, same organism, after > gap_days | New event | | Same site, same day, different organisms | Separate events |

Event IDs are numbered GLOBALLY per patient in chronological order.

Value

Original data frame with event_id column added.

prep_create_resistance_profile
<i>Create Resistance Profile</i>

Description

Generates a resistance profile string for each event summarizing resistance patterns. Useful for identifying common resistance phenotypes.

Usage

```
prep_create_resistance_profile(  
  data,  
  event_col = "event_id",  
  antibiotic_col = "antibiotic_normalized",  
  susceptibility_col = "antibiotic_value",  
  format = "resistant_list",  
  class_col = "antibiotic_class"  
)
```

Arguments

data	Data frame with resistance data
event_col	Character. Event ID column. Default "event_id".
antibiotic_col	Character. Antibiotic column. Default "antibiotic_normalized".
susceptibility_col	Character. Susceptibility column. Default "antibiotic_value".
format	Character. Output format: - "resistant_list": List resistant drugs only (default) - "full_pattern": Full S/R pattern string - "class_summary": Resistant classes only
class_col	Character. Class column (required for format = "class_summary"). Default "antibiotic_class".

Value

Data frame with resistance_profile column added

Examples

```
## Not run:
# List resistant drugs
data_with_profile <- prep_create_resistance_profile(data)

# Full S/R pattern
data_with_profile <- prep_create_resistance_profile(
  data,
  format = "full_pattern"
)

# Resistant classes only
data_with_profile <- prep_create_resistance_profile(
  data,
  format = "class_summary"
)

## End(Not run)
```

```
prep_create_wide_ast_matrix
      Create Wide Format AST Matrix
```

Description

Converts long format (one row per organism-antibiotic) to wide format (one row per event with antibiotic columns). Useful for analysis and machine learning applications.

Usage

```
prep_create_wide_ast_matrix(
  data,
  event_col = "event_id",
  antibiotic_col = "antibiotic_normalized",
  susceptibility_col = "antibiotic_value",
  prefix = "abx_",
  keep_cols = c("patient_id", "organism_normalized", "date_of_culture")
)
```

Arguments

<code>data</code>	Data frame in long format
<code>event_col</code>	Character. Event ID column. Default "event_id".
<code>antibiotic_col</code>	Character. Antibiotic/class column to pivot. Default "antibiotic_normalized".
<code>susceptibility_col</code>	Character. Susceptibility column. Default "antibiotic_value".
<code>prefix</code>	Character. Prefix for pivoted columns. Default "abx_".
<code>keep_cols</code>	Character vector. Additional columns to keep from original data. Default c("patient_id", "organism_normalized", "date_of_culture").

Value

Wide format data frame (one row per event)

Examples

```
## Not run:
# Basic wide format
wide_data <- prep_create_wide_ast_matrix(data)

# Class-level wide format
wide_data <- prep_create_wide_ast_matrix(
  data,
  antibiotic_col = "antibiotic_class",
  prefix = "class_"
)

## End(Not run)
```

```
prep_decode_antibiotic_code
```

Decode Antibiotic Short Codes to Full Names

Description

Converts antibiotic short codes (e.g. AMK, AMP, AMPSUL) to full antibiotic names using a caller-supplied code map CSV.

Usage

```
prep_decode_antibiotic_code(
  data,
  code_col = "antibiotic_name_raw",
  map_path = NULL,
  output_col = "antibiotic_name_std"
)
```

Arguments

data	Data frame with a column of antibiotic codes.
code_col	Character. Column containing antibiotic codes. Default "antibiotic_name_raw".
map_path	Character. Path to the code map CSV file. Must have columns code and antibiotic_name_full. If NULL, falls back to inst/extdata/antibiotic_code_map.csv if it exists.
output_col	Character. Name of the output column for decoded names. Default "antibiotic_name_std".

Details

The code map must have columns `code` and `antibiotic_name_full`. This function is dataset-agnostic: the code map and the column containing the codes are supplied by the caller (typically from the analysis layer).

Value

Data frame with `output_col` column added.

```
prep_deduplicate_ast  Handle Duplicate AST Results
```

Description

Deduplicates AST records in two sequential steps:

1. **Exact-duplicate removal:** drops rows that are fully identical across all five key columns (patient + organism + antibiotic + date + value). These are true redundant records with no ambiguity.
2. **Conflict resolution:** handles the remaining cases where the same patient + organism + antibiotic + date group has *different* values.

Usage

```
prep_deduplicate_ast(
  data,
  mode = c("detect", "remove"),
  strategy = c("resistant_wins", "susceptible_wins", "first"),
  patient_col = "patient_id",
  organism_col = "organism_normalized",
  antibiotic_col = "antibiotic_normalized",
  date_col = "culture_date",
  ast_col = "ast_value_harmonized"
)
```

Arguments

<code>data</code>	Data frame with AST data in long format.
<code>mode</code>	Character. "detect" (flag only) or "remove" (flag then resolve). Default "detect".
<code>strategy</code>	Character. Resolution strategy used only when <code>mode = "remove"</code> . One of "resistant_wins" (default), "susceptible_wins", or "first".
<code>patient_col</code>	Character. Patient ID column. Default "patient_id".
<code>organism_col</code>	Character. Organism column. Default "organism_normalized".
<code>antibiotic_col</code>	Character. Antibiotic column. Default "antibiotic_normalized".
<code>date_col</code>	Character. Culture date column. Default "culture_date".
<code>ast_col</code>	Character. Harmonized AST value column. Default "ast_value_harmonized".

Details

- "detect" After exact-dedup, flags conflicting rows with `is_ast_duplicate = TRUE` and prints a QC summary. Returns the data with the flag column so you can inspect before deciding how to resolve.
- "remove" Runs both steps: removes exact duplicates, then applies strategy to keep one row per key combination for any remaining conflicts.

Value

- `mode = "detect"`: original data with `is_ast_duplicate` logical column added.
- `mode = "remove"`: data with conflicts resolved (one row per key) and no flag column.

prep_deduplicate_events
<i>Deduplicate Events</i>

Description

Removes duplicate rows within groups. Two modes:

Usage

```
prep_deduplicate_events(  
  data,  
  event_col = "event_id",  
  organism_col = "organism_normalized",  
  antibiotic_col = "antibiotic_normalized",  
  key_cols = NULL,  
  keep = "first"  
)
```

Arguments

data	Data frame.
event_col	Character. Event ID column (event-aware mode). Default "event_id".
organism_col	Character. Organism column (event-aware mode). Default "organism_normalized".
antibiotic_col	Character. Antibiotic column (event-aware mode). Default "antibiotic_normalized".
key_cols	Character vector. When supplied, switches to generic mode and uses these columns for duplicate detection. NULL uses event-aware mode. Default NULL.
keep	Character. "first", "last", or "none" (drop all duplicates). Default "first".

Details

Event-aware (default) Groups by event_col + organism_col + antibiotic_col and keeps first/last antibiotic test per group. Requires event_col to be present.

Generic (key_cols supplied) Groups by key_cols and drops duplicates. When keep = "none", both copies of every duplicate are removed. Replaces the former remove_duplicate_rows() helper.

Value

Deduplicated data frame.

```
prep_derive_dob_from_components
```

Derive Date of Birth from Age Components

Description

Stewardship-specific helper: DOB is sometimes stored as separate year/month/day components or as a decimal age in years. This function reconstructs an approximate DOB or derives age directly when DOB is absent.

Usage

```
prep_derive_dob_from_components(
  data,
  dob_year_col = "dob_year",
  dob_month_col = "dob_month",
  dob_day_col = "dob_day",
  age_years_col = "age_years",
  reference_date_col = "admission_date",
  dob_output_col = "dob"
)
```

Arguments

data	Data frame.
dob_year_col	Character. Year component column. Default "dob_year".
dob_month_col	Character. Month component column. Default "dob_month".
dob_day_col	Character. Day component column. Default "dob_day".
age_years_col	Character. Decimal age in years. Default "age_years".
reference_date_col	Character. Reference date for age-based DOB estimation. Default "admission_date".
dob_output_col	Character. Output DOB column. Default "dob".

Details

Strategies (applied in order):

1. If dob_year_col, dob_month_col, and dob_day_col are present, assemble into a Date.
2. If age_years_col is present and a reference date exists, estimate DOB as $\text{reference_date} - \text{age_years} * 365.25$.

Value

Data frame with dob column added/populated.

prep_derive_hai_cai	<i>Derive HAI/CAI Infection Type</i>
---------------------	--------------------------------------

Description

Infers Community-Acquired (CAI) vs Hospital-Acquired (HAI) infection using the admission-to-culture date gap (default cutoff: 2 days / 48 hours).

Usage

```
prep_derive_hai_cai(
  data,
  infection_type_col = "infection_type",
  admission_col = "date_of_admission",
  culture_col = "date_of_culture",
  hai_cutoff = 2,
  overwrite = FALSE
)
```

Arguments

data	Data frame.
infection_type_col	Character. Infection type column. Default "infection_type".
admission_col	Character. Admission date column. Default "date_of_admission".
culture_col	Character. Culture date column. Default "date_of_culture".
hai_cutoff	Numeric. Days after admission to classify as HAI. Default 2.
overwrite	Logical. Recalculate even when already present. Default FALSE.

Value

Data frame with infection_type enriched.

prep_derive_icu_flag *Derive ICU Flag*

Description

Sets icu_flag = TRUE where ward/department information indicates the patient was in an ICU at the time of culture.

Usage

```
prep_derive_icu_flag(  
  data,  
  ward_col = "ward_icu",  
  dept_col = "hospital_department"  
)
```

Arguments

data	Data frame.
ward_col	Character. Ward/unit column. Default "ward_icu".
dept_col	Character. Department column. Default "hospital_department".

Value

Data frame with icu_flag logical column added.

prep_derive_los_from_dates
 Derive Length of Stay from Date Columns

Description

Computes LOS when missing (or always when overwrite = TRUE). Adds los_method and los_confidence audit columns. For AIIMS data, uses unit_admission_date + unit_duration_days as a fallback when primary date columns are absent.

Usage

```
prep_derive_los_from_dates(  
  data,  
  admission_col = "admission_date",  
  outcome_col = "outcome_date",  
  los_col = "los_days",  
  unit_admission_col = "unit_admission_date",  
  unit_duration_col = "unit_duration_days",  
  overwrite = FALSE  
)
```

Arguments

data	Data frame.
admission_col	Character. Admission date column. Default "admission_date".
outcome_col	Character. Outcome/discharge date column. Default "outcome_date".
los_col	Character. Target LOS column to populate. Default "los_days".
unit_admission_col	Character. AIIMS unit admission date fallback. Default "unit_admission_date".
unit_duration_col	Character. AIIMS unit duration fallback. Default "unit_duration_days".
overwrite	Logical. Recalculate even when los_col is already present. Default FALSE.

Details

- Priority order:
1. outcome_col - admission_col
 2. unit_admission_date + unit_duration_days (AIIMS fallback)
 3. Already-present los_col value retained as-is

Value

Data frame with los_col, los_method, and los_confidence populated where possible.

prep_detect_schema_drift
<i>Detect Schema Drift Across Centres</i>

Description

Compares column sets across a list of centre data frames. Flags columns present in some centres but not others.

Usage

```
prep_detect_schema_drift(data_list, reference_centre = NULL)
```

Arguments

data_list	Named list of data frames (one per centre).
reference_centre	Character or NULL. Name of the centre to treat as the reference schema. If NULL, the union of all columns is used.

Value

Tibble: column | present_in (comma-separated centres) | missing_from | n_centres_present

prep_diagnosis_text	<i>Prepare Diagnosis Text</i>
---------------------	-------------------------------

Description

Combines a primary diagnosis column with an optional fallback column, trims whitespace, and collapses repeated separators. Optionally appends the organism name to the text (as the notebook does for context-aware embedding).

Usage

```
prep_diagnosis_text(  
  data,  
  diagnosis_col,  
  fallback_col = NULL,  
  output_col = "diagnosis_text",  
  include_organism = FALSE,  
  organism_col = "organism_normalized",  
  keep_original = TRUE  
)
```

Arguments

data	Data frame.
diagnosis_col	Character. Primary diagnosis column.
fallback_col	Character or NULL. Used when diagnosis_col is NA. Default NULL.
output_col	Character. Name for the output text column. Default "diagnosis_text".
include_organism	Logical. Append organism name to the text. Default FALSE.
organism_col	Character. Organism column. Used only when include_organism = TRUE. Default "organism_normalized".
keep_original	Logical. Keep the original diagnosis columns. Default TRUE.

Value

Data frame with output_col added.

prep_extract_genus	<i>Extract Genus from Organism Name</i>
--------------------	---

Description

Extract Genus from Organism Name

Usage

```
prep_extract_genus(data, organism_col = "organism_normalized")
```

Arguments

- data Data frame.
- organism_col Character. Organism column. Default "organism_normalized".

Value

Data frame with org_genus column added.

prep_extract_species	<i>Extract Species from Organism Name</i>
----------------------	---

Description

Extract Species from Organism Name

Usage

```
prep_extract_species(data, organism_col = "organism_normalized")
```

Arguments

- data Data frame.
- organism_col Character. Organism column. Default "organism_normalized".

Value

Data frame with org_species column added.

```
prep_fill_age
```

Fill Missing Age Values

Description

Fills missing Age by computing from DOB and a reference date. Tracks derivation method and confidence for each record.

Usage

```
prep_fill_age(  
  data,  
  age_col = "Age",  
  dob_col = "DOB",  
  date_col = "date_of_culture",  
  overwrite = FALSE  
)
```

Arguments

data	Data frame.
age_col	Character. Age column. Default "Age".
dob_col	Character. Date of birth column. Default "DOB".
date_col	Character. Reference date column. Default "date_of_culture".
overwrite	Logical. Recalculate even when Age is present. Default FALSE.

Value

Data frame with Age enriched plus age_method and age_confidence columns.

```
prep_filter_analysis_ready
```

Filter Analysis-Ready Records

Description

Keeps records that meet the full burden-analysis readiness criteria:

- Patient ID present
- Culture date present
- Organism name present and standardized
- Antibiotic name present and standardized
- Harmonized AST value in S/I/R
- Not flagged as a contaminant (if column present)

Usage

```
prep_filter_analysis_ready(  
  data,  
  patient_col = "patient_id",  
  culture_date_col = "culture_date",  
  organism_col = "organism_name",  
  antibiotic_col = "antibiotic_name",  
  ast_col = "ast_value_harmonized",  
  exclude_contaminants = TRUE,  
  contaminant_col = "contaminant_flag"  
)
```

Arguments

- data Data frame after preprocessing.
- patient_col Character. Patient ID column. Default "patient_id".
- culture_date_col Character. Culture date column. Default "culture_date".
- organism_col Character. Organism column. Default "organism_name".
- antibiotic_col Character. Antibiotic column. Default "antibiotic_name".
- ast_col Character. Harmonized AST column. Default "ast_value_harmonized".
- exclude_contaminants Logical. Exclude rows flagged as contaminants. Default TRUE.
- contaminant_col Character. Contaminant flag column. Default "contaminant_flag".

Value

Filtered data frame.

prep_filter_fungal	<i>Filter Fungal Organisms</i>
--------------------	--------------------------------

Description

Removes rows where the organism belongs to a fungal group. Logs the count of rows removed.

Usage

```
prep_filter_fungal(  
  data,  
  group_col = "organism_group",  
  organism_col = "organism_normalized"  
)
```

Arguments

data	Data frame.
group_col	Character. Organism group column. Default "organism_group".
organism_col	Character. Normalized organism column used as fallback when group_col is absent. Default "organism_normalized".

Value

Data frame with fungal rows removed.

prep_filter_minimally_usable
Filter Minimally Usable Records

Description

Keeps records that have the bare minimum fields needed for any analysis: a patient identifier, a culture date, an organism name, and at least one non-NA AST result. This is a looser filter than prep_filter_analysis_ready().

Usage

```
prep_filter_minimally_usable(  
  data,  
  patient_col = "patient_id",  
  culture_date_col = "culture_date",  
  organism_col = "organism_name",  
  ast_col = "ast_value_harmonized"  
)
```

Arguments

data	Data frame after preprocessing.
patient_col	Character. Patient ID column. Default "patient_id".
culture_date_col	Character. Culture date column. Default "culture_date".
organism_col	Character. Organism column. Default "organism_name".
ast_col	Character. Harmonized AST column. Default "ast_value_harmonized".

Value

Filtered data frame with only minimally usable rows.

prep_flag_contaminants
<i>Flag Contaminant Organisms</i>

Description

Identifies likely contaminant organisms using multi-path logic (provided flags, device-based context, or heuristic organism+specimen matching).

Usage

```
prep_flag_contaminants(  
  data,  
  method = "auto",  
  context = c("general", "icu_device"),  
  organism_col = "organism_normalized",  
  specimen_col = "specimen_type"  
)
```

Arguments

data	Data frame.
method	Character. "auto" (try all methods), "device_based", "heuristic", or "provided". Default "auto".
context	Character. "general" or "icu_device". Default "general".
organism_col	Character. Normalized organism column. Default "organism_normalized".
specimen_col	Character. Specimen type column. Default "specimen_type".

Details

When context = "icu_device" (AIIMS ICU BSI data), CoNS in blood with a central line present AND line in place > 2 days are classified as CLABSI pathogens rather than contaminants.

Value

Data frame with is_contaminant, contaminant_confidence, and contaminant_method columns.

```
prep_flag_hai_inferred
```

Flag HAI Inferred vs Observed

Description

Adds an `infection_type_src` column indicating whether the infection type was present in the raw data ("observed"), derived from dates ("inferred"), or could not be determined ("unknown").

Usage

```
prep_flag_hai_inferred(
  data,
  infection_type_col = "infection_type",
  infection_type_method_col = "infection_type_method"
)
```

Arguments

<code>data</code>	Data frame.
<code>infection_type_col</code>	Character. Raw/observed infection type column. Default "infection_type".
<code>infection_type_method_col</code>	Character. Method column added by <code>prep_derive_hai_cai()</code> . Default "infection_type_method".

Value

Data frame with `infection_type_src` column added.

```
prep_flag_invalid_ast
```

Flag Invalid AST Values

Description

Adds a logical column `is_ast_invalid` that is TRUE when the harmonized AST value is not in `c("S", "I", "R")` and not NA.

Usage

```
prep_flag_invalid_ast(data, col = "ast_value_harmonized")
```

Arguments

<code>data</code>	Data frame.
<code>col</code>	Character. Harmonized AST column. Default "ast_value_harmonized".

Value

Data frame with is_ast_invalid column added.

prep_flag_organism_unmatched
<i>Flag Organisms Unmatched in Reference</i>

Description

Flags organisms that could not be matched to the reference organisms.csv. Adds a logical column is_organism_unmatched.

Usage

```
prep_flag_organism_unmatched(data, organism_col = "organism_normalized")
```

Arguments

- | | |
|--------------|---|
| data | Data frame with organism_normalized column. |
| organism_col | Character. Standard organism column. Default "organism_normalized". |

Value

Data frame with is_organism_unmatched column.

prep_flag_polymicrobial
<i>Flag Polymicrobial Infections</i>

Description

Flags polymicrobial status by counting distinct organisms per patient (optionally scoped within facility and/or syndrome).

Usage

```
prep_flag_polymicrobial(  
  data,  
  patient_col = "patient_id",  
  organism_col = "organism_normalized",  
  facility_col = NULL,  
  facility_name = NULL,  
  syndrome_col = NULL,  
  syndrome_name = NULL  
)
```

```

flag_polymicrobial(
  data,
  patient_col = "patient_id",
  organism_col = "organism_normalized",
  facility_col = NULL,
  facility_name = NULL,
  syndrome_col = NULL,
  syndrome_name = NULL
)

```

Arguments

<code>data</code>	Data frame.
<code>patient_col</code>	Character. Patient ID column. Default "patient_id".
<code>organism_col</code>	Character. Organism column. Default "organism_normalized".
<code>facility_col</code>	Character or NULL. Facility column for scoped counting.
<code>facility_name</code>	Character or NULL. Filter to this facility before flagging.
<code>syndrome_col</code>	Character or NULL. Syndrome column for scoped counting.
<code>syndrome_name</code>	Character or NULL. Filter to this syndrome before flagging.

Value

Data frame with `n_organisms` and `is_polymicrobial` (0/1).

`prep_flag_readmission` *Flag and Classify Readmissions*

Description

For each patient, classifies admissions into:

index First admission for this patient.

linked_readmission Readmission within `gap_linked_days` – treated as the same episode. Excluded from HAI incidence counts.

new_readmission Readmission between `gap_linked_days` and `gap_new_days`.

late_readmission Readmission after `gap_new_days` – fully new event.

Usage

```

prep_flag_readmission(
  data,
  patient_col = "patient_id",
  admission_col = "admission_date",
  gap_linked_days = 30,
  gap_new_days = 90
)

```

Arguments

<code>data</code>	Data frame.
<code>patient_col</code>	Character. Patient ID column. Default "patient_id".
<code>admission_col</code>	Character. Admission date column. Default "admission_date".
<code>gap_linked_days</code>	Numeric. Gap threshold for linked readmissions. Default 30.
<code>gap_new_days</code>	Numeric. Gap threshold for new vs late readmissions. Default 90.

Value

Data frame with `readmission_class` column added.

```
prep_get_contaminant_list
```

Get Contaminant List from Reference File

Description

Loads contaminant organisms from `common_commensals.csv` and filters by syndrome and/or specimen type.

Usage

```
prep_get_contaminant_list(
  syndrome = NULL,
  specimen_type = NULL,
  return_all = FALSE
)
```

Arguments

<code>syndrome</code>	Character. Syndrome name to filter (e.g., "Bloodstream infections").
<code>specimen_type</code>	Character. Specimen type to filter (e.g., "Blood culture").
<code>return_all</code>	Logical. If TRUE, returns all contaminants across syndromes. Default FALSE.

Value

A list with `names` (character vector) and `patterns` (list of pattern information for flexible matching).

prep_harmonize_ast	<i>Harmonize AST Values</i>
--------------------	-----------------------------

Description

Wrapper that applies the full AST cleaning pipeline in sequence:

1. `prep_clean_ast_values()` - extract clean S/I/R from raw strings (handles exact matches, regex patterns, and keyword scanning in one pass)
2. `prep_recode_intermediate_ast()` - resolve I values

Retains the original `ast_col` column as `ast_value_raw` and adds `ast_value_harmonized` and `intermediate_recoded`.

Usage

```
prep_harmonize_ast(  
  data,  
  antibiotic_col = "antibiotic_name_std",  
  ast_col = "ast_value_raw",  
  colistin_to_s = TRUE,  
  others_to_r = TRUE  
)
```

Arguments

<code>data</code>	Data frame with AST data.
<code>antibiotic_col</code>	Character. Standardized antibiotic name column. Default "antibiotic_name_std".
<code>ast_col</code>	Character. Raw AST value column. Default "ast_value_raw".
<code>colistin_to_s</code>	Logical. Recode Colistin I -> S. Default TRUE.
<code>others_to_r</code>	Logical. Recode non-Colistin I -> R. Default TRUE.

Value

Data frame with `ast_value_harmonized` and `intermediate_recoded` columns added.

prep_infer_department *Infer Hospital Department*

Description

Heuristically infers hospital department from specimen type, diagnosis, and patient age. Low-confidence inference; used only to fill missing values.

Usage

```
prep_infer_department(  
  data,  
  department_col = "hospital_department",  
  specimen_col = "specimen_type",  
  diagnosis_col = "diagnosis_1",  
  age_col = "Age",  
  overwrite = FALSE  
)
```

Arguments

data	Data frame.
department_col	Character. Department column. Default "hospital_department".
specimen_col	Character. Specimen type column. Default "specimen_type".
diagnosis_col	Character. Diagnosis column. Default "diagnosis_1".
age_col	Character. Age column. Default "Age".
overwrite	Logical. Overwrite existing values. Default FALSE.

Value

Data frame with hospital_department enriched.

prep_inventory_columns
Inventory Columns

Description

Returns a tibble describing every column: class, distinct values, and missingness. Used to detect completely empty columns and schema drift.

Usage

```
prep_inventory_columns(data)
```

Arguments

data Data frame.

Value

Tibble: col_name | class | n_distinct | pct_missing | has_any_value

prep_is_contaminant	<i>Check if Organism is a Contaminant</i>
---------------------	---

Description

Checks whether an organism name matches any known contaminant for a specific syndrome or specimen type using flexible pattern matching.

Usage

```
prep_is_contaminant(organism_name, syndrome = NULL, specimen_type = NULL)
```

Arguments

organism_name Character vector of organism name(s) to check.
 syndrome Character. Optional syndrome filter.
 specimen_type Character. Optional specimen type filter.

Value

Logical vector (same length as organism_name).

prep_log_source	<i>Log Data Source Provenance</i>
-----------------	-----------------------------------

Description

Records metadata about the source of data entering the pipeline. Returns a provenance metadata list attached as an attribute.

Usage

```
prep_log_source(  
  data,  
  study_type = "generic",  
  centre_name = NULL,  
  file_path = NULL,  
  sheet_name = NULL  
)
```

Arguments

data	Data frame.
study_type	Character. One of "stewardship", "surveillance", "aiims_icu_bsi", "generic".
centre_name	Character. Name of the contributing centre.
file_path	Character. Path to the source file (optional).
sheet_name	Character. Sheet name within Excel file (optional).

Value

Data frame with .provenance attribute attached.

```
prep_map_diagnosis_to_icd
```

Map Diagnosis Text to ICD Candidates

Description

Maps free-text diagnosis strings to ICD-10 candidate descriptions using one of three methods:

Usage

```
prep_map_diagnosis_to_icd(
  data,
  text_col = "diagnosis_text",
  reference = NULL,
  method = c("exact", "fuzzy", "python_embedding"),
  icd_desc_col = "description3",
  icd_code_col = "icd_code_who_eq",
  top_k = 5L,
  threshold = 0,
  model = "FremyCompany/BioLORD-2023",
  id_col = NULL
)
```

Arguments

data	Data frame.
text_col	Character. Column containing prepared diagnosis text (output of prep_diagnosis_text()). Default "diagnosis_text".
reference	Data frame or NULL. ICD-10 reference table. If NULL, loads from inst/extdata/icd10_who.csv.
method	Character. One of "exact", "fuzzy", "python_embedding". Default "exact".
icd_desc_col	Character. Column in reference to match against. Default "description3" (most concise ICD labels).
icd_code_col	Character. Column in reference holding ICD codes. Default "icd_code_who_eq".

top_k	Integer. Maximum ICD candidates to return per input string. Default 5. Ignored for "exact" (returns all exact matches).
threshold	Numeric. Minimum similarity score (0-1) to retain a candidate. Default 0.0 (keep all). For "fuzzy": similarity is 1 - normalised_distance.
model	Character. Sentence-transformers model name. Used only for "python_embedding". Default "FreemyCompany/BioLORD-2023".
id_col	Character or NULL. Identifier column to carry through into the output. Default NULL (output contains only match columns).

Details

"exact" Case-insensitive exact string match against ICD descriptions. Fast, high precision, low recall.

"fuzzy" String distance matching via **stringdist**. Handles typos and minor variations. Requires **stringdist**.

"python_embedding" Semantic embedding similarity using the Python alethia package via **reticulate**. Highest recall. Requires Python, alethia, and a sentence-transformers model.

Value

Long data frame with columns: diagnosis_text, icd_prediction, icd_code, icd_score, icd_rank, icd_method. If id_col is supplied, it is included as the first column.

```
prep_map_icd_to_syndrome
```

Map ICD Codes to Infectious Syndromes

Description

Joins an ICD-to-syndrome reference table onto the data. The reference must have at minimum two columns: an ICD code/description column and a syndrome column. Typically this is a manually curated mapping CSV.

Usage

```
prep_map_icd_to_syndrome(
  data,
  icd_col = "icd_code",
  icd_to_syndrome_ref,
  ref_icd_col = "icd_code",
  ref_syndrome_col = "infectious_syndrome",
  output_col = "syndrome",
  unmatched_label = NA_character_
)
```

Arguments

data	Data frame (output of prep_map_diagnosis_to_icd()) or any data frame with an ICD column).
icd_col	Character. Column in data holding ICD codes or descriptions to match on. Default "icd_code".
icd_to_syndrome_ref	Data frame. Reference table with at minimum icd_col values and a syndrome column.
ref_icd_col	Character. The ICD column name in icd_to_syndrome_ref. Default "icd_code".
ref_syndrome_col	Character. The syndrome column name in the reference. Default "infectious_syndrome".
output_col	Character. Name for the syndrome column added to data. Default "syndrome".
unmatched_label	Character. Label for rows with no syndrome match. Default NA_character_.

Value

Data frame with output_col added.

```
prep_missingness_report
```

Missingness Report

Description

Generates a per-column summary of missing values. Flags columns where missingness exceeds a threshold.

Usage

```
prep_missingness_report(data, threshold = 20, cols = NULL)
```

Arguments

data	Data frame.
threshold	Numeric. Percentage threshold (0-100) above which a column is flagged as high-missingness. Default 20.
cols	Character vector. Subset of columns to report on. NULL = all.

Value

Data frame with columns: col_name, n_total, n_missing, pct_missing, is_high_missing.

```
prep_parse_date_column
```

Detect and decode encrypted or non-standard date columns

Description

Inspects a column for date-like content and automatically selects the correct parsing strategy (Excel serial, Unix timestamp ms, ISO/DMY/MDY strings, reversed YYYYDDMM).

Usage

```
prep_parse_date_column(x, col_name = "date", table_label = "table")
```

Arguments

x	Vector (character, numeric, or Date-like) to parse.
col_name	Character. Column name used in messages.
table_label	Character. Table label used in messages.

Value

A Date vector of the same length as x.

```
prep_pivot_ast_wide_to_long
```

Convert Wide Format to Long Format

Description

Converts wide format data (where each antibiotic is a column) to long format (one row per organism-antibiotic combination). This is the first step before normalization and analysis.

Usage

```
prep_pivot_ast_wide_to_long(
  data,
  antibiotic_cols = NULL,
  pattern = NULL,
  id_cols = c("patient_id", "event_id", "organism_name", "date_of_culture"),
  antibiotic_name_col = "antibiotic_name",
  antibiotic_value_col = "antibiotic_value",
  remove_missing = TRUE,
  create_event_id = FALSE
)
```

Arguments

<code>data</code>	Data frame in wide format (antibiotics as columns)
<code>antibiotic_cols</code>	Character vector. Names of antibiotic columns to pivot. If NULL, will auto-detect based on pattern. Default NULL.
<code>pattern</code>	Character. Regex pattern to identify antibiotic columns if <code>antibiotic_cols</code> not provided. Default NULL (no auto-detection).
<code>id_cols</code>	Character vector. Columns to keep as identifiers (not pivoted). Default <code>c("patient_id", "event_id", "organism_name", "date_of_culture")</code> .
<code>antibiotic_name_col</code>	Character. Name for the new column containing antibiotic names. Default <code>"antibiotic_name"</code> .
<code>antibiotic_value_col</code>	Character. Name for the new column containing susceptibility results. Default <code>"antibiotic_value"</code> .
<code>remove_missing</code>	Logical. Remove rows where <code>antibiotic_value</code> is NA, empty, or <code>"-"</code> . Default TRUE.
<code>create_event_id</code>	Logical. Create <code>event_id</code> column if it doesn't exist (uses row numbers). Default FALSE.

Value

Data frame in long format

Examples

```
## Not run:
# Specify antibiotic columns explicitly
long_data <- prep_pivot_ast_wide_to_long(
  data = raw_data,
  antibiotic_cols = c("AMIKACIN", "GENTAMICIN", "CIPROFLOXACIN")
)

# Auto-detect columns by pattern (columns 12-53)
long_data <- prep_pivot_ast_wide_to_long(
  data = raw_data,
  antibiotic_cols = names(raw_data)[12:53]
)

# Auto-detect uppercase antibiotic names
long_data <- prep_pivot_ast_wide_to_long(
  data = raw_data,
  pattern = "^[A-Z]+$"
)

## End(Not run)
```

```
prep_recode_intermediate_ast
```

Recode Intermediate (I) Susceptibility Values

Description

Converts "I" (Intermediate) values to either "S" or "R" based on antibiotic. For Colistin: I -> S (clinical guideline). For all other antibiotics: I -> R (conservative surveillance approach).

Usage

```
prep_recode_intermediate_ast(
  data,
  antibiotic_col = "antibiotic_name",
  value_col = "antibiotic_value",
  colistin_to_s = TRUE,
  others_to_r = TRUE
)
```

Arguments

<code>data</code>	Data frame with antibiotic susceptibility data.
<code>antibiotic_col</code>	Character. Column with antibiotic names. Default "antibiotic_name".
<code>value_col</code>	Character. Column with S/I/R values. Default "antibiotic_value".
<code>colistin_to_s</code>	Logical. Convert Colistin I to S. Default TRUE.
<code>others_to_r</code>	Logical. Convert other antibiotics' I to R. Default TRUE.

Value

Data frame with recoded intermediate values.

```
prep_reconcile_hai_observed_inferred
```

Reconcile Observed and Inferred HAI/CAI Classification

Description

Where both an observed (raw) infection type and an inferred (date-derived) type exist, flags rows where they disagree. Retains observed classification by default and marks discordant records.

Usage

```
prep_reconcile_hai_observed_inferred(  
  data,  
  observed_col = "infection_type_observed",  
  inferred_col = "infection_type_inferred",  
  output_col = "infection_type"  
)
```

Arguments

data	Data frame.
observed_col	Character. Observed infection type column. Default "infection_type_observed".
inferred_col	Character. Inferred infection type column. Default "infection_type_inferred".
output_col	Character. Reconciled output column. Default "infection_type".

Value

Data frame with reconciled infection_type and hai_discordant logical flag.

prep_report_capabilities

Report Preprocessing Capabilities

Description

Prints a human-readable summary of which preprocessing steps can be run given the current data columns.

Usage

```
prep_report_capabilities(data)
```

Arguments

data	Data frame after column mapping. Or pass the result of detect_preprocessing_capabilities() directly.
------	--

Value

Invisibly returns the capabilities vector.

```
prep_split_poly_episode
```

Split Polymicrobial Episodes by Strategy

Description

Applies a chosen strategy for handling polymicrobial episodes:

fractional Keep all rows; poly_weight is applied downstream during DALY calculation.

exclude Drop polymicrobial episodes from the main analysis frame.

separate Retain polymicrobial episodes in a separate data frame stored in `attr(result, "poly_data")` for sensitivity analyses.

Usage

```
prep_split_poly_episode(
  data,
  strategy = c("fractional", "exclude", "separate"),
  polymicrobial_col = "is_polymicrobial"
)
```

Arguments

<code>data</code>	Data frame with <code>is_polymicrobial</code> column (output of <code>prep_flag_polymicrobial()</code>).
<code>strategy</code>	Character. "fractional" (default), "exclude", or "separate".
<code>polymicrobial_col</code>	Character. Polymicrobial flag column (0/1). Default "is_polymicrobial".

Value

Data frame with polymicrobial rows handled per strategy. When `strategy = "separate"`, the poly subset is in `attr(result, "poly_data")`.

```
prep_standardize_antibiotics
```

Normalize Antibiotic Names

Description

Standardizes antibiotic names using fuzzy matching against WHO AWaRe reference.

Usage

```
prep_standardize_antibiotics(
  data,
  antibiotic_col = "antibiotic_name",
  who_table = NULL,
  add_class = TRUE,
  add_aware = TRUE
)
```

Arguments

<code>data</code>	Data frame with antibiotic data.
<code>antibiotic_col</code>	Character. Column name with antibiotic names. Default "antibiotic_name".
<code>who_table</code>	Data frame. WHO AWaRe classification table. If NULL, loads from inst/extdata/WHO_aware_class.csv.
<code>add_class</code>	Logical. Add antibiotic_class column. Default TRUE.
<code>add_aware</code>	Logical. Add aware_category column. Default TRUE.

Value

Data frame with antibiotic_normalized, antibiotic_class, aware_category.

```
prep_standardize_column_names
```

Standardize Column Names to Package Convention

Description

Convenience wrapper around `prep_build_column_map()` + `prep_apply_column_map()`. Builds an exact-match map from mapping aliases, optionally extends it with fuzzy matching, then applies the rename in one call.

Usage

```
prep_standardize_column_names(
  data,
  mapping = default_column_mappings,
  fuzzy_match = TRUE,
  fuzzy_threshold = 0.3,
  interactive = FALSE
)
```

Arguments

data	A data frame with raw column names.
mapping	Named list of standard_name -> aliases vectors. Default uses default_column_mappings.
fuzzy_match	Logical. Enable fuzzy matching for unresolved standards. Default TRUE.
fuzzy_threshold	Numeric. Max Jaro-Winkler distance (0-1). Default 0.3.
interactive	Logical. Prompt user to confirm fuzzy matches. Default FALSE.

Details

For more control (e.g. dataset-specific maps, post-rename assertions) use the staged API directly:
 prep_build_column_map() -> prep_apply_column_map() -> prep_assert_standard_names().

Value

List: data (renamed), mapping_log, unmapped.

```
prep_standardize_final_outcome
```

Standardize Final Outcome Column

Description

Normalizes outcome variants to a two-level controlled vocabulary: "Survived", "Died", or NA.

Usage

```
prep_standardize_final_outcome(data, col = "final_outcome")
```

Arguments

data	Data frame.
col	Character. Outcome column. Default "final_outcome".

Details

Survived Survived / Alive / Discharge / Discharged / Recovered

Died Died / Death / Expired / Deceased / Dead

NA Unknown / Missing / Absconded / LAMA / DAMA

Value

Data frame with outcome_std column added.

```
prep_standardize_infection_type
```

Standardize Infection Type Column

Description

Normalizes HAI/CAI variants to a controlled vocabulary: "HAI", "CAI", or NA.

Usage

```
prep_standardize_infection_type(data, col = "infection_type")
```

Arguments

data	Data frame.
col	Character. Infection type column. Default "infection_type".

Value

Data frame with standardized infection_type column.

```
prep_standardize_organisms
```

Normalize Organism Names

Description

Standardizes organism names using organisms.csv reference file. Automatically handles abbreviations (E. coli), case variations, and typos.

Usage

```
prep_standardize_organisms(
  data,
  organism_col = "organism_name",
  add_organism_group = TRUE,
  add_resistance_flags = TRUE
)
```

Arguments

data	Data frame with organism column.
organism_col	Character. Organism column name. Default "organism_name".
add_organism_group	Logical. Add organism_group column. Default TRUE.
add_resistance_flags	Logical. Add MRSA/MRCONS flags. Default TRUE.

Value

Data frame with organism_normalized, organism_group, and optionally resistance flag columns.

```
prep_standardize_outcome
```

Standardize Outcome Values (deprecated)

Description

Deprecated. Use prep_standardize_final_outcome() instead, which adds outcome_std ("Survived"/"Died"/NA) without overwriting the original column.

Usage

```
prep_standardize_outcome(data, col = "final_outcome")
```

Arguments

data	Data frame containing outcome column.
col	Character. Name of outcome column. Default "final_outcome".

Value

Data frame with outcome_std column added.

```
prep_standardize_sex
```

Standardize Sex Values

Description

Maps various gender/sex representations to standard "M" or "F".

Usage

```
prep_standardize_sex(data, col = "gender")
```

Arguments

data	Data frame containing gender column.
col	Character. Name of gender column. Default "gender".

Value

Data frame with standardized gender values.

```
prep_standardize_specimens
```

Normalize Specimen/Sample Type

Description

Normalizes specimen/sample type names and adds sample_category and a collapsed sterile_classification from the reference CSV file. Includes rule-based text cleaning and fuzzy matching for common shorthand and minor misspellings.

Usage

```
prep_standardize_specimens(  
  data,  
  specimen_col = "specimen_type",  
  add_categories = TRUE  
)
```

Arguments

data	Data frame with specimen column.
specimen_col	Character. Specimen column name. Default "specimen_type".
add_categories	Logical. Add sample_category and sterile_classification. sterile_classification is collapsed to Sterile, Non-Sterile, or Others/Ambiguous using specimen-specific rules for body-fluid, drain/device-associated, and unspecified samples. Default TRUE.

Value

Data frame with specimen_normalized, sample_category, and sterile_classification columns.

```
prep_validate_analysis_ready
```

Validate Analysis-Ready Dataset

Description

Final gate check before burden estimation. Confirms that the minimum standard AMR schema fields are present and have acceptable completeness.

Usage

```
prep_validate_analysis_ready(
  data,
  min_rows = 10L,
  max_missing_pct = 30,
  stop_on_failure = FALSE
)
```

Arguments

<code>data</code>	Analysis-ready data frame.
<code>min_rows</code>	Integer. Minimum acceptable row count. Default 10.
<code>max_missing_pct</code>	Numeric. Maximum percent missing allowed for core fields. Default 30.
<code>stop_on_failure</code>	Logical. Stop if validation fails. Default FALSE.

Value

Invisibly returns a list: passes (logical), issues (character vector).

```
prep_validate_date_logic
```

Validate Date Logic

Description

Checks that date sequences and derived values are internally consistent. Consolidates all cross-column date/age checks in one place: 1. admission_date <= culture_date <= outcome_date 2. dob <= admission_date 3. age in [0, 120] 4. age vs DOB-computed age within 2-year tolerance 5. "Died" rows must have an outcome date

Usage

```
prep_validate_date_logic(
  data,
  admission_col = "admission_date",
  culture_col = "culture_date",
  outcome_col = "outcome_date",
  dob_col = "dob",
  age_col = "age",
  outcome_value_col = "final_outcome"
)
```

Arguments

data	Data frame with standard column names.
admission_col	Character. Admission date column. Default "admission_date".
culture_col	Character. Culture/event date column. Default "culture_date".
outcome_col	Character. Outcome date column. Default "outcome_date".
dob_col	Character. Date of birth column. Default "dob".
age_col	Character. Age column. Default "age".
outcome_value_col	Character. Outcome value column for death check. Default "final_outcome".

Details

Reports violations as warnings; does not drop or modify rows.

Value

Invisibly returns data. Prints violation summary.

```
prep_validate_table    Run all pre-join sanity checks for one table
```

Description

Convenience wrapper that calls `prep_check_columns()`, `prep_check_keys()`, and `prep_coerce_dates()` in sequence. Returns the date-coerced data along with a check report.

Usage

```
prep_validate_table(
  data,
  required_cols = character(),
  key_col = NULL,
  expected_types = character(),
  date_cols = NULL,
  table_label = "table",
  stop_on_missing = TRUE
)
```

Arguments

data	Data frame.
required_cols	Character vector of required column names.
key_col	Character. Primary/foreign key column to quality-check.
expected_types	Named character vector of expected column classes.

date_cols Character vector of date column names to coerce. NULL triggers auto-detection.
table_label Character. Label used in all messages.
stop_on_missing Logical. Passed to `prep_check_columns()`.

Value

A list with data (date-coerced) and report (check summary).

prepare_los_data	<i>Prepare LOS Dataset</i>
------------------	----------------------------

Description

Calculates length of stay from admission and outcome dates, filters to discharged patients with valid LOS within a plausible range.

Usage

```
prepare_los_data(
  data,
  admission_col = "date_of_admission",
  outcome_date_col = "final_outcome_date",
  outcome_col = "final_outcome",
  patient_id_col = "PatientInformation_id",
  max_los = 200
)
```

Arguments

data Data frame with patient-level records.
admission_col Character. Column name for admission date. Default "date_of_admission".
outcome_date_col Character. Column name for outcome/discharge date. Default "final_outcome_date".
outcome_col Character. Column name for outcome status. Default "final_outcome".
patient_id_col Character. Column name for patient identifier. Default "PatientInformation_id".
max_los Numeric. Maximum plausible LOS in days. Default 200.

Value

Data frame with one row per patient-admission, including LOS_days.

```
prepare_random_effects
```

Build the generic random-effect representation for an arbitrary number of blocks

Description

Flattens an arbitrary number of declared random-intercept blocks into the representation used by the generic Stan models and by every downstream mu-reconstruction site (profile generation, DALY aggregation, validation).

Usage

```
prepare_random_effects(
  data,
  random_effects,
  min_repeated_levels = NULL,
  singleton_threshold = 0.9,
  on_mostly_singleton = c("warn", "stop")
)
```

Arguments

<code>data</code>	Data frame containing every declared block's <code>group_col</code> .
<code>random_effects</code>	Character vector (legacy) or list-of-blocks (see <code>.normalize_random_effects_spec()</code>).
<code>min_repeated_levels</code>	Optional integer. If supplied, a block whose number of levels with ≥ 2 observations falls below this is checked INDEPENDENTLY of <code>singleton_threshold</code> below (previously this was bugged: nested inside the <code>singleton_fraction</code> check, so it was silently skipped whenever <code>singleton_fraction</code> fell at or below the threshold even if <code>n_repeated_levels</code> was itself too low). Triggers a stop (via <code>on_mostly_singleton = "stop"</code>) or warning. NULL (default) never checks this – eligibility gating belongs in the analysis layer (see <code>anumaan-analysis</code> 's <code>random_effect_eligibility.R</code>), not hardcoded into the package.
<code>singleton_threshold</code>	Numeric in (0, 1]. A block whose fraction of singleton levels (exactly 1 observation) exceeds this triggers a separate warning/stop, evaluated independently of <code>min_repeated_levels</code> . Default 0.90.
<code>on_mostly_singleton</code>	"warn" (default) or "stop" – applies to BOTH the <code>min_repeated_levels</code> check and the <code>singleton_threshold</code> check.

Value

An object of class "amr_random_effects": block metadata, level maps, per-event flattened group indices, nesting diagnostics.

preprocess_for_profiles

Preprocess AST Data for Resistance Profile Estimation (Pathway 1)

Description

Orchestrates the full preprocessing pipeline required before marginal resistance computation and convex optimisation. Calls existing prep_* package functions in sequence and returns a wide tibble (one row per isolate) with antibiotic-class columns ready for compute_marginals_from_data().

Usage

```
preprocess_for_profiles(
  data,
  col_map = list(isolate_col = "isolate_id", pathogen_col = "pathogen", ast_col =
    "ast_value", patient_col = "patient_id", date_col = "date_of_culture", geography_col
    = "state", specimen_col = "specimen_type", age_col = "age", dob_col = "dob",
    antibiotic_col = "antibiotic_name", class_col = "antibiotic_class", location_col =
    NULL, outcome_col = NULL),
  panel_map,
  stratify_by = NULL,
  outcome_col = NULL,
  who_table = NULL
)
```

Arguments

data	Data frame. Raw AST data in long or wide format.
col_map	Named list mapping logical roles to actual column names. Same structure as in validate_profile_inputs(). Default names follow standard package conventions.
panel_map	Named list. Mandatory. Maps each pathogen name to a character vector of antibiotic class names to include in profile estimation. Classes not in the panel are excluded and logged with reason code "not_in_user_panel". Example: list("Klebsiella pneumoniae" = c("3GC", "Carbapenems", "Fluoroquinolones")).
stratify_by	Character vector or NULL. Stratification dimensions to carry through to output. Valid values: "geography", "year", "outcome". Each requested dimension adds the corresponding column to the wide output. Default NULL.
outcome_col	Character or NULL. Column name for patient outcome (e.g. "final_outcome"). Required when "outcome" is in stratify_by. Default NULL.
who_table	Data frame or NULL. Custom WHO AWaRe classification table passed to prep_standardize_antibiotic. If NULL, uses the built-in inst/extdata/WHO_aware_class.csv. Default NULL.

Details

Pipeline steps (in order):

1. Input validation via `validate_profile_inputs()`.
2. Wide -> long conversion if wide format detected (`prep_pivot_ast_wide_to_long()`).
3. AST value harmonisation – clean messy strings, resolve I values (`prep_harmonize_ast()`).
4. Antibiotic name standardisation if no class column (`prep_standardize_antibiotics()`).
5. Class assignment if still absent (`prep_classify_antibiotic_class()`).
6. Class-level collapse: any R in class => class = R (`prep_collapse_class_level()`). Counting unit is `isolate_id`.
7. Year derivation from `date_col` when "year" is in `stratify_by`.
8. Organism-specific panel filter: retain only user-specified classes per pathogen.
9. Long -> wide pivot with antibiotic-class columns (`prep_create_wide_ast_matrix()`).

Counting unit: All statistics are computed per unique `isolate_id`. A patient with multiple isolates contributes one count per isolate, not one count per patient.

Value

A named list:

`data_wide` Tibble. One row per `isolate_id`. Columns: all metadata columns + one column per antibiotic class (values S / R / NA).

`preprocessing_log` Tibble. One row per pipeline step with columns `step`, `n_rows_in`, `n_rows_out`, `message`, `status`.

`panel_exclusions` Tibble. All organism-class combinations removed by the panel filter, with `reason_code` and `reason_text`.

`col_map_resolved` Named list. Updated `col_map` with any column names derived during preprocessing (e.g. the harmonised AST column name, the resolved class column name).

`detected_format` Character. "long" or "wide".

Examples

```
## Not run:
result <- preprocess_for_profiles(
  data = raw_ast,
  col_map = list(
    isolate_col = "isolate_id",
    pathogen_col = "organism",
    ast_col = "result",
    patient_col = "pid",
    date_col = "culture_date",
    geography_col = "state",
    specimen_col = "specimen",
    age_col = "age",
    antibiotic_col = "drug_name"
  ),
```

```

panel_map = list(
  "Klebsiella pneumoniae" = c("3GC", "Carbapenems", "Fluoroquinolones"),
  "Escherichia coli"      = c("3GC", "Fluoroquinolones", "Aminoglycosides")
),
stratify_by = c("geography", "year"),
outcome_col = "final_outcome"
)

result$data_wide # ready for compute_marginals_from_data()
result$preprocessing_log # step-by-step row counts
result$panel_exclusions # classes dropped per organism

## End(Not run)

```

```
print.amr_config      Print AMR Configuration
```

Description

Print AMR Configuration

Usage

```
## S3 method for class 'amr_config'
print(x, ...)
```

Arguments

x	An amr_config object
...	Additional arguments (not used)

```
print.amr_preprocessing_report
      Print Preprocessing Report
```

Description

Print Preprocessing Report

Usage

```
## S3 method for class 'amr_preprocessing_report'
print(x, ...)
```

Arguments

x	An amr_preprocessing_report object
...	Additional arguments (not used)

<code>print.amr_result</code>	<i>Print Method for AMR Preprocessing Results</i>
-------------------------------	---

Description

Print Method for AMR Preprocessing Results

Usage

```
## S3 method for class 'amr_result'
print(x, ...)
```

Arguments

<code>x</code>	amr_result object
<code>...</code>	Additional arguments (not used)

Value

Invisibly returns the object

<code>re_contribution</code>	<i>Compute each event's total random-effect contribution from a flattened <code>re_effect[D, total_re_levels]</code> matrix (as emitted by the generic Stan models' generated <code>re_effect</code>, or reconstructed R-side from posterior draws of <code>z_re/tau_re/L_corr_re</code>). This is the SINGLE generic helper every downstream mu-reconstruction site should call instead of hand-summing <code>hospital_effect/patient_effect/admission_effect</code>.</i>
------------------------------	--

Description

Compute each event's total random-effect contribution from a flattened `re_effect[D, total_re_levels]` matrix (as emitted by the generic Stan models' generated `re_effect`, or reconstructed R-side from posterior draws of `z_re/tau_re/L_corr_re`). This is the SINGLE generic helper every downstream mu-reconstruction site should call instead of hand-summing `hospital_effect/patient_effect/admission_effect`.

Usage

```
re_contribution(re_effect, flat_group_index)
```

Arguments

<code>re_effect</code>	D x total_re_levels numeric matrix (one posterior draw).
<code>flat_group_index</code>	n_events x R integer matrix (see <code>prepare_random_effects()</code> \$flat_group_index), or a single event's R-length integer vector.

Value

If flat_group_index is a matrix: n_events x D matrix of summed RE contributions. If a vector: length-D vector for one event.

round_to_sum	<i>Largest-Remainder Rounding</i>
--------------	-----------------------------------

Description

Rounds a numeric vector so that the individual rounded values sum exactly to a target total. Uses the largest-remainder method (Hamilton method). Input is coerced with as.numeric(); non-numeric values become NA.

Usage

```
round_to_sum(x, target = round(sum(x, na.rm = TRUE)))
```

Arguments

- x Numeric vector to round.
- target Integer target sum. Default is round(sum(x, na.rm = TRUE)).

Value

Integer vector of the same length as x whose sum equals target across non-missing entries; missing values are preserved as NA.

Examples

```
round_to_sum(c(3.3, 3.3, 3.4), target = 10)
```

run_preprocess	<i>AMR Data Preprocessing Pipeline</i>
----------------	--

Description

Master function that orchestrates the complete AMR data preprocessing pipeline. Applies standardization, enrichment, and derivation in sequence.

Usage

```
run_preprocess(
  data,
  config = NULL,
  phases = "all",
  verbose = TRUE,
  validate = TRUE,
  generate_report = TRUE
)
```

Arguments

data	Data frame. Raw AMR dataset.
config	Configuration object from <code>amr_config()</code> . If NULL, uses defaults.
phases	Character vector. Which phases to run: "standardize", "enrich", "derive", or "all" (default). Allows partial pipeline execution.
verbose	Logical. Print detailed progress messages. Default TRUE.
validate	Logical. Run validation checks before and after. Default TRUE.
generate_report	Logical. Generate preprocessing report. Default TRUE.

Details

Pipeline phases: 1. **Standardization**: Column mapping, value normalization, date parsing 2. **Enrichment**: Derive missing optional variables (Age, LOS, infection type) 3. **Derivation**: Create analytical variables (event IDs, MDR/XDR, weights)

Value

List with class "amr_result": - data: Preprocessed data frame - config: Configuration used - log: List of processing logs and summaries - report: Preprocessing report (if `generate_report = TRUE`) - metadata: Pipeline execution metadata

Examples

```
## Not run:
# Full pipeline with defaults
result <- run_preprocess(raw_data)
clean_data <- result$data

# Custom configuration
config <- amr_config(
  hai_cutoff = 3,
  mdr_definition = "CDC",
  fuzzy_match = TRUE
)
result <- run_preprocess(raw_data, config = config)

# Run only standardization phase
```

```
result <- run_preprocess(raw_data, phases = "standardize")

# Summary
summary(result)

## End(Not run)
```

safe_fit	<i>Safely Fit a Distribution</i>
----------	----------------------------------

Description

Wrapper around `fitdistrplus::fitdist` that returns `NULL` instead of throwing an error when fitting fails.

Usage

```
safe_fit(x, dist)
```

Arguments

- x Numeric vector of positive values (e.g., LOS in days).
- dist Character. Distribution name: "weibull", "lnorm", or "gamma".

Value

A `fitdist` object, or `NULL` on failure.

Examples

```
safe_fit(rlnorm(100), "lnorm")
```

select_resistance_class	<i>Select Resistance Class for Burden Attribution</i>
-------------------------	---

Description

Selects a single resistance class per event using beta-lactam hierarchy and relative risk (RR) values. Prevents double-counting in DALY burden estimation by choosing the most clinically relevant resistant class.

Usage

```
select_resistance_class(  
  data,  
  event_col = "event_id",  
  class_col = "antibiotic_class",  
  susceptibility_col = "antibiotic_value",  
  rr_col = "rr_value",  
  hierarchy = NULL,  
  filter_resistant = TRUE  
)
```

Arguments

data	Data frame with class-level resistance and RR columns.
event_col	Character. Event ID column. Default "event_id".
class_col	Character. Antibiotic class column. Default "antibiotic_class".
susceptibility_col	Character. Susceptibility column. Default "antibiotic_value".
rr_col	Character. RR value column. Default "rr_value". If missing, only hierarchy is used.
hierarchy	Named numeric vector. Custom hierarchy (class name -> rank). If NULL, uses get_beta_lactam_hierarchy().
filter_resistant	Logical. If TRUE, only consider resistant (R) classes. Default TRUE.

Details

Selection order: 1. Beta-lactam hierarchy rank (Carbapenems > 4GC > 3GC > ...) 2. RR value (higher relative risk = higher priority) 3. Alphabetical tie-breaker

Value

Data frame filtered to one resistance class per event.

shorten_drug_class	<i>Shorten Antibiotic Class Names</i>
--------------------	---------------------------------------

Description

Maps long antibiotic class names to common abbreviations used in GBD-style figures.

Usage

```
shorten_drug_class(x)
```

Arguments

x Character vector of antibiotic class names.

Value

Character vector of shortened names.

Examples

```
shorten_drug_class(c("Carbapenems", "Third-generation-cephalosporins"))
```

```
simulate_probit_mixed_predictive
```

*Simulate mixed predictive replicate datasets: new random-effect levels
for a requested block, retaining fitted hyperparameters*

Description

For each requested block r in `blocks_to_replicate`, retains the posterior draws of $\tau_r^{(s)}$ and the block's own fitted cross-class correlation $R_{block[r]}^{(s)}$ (a generated quantity already stored in `fitted_model$draws`), but draws `n_new_levels_per_block` ENTIRELY NEW synthetic level effects

$$u_{r,new}^{(s)} \sim MVN(0, \Sigma_r^{(s)})$$

per posterior state, where $\Sigma_r^{(s)} = \text{diag}(\tau_r^{(s)}) R_{block[r]}^{(s)} \text{diag}(\tau_r^{(s)})$. Every real fitted event is deterministically (seeded) reassigned to one of the `n_new_levels_per_block` new synthetic levels for each replicated block, so a complete replicate outcome matrix can be generated exactly as in [simulate_probit_posterior_predictive](#). Blocks NOT in `blocks_to_replicate` use their FITTED posterior random effects (`unreplicated_block_behavior = "retain_fitted"`, default) or cause an error ("error") if any declared block is left unhandled.

Usage

```
simulate_probit_mixed_predictive(
  fitted_model,
  blocks_to_replicate,
  n_states = 500L,
  seed = 123L,
  n_new_levels_per_block = 1L,
  unreplicated_block_behavior = c("retain_fitted", "error")
)
```

Arguments

fitted_model	A fitted model object as returned by <code>fit_bayesian_multivariate_probit</code> .
blocks_to_replicate	Character vector; a subset of <code>fitted_model\$random_effects_prep\$block_names</code> to draw NEW levels for.
n_states	Integer; number of posterior states to use. Default 500.
seed	Integer seed (deterministic).
n_new_levels_per_block	Integer; how many new synthetic levels to draw per replicated block. Default 1.
unreplicated_block_behavior	"retain_fitted" (default) uses fitted posterior random effects for every declared block not in <code>blocks_to_replicate</code> ; "error" requires every declared block to be listed in <code>blocks_to_replicate</code> .

Details

This is NOT standard posterior predictive checking (which conditions on the observed groups' own fitted random effects) – do not conflate the two. `blocks_to_replicate` is currently exercised for a single block at a time in this package's own tests (the hospital-level use case explicitly scoped by Part 13 of the predictive-checking specification); multi-block replication is implemented generically but not yet covered by synthetic recovery tests.

Value

An object of class `c("amr_mixed_predictive_draws", "amr_ppc_draws")` with `setup`, `generate_state`, and the other fields shared with `simulate_probit_posterior_predictive`'s return value, plus `blocks_to_replicate`, `unreplicated_blocks`, `n_new_levels_per_block`, `unreplicated_block_behavior`.

References

Stan Development Team. "Posterior and Prior Predictive Checks." Stan User's Guide. <https://mc-stan.org/docs/stan-users-guide/posterior-predictive-checks.html>

`simulate_probit_posterior_predictive`

Simulate posterior predictive replicate datasets from a fitted probit model

Description

For each of `n_states` saved posterior states s and each fitted event e , reconstructs

$$\mu_e^{(s)} = X_e \beta^{(s)} + \sum_r u_{r, g_r(e)}^{(s)}$$

using the existing generic random-effect machinery ([prepare_random_effects/re_contribution](#) – never hardcoded block names), then draws a COMPLETE replicate outcome matrix $Y_{rep}^{(s)}$ unconditionally from $P(Y \mid \mu^{(s)}, \Omega^{(s)})$: Bernoulli($\Phi(\mu)$) per class for identity residual models, or $Z \sim \text{MVN}(\mu, \Omega)$; $Y = I(Z > 0)$ for correlated residual models (see Stan User’s Guide, <https://mc-stan.org/docs/stan-users-guide/posterior-predictive-checks.html>).

Usage

```
simulate_probit_posterior_predictive(
  fitted_model,
  n_states = 500L,
  seed = 123L,
  preserve_observation_mask = TRUE,
  return_replicates = FALSE,
  replicate_random_effects = FALSE,
  random_effect_blocks_to_replicate = NULL
)
```

Arguments

fitted_model	A fitted model object as returned by fit_bayesian_multivariate_probit (or the equivalent fit_light object saved by the analysis layer with \$fit stripped – this function never touches \$fit, only stored draws and design matrices).
n_states	Integer; number of saved posterior states to use (a random subsample, seeded, if fewer than the total number of saved draws). Default 500; use 1000-2000 for shortlisted models.
seed	Integer seed for state subsampling and stochastic generation.
preserve_observation_mask	Logical, default TRUE. If TRUE, the returned generator’s Y_rep slot masks the complete generated replicate to exactly the same tested/untested cell pattern as the real observed AST matrix, so that discrepancy statistics computed downstream compare like with like. The COMPLETE (unmasked) replicate is always separately available as Y_rep_complete for statistics that genuinely need it (e.g. complete-profile statistics restricted to adequately-supported fully-observed subsets) – the two are never compared without explicit qualification.
return_replicates	Logical, default FALSE. If TRUE, materialises full $S \times N_{\text{events}} \times D$ replicate arrays (Y_rep_array, masked per preserve_observation_mask, and Y_rep_complete_array, always unmasked) – small debug datasets only; refuses above a fixed element-count ceiling.
replicate_random_effects, random_effect_blocks_to_replicate	Reserved for API compatibility with the (distinct) mixed predictive check. Not implemented here – passing a non-default value errors with a pointer to simulate_probit_mixed_predictive rather than silently being ignored.

Details

This function NEVER calls the internal DALY conditional-completion helper (`.gibbs_conditional_profile_probs()`) – that function samples missing cells conditional on an event’s own observed AST signs, which is a

fundamentally different (and, for this purpose, wrong) generative mechanism. See the file header of `probit_posterior_predictive.R` and `test-probit-posterior-predictive.R` for the explicit distinction and regression test.

By default (`return_replicates = FALSE`), no $S \times N_{\text{events}} \times D$ array is ever materialised – this function returns a lightweight generator object (class `"amr_ppc_draws"`) that `compute_probit_ppc_statistics` drives one state at a time, streaming, to bound peak memory at $O(N_{\text{events}} \times D)$ rather than $O(S \times N_{\text{events}} \times D)$. Each state's replicate is generated under a temporary, locally-seeded RNG state derived deterministically from (`seed`, `s`) alone (restoring the caller's global RNG state afterwards) – so a given state's draw depends only on `seed`, `s`, and that state's posterior parameters, never on call order, on interleaving with any other generator object, or on any other RNG use elsewhere in the calling session. Draws are cached per state, so calling the generator repeatedly for the same `s` is idempotent.

`return_replicates = TRUE` additionally materialises the full replicate arrays and is intended for small debugging datasets only; a safety ceiling refuses to silently run this on a large realistic dataset (reduce `n_states` or leave `return_replicates = FALSE` instead).

Value

An object of class `"amr_ppc_draws"`: a list with `setup` (internal draw-setup object), `generate_state` (a function of one integer argument `s` returning `list(Y_rep_complete, Y_rep)` for that state), `n_states_used`, `seed_used`, `preserve_observation_mask`, `residual_structure`, `return_replicates`, and (only when `return_replicates = TRUE`) `Y_rep_array/Y_rep_complete_array`.

References

Stan Development Team. "Posterior and Prior Predictive Checks." Stan User's Guide. <https://mc-stan.org/docs/stan-users-guide/posterior-predictive-checks.html>

`simulate_probit_prior_predictive`

Simulate prior predictive replicate datasets for a probit model

Description

Draws `n_states` independent parameter states directly from the declared priors (no NUTS/no conditioning on outcomes) and generates a complete replicate outcome matrix from each, using the REAL fixed-effect design matrix and random-effect grouping structure of `fitted_model` as conditioning predictors (per Stan User's Guide-style prior predictive simulation). Reuses the identical generative mechanism as `simulate_probit_posterior_predictive` (Bernoulli($\Phi(\mu)$) for identity residual, $Z \sim \text{MVN}(\mu, \Omega)$; $Y = I(Z > 0)$ for correlated residual) – only the source of θ differs (prior vs. posterior draws).

Usage

```
simulate_probit_prior_predictive(
  fitted_model,
  n_states = 1000L,
  seed = 123L,
  prior_config_override = NULL,
  preserve_observation_mask = TRUE,
  return_replicates = FALSE
)
```

Arguments

fitted_model A fitted model object as returned by [fit_bayesian_multivariate_probit](#) – supplies the real X_{event} design matrix, random-effect grouping structure, class panel, and `prior_config_used` (the priors to replicate exactly). Only design/configuration fields are read; no posterior draws are used.

n_states Integer; number of independent prior states to draw. Default 1000.

seed Integer seed (deterministic).

prior_config_override Optional named list overriding any of `beta_sd`, `tau_sd`, `lkj_eta` from `fitted_model$prior_config_used` – e.g. to prior-predictive-check a CANDIDATE prior before fitting.

preserve_observation_mask, return_replicates See [simulate_probit_posterior_predictive](#).

Details

Returns an object of class `c("amr_prior_predictive_draws", "amr_ppc_draws")` – structurally compatible with [compute_probit_ppc_statistics](#) (which accepts any "amr_ppc_draws"-classed object), so the same AMR-specific discrepancy-statistic machinery can summarise the prior predictive distribution's own properties (see [compute_prior_predictive_status](#), which additionally derives the prior-specific plausibility summaries in Part 11 of the predictive-checking specification: fraction of event-class probabilities near 0/1, fraction of degenerate all-resistant/all-susceptible profiles, and hospital-level spread).

Value

An "amr_prior_predictive_draws"/"amr_ppc_draws" object with the same shape as [simulate_probit_posterior_predictive](#) return value.

References

Stan Development Team. "Posterior and Prior Predictive Checks." Stan User's Guide. <https://mc-stan.org/docs/stan-users-guide/posterior-predictive-checks.html>

`summarise_distribution`*Summarise a Fitted Distribution*

Description

Extracts mean, median, SD, and parameter values from a `fitdist` object for Weibull, Lognormal, or Gamma distributions.

Usage

```
summarise_distribution(fit, dist)
```

Arguments

<code>fit</code>	A <code>fitdist</code> object from <code>fitdistrplus::fitdist()</code> .
<code>dist</code>	Character. One of "weibull", "lnorm", or "gamma".

Value

Data frame with columns `Mean_LOS`, `Median_LOS`, `SD_LOS`, `Parameters`.

Examples

```
fit <- fitdistrplus::fitdist(rlnorm(200), "lnorm")
summarise_distribution(fit, "lnorm")
```

`summarize_column_mapping`*Summarize Column Mapping*

Description

Documents how original column names were mapped to standard names.

Usage

```
summarize_column_mapping(raw_data, processed_data, mapping_log)
```

Arguments

<code>raw_data</code>	Original data
<code>processed_data</code>	Processed data
<code>mapping_log</code>	Column mapping log from preprocessing

Value

Data frame with mapping details

```
summarize_data_quality
```

Summarize Data Quality

Description

Computes data quality metrics before and after preprocessing.

Usage

```
summarize_data_quality(raw_data, processed_data, include_plots = TRUE)
```

Arguments

```
raw_data          Original data
processed_data    Processed data
include_plots     Logical. Generate plots. Default TRUE.
```

Value

List with quality metrics

```
summarize_fit_correlation_matrix
```

Summarize a posterior $D \times D$ correlation matrix from stored fit draws

Description

Extracts posterior mean/median/CI for each off-diagonal cell of a named correlation-matrix generated quantity (e.g. "Omega", "R_hospital", "R_patient", "R_admission"). These are three distinct quantities and must not be combined into one table: Omega is the event-level residual cross-class correlation (only estimated when residual_structure == "correlated"); R_hospital/R_patient/R_admission are the correlation, across antibiotic classes, of that random effect's own tendencies (estimated regardless of residual_structure, since the hospital/patient/admission random effects always use a diag_pre_multiply(tau, L_corr) parameterisation). Returns NULL if the requested matrix_var is not present in fit\$draws (e.g. Omega on an identity-residual fit, or R_admission on a 1- or 2-RE fit).

Usage

```
summarize_fit_correlation_matrix(
  fit,
  matrix_var,
  class_cols,
  ci_level = 0.95,
  block_index = NULL
)
```

Arguments

fit	A fitted_model object (or its lightweight saved form) with a \$draws posterior::draws_array retaining the requested matrix_var.
matrix_var	Character. One of "Omega", "R_hospital", "R_patient", "R_admission".
class_cols	Character vector of antibiotic class column names, in the same order used to fit the model (dimension order of matrix_var).
ci_level	Numeric credible-interval level. Default 0.95.
block_index	Integer or NULL. When the fit used the generic random-effect architecture (Stage 1), per-block correlation matrices are emitted as R_block[r,i,j] (block index r as the FIRST subscript), not R_hospital[i,j]/R_patient[i,j]/etc. Pass matrix_var = "R_block" and the 1-based block index here to look up a specific block's correlation matrix; leave NULL for 2-index variables (Omega, or legacy R_hospital etc. from a fit made with the old hardcoded 1re/2re/3re Stan models).

Value

A tibble with one row per off-diagonal class pair, or NULL if matrix_var was not estimated for this fit.

summarize_raw_data	<i>Summarize Raw Data</i>
--------------------	---------------------------

Description

Creates summary statistics for the original input dataset.

Usage

```
summarize_raw_data(data)
```

Arguments

data	Raw input data frame
------	----------------------

Value

List with raw data summaries

summarize_transformations	
	<i>Summarize Data Transformations</i>

Description

Documents all transformations applied to data values.

Usage

```
summarize_transformations(raw_data, processed_data, preprocessing_log)
```

Arguments

raw_data	Original data
processed_data	Processed data
preprocessing_log	Full preprocessing log

Value

List with transformation summaries

summary.amr_result	<i>Summary Method for AMR Preprocessing Results</i>
--------------------	---

Description

Prints a formatted summary of preprocessing pipeline results.

Usage

```
## S3 method for class 'amr_result'  
summary(object, ...)
```

Arguments

object	amr_result object from run_preprocess()
...	Additional arguments (not used)

Value

Invisibly returns the object

Examples

```
## Not run:
result <- run_preprocess(data)
summary(result)

## End(Not run)
```

```
validate_aggregate_inputs
```

Validate Pre-computed Aggregate Marginal Inputs

Description

Validates a pre-computed marginal resistance table (e.g. from GBD or a surveillance network) before it is passed to `estimate_profiles_convex()`. Checks prevalence bounds, duplicate rows, zero-denominator cells, impossible pairwise values, and minimum class counts per pathogen.

Usage

```
validate_aggregate_inputs(
  marginals,
  pairwise = NULL,
  pathogen_col = "pathogen",
  class_col = "antibiotic_class",
  rate_col = "marginal_resistance",
  n_tested_col = "n_tested",
  n_resistant_col = "n_resistant",
  class1_col = "antibiotic_class_1",
  class2_col = "antibiotic_class_2",
  pairwise_rate_col = "pairwise_resistance_prevalence",
  min_classes_per_pathogen = 2L
)
```

Arguments

<code>marginals</code>	Data frame with at minimum columns <code>pathogen_col</code> , <code>class_col</code> , and <code>rate_col</code> . Optionally also <code>n_tested_col</code> and <code>n_resistant_col</code> .
<code>pairwise</code>	Data frame or <code>NULL</code> . Pairwise co-resistance table with columns <code>pathogen_col</code> , <code>class1_col</code> , <code>class2_col</code> , and <code>pairwise_rate_col</code> . Default <code>NULL</code> .
<code>pathogen_col</code>	Character. Default <code>"pathogen"</code> .
<code>class_col</code>	Character. Default <code>"antibiotic_class"</code> .
<code>rate_col</code>	Character. Marginal resistance rate column. Default <code>"marginal_resistance"</code> .
<code>n_tested_col</code>	Character or <code>NULL</code> . Default <code>"n_tested"</code> .
<code>n_resistant_col</code>	Character or <code>NULL</code> . Default <code>"n_resistant"</code> .

class1_col Character. Default "antibiotic_class_1".
 class2_col Character. Default "antibiotic_class_2".
 pairwise_rate_col
 Character. Default "pairwise_resistance_prevalence".
 min_classes_per_pathogen
 Integer. Pathogens with fewer classes are flagged. Default 2L.

Value

Tibble of check results (same schema as `validate_profile_inputs()`). Stops on any "fail".

validate_complete_profile_calibration

Observed-versus-Model Complete-Profile Validation

Description

For hospital x pathogen panels (from `.resolve_profile_class_panel()`, the same approved-eligibility panel used for DALY profiles) with at least `min_complete_events` events that have **every** panel class actually observed (no imputation involved), compares the empirical complete-profile frequency distribution against the model-implied profile probability distribution for that same event cohort. With identity residuals it uses the analytic product of per-class $\Phi(\mu_{ed})$; with correlated residuals it uses correlated MVN simulation with L_{Ω} , as if the classes had not been observed – i.e. the model's unconditional prediction, deliberately NOT using `compute_event_profile_probabilities()`'s observed-cell-preserving logic, since the point here is to check whether the model's predictions agree with what was actually measured). Panels without enough complete events are skipped and the reason is recorded in the status column rather than silently omitted.

Usage

```
validate_complete_profile_calibration(
  fitted_model,
  n_posterior_draws_for_validation = 2000L,
  seed = 123L,
  ci_level = 0.95,
  min_complete_events = 30L,
  n_mc_profile_replicates = 200L
)
```

Arguments

fitted_model List returned by `fit_bayesian_multivariate_probit()`.
 n_posterior_draws_for_validation
 Integer. Posterior draws used. Default 2000L.
 seed Integer. Random seed (draw subsampling only). Default 123L.
 ci_level Numeric. Credible interval coverage. Default 0.95.

min_complete_events

Integer. Minimum number of fully-observed-panel events required to evaluate a hospital-pathogen panel. Default 30L.

n_mc_profile_replicates

Integer. Correlated residual structure only: inner Monte Carlo replicate count M used per posterior draw per complete event to estimate model-implied full-profile probabilities via $Z = \mu + L_{\Omega}\epsilon$, $Y = I(Z > 0)$ (the same simulation mechanism `.ppc_generate_correlated()` uses). Worst-case Monte Carlo SE on a cohort of n complete events is approximately $0.5/\sqrt{Mn}$; the default 200 keeps that under ~0.7pp even at the `min_complete_events` floor of 30, while remaining far cheaper than reusing `n_posterior_draws_for_validation` (e.g. 2000) as the inner replicate count would be. Ignored for identity residual structure, where profile probabilities are computed exactly (the independent product of $\Phi(\mu_d)$ terms), not simulated.

Value

Tibble with one row per hospital x pathogen x profile for evaluated panels (`status = "evaluated"`: `empirical_count/frequency`, `model_frequency_mean/lower/upper`, `absolute_error`, `interval_contains_observed`) and one row per skipped panel (`status` starts with `"skipped_"`, numeric columns NA).

validate_compute_backend

Validate a selectable Stan compute backend configuration

Description

Validates a compute configuration for CPU or OpenCL execution. This checks only package/toolchain availability and the structure of the requested configuration; it does not inspect machine-specific GPU inventories.

Usage

```
validate_compute_backend(compute = list())
```

Arguments

`compute` Named list. Supported fields are `backend`, `opencl_platform_id`, `opencl_device_id`, and `allow_cpu_fallback`.

Value

A canonical validated list with fields `backend`, `opencl_platform_id`, `opencl_device_id`, and `allow_cpu_fallback`.

validate_config	<i>Validate AMR Configuration</i>
-----------------	-----------------------------------

Description

Checks that configuration parameters are valid.

Usage

```
validate_config(config)
```

Arguments

config	An amr_config object
--------	----------------------

Value

Logical. TRUE if valid, stops with error if invalid.

validate_data_quality	<i>Validate Data Quality</i>
-----------------------	------------------------------

Description

Runs quality checks: minimum row count, required column presence, and maximum missing-value percentage per column.

Usage

```
validate_data_quality(  
  data,  
  min_rows = 10,  
  max_missing_pct = 50,  
  required_cols = c("patient_id", "organism_normalized"),  
  stop_on_failure = FALSE  
)
```

Arguments

data	Data frame to validate.
min_rows	Integer. Minimum acceptable number of rows. Default 10.
max_missing_pct	Numeric. Maximum acceptable percent missing per column (0-100). Default 50.
required_cols	Character vector. Columns that must be present.
stop_on_failure	Logical. Stop with error on failure. Default FALSE.

Value

List with quality assessment.

validate_marginal_calibration

Observed-versus-Model Marginal Resistance Validation

Description

For every hospital x pathogen x antibiotic-class cell with adequate observed testing support, compares the observed marginal resistance rate against the model's fitted probability $\Phi(\mu_{ed})$, averaged over the same tested-event cohort and over posterior draws. This is a model-validation calculation: it uses the fitted probability for TESTED cells (unlike `compute_event_profile_probabilities()`, which never overwrites an observed value). A calibration residual near 0 and interval coverage near the nominal CI level indicate the model reproduces the observed marginal resistance rates; it does not by itself validate joint (pairwise/profile) structure.

Usage

```
validate_marginal_calibration(
  fitted_model,
  n_posterior_draws_for_validation = 2000L,
  seed = 123L,
  ci_level = 0.95,
  min_tested = NULL,
  min_resistant = NULL,
  min_susceptible = NULL
)
```

Arguments

`fitted_model` List returned by `fit_bayesian_multivariate_probit()`.

`n_posterior_draws_for_validation` Integer. Posterior draws used. Default 2000L.

`seed` Integer. Random seed (draw subsampling only). Default 123L.

`ci_level` Numeric. Credible interval coverage. Default 0.95.

`min_tested, min_resistant, min_susceptible` Integer or NULL. Eligibility thresholds. When all NULL (default), eligibility is taken from `fitted_model$eligibility_report$marginal` (the same approved rules used for profile panels) rather than re-deriving new thresholds. Supplying any of these overrides that report and applies the supplied thresholds instead.

Value

Tibble, one row per eligible hospital x pathogen x class cell: `n_events`, `n_tested`, `n_resistant`, `n_susceptible`, `observed_resistance`, `model_resistance_mean/lower/upper`, `absolute_error`, `calibration_residual`, `interval_contains_observed`.

validate_pairwise_calibration

Observed-versus-Model Pairwise Co-resistance Validation

Description

For every adequately co-tested hospital x pathogen x class-pair combination, compares the observed pairwise resistance rate ($n_{RR}/n_{\text{cotested}}$) against the model's implied joint probability, computed differently by residual structure:

Identity $P(Y_{d_1} = 1, Y_{d_2} = 1 \mid \theta) = \Phi(\mu_{ed_1}) \Phi(\mu_{ed_2})$ – the two classes are combined only through the shared fixed and random effects in μ , not through any residual correlation (the identity model has none). **This checks whether the identity-residual model reproduces observed pairwise resistance through shared covariates and random effects; it is not evidence that a residual-independence assumption is correct.** Persistent pairwise miscalibration despite good marginal calibration suggests unmodelled class-to-class dependence that a correlated-residual fit might capture.

Correlated $P(Y_{d_1} = 1, Y_{d_2} = 1 \mid \theta) = \Phi_2(\mu_{ed_1}, \mu_{ed_2}; \rho_{d_1 d_2})$ – the standard bivariate-probit identity $P(Z_1 > 0, Z_2 > 0) = \Phi_2(\mu_1, \mu_2; \rho)$ for $(Z_1, Z_2) \sim N((\mu_1, \mu_2), [[1, \rho], [\rho, 1]])$, with $\rho_{d_1 d_2} = \Omega_{d_1 d_2}$ from the fitted residual correlation matrix (computed via **pbivnorm**). This actually uses the fitted correlation structure, unlike the identity-model product formula – checking whether Ω reproduces observed co-resistance.

Usage

```
validate_pairwise_calibration(
  fitted_model,
  n_posterior_draws_for_validation = 2000L,
  seed = 123L,
  ci_level = 0.95,
  min_cotested = NULL
)
```

Arguments

fitted_model	List returned by <code>fit_bayesian_multivariate_probit()</code> .
n_posterior_draws_for_validation	Integer. Posterior draws used. Default 2000L.
seed	Integer. Random seed (draw subsampling only). Default 123L.
ci_level	Numeric. Credible interval coverage. Default 0.95.
min_cotested	Integer or NULL. Eligibility threshold. When NULL (default), eligibility is taken from <code>fitted_model\$eligibility_report\$pairwise</code> .

Value

Tibble, one row per eligible hospital x pathogen x class-pair: `n_cotested`, `observed_RR/RS/SR/SS`, `observed_pairwise_resistance`, `model_pairwise_mean/lower/upper`, `absolute_error`, `interval_contains_observed`

 validate_profile_inputs

Validate Inputs for Resistance Profile Estimation (Pathway 1)

Description

Checks that all mandatory columns exist, detects long vs wide format, validates AST content, and verifies that columns required for any requested stratification are present. Returns a tibble of check results and stops on any failed mandatory check.

Usage

```
validate_profile_inputs(
  data,
  col_map = list(isolate_col = "isolate_id", pathogen_col = "pathogen", ast_col =
    "ast_value", patient_col = "patient_id", date_col = "date_of_culture", geography_col
    = "state", specimen_col = "specimen_type", age_col = "age", dob_col = "dob",
    antibiotic_col = "antibiotic_name", class_col = "antibiotic_class", location_col =
    NULL, outcome_col = NULL),
  stratify_by = NULL,
  outcome_col = NULL
)
```

Arguments

data	Data frame to validate.
col_map	Named list mapping logical roles to actual column names. Defaults use standard package column names. Set any entry to NULL to skip that check (only valid for optional roles).
stratify_by	Character vector or NULL. Requested stratification dimensions. Accepted values: "geography", "year", "outcome". Triggers additional column checks for each dimension. Default NULL.
outcome_col	Character or NULL. Outcome column name. Required when "outcome" is in stratify_by. Can also be supplied via col_map\$outcome_col. Default NULL.

Details

Mandatory columns (always):

- isolate_col – unique isolate identifier; the counting unit for all resistance statistics. One patient may have multiple isolates.
- pathogen_col – organism / pathogen name.
- ast_col – susceptibility result; must contain S / I / R.
- patient_col – patient identifier (retained for context, not counted).
- date_col – culture date; used to derive year when "year" is in stratify_by.

- geography_col – geographic unit (state / district).
- specimen_col – specimen type (blood, urine, etc.).
- age_col OR dob_col – at least one must be present.

Mandatory in long format only:

- antibiotic_col – antibiotic / drug name column.

Optional (warn if absent, do not stop):

- class_col – antibiotic class; derived from antibiotic_col via WHO AWaRe reference if absent.
- location_col – ward / ICU / OPD; retained for subgroup outputs.
- outcome_col – patient outcome; required for outcome stratification.

Value

A tibble with columns check_name, status ("pass" / "warn" / "fail"), message, and n_affected. The attribute detected_format ("long" or "wide") is attached. Stops with an informative message if any mandatory check fails.

Examples

```
## Not run:
report <- validate_profile_inputs(
  data = my_ast_data,
  col_map = list(
    isolate_col = "isolate_id",
    pathogen_col = "organism",
    ast_col = "result",
    patient_col = "pid",
    date_col = "culture_date",
    geography_col = "state",
    specimen_col = "specimen",
    age_col = "age_years",
    dob_col = NULL,
    antibiotic_col = "drug_name",
    class_col = NULL,
    location_col = "ward",
    outcome_col = "discharge_status"
  ),
  stratify_by = c("geography", "year"),
  outcome_col = "discharge_status"
)

## End(Not run)
```

`validate_required_fields`*Validate required fields exist and meet completeness threshold*

Description

Validate required fields exist and meet completeness threshold

Usage

```
validate_required_fields(  
  data,  
  required_cols,  
  stop_on_failure = TRUE,  
  allow_na = FALSE,  
  min_completeness = 0.8  
)
```

Arguments

<code>data</code>	Data frame.
<code>required_cols</code>	Character vector of required column names.
<code>stop_on_failure</code>	Logical. Stop if validation fails. Default TRUE.
<code>allow_na</code>	Logical. Skip completeness checks if TRUE. Default FALSE.
<code>min_completeness</code>	Numeric (0-1). Minimum completeness required. Default 0.8.

Value

List with validation result details.

Index

* datasets

- default_column_mappings, 76
- aggregate_profiles_for_daly, 6
- amr_config, 8
- assess_gibbs_profile_stability, 10
- bootstrap_profiles_convex, 11
- build_constraint_matrix, 13
- check_profile_constraints, 14
- class_display_label, 16
- class_pair_label, 17
- class_short_label, 17
- compare_distribution_aic, 18
- compute_event_profile_probabilities, 18
- compute_hospital_daly, 20
- compute_marginal_resistance, 21
- compute_marginals_from_data, 23
- compute_pairwise_coresistance, 24
- compute_pairwise_from_data, 26
- compute_polymicrobial_weight (prep_compute_poly_weights), 155
- compute_posterior_predictive_status, 27
- compute_prior_predictive_fingerprint, 28
- compute_prior_predictive_status, 29, 209
- compute_probit_ppc_statistics, 28, 30, 127, 208, 209
- compute_processing_statistics, 32
- compute_profile_validation_status, 32
- compute_resistance_profiles, 34
- daly_add_rr_mappings, 37
- daly_assign_rr_to_profiles, 37
- daly_calc_case_fatality, 39
- daly_calc_cr_l, 40
- daly_calc_deaths_by_cause, 42
- daly_calc_deaths_by_syndrome, 43
- daly_calc_incidence_from_cfr, 46
- daly_calc_infection_fraction, 47
- daly_calc_paf_los, 48
- daly_calc_paf_mortality, 49
- daly_calc_pathogen_fraction_fatal, 51
- daly_calc_resistance_prevalence_fatal, 52
- daly_calc_syndrome_fraction, 53
- daly_calc_yld_associated, 54
- daly_calc_yld_attributable, 56
- daly_calc_yll_associated, 58
- daly_calc_yll_attributable, 61
- daly_compute_patient_los, 63
- daly_count_incident_cases, 65
- daly_derive_hai_cai_for_mortality, 66
- daly_filter_profiles_to_rr_classes, 67
- daly_fit_los_rr, 68, 72
- daly_fit_los_rr_distribution, 70
- daly_fit_mortality_rr, 72
- daly_get_top_pathogens, 75
- daly_load_rr_reference, 76
- default_column_mappings, 76
- detect_preprocessing_capabilities, 77
- eda_theme, 77
- enumerate_binary_profiles, 30, 78
- estimate_profiles_convex, 78
- estimate_resistance_profiles, 80
- export_report, 82
- fit_bayesian_multivariate_probit, 29, 83, 206, 207, 209
- fit_distributions, 86
- flag_polymicrobial (prep_flag_polymicrobial), 174
- generate_preprocessing_report, 87

- get_age_bins, 88
- get_los_by_resistance, 88
- hospital_display_label, 89
- infer_patient_syndrome_wide, 90
- mask_and_validate_ast, 91
- plot_abx_complex_heatmap, 93
- plot_abx_heatmap, 93, 95
- plot_abx_susceptibility, 97
- plot_age_ridge, 98
- plot_burden_by_hospital, 100
- plot_burden_by_organism, 101
- plot_death_discharged, 103
- plot_hai_cai_by_facility, 105
- plot_infection_type_by_location, 106
- plot_location_by_facility, 108
- plot_los_by_agebin, 109
- plot_los_distributions, 111
- plot_los_ridge, 112
- plot_mono_poly_by_facility, 113
- plot_omega_convergence_heatmap, 115
- plot_omega_correlation_heatmap, 115
- plot_omega_degeneracy_diagnostic, 116
- plot_omega_summary_table, 117
- plot_outcome_by_agebin, 118
- plot_outcome_by_organism, 119
- plot_outcome_by_year, 121
- plot_outcome_distribution, 123
- plot_patients_by_hospital, 124
- plot_probit_beta_family_diagnostics, 125
- plot_probit_diagnostics, 126
- plot_probit_fixed_effect_diagnostics, 126
- plot_probit_posterior_predictive_checks, 127
- plot_probit_worst_offender_diagnostics, 128
- plot_probit_worst_parameters, 129
- plot_resistance_by_agebin, 129
- plot_resistance_by_organism, 131
- plot_resistance_by_sample, 132
- plot_syndrome_distribution, 134
- plot_top_organisms, 135
- plot_yld_heatmap, 136
- plot_yll_heatmap, 137
- prep_apply_column_map, 139
- prep_assert_standard_names, 139
- prep_assign_age_bins, 140
- prep_assign_facility_type, 141
- prep_assign_organism_group, 142
- prep_assign_patient_syndrome, 143
- prep_attrition_flow, 144
- prep_build_column_map, 145
- prep_build_fatal_cohort, 146
- prep_build_nonfatal_cohort, 146
- prep_check_columns, 147
- prep_check_keys, 148
- prep_check_organism_ast_consistency, 148
- prep_classify_antibiotic_class, 149
- prep_classify_aware, 150
- prep_classify_mdr_xdr, 150
- prep_classify_mortality, 151
- prep_classify_readmission, 152
- prep_clean_ast_values, 153
- prep_clean_optional_columns, 153
- prep_coerce_dates, 154
- prep_collapse_class_level, 154
- prep_compute_poly_weights, 155
- prep_create_event_ids, 156
- prep_create_resistance_profile, 158
- prep_create_wide_ast_matrix, 159
- prep_decode_antibiotic_code, 160
- prep_deduplicate_ast, 161
- prep_deduplicate_events, 162
- prep_derive_dob_from_components, 163
- prep_derive_hai_cai, 164
- prep_derive_icu_flag, 165
- prep_derive_los_from_dates, 165
- prep_detect_schema_drift, 166
- prep_diagnosis_text, 167
- prep_extract_genus, 168
- prep_extract_species, 168
- prep_fill_age, 169
- prep_filter_analysis_ready, 169
- prep_filter_fungal, 170
- prep_filter_minimally_usable, 171
- prep_flag_contaminants, 172
- prep_flag_hai_inferred, 173
- prep_flag_invalid_ast, 173
- prep_flag_organism_unmatched, 174
- prep_flag_polymicrobial, 174
- prep_flag_readmission, 175

prep_get_contaminant_list, 176
prep_harmonize_ast, 177
prep_infer_department, 178
prep_inventory_columns, 178
prep_is_contaminant, 179
prep_log_source, 179
prep_map_diagnosis_to_icd, 180
prep_map_icd_to_syndrome, 181
prep_missingness_report, 182
prep_parse_date_column, 183
prep_pivot_ast_wide_to_long, 183
prep_recode_intermediate_ast, 185
prep_reconcile_hai_observed_inferred, 185
prep_report_capabilities, 186
prep_split_poly_episode, 187
prep_standardize_antibiotics, 187
prep_standardize_column_names, 188
prep_standardize_final_outcome, 189
prep_standardize_infection_type, 190
prep_standardize_organisms, 190
prep_standardize_outcome, 191
prep_standardize_sex, 191
prep_standardize_specimens, 192
prep_validate_analysis_ready, 192
prep_validate_date_logic, 193
prep_validate_table, 194
prepare_los_data, 195
prepare_random_effects, 196, 207
preprocess_for_profiles, 197
print.amr_config, 199
print.amr_preprocessing_report, 199
print.amr_result, 200

re_contribution, 200, 207
round_to_sum, 201
run_preprocess, 201

safe_fit, 203
select_resistance_class, 203
shorten_drug_class, 204
simulate_probit_mixed_predictive, 205, 207
simulate_probit_posterior_predictive, 30, 31, 127, 205, 206, 206, 208, 209
simulate_probit_prior_predictive, 28, 29, 208
summarise_distribution, 210
summarize_column_mapping, 210
summarize_data_quality, 211
summarize_fit_correlation_matrix, 211
summarize_raw_data, 212
summarize_transformations, 213
summary.amr_result, 213

validate_aggregate_inputs, 214
validate_complete_profile_calibration, 215
validate_compute_backend, 216
validate_config, 217
validate_data_quality, 217
validate_marginal_calibration, 218
validate_pairwise_calibration, 219
validate_profile_inputs, 220
validate_required_fields, 222