Package: mlumr (via r-universe)

October 13, 2026

Title Multilevel Unanchored Meta-Regression for Indirect Treatment Comparisons

Version 0.2.0

Description Bayesian multilevel unanchored meta-regression (ML-UMR) for indirect treatment comparisons using individual patient data (IPD) and aggregate data (AgD). Implements shared prognostic factor assumption (SPFA) and relaxed SPFA models for binary, continuous, count, and time-to-event (survival) outcomes via 'Stan', including parametric and flexible M-spline survival baselines. Also provides a one-arm simulated treatment comparison (STC) via comparator-population G-computation and naive unadjusted benchmarks. ML-UMR is an adaptation of the ML-NMR methodology (Phillippo et al. 2020, <doi:10.1111/rssa.12579>) implemented in the 'multinma' package (GPL-3) to the unanchored two-trial case. The public API uses familiar names for data setup, integration and effect summaries, but ML-NMR and ML-UMR specifications and fitted objects are not interchangeable because their evidence structures and models differ.

License GPL-3

URL https://github.com/choxos/mlumr, https://choxos.github.io/mlumr/

BugReports https://github.com/choxos/mlumr/issues

Encoding UTF-8

LazyData true

LazyDataCompression xz

Roxygen list(markdown = TRUE)

Biarch true

Depends R (>= 4.1.0)

Imports methods, stats, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), rstantools (>= 2.3.0), randtoolbox, survival (>= 3.1-0), splines2 (>= 0.4.1), ggplot2 (>= 3.4.0)

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

Suggests testthat (>= 3.2.0), ggsurvfit, knitr, rmarkdown, posterior, bayesplot, loo, Matrix, cmdstanr, withr, flexsurv, detectseparation, xfun, R.rsp

Additional_repositories https://mc-stan.org/r-packages

SystemRequirements GNU make

Config/testthat/edition 3

VignetteBuilder knitr, R.rsp

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make

Repository https://choxos.r-universe.dev

Date/Publication 2026-10-13 15:05:17 UTC

RemoteUrl https://github.com/choxos/mlumr

RemoteRef HEAD

RemoteSha ae574a035c2599852aef61740cecf1b2727fa4a6

Contents

add_integration . . . 3
calculate_dic . . . 5
calculate_loo . . . 5
calculate_waic . . . 7
caries_agd . . . 8
caries_ipd . . . 8
check_diagnostics . . . 10
check_identification . . . 10
check_integration . . . 11
combine_data . . . 12
compare_models . . . 13
conditional_effects . . . 14
conditional_predict . . . 16
dbern . . . 17
default_priors . . . 17
distr . . . 18
GammaDist . . . 19
geom_km . . . 20
logitNormal . . . 21
make_knots . . . 22
marginal_effects . . . 23
mlumr . . . 26
mlumr_engine . . . 30
mlumr_forest . . . 31

naive . . . 32
ndmm_agd . . . 33
ndmm_agd_covs . . . 34
ndmm_ipd . . . 35
pbern . . . 36
plot.mlumr_conditional_effects . . . 36
plot.mlumr_marginal_effects . . . 37
plot.mlumr_prediction . . . 38
plot_prior_posterior . . . 39
predict.mlumr_fit . . . 40
prior_cauchy . . . 42
prior_exponential . . . 42
prior_normal . . . 43
prior_sensitivity . . . 44
prior_student_t . . . 46
prior_summary . . . 47
psoriasis_agd . . . 48
psoriasis_ipd . . . 48
qbern . . . 49
set_agd . . . 50
set_agd_surv . . . 52
set_ipd . . . 54
shoulder_agd . . . 55
shoulder_ipd . . . 56
stc . . . 57
unnest_integration . . . 59

Index **60**

`add_integration` *Add numerical integration points*

Description

Generate quasi-Monte Carlo integration points using Sobol sequences and a Gaussian copula to account for correlations between covariates in the AgD.

Usage

```
add_integration(
  data,
  n_int = 64,
  cor = NULL,
  cor_adjust = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

data	An mlumr_data object from combine_data()
n_int	Number of integration points (default 64; use powers of 2). More points can improve quasi-Monte Carlo integration of the AgD likelihood and comparator-population estimands. Increase it when check_integration() shows numerical sensitivity. Wider posterior intervals alone indicate neither an inadequate grid nor a need for more points. Larger values cost more sampling time.
cor	Correlation matrix for covariates, on the covariate scale. If NULL (the default) it is estimated from the IPD; see the correlation-transport note in Details.
cor_adjust	Adjustment method: "spearman", "pearson", or "none"
verbose	Logical; if FALSE, suppresses progress messages.
...	Distribution specifications for each covariate using distr()

Details

The correlation structure is assumed to transport. Aggregate data report marginal summaries only, so with cor = NULL the within-row correlation is estimated from the IPD and applied to every comparator row, as in ML-NMR (Phillippo et al. 2020). The assumption is untestable from the data; supply cor from an external source to vary it, and use check_integration() to confirm the realized moments and correlations.

cor_adjust maps the covariate-scale correlation onto the Gaussian copula: the Spearman map is exact for continuous margins, the Pearson map holds for Gaussian margins, and pairs involving a binary margin use prevalence-independent heuristics. A nonbinary discrete margin (a count or an ordered category) is treated as continuous and its realized association need not match the target; add_integration() warns when it sees one. "none" passes a latent Gaussian-copula matrix through unchanged.

Value

An mlumr_data object with integration points added

Examples

```
## Not run:
dat <- add_integration(
  dat,
  n_int = 64,
  x1 = distr(qnorm, mean = x1_mean, sd = x1_sd),
  x2 = distr(qbern, prob = x2_mean)
)

## End(Not run)
```

calculate_dic — *Calculate DIC for model comparison*

Description

Computes the Deviance Information Criterion using the variance-based effective-parameters formula from Gelman et al. (2004): `pD = 0.5 * Var(D)`. This is more stable than the plug-in alternative for multimodal posteriors.

Usage

```
calculate_dic(object)
```

Arguments

`object`	An `mlumr_fit` object

Details

DIC is retained for backward compatibility and rough comparison. For principled Bayesian model comparison, prefer `calculate_loo()` or `calculate_waic()` (Vehtari, Gelman, Gabry 2017).

Value

A list of class `mlumr_dic` with components `DIC`, `pD`, `D_bar`, `n_obs` and `model`.

Examples

```
## Not run:
dic_spfa <- calculate_dic(fit_spfa)
dic_relaxed <- calculate_dic(fit_relaxed)

## End(Not run)
```

calculate_loo — *Calculate LOO-CV for an mlumr_fit*

Description

Computes approximate leave-one-out cross-validation (PSIS-LOO, Vehtari, Gelman, Gabry 2017) using the pointwise log-likelihoods stored by the Stan models. Returns a `loo` object from the `loo` package.

Usage

```
calculate_loo(
  object,
  survival_unit = c("observation", "arm", "aggregate"),
  ...
)
```

Arguments

`object`	An `mlumr_fit` object.
`survival_unit`	For survival fits, the LOO/WAIC pointwise unit: `"observation"` (default; per reconstructed comparator pseudo-individual, optimistic), `"arm"` (group the comparator pseudo-IPD by comparator arm, so each external arm is one held-out unit), or `"aggregate"` (all comparator pseudo-IPD as a single external-evidence unit). The index IPD always stays per-individual. Ignored for non-survival families.
`...`	Further arguments passed to `loo::loo()`; `r_eff` is computed from the fit's chains.

Details

Pareto-k diagnostics: values > 0.7 indicate observations for which the PSIS approximation is unreliable; the printed output flags these. Typical remedies are running more iterations or, for highly influential AgD rows, refitting without the offending observation to check sensitivity. Moment matching (`loo::loo_moment_match()`) needs the fitted model rather than a log-likelihood matrix, so `moment_match` is refused.

Value

An object of class `psis_loo` (see `loo::loo()`).

Note

AgD rows are treated as independent observations. Subgroup rows from one study share no clustering term, so their effective sample sizes are inflated and Pareto-k warnings understated; corroborate with `prior_sensitivity()` or by refitting without suspect rows.

Survival fits. The comparator enters as reconstructed pseudo-IPD, so the default pointwise unit is one pseudo-individual and the criteria are optimistic relative to leaving out the comparator arm. Set `survival_unit = "arm"` or `"aggregate"` to hold out whole comparator arms or all of the external evidence instead.

Examples

```
## Not run:
loo_spfa <- calculate_loo(fit_spfa)
print(loo_spfa)

## End(Not run)
```

calculate_waic *Calculate WAIC for an mlumr_fit*

Description

Watanabe-Akaike Information Criterion (Watanabe 2010) based on the pointwise log-likelihoods. WAIC is asymptotically equivalent to LOO-CV; prefer `calculate_loo()` when Pareto-k is well-behaved.

Usage

```
calculate_waic(
  object,
  survival_unit = c("observation", "arm", "aggregate"),
  ...
)
```

Arguments

`object`	An `mlumr_fit` object.
`survival_unit`	For survival fits, the WAIC pointwise unit: `"observation"` (default), `"arm"`, or `"aggregate"` (see `calculate_loo()` for details). Ignored for non-survival families.
`...`	Further arguments passed to `loo::waic()`.

Value

An object of class `waic` (see `loo::waic()`).

Note

As with `calculate_loo()`, each AgD row is treated as an independent observation. WAIC will be optimistic for AgD rows that share a study (see the note on `calculate_loo()`). For **survival** fits the AgD pointwise unit is a reconstructed pseudo-individual, so WAIC is at the pseudo-individual level (see the survival note on `calculate_loo()`).

Examples

```
## Not run:
waic_spfa <- calculate_waic(fit_spfa)

## End(Not run)
```

caries_agd — *Dental caries: simulated comparator aggregate data (count)*

Description

Aggregate summary of the comparator treatment (nano-silver fluoride, NSF) for the count ML-UMR example; pair with `caries_ipd`. Not real patient data.

Usage

```
caries_agd
```

Format

A data frame with 1 row and 10 columns: `study`, `treatment`, `n`, count outcome `r` (total dmft) with exposure `E` (n children), and covariate summaries (`age_mean`/`age_sd`, `gender_prop` (a proportion), `log_cfu_mean`/`log_cfu_sd`).

Source

Simulated; see `caries_ipd` for the modeling basis.

caries_ipd — *Dental caries: simulated index IPD (count)*

Description

Simulated individual patient data for the index treatment (silver diamine fluoride, SDF) in a count-outcome ML-UMR example; pair with `caries_agd`. Not real patient data.

Usage

```
caries_ipd
```

Format

A data frame with 103 rows and 8 columns:

study study label (`"Ammar 2025"`)

treatment index treatment label (`"SDF"`)

subject row identifier

age age in years

gender 1/0 indicator, as coded in the source trial, which does not record which level is which. Only the covariate's distribution matters for the adjustment, so the labeling does not affect the estimand.

log_cfu log baseline salivary *S. mutans* count, `log(CFU/mL + 1)`

dmft decayed-missing-filled-teeth count (the count outcome)

exposure Poisson offset, 1 per child. `dmft` is a whole-mouth count with no time at risk and no per-tooth denominator, so the rate is "dmft per child" and the offset is 1. The column is present because `set_ipd` requires `exposure` for the Poisson family; `log(1) = 0`, so it contributes nothing to the linear predictor. The comparator's `E` is the matching total, one unit per child.

`log_cfu` is bimodal: 9 of the 103 children (8.7\ detectable baseline count and so sit at exactly 0, and the rest are concentrated near 10.9 (SD 0.8). The comparator arm declares this covariate as a single normal distribution (`log_cfu_mean` 11.0, `log_cfu_sd` 0.7) because the source trial's comparator arm contains no such zeros. Integrating a normal over the comparator therefore puts almost no weight where the index arm's zero mode sits, so effects adjusted for `log_cfu` extrapolate over that part of the covariate space. This is a property of the source trial, not of the synthesis; treat it as a worked illustration of a covariate whose arms are not fully overlapping.

Details

The index and comparator arms come from the same trial, so both carry the study label `"Ammar 2025"`. Splitting one randomized trial into a single-arm IPD source and a single-arm aggregate source is what makes this an unanchored example that still has a full-data comparison to be checked against: fitting the two arms together on the complete synthetic records gives the quantity the unanchored methods are trying to recover once one arm has been reduced to summaries. That is a data-reduction benchmark. It is not a randomized reference and not a known population causal effect. The generator draws treatment first and then synthesizes the covariates conditional on it, so the synthetic arms are not a fresh randomized assignment independent of the generated baseline variables; and the full-data estimate carries its own sampling error. Validating an estimator against a causal target would need a specified covariate distribution, potential-outcome mechanism, assignment rule and estimand, which these records do not provide. Here the treatment contrast itself was imposed: `dmft` was a balanced baseline characteristic in the source trial and the index arm's synthetic counts were thinned binomially to create an effect, so the benchmark recovers a contrast this package manufactured, not one the trial reported. Because the label is shared, `combine_data` warns that IPD and AgD come from the same study; that warning is expected here and is the honest reading of the data. It does not fire for `psoriasis_ipd` or `ndmm_ipd`, whose arms really do come from different trials.

Source

Simulated (not real patient data), generated with the **synthpop** package (sequential CART; Nowok, Raab and Dibben 2016, doi:10.18637/jss.v074.i11) from the silver-diamine-fluoride vs nano-silver-fluoride dental caries RCT (Ammar et al., *BMC Oral Health* 2025;25:945, CC BY 4.0; data Synapse syn43185346), preserving the covariate relationships; fidelity validated with the **syntheticdata** package. A plausible caries-arresting SDF effect was imposed for illustration (dmft was a balanced baseline characteristic in the source trial). See `data-raw/simulate_external_data.R`.

check_diagnostics *Check the sampler diagnostics of a fit*

Description

Warns about divergent transitions, iterations that hit the maximum tree depth, chains that did not come back, split-Rhat above 1.01 or 1.05, and bulk or tail effective sample sizes below 400. A diagnostic the backend did not supply is reported as unavailable rather than read as clean. `mlumr()` runs this check after sampling; call it again on a stored fit to see the same verdict.

Usage

```
check_diagnostics(fit)
```

Arguments

`fit`	An `mlumr_fit` object.

Value

`NULL`, invisibly; called for its warnings and messages.

See Also

`mlumr()`, `prior_sensitivity()`.

check_identification *Can the aggregate data identify the comparator coefficients?*

Description

In the relaxed model the comparator coefficients `beta_comparator` are informed only by the aggregate rows. With `K` covariates there are `K + 1` comparator parameters, so at least `K + 1` distinct aggregate rows are needed, and under an identity link the rows must also differ in every covariate direction.

Usage

```
check_identification(x, verbose = TRUE, link = NULL)
```

Arguments

`x`	An `mlumr_data` object or a fitted relaxed `mlumr_fit`.
`verbose`	Print a readable report (default `TRUE`).
`link`	Planned link for an unfitted data object. Defaults to the family default. A fitted object always uses its stored link.

Details

The subgroup mean profiles are centered, divided by the IPD covariate SDs and decomposed. `cond_inv` is the ratio of the smallest to the largest singular value and goes to 0 as the rows collapse onto a lower-dimensional set. `eff_dim` is the participation ratio of the squared singular values, the number of directions the rows effectively spread along, from 1 to `K`; it is 0 when the rows do not vary or cannot be decomposed. `spread` is the RMS distance of the rows from their center along the dominant direction, in IPD SDs; it supplies the absolute scale `cond_inv` lacks. For a normal identity-link model the subgroup means are the aggregate design and the screen flags `cond_inv` < `0.2` or `spread` < `0.05`, which are package heuristics. For other links the integrated response also depends on each row's covariate distribution, so the geometry is descriptive only and `flagged` is `NA` unless there are too few rows. Reconstructed survival curves are refused, since a curve is not one scalar summary per row. Neither measure sees subgroup sizes or outcome precision, so confirm any verdict with the coefficient posterior and `prior_sensitivity()`. The subgroup-identification vignette works through the cases.

Value

Invisibly, a list with `n_rows`, `n_distinct` (rows that do not repeat another's integration grid), `n_cov`, `n_rows_needed` (`K` + `1`), `cond_inv`, `eff_dim`, `spread`, `singular_values`, `means` (the scaled, centered subgroup mean matrix), `diagnostic_scope` (`"identity"` or `"descriptive"`) and `flagged`.

See Also

`mlumr()` for `model = "relaxed"`; `prior_sensitivity()`.

Examples

```
## Not run:
dat <- add_integration(combine_data(ipd, agd), n_int = 64, ...)
check_identification(dat)

## End(Not run)
```

---

check_integration *Check integration point adequacy*

Description

Compare integration results at the current `n_int` against a doubled resolution to assess numerical accuracy. Large discrepancies indicate that `n_int` should be increased. Because the Sobol sequence is nested (the doubled set contains the current set), this current-vs-doubled difference is a convergence heuristic, not an error bound. Agreement between the two grids does not establish accuracy for rare discrete margins or for a final treatment-effect estimand.

Usage

```
check_integration(
  data,
  ...,
  cor = NULL,
  cor_adjust = NULL,
  check_joint = TRUE,
  verbose = TRUE
)
```

Arguments

`data`	An `mlumr_data` object with integration points
`...`	Distribution specifications (same as passed to `add_integration()`)
`cor`	Correlation matrix (same as passed to `add_integration()`)
`cor_adjust`	Adjustment method (same as passed to `add_integration()`)
`check_joint`	If `TRUE` (default), also compare pairwise correlation matrices between the current and doubled `n_int`, and the maximum per-AgD-row absolute deviation from the user-supplied `cor`. The pairwise comparison catches cases where marginals converge but joint dependence structure does not (rare in practice for QMC with sensible `cor_adjust` but worth flagging when `n_int` is small).
`verbose`	Logical; if `FALSE`, suppresses printed diagnostic messages.

Value

A list with `marginals`, a data frame of grid means and SDs at the current and doubled `n_int` against the declared targets, and `verdict`, whose entries are `"stable"` or `"close"` when a comparison met the heuristic, `"review"` when it did not, `"partial"` when the measured correlation pairs passed but some pair could not be measured, and `"unavailable"` when there was nothing finite to compare (a latent matrix under `cor_adjust = "none"` is never compared). With `check_joint = TRUE` and two or more covariates it also holds `correlations`, the pairwise correlations per AgD row on both grids, and `correlation_pairs`, which counts the pairs measured and names the rest. A binary margin's target SD is `sqrt(p * (1 - p))` from the declared mean, and grid SDs are population SDs.

`combine_data` *Combine IPD and AgD for unanchored comparison*

Description

Combine IPD and AgD for unanchored comparison

Usage

```
combine_data(ipd, agd)
```

Arguments

ipd	An mlumr_ipd object from set_ipd()
agd	An mlumr_agd object from set_agd()

Value

An object of class mlumr_data

Examples

```
## Not run:
dat <- combine_data(ipd, agd)

## End(Not run)
```

---

compare_models *Compare fitted ML-UMR models*

Description

Compare two or more mlumr_fit objects by DIC (default), LOO, or WAIC. For LOO/WAIC, loo::loo_compare() is used under the hood; the output is the standard loo_compare table. The pointwise scores are paired observation by observation, so the fits must be of the same data in the same row order; fits whose stored outcomes differ are refused. For DIC the return is a data frame ordered by DIC.

Usage

```
compare_models(
  ...,
  criterion = c("dic", "loo", "waic"),
  survival_unit = c("observation", "arm", "aggregate")
)
```

Arguments

...	Two or more mlumr_fit objects. For DIC, mlumr_dic objects are also accepted.
criterion	One of "dic" (default), "loo", or "waic". LOO and WAIC require the optional loo package.
survival_unit	For survival fits compared by "loo"/"waic", the pointwise unit forwarded to calculate_loo() / calculate_waic(): "observation" (default; per reconstructed comparator pseudo-individual, optimistic), "arm", or "aggregate". Choose "arm" or "aggregate" to select on whole-external-arm predictive fit. Ignored for non-survival families and for criterion = "dic".

Details

DIC is the default for backward compatibility and because it has no additional package dependencies. For principled Bayesian model comparison, LOO (Vehtari, Gelman, Gabry 2017) is preferred and requires the optional `loo` package.

Value

For `"loo"` / `"waic"`: a `compare.loo` table from `loo::loo_compare()`. For `"dic"`: a data frame (invisibly) with columns `Model`, `DIC`, `pD`, `Delta_DIC`.

---

conditional_effects *Conditional treatment effects*

Description

Compute conditional (individual-level) treatment effects at specific covariate values from a fitted ML-UMR model. Unlike `marginal_effects()`, which averages over a population's covariate distribution, conditional effects evaluate the treatment effect at a particular covariate profile.

Usage

```
conditional_effects(
  object,
  newdata = NULL,
  effect = "all",
  summary = TRUE,
  probs = c(0.025, 0.5, 0.975)
)
```

Arguments

`object` An `mlumr_fit` object

`newdata` Data frame of covariate values at which to compute effects. Each row defines one covariate profile. Column names must match the covariates used in fitting. If `NULL` (default), uses the covariate means from the IPD as a single reference profile.

`effect` Which effect measure. For binomial: `"all"`, `"link_effect"`, `"rd"`, or `"rr"`. For normal: `"all"` or `"md"`. For Poisson: `"all"` or `"rr"`. The legacy value `"lor"` is accepted as an alias for `"link_effect"` when the fitted link is logit.

For **survival**, `exp(eta_index - eta_comparator)` is a conditional hazard ratio (`"hr"`, proportional-hazards distributions) or time ratio (`"tr"`, accelerated failure time distributions) only when the two studies share a baseline shape (`aux_by = "none"`, or any exponential fit). The two are different measures: `"tr"` on a proportional-hazards fit and `"hr"` on an AFT fit are errors, whose message gives the conversion where one exists (`TR = HR^(-1/shape)` for a Weibull, applied draw by draw; `TR = 1/HR` for an exponential). Under the default study-specific

shapes an explicit "hr" or "tr" is an error, since the baseline ratio does not cancel; "all" returns the contrast under the label "EXP_ETA_CONTRAST" with a warning. The RMST effects from marginal_effects() are the recommended alternative, and predict(type = "loghr") gives the population-standardized curve.

summary — Return summary statistics (TRUE) or full posterior draws (FALSE)

probs — Quantiles for summary (default c(0.025, 0.5, 0.975))

Details

For SPFA models, the conditional link-scale treatment effect is constant across all covariate values because the shared beta cancels in the treatment contrast on the fitted link scale. However, risk difference (RD) and risk ratio (RR) still vary with covariates because they depend on absolute probability levels. For relaxed models, all conditional effects vary with covariate values because the index and comparator treatments have different regression coefficients.

For binomial, normal and Poisson the conditional link-scale effect is eta_index - eta_comparator; for survival the contrast is exponentiated (null 1) and labeled "EXP_ETA_CONTRAST" when the baseline shapes differ. A conditional effect is evaluated at one profile, so unlike marginal_effects() and predict.mlumr_fit() there is no averaging over a population and no gap between E[g^{-1}(eta)] and g^{-1}(E[eta]).

Value

A data frame. If summary = TRUE, contains columns profile, effect, mean, sd and quantile columns. If summary = FALSE, returns a single combined data frame of full posterior draws with a profile column indicating which covariate profile each draw belongs to.

See Also

marginal_effects() for population-averaged treatment effects; conditional_predict() for absolute predictions at specific profiles; predict.mlumr_fit() for population-level predictions.

Examples

```
## Not run:
# Conditional effects at IPD covariate means (default)
conditional_effects(fit)

# At specific covariate values
conditional_effects(fit, newdata = data.frame(age = 60, sex = 1))

# Multiple profiles
profiles <- data.frame(age = c(50, 60, 70), sex = c(0, 0, 1))
conditional_effects(fit, newdata = profiles)

## End(Not run)
```

conditional_predict *Conditional predictions*

Description

Generate absolute predictions at specific covariate values for both treatments.

Usage

```
conditional_predict(
  object,
  newdata = NULL,
  type = NULL,
  summary = TRUE,
  probs = c(0.025, 0.5, 0.975)
)
```

Arguments

object	An mlumr_fit object
newdata	Data frame of covariate values. If NULL, uses IPD covariate means.
type	"response" for probabilities, means, or rates; "link" for the fitted linear-predictor scale. NULL, the default, resolves to "response" for binomial, normal and Poisson fits. Ignored for survival fits, which return the conditional survival probability S(t | x) at each fitted prediction time.
summary	Return summary (TRUE) or full draws (FALSE)
probs	Quantiles for summary

Value

A data frame with predictions for each treatment at each profile. For survival fits the summary has one row per profile, treatment, and time; with summary = FALSE the draws are rows and the prediction times are t_* columns.

See Also

conditional_effects() for covariate-conditional treatment *effects*; predict.mlumr_fit() for population-level predictions.

Examples

```
## Not run:
conditional_predict(fit)
conditional_predict(fit, newdata = data.frame(age = 60, sex = 1))

## End(Not run)
```

dbern — *Bernoulli PMF*

Description

Bernoulli PMF

Usage

```
dbern(x, prob, log = FALSE)
```

Arguments

x	Vector of values
prob	Success probability
log	Logical; if TRUE, return log-density

Value

Numeric vector

default_priors — *Default priors used by* mlumr()

Description

These accessors return the current default priors used by mlumr(), tagged with $default = TRUE and the package version. For family = "normal" their scales are multiples of the IPD outcome SD wherever the parameter is in outcome units; see prior_normal(). prior_summary() prints the version so cross-release reproducibility is diagnosable: if a later release changes a default, fits produced with an older version will still carry the correct $version tag.

Usage

```
default_prior_intercept()

default_prior_beta()

default_prior_sigma()

default_prior_aux()

default_prior_smooth()
```

Details

default_prior_aux() and default_prior_smooth() apply to the survival family only. prior_aux is a half-normal(0, 2) on the shape/scale parameter(s) of parametric survival distributions (Weibull/Gompertz/gamma shape, log-normal sdlog, generalized-gamma shapes). prior_smooth is a half-normal(0, 1) on the random-walk smoothing SD of the M-spline / piecewise-exponential baseline hazard.

Value

A prior list (see prior_normal()).

Examples

```
default_prior_intercept()
default_prior_beta()
default_prior_sigma()
```

---

distr *Specify a marginal distribution*

Description

Used to specify marginal distributions for covariates when adding integration points. Wraps an inverse CDF (quantile) function with its parameters.

Usage

```
distr(qfun, ...)
```

Arguments

qfun	Inverse CDF function (e.g., qnorm, qbern)
...	Parameters of the distribution, can reference column names in AgD

Value

A list with class "mlumr_distr" containing the distribution specification

Examples

```
# Normal distribution
distr(qnorm, mean = 0, sd = 1)

# Bernoulli distribution with probability 0.3
distr(qbern, prob = 0.3)
```

GammaDist *The Gamma distribution, parameterized by mean and standard deviation*

Description

Density, distribution, and quantile functions for the gamma distribution, accepting either the native shape / rate (or scale) parameters or a mean and sd, which override them. Useful with distr() and add_integration(), where covariate moments come straight from a published baseline table. The moment parameterization uses shape = (mean / sd)^2 and rate = mean / sd^2.

Usage

```
qgamma(
  p,
  shape,
  rate = 1,
  scale = 1/rate,
  lower.tail = TRUE,
  log.p = FALSE,
  ...,
  mean,
  sd
)

pgamma(
  q,
  shape,
  rate = 1,
  scale = 1/rate,
  lower.tail = TRUE,
  log.p = FALSE,
  ...,
  mean,
  sd
)

dgamma(x, shape, rate = 1, scale = 1/rate, log = FALSE, ..., mean, sd)
```

Arguments

p	Vector of probabilities.
shape, rate, scale	See stats::GammaDist.
lower.tail, log.p, log	See stats::GammaDist.
...	Must be empty; it keeps mean and sd out of partial matching.

mean, sd	Mean and standard deviation, overriding shape and rate / scale when both are supplied. Both must be named in full.
x, q	Vector of quantiles.

Value

A numeric vector, as the corresponding **stats** function.

See Also

distr(), add_integration(), qbern()

Examples

```
# Equivalent specifications
qgamma(0.5, mean = 65, sd = 8)
qgamma(0.5, shape = (65 / 8)^2, rate = 65 / 8^2)
```

---

geom_km *Observed Kaplan-Meier layer for survival overlays*

Description

Returns ggplot2 layers drawing the observed Kaplan-Meier step curves (the index IPD and the reconstructed comparator pseudo-IPD) of a survival mlumr_data, colored by treatment so they line up with a survival prediction plot. The mlumr analogue of multinma's geom_km(), so a predicted-versus-observed figure is just plot(predict(fit, type = "survival")) + geom_km(data).

Usage

```
geom_km(
  data,
  treatments = NULL,
  population = NULL,
  marks = TRUE,
  linewidth = 0.4,
  ...
)
```

Arguments

data	An mlumr_data survival object from combine_data().
treatments	Optional character vector of treatment labels to draw. By default both observed arms are drawn. This cannot separate the arms when both carry the same label; use population there. A label that names no observed arm is refused rather than drawn as nothing.

population	Optional cohort to draw, "Index" and/or "Comparator". Selects the arm itself rather than its display name, so population = "Comparator" overlays only the comparator KM on a comparator-population prediction whatever the treatments are called. Only the selected cohorts are examined: a left- or interval-censored observation in a cohort that is not drawn does not stop the plot, and one in a cohort that is drawn refuses it.
marks	Logical; draw censoring marks (default TRUE).
linewidth	Step line width (default 0.4).
...	Passed to ggplot2::geom_step().

Value

A list of ggplot2 layers (a step layer, plus a censoring-mark layer when marks = TRUE) to add to a plot with +. The layers carry the population each observed arm was measured in, so on a plot faceted by population each curve appears only in its own panel. A plot standardized to a newdata target therefore shows no observed curve, which is correct: no arm was observed in that population.

See Also

plot.mlumr_prediction()

Examples

```
## Not run:
plot(predict(fit, type = "survival")) + geom_km(dat)
plot(predict(fit, type = "survival")) + geom_km(dat, population = "Comparator")

## End(Not run)
```

---

logitNormal *The logit-Normal distribution*

Description

Density, distribution, and quantile functions for the logit-normal distribution: the distribution of plogis(z) where z is normal with mean mu and standard deviation sigma. It is the natural marginal for a covariate reported as a proportion on (0, 1), such as percent body surface area.

Usage

```
dlogitnorm(x, mu = 0, sigma = 1, log = FALSE, ..., mean, sd)

plogitnorm(q, mu = 0, sigma = 1, ..., mean, sd)

qlogitnorm(p, mu = 0, sigma = 1, ..., mean, sd)
```

Arguments

x, q	Vector of quantiles, in (0, 1).
mu, sigma	Location and scale, on the logit scale.
log	Return the log density. Positional, as in stats::dnorm().
...	For plogitnorm() and qlogitnorm(), passed to the underlying **stats** normal function (stats::pnorm(), stats::qnorm()), so lower.tail and log.p work as usual. dlogitnorm() builds its density from stats::dnorm() and a Jacobian rather than delegating, so it has nothing to forward and refuses anything passed here; in its signature ... serves only to keep mean and sd from matching positionally.
mean, sd	Mean and standard deviation on the (0, 1) scale, overriding mu and sigma when both are supplied.
p	Vector of probabilities.

Details

For convenience the distribution may be given by its mean and sd on the natural (0, 1) scale instead of mu and sigma on the logit scale. There is no closed form for that reparameterization, so mu and sigma are found numerically; supply mu / sigma directly if you have them.

Value

A numeric vector.

See Also

distr(), add_integration(), GammaDist

Examples

```
qlogitnorm(0.5, mean = 0.34, sd = 0.19)
```

---

make_knots *Choose M-spline knots for a flexible-baseline survival model*

Description

Place boundary and internal knots for the M-spline baseline hazard used by the flexible survival models (distribution = "mspline" or "pexp"). Knots are chosen from the pooled event/censoring times of the index IPD and the reconstructed comparator pseudo-IPD.

Usage

```
make_knots(data, n_knots = 7, type = c("quantile", "equal"))
```

Arguments

data	An mlumr_data object (survival family) from combine_data().
n_knots	Number of internal knots (default 7; capped at 50). Use a smaller value when events are scarce; the recommended rule of thumb is to keep the number of spline coefficients (n_knots + degree + 1) below half the number of events.
type	Internal-knot placement: "quantile" (default, at quantiles of the pooled event times) or "equal" (evenly spaced between the boundaries).

Details

The lower boundary knot is fixed at 0 (not the minimum delayed-entry time), so the cumulative hazard is anchored at H(0) = 0 and stays continuous with the backward constant-hazard extrapolation; delayed-entry times (> 0) are evaluated on the basis. The upper boundary knot is the maximum observed time across both data sources. The M-spline basis is normalized so that the baseline cumulative hazard equals 1 at the upper boundary; the hazard scale is carried by the model intercepts.

Value

A list with internal (internal knot locations), boundary (lower and upper boundary knots) and n_knots (the realized number of internal knots after dropping any that coincide with the boundaries).

See Also

mlumr() with distribution = "mspline".

Examples

```
## Not run:
# Build a survival network, then choose M-spline baseline knots:
dat <- combine_data(index_ipd, comparator_agd)
knots <- make_knots(dat, n_knots = 5)
knots$boundary  # lower (0) and upper boundary knots
knots$internal  # internal knot locations

## End(Not run)
```

---

marginal_effects *Marginal treatment effects*

Description

Extract marginal treatment effects from a fitted ML-UMR model. For binomial: log odds ratio, risk difference, risk ratio. For normal: mean difference. For poisson: rate ratio. For survival: the hazard ratio (proportional-hazards distributions, labeled HR), the time ratio (accelerated-failure-time distributions with one shared shape and one shared coefficient vector, labeled TR), or the exponentiated linear-predictor contrast (EXP_DELTA_ETA, where neither of those holds); all natural-scale, null 1,

like the poisson rate ratio. Plus the restricted-mean-survival-time difference (RMSTD, null 0) and the RMST ratio (RMSTR, null 1), both reported with the restriction time in a horizon column. For the time-varying log hazard ratio curve (null 0) use predict(type = "loghr").

Usage

```
marginal_effects(
  object,
  population = c("both", "index", "comparator"),
  effect = "all",
  summary = TRUE,
  probs = c(0.025, 0.5, 0.975),
  newdata = NULL,
  at_time = NULL
)
```

Arguments

object	An mlumr_fit object
population	Which population: "both" (default), "index", or "comparator". The **index** population is normally the decision-relevant target for health technology assessment, since cost-effectiveness models are built for the population the decision is about; report it as the primary estimand and the comparator population alongside. Ignored when newdata is supplied (the effect is standardized to the newdata target population instead).
effect	Which effect measure. For binomial: "all", "lor", "rd", or "rr". For normal: "all" or "md". For poisson: "all" or "rr". For survival: "all", "rmstd", "rmstr", and the one scalar name the fit's contrast is, "hr" for a proportional-hazards fit, "tr" for a shared-shape SPFA accelerated-failure-time fit and "exp_delta_eta" otherwise. There are no aliases: requesting a scale the fit cannot supply is an error naming the one it can.
summary	Return summary (TRUE) or full draws (FALSE)
probs	Quantiles for summary
newdata	Optional data frame of covariate profiles defining a target population to standardize the effect to by g-computation (Chandler and Ishak, Eq 9 and 10); population is then ignored. Column names must match the model covariates. The same survival scalar selector applies as without newdata.
at_time	Evaluation time for the scalar marginal hazard ratio of a proportional-hazards fit whose two studies have different baseline shapes. Snapped to the nearest fitted prediction time, with a message. NULL uses the first prediction time. Under a shared baseline the scalar is the t -> 0 limit, so only at_time = 0 is accepted; an error for AFT fits.

Details

For survival proportional-hazards fits the scalar "hr" is always a marginal hazard ratio at one time, recorded in the at_time column: the t -> 0 limit under a shared baseline shape (where an SPFA

fit's value coincides with the conditional hazard ratio, since the shared coefficients cancel), and the value at the first prediction time, or at `at_time`, under study-specific shapes. Hazard ratios are non-collapsible, so the marginal ratio is time-varying; `predict(type = "loghr")` gives the curve, and the RMST effects are collapsible within a population.

For accelerated-failure-time fits the scalar is `exp(E_X[eta_index(X)] - E_X[eta_comparator(X)])`. With one shared shape and SPFA coefficients it is a population time ratio (`TR`): every individual's survival time is accelerated by the same factor. With differing shapes or treatment-specific coefficients no single acceleration factor exists and it is labeled `EXP_DELTA_ETA`; use RMST effects or explicitly indexed survival quantiles there. Neither carries an evaluation time. See `vignette("survival-outcomes", "mlumr")`.

For binomial fits `"lor"` is always a logit-scale marginal odds ratio, whatever the fitted link, so for a probit or cloglog fit it is on a different scale than `naive()` and `stc()` `$estimate`.

Relaxed-model index-population estimands average `beta_comparator` over the IPD covariate distribution, outside the support it was identified on, so they are wider and more prior-sensitive than the comparator-population ones. `marginal_effects()` says so once per call; suppress with `options(mlumr.quiet_relaxed_index = TRUE)`.

Value

A data frame. With `summary = FALSE` the raw posterior draws are returned as a plain data frame whose column names carry the effect scale (`lor_*`, `rr_*`, `delta_*`, `hr_*` / `tr_*` / `exp_delta_eta_*`, `rmst*`); with `summary = TRUE` the `effect` column names the measure. For survival, RMST rows carry a `horizon` column and the scalar rows an `at_time` column (attributes of the same names on the raw-draw frame).

See Also

`predict.mlumr_fit()` for absolute predictions; `conditional_effects()` for covariate-conditional effects at specific profiles; `prior_sensitivity()` to check how strongly the marginal effect depends on `prior_beta`.

Examples

```
## Not run:
# All effect measures for both populations
marginal_effects(fit)

# Only the log odds ratio in the index population
marginal_effects(fit, population = "index", effect = "lor")

# Transport the effect to an external (e.g. jurisdiction-specific) population
marginal_effects(fit, newdata = target_population_covariates)

# Full posterior draws rather than summary statistics
marginal_effects(fit, summary = FALSE)

## End(Not run)
```

mlumr *Fit ML-UMR Model*

Description

Fit a Bayesian multilevel unanchored meta-regression model using individual patient data (IPD) and aggregate data (AgD). Supports binary, continuous, count, and time-to-event outcomes.

Usage

```
mlumr(
  data,
  model = c("spfa", "relaxed"),
  link = NULL,
  prior_intercept = default_prior_intercept(),
  prior_beta = default_prior_beta(),
  prior_sigma = default_prior_sigma(),
  distribution = NULL,
  prior_aux = NULL,
  prior_smooth = NULL,
  n_knots = 7L,
  knots = NULL,
  mspline_degree = NULL,
  aux_by = ".study",
  pred_times = NULL,
  rmst_horizon = NULL,
  n_rmst_grid = 100L,
  center = TRUE,
  qr = FALSE,
  chains = 4,
  iter = 2000,
  warmup = 1000,
  seed = NULL,
  adapt_delta = 0.95,
  max_treedepth = 15,
  refresh = 200,
  engine = NULL,
  verbose = TRUE,
  prior_beta_comparator = NULL,
  prior_aux2 = NULL,
  ...
)
```

Arguments

`data`	An `mlumr_data` object with integration points (from `add_integration()`)

model
: Model type: "spfa" (shared prognostic factor assumption) or "relaxed" (treatment-specific coefficients). Default "spfa".

link
: Link function. For binomial: "logit" (default), "probit", or "cloglog". For normal: "identity" (default) or "log". For poisson and survival: "log" (default, only option). If NULL, uses the canonical default for the family.

prior_intercept
: Prior for treatment intercepts. Default from default_prior_intercept() (prior_normal(0, 10)), on the linear-predictor scale; for family = "normal" with the identity link, where the intercepts are in outcome units, normal(0, 10 * sd(y)) with sd(y) the IPD outcome SD. See prior_normal().

prior_beta
: Prior for regression coefficients. A single prior broadcast to all covariates, or a list of priors of length n_cov sharing one family (and, for Student-t, one df). Default from default_prior_beta() (prior_normal(0, 2.5), times sd(y) for family = "normal" with the identity link). Set autoscale = TRUE on the prior to divide the scale by each covariate's empirical SD (and, for the normal identity link, multiply it by sd(y)). For model = "spfa" this is the prior on the shared beta; for model = "relaxed" on beta_index, with beta_comparator taking prior_beta_comparator.

prior_sigma
: Prior for residual SD (normal family only). Default from default_prior_sigma(), a half-normal (through the Stan <lower=0> constraint) with scale 2.5 * sd(y), the residual SD being in outcome units under either link. prior_exponential() is also supported for sigma.

distribution
: For family = "survival" only: the survival distribution. Proportional hazards: "exponential", "weibull" (default), "gompertz" (positive shape, so an increasing hazard). Accelerated failure time: "exponential-aft", "weibull-aft", "lognormal", "loglogistic", "gamma", "gengamma" (the positive Lawless Q subfamily). Flexible baseline hazard: "mspline" and "pexp" (piecewise exponential). "gengamma" is the least numerically robust option, so inspect its MCMC diagnostics. Must be NULL for other families.

prior_aux
: For family = "survival" parametric distributions: prior for the shape or scale parameter(s), half-normal, half-t or exponential through the <lower=0> constraint. Default default_prior_aux(). The Gompertz shape has units of 1 / time, so set this explicitly for a Gompertz baseline and check it against the time unit.

prior_smooth
: For family = "survival" flexible baselines ("mspline"/"pexp"): prior for the random-walk smoothing SD. Default default_prior_smooth().

n_knots
: For family = "survival" flexible baselines: number of internal spline knots (default 7). See make_knots().

knots
: Optional custom knots for a flexible survival baseline. With a shared baseline (aux_by = "none"), supply one make_knots() result. With study-specific baselines, supply list(index = ..., comparator = ...), where each element has the same structure and coefficient count.

mspline_degree
: For family = "survival" flexible baselines: spline degree override (default derived from distribution: 3 for "mspline", 0 for "pexp").

aux_by
: For family = "survival": how the baseline hazard is shared between the two studies, the unanchored analogue of multinma::nma()'s aux_by. ".study"

(the default, and what NULL means) gives each study its own baseline shape, with its own knots over its own observed support for a flexible baseline. "none" gives both studies one shared shape, a stronger assumption that buys precision; fit it as a sensitivity analysis when the two Kaplan-Meier curves plainly share a shape. Each study contributes one arm, so a study-specific and a treatment-specific baseline shape cannot be told apart, and a stratified fit carries each study's shape with its treatment when predictions are transported; where that is doubtful, prefer the RMST estimands and report both settings. With the stratified default the marginal hazard ratio varies with time, so marginal_effects() reports it at one at_time.

pred_times
: For family = "survival": times at which survival, hazard and cumulative-hazard predictions are produced. If NULL, a grid up to the maximum observed time is used.

rmst_horizon
: For family = "survival": the upper time limit for the restricted mean survival time. If NULL, the maximum observed time, except for a flexible baseline stratified by study, where it defaults to the follow-up both studies observed so the headline RMST does not extrapolate the shorter study. A longer horizon warns.

n_rmst_grid
: For family = "survival": number of equally spaced nodes (default 100) on [0, rmst_horizon] for the trapezoidal RMST integral. Increase for sharp early hazards or long horizons.

center
: Logical (default TRUE). Center the covariates about the pooled IPD and population-weighted declared AgD means before fitting. The likelihood is unchanged and sampling is usually easier, but prior_intercept then applies to the intercept at the pooled covariate mean. Set FALSE to fit on the raw covariate scale. A fit whose centering rounds two integration points onto one is refused.

qr
: Logical (default FALSE). Apply a thin-QR reparameterization to the combined design matrix, which decorrelates its columns for HMC. The priors still apply to the original coefficients. Useful with many correlated or ill-scaled covariates.

chains
: Number of MCMC chains (default 4)

iter
: Total iterations per chain (default 2000)

warmup
: Number of warmup iterations (default 1000)

seed
: Random seed for reproducibility. If NULL (default), the fixed seed 2026 is used and a warning says so. The seed used is reported in the fitting messages.

adapt_delta
: Target acceptance rate (default 0.95)

max_treedepth
: Maximum tree depth for NUTS (default 15)

refresh
: How often to print progress (0 = silent, default 200)

engine
: Stan backend: "rstan" (default) or "cmdstanr". If NULL, uses the engine set by mlumr_engine(). See mlumr_engine() for setup.

verbose
: Logical; if FALSE, suppresses mlumr progress messages. Stan sampler progress is still controlled by refresh.

prior_beta_comparator
: Relaxed model only: prior for the comparator-arm coefficients beta_comparator, with the same specification rules as prior_beta and any supported family. NULL

(the default) reuses prior_beta. beta_comparator is informed only by the aggregate likelihood, so a tighter prior here regularizes the index-population estimand; see check_identification() and prior_sensitivity(). Ignored for model = "spfa".

prior_aux2 For distribution = "gengamma" only: prior for the second auxiliary parameter. NULL (the default) reuses prior_aux. Supplying it for any other distribution warns and has no effect.

... Additional arguments passed to the Stan sampling function (rstan::sampling() or cmdstanr's $sample() method)

Details

The model assumes that all AgD rows come from the same comparator treatment and that, conditional on covariates, there is no between-study heterogeneity. No random effects for study-level heterogeneity are included.

AgD scale (family = "normal"). The AgD likelihood is y_agd ~ normal(E[exp(eta)], se_agd) under link = "log" and y_agd ~ normal(E[eta], se_agd) under link = "identity"; in both cases set_agd() expects outcome_mean and outcome_se on the arithmetic scale.

The comparator population is the size-weighted mixture of its aggregate rows. Comparator-population predictions weight each row by the population it represents: n_agd for binomial, outcome_n for normal (required for more than one row) and E_agd for poisson. These are mixing weights, separate from the likelihood's precision weights, so splitting a comparator population into subgroup rows leaves the estimand unchanged.

Identifying the relaxed model. beta_comparator is informed only by the aggregate likelihood. Jointly defined subgroup rows, one set_agd() row per stratum, are what can separate it from the comparator intercept; a single aggregate summary constrains one combination of them. Use check_identification() before fitting and prior_sensitivity() after; the subgroup-identification vignette works through the geometry.

Normal outcomes. A normal fit whose IPD covariates reproduce the outcome exactly has an improper posterior for the residual SD, so mlumr() refuses a constant outcome and a least-squares fit that is exact to numerical precision (residual sum of squares at most 1e-12 of the total) before sampling, and warns when the design is saturated. Everything else is left to the sampler and its diagnostics.

Value

An object of class mlumr_fit

See Also

prior_sensitivity(), check_identification(), set_agd(), prior_summary().

Examples

```
## Not run:
# Binary SPFA model
fit_spfa <- mlumr(dat, model = "spfa")
```

```
# Relaxed SPFA (allows effect modification)
fit_relaxed <- mlumr(dat, model = "relaxed")

## End(Not run)
```

`mlumr_engine` *Get or set the Stan engine*

Description

mlumr fits its models through rstan by default. `mlumr_engine("cmdstanr")` switches to cmdstanr for the session; the choice is stored in `options(mlumr.stan_engine)`, so a permanent default belongs in `.Rprofile`. When cmdstanr or CmdStan is missing, an interactive session is offered their installation (cmdstanr from stan-dev's maintained repository); otherwise the install commands are printed and the engine is left unchanged.

Usage

```
mlumr_engine(engine = NULL)
```

Arguments

`engine`	`"rstan"` or `"cmdstanr"`, matched exactly. `NULL` (the default) returns the current engine without changing it.

Value

The current engine, invisibly when setting.

Examples

```
mlumr_engine()
## Not run:
mlumr_engine("cmdstanr")

## End(Not run)
```

mlumr_forest *Forest plot of a small set of estimates*

Description

A one-call forest plot for comparing a handful of estimates supplied as a data frame, e.g. several methods (naive / STC / ML-UMR) or several covariate profiles. Rows are drawn top-to-bottom in the order given. Returns a `ggplot` object, so further ggplot2 layers compose with `+`. This keeps the method-comparison forests in the vignettes to a single line instead of a hand-built `ggplot()` stack.

Usage

```
mlumr_forest(
  data,
  ref_line = NULL,
  log_x = FALSE,
  x = NULL,
  title = NULL,
  subtitle = NULL,
  color = "#3B6B9A",
  clip = TRUE,
  ...
)
```

Arguments

`data`	A data frame with one row per estimate. Columns are matched flexibly (first match wins): the row label from `label` / `method` / `Method` / `Comparison` (else the first character/factor column); the point estimate from `est` / `estimate` / `mean`; the interval bounds from `lo`/`hi`, `q2.5`/`q97.5`, `ci_lower`/`ci_upper`, `conf.low`/`conf.high`, or `lower`/`upper`.
`ref_line`	Null-effect reference line. By default it is read from the `effect` column when the label is one the package produces (`1` for a ratio measure, `0` for a difference); otherwise `1` when `log_x = TRUE` and `0` otherwise. Pass it explicitly for a measure this does not name. Kept inside the clipping window.
`log_x`	Draw the x axis on a log10 scale (for ratio measures).
`x`, `title`, `subtitle`	Axis label and titles (passed to `ggplot2::labs()`).
`color`	Point and interval color.
`clip`	Logical; if `TRUE` (default), when one or two intervals are far wider than the rest the x axis is clipped to the bulk of the estimates and the over-wide interval's clipped end(s) are drawn with an arrow, so a single very uncertain estimate does not compress all the others into a sliver.
`...`	Unused.

Details

All rows share one axis, so they must be on one effect scale: a frame with an `effect` column naming more than one measure is rejected rather than drawn against a single reference that cannot be right for both.

Value

A `ggplot` object.

See Also

`marginal_effects()`, `naive()`, `stc()`

Examples

```
## Not run:
forest_df <- data.frame(
  label = c("Naive", "STC", "ML-UMR SPFA"),
  est = c(res_naive$estimate, res_stc$estimate, me$mean),
  lo = c(res_naive$ci_lower, res_stc$ci_lower, me$q2.5),
  hi = c(res_naive$ci_upper, res_stc$ci_upper, me$q97.5)
)
mlumr_forest(forest_df, ref_line = 0, x = "Log odds ratio")

## End(Not run)
```

---

naive — *Naive unadjusted indirect comparison*

Description

Compute an unadjusted (naive) indirect treatment comparison by comparing crude outcomes from the IPD and AgD without any covariate adjustment. The index outcome remains marginal over the index-study population and the comparator outcome remains marginal over the comparator population. The contrast therefore has no single standardized target population. It returns the link-scale contrast plus the two observed marginal outcomes and available natural-scale contrasts.

Usage

```
naive(data, link = NULL, conf_level = 0.95)
```

Arguments

`data`	An `mlumr_data` object from `combine_data()`

link	Link function. For binomial: "logit" (default), "probit", or "cloglog". For normal/poisson: ignored (identity/log always used). For survival: ignored (an unadjusted Cox proportional-hazards log hazard ratio is returned). The naive Cox benchmark accepts only right-censored / event data (optionally with delayed entry); left- or interval-censored data (which the Bayesian mlumr() model supports) are rejected. If NULL, uses the canonical default.
conf_level	Confidence level for the interval (default 0.95)

Details

The two arms are observed directly, so their intervals are exact: the Clopper-Pearson interval for a binomial proportion and the Garwood interval for a Poisson rate, pooled across aggregate rows. The arm standard errors and the contrasts use the boundary pseudo-count (r + 0.5) / (n + 1) when an arm has zero or all events, or 0.5 events for a zero Poisson count; the reported crude proportions and rates are unchanged. The link-scale contrast, the log risk ratio and the risk difference get Wald intervals, whose coverage is approximate.

Scale note: $estimate (and the binomial $log_rr) is on the link / log scale, where the null is 0. To compare against the natural-scale risk ratio or rate ratio from marginal_effects() (where the null is 1), exponentiate it (e.g. exp(result$estimate)).

Value

An object of class mlumr_naive

Normal-family weighting

Across multiple AgD rows the normal-family comparator mean here is population weighted using outcome_n, matching the Bayesian ML-UMR comparator-population estimand. outcome_n is required when there is more than one row; a single row has weight one. The comparator-mean variance combines independent, mutually exclusive strata as sum(w^2 * se^2) using normalized population weights. The same weighting applies to stc().

Examples

```
## Not run:
result <- naive(dat)
print(result)

## End(Not run)
```

---

ndmm_agd — *Newly diagnosed multiple myeloma: comparator pseudo-IPD (survival)*

Description

Reconstructed Kaplan-Meier pseudo-IPD for the comparator arm (Morgan 2012, thalidomide), the survival AgD comparator; pair with ndmm_ipd and ndmm_agd_covs.

Usage

```
ndmm_agd
```

Format

A data frame with 408 rows and 5 columns: `study`, `treatment`, `subject`, `eventtime`, `status`.

Source

The Morgan-2012 thalidomide arm of `multinma::ndmm_agd` (GPL-3; Phillippo, *multinma*, 2024); event/censoring times reconstructed from published Kaplan-Meier curves. See `ndmm_ipd` for trial sources.

`ndmm_agd_covs` *Newly diagnosed multiple myeloma: comparator covariate summaries (survival)*

Description

Published covariate summaries for the comparator arm (Morgan 2012, thalidomide), supplying the covariate moments for the AgD integration points.

Usage

```
ndmm_agd_covs
```

Format

A data frame with 1 row and 7 columns: `study`, `treatment`, `age_mean` / `age_sd`, and the proportions `iss_stage3_prop`, `response_cr_vgpr_prop`, `male_prop`.

Source

The Morgan-2012 thalidomide row of `multinma::ndmm_agd_covs` (GPL-3; Phillippo, *multinma*, 2024), subset to the columns used in the survival vignette. See `ndmm_ipd` for trial sources.

`ndmm_ipd` *Newly diagnosed multiple myeloma: index individual patient data (survival)*

Description

Individual patient data for the index arm (McCarthy 2012, lenalidomide), the survival (progression-free survival) ML-UMR worked example; pair with `ndmm_agd` and `ndmm_agd_covs`.

Usage

```
ndmm_ipd
```

Format

A data frame with 231 rows and 9 columns:

study, treatment study and treatment labels

subject row identifier

age, iss_stage3, response_cr_vgpr, male patient covariates

eventtime, status progression-free survival time and event indicator (1 = event, 0 = censored)

Details

The two source trials are not genuinely disconnected: McCarthy 2012 and Morgan 2012 both include a placebo arm. Those arms are omitted here deliberately, so that this pair of datasets poses the unanchored problem ML-UMR exists for.

McCarthy 2012 is the index arm because population overlap, not sample size, governs how far an unanchored comparison has to extrapolate. Unanchored MAIC effective sample size against the Morgan 2012 thalidomide arm is 35.8 of 231 (15.5\ lenalidomide arm, the other candidate for this comparison. See `data-raw/prepare_multinma_subsets.R` for the full pair ranking.

Source

The McCarthy-2012 lenalidomide arm of `multinma::ndmm_ipd` (GPL-3; Phillippo, *multinma*, 2024), subset to the columns used in the survival vignette. multinma provides *simulated* IPD resembling published newly-diagnosed-multiple-myeloma trials; the vignette compares lenalidomide (McCarthy et al. 2012) with thalidomide (Morgan et al. 2012). See `data-raw/prepare_multinma_subsets.R`.

pbern *Bernoulli CDF*

Description

Bernoulli CDF

Usage

```
pbern(q, prob, lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	Vector of quantiles
prob	Success probability
lower.tail	Logical
log.p	Logical

Value

Numeric vector

plot.mlumr_conditional_effects *Plot covariate-conditional treatment effects*

Description

Point-interval of the conditional effect at each covariate profile.

Usage

```
## S3 method for class 'mlumr_conditional_effects'
plot(x, ref_line = NULL, ...)
```

Arguments

x	A `conditional_effects()` result.
ref_line	Numeric null-effect reference line. By default it is chosen per facet from the effect label: 1 for the natural-ratio measures (RR, HR, TR, and the exponentiated contrast reported when the study baselines differ) and 0 for the additive ones (RD, MD, LINK_EFFECT). Pass a single value to override for all panels.
...	Unused.

Value

A ggplot object.

See Also

conditional_effects()

---

plot.mlumr_marginal_effects

Forest plot of population-standardized marginal effects

Description

Plots the point estimate and credible interval for each effect measure, grouped by target population (index / comparator). The interval's coverage is read from the quantile columns present, so it matches the probs the result was summarized with. Panels showing only ratio measures are drawn on a log axis, where reciprocal effects sit at equal distances from the null. The mlumr analogue of plot(multinma::relative_effects(fit)).

Usage

```
## S3 method for class 'mlumr_marginal_effects'
plot(x, ref_line = NULL, ...)
```

Arguments

x	A marginal_effects() result.
ref_line	Numeric null-effect reference line. By default 0 for difference/log measures and 1 for natural ratio measures (RR), drawn per facet. Pass a single value to override for all panels.
...	Unused.

Value

A ggplot object.

See Also

marginal_effects()

Examples

```
## Not run:
plot(marginal_effects(fit, effect = "all"))

## End(Not run)
```

plot.mlumr_prediction *Plot absolute predictions from a fitted ML-UMR model*

Description

Dispatches on the prediction `type`: time-indexed types (`"survival"`, `"hazard"`, `"cumhaz"`, `"loghr"`) are drawn as curves with credible bands at the coverage the result was summarized with; scalar types (`"rmst"`, `"median"`, `"response"`) as point-intervals. The mlumr analogue of `plot(predict(multinma_fit))`.

Usage

```
## S3 method for class 'mlumr_prediction'
plot(x, ref_line = NULL, ...)
```

Arguments

`x`	A `predict()` result (an `mlumr_prediction`).
`ref_line`	Optional numeric null-reference line(s). Drawn as horizontal line(s) for curve types and vertical line(s) for scalar types, mirroring the `plot(predict(fit), ref_line = c(0, 1))` idiom (e.g. probability bounds for `type = "response"`). The log-hazard-ratio curve defaults to `ref_line = 0`.
`...`	Unused.

Value

A `ggplot` object (compose further layers, e.g. a KM overlay, with `+`).

See Also

`predict.mlumr_fit()`, `geom_km()`

Examples

```
## Not run:
plot(predict(fit, type = "survival")) + geom_km(dat)
plot(predict(fit, type = "response"), ref_line = c(0, 1))
plot(predict(fit, type = "loghr"))

## End(Not run)
```

plot_prior_posterior *Prior-versus-posterior overlay*

Description

Plots the posterior density of the named parameters with their prior density overlaid, reading the prior from the fit. The mlumr analogue of `multinma::plot_prior_posterior()`. Intended for parameters with a known, un-autoscaled prior (the treatment intercepts `mu_index` / `mu_comparator` under the default `prior_intercept`).

Usage

```
plot_prior_posterior(object, pars = c("mu_index", "mu_comparator"), ...)
```

Arguments

`object`	An `mlumr_fit`.
`pars`	Character vector of parameter (draw column) names. Default the treatment intercepts `c("mu_index", "mu_comparator")`.
`...`	Unused.

Value

A `ggplot` object.

See Also

`prior_summary()`, `mlumr()`

Examples

```
## Not run:
plot_prior_posterior(fit)
plot_prior_posterior(fit, pars = c("mu_index", "mu_comparator"))

## End(Not run)
```

predict.mlumr_fit *Predictions from ML-UMR model*

Description

Generate population-average absolute-outcome predictions in the index and comparator populations.

Usage

```
## S3 method for class 'mlumr_fit'
predict(
  object,
  population = c("both", "index", "comparator"),
  type = NULL,
  summary = TRUE,
  probs = c(0.025, 0.5, 0.975),
  times = NULL,
  newdata = NULL,
  ...
)
```

Arguments

object	An mlumr_fit object
population	Which population: "both", "index", or "comparator"
type	Prediction type. For binomial/normal/poisson: "response" (default) or "link". For survival: "survival" (default), "hazard", "cumhaz", "rmst" (restricted mean survival time), "median" (median survival, obtained by linear interpolation on the fitted pred_times grid; if the true median precedes the first grid point it is interpolated between the known exact point S(0) = 1 and (pred_times[1], S(pred_times[1])), so it is never reported as later than the first grid time, though a denser pred_times near zero still resolves very early medians better), or "loghr" (time-varying marginal log hazard ratio of index vs comparator at each fitted time, per population). For "response": probabilities (binomial), means (normal), or rates (poisson). For "link": the fitted link applied to the population-standardized response mean, g(E[g^{-1}(eta)]). This is the marginal link-scale prediction used by G-computation; it is generally not the mean conditional linear predictor E[eta].
summary	Return summary statistics (TRUE) or full posterior draws (FALSE)
probs	Quantiles for summary (default c(0.025, 0.5, 0.975))
times	For survival fits, an optional vector of times at which to report curve predictions; each is matched to the nearest fitted pred_times grid point. If NULL, all fitted times are returned. When supplied, the result has one row per requested time for each treatment and population cell, in the order requested and including repeats, with a requested_time

column beside `time`. With `summary = FALSE` the mapping is carried as the `requested_time` and `used_time` attributes instead, one entry per time column. Refit with `pred_times` containing the exact times to avoid the approximation.

`newdata` Optional data frame of covariate profiles defining an arbitrary **target population**. When supplied, per-treatment absolute predictions are standardized to this population by g-computation (averaging model-based predictions over the rows at each posterior draw), and `population` is ignored. Supports `type = "response"/"link"` (binomial/normal/poisson) and `type = "survival"/"hazard"/"cumhaz"/"rmst"/"m` (survival). Rows outside the fitted covariate support are model-based extrapolation; overlap is not checked.

`...` Additional arguments (unused)

Details

Marginalization on non-identity links. For `type = "response"` the reported values are `E[g^{-1}(eta)]`, the population-average prediction for an individual drawn from that population, not `g^{-1}(E[eta])`; the expectation runs over the IPD individuals for the index population and over the integration points for the comparator one. `type = "link"` applies the fitted link after that marginalization.

Value

A data frame with predictions. `type = "rmst"` adds a `horizon` column with the restriction time actually integrated to, since RMST at different horizons is a different estimand. The plot methods require `summary = TRUE`; with `summary = FALSE` the raw posterior draws are returned as a plain data frame. For `type = "median"` the summary is conditional on the median being reached, and `p_not_reached` gives the posterior probability that it is not.

See Also

`marginal_effects()` for treatment-effect summaries; `conditional_predict()` and `conditional_effects()` for predictions at specific covariate profiles.

Examples

```
## Not run:
# Absolute predictions for both populations:
predict(fit, population = "both")
# Survival RMST, and transport to a target covariate distribution:
predict(fit, type = "rmst")
predict(fit, newdata = target_population)

## End(Not run)
```

prior_cauchy *Specify a Cauchy prior*

Description

Cauchy is Student-t with `df = 1`, a wrapper around `prior_student_t()`. Its very heavy tails can slow sampling; a Student-t with 3 to 7 degrees of freedom is usually preferred for regression coefficients.

Usage

```
prior_cauchy(mean = 0, sd = 2.5, autoscale = FALSE)
```

Arguments

`mean`	Prior location (default 0).
`sd`	Prior scale (default 2.5).
`autoscale`	See `prior_normal()`. Default `FALSE`.

Value

A list with components `distribution = "student_t"`, `df = 1`, `mean`, `sd`, `autoscale`.

Examples

```
prior_cauchy(mean = 0, sd = 2.5)
```

prior_exponential *Specify an exponential prior*

Description

Exponential prior on a positive scalar. Currently supported for `prior_sigma` (normal-family residual SD) only; rejected for unconstrained intercepts and regression coefficients. `prior_exponential(rate = 1)` has mean 1 and is a reasonable weakly-informative choice when the outcome is standardized.

Usage

```
prior_exponential(rate = 1)
```

Arguments

`rate`	Rate parameter (default 1). Larger `rate` = tighter prior concentrated near zero.

Value

A list with components `distribution = "exponential"`, `rate`, and placeholders (`mean = 0`, `sd = 1 / rate`) so the same Stan-field translation works.

Examples

```
prior_exponential(rate = 1)
```

---

prior_normal *Specify a normal prior*

Description

Construct a normal prior for passing to `mlumr()` via `prior_intercept`, `prior_beta`, or `prior_sigma`. The resulting list is consumed by the Stan models.

Usage

```
prior_normal(mean = 0, sd = 10, autoscale = FALSE)
```

Arguments

`mean`	Prior mean (default 0).
`sd`	Prior standard deviation (default 10). The default matches the historical "very weak" scale; explicit tighter values are recommended for regression coefficients (see Details).
`autoscale`	If `TRUE` and this prior is passed as `prior_beta`, the scale is divided by each covariate's empirical SD so the prior is weakly-informative regardless of predictor scaling, and for `family = "normal"` with the identity link also multiplied by the IPD outcome SD, since the coefficients are then in outcome units. Default `FALSE` to preserve backward-compatible behavior; set to `TRUE` explicitly when passing unstandardized predictors. Ignored for `prior_intercept` and `prior_sigma`.

Value

A list with components `distribution`, `mean`, `sd`, `df`, `autoscale`.

Choosing a scale

The default intercept prior `normal(0, 10)` is very weak on the link scale, and the data usually constrain the intercept strongly. The coefficient default `normal(0, 2.5)` is a generic starting value on the link scale per unit of covariate, not a calibrated choice; use `autoscale = TRUE` for predictors on different scales and calibrate with prior predictive checks (Gelman et al., 2008; the Stan prior-choice wiki). `prior_sigma` is a normal truncated at zero through the Stan `<lower=0>` constraint, a half-normal at the default mean of 0.

For `family = "normal"` the identity link is not unit-free: the intercepts and coefficients are in the outcome's units, and so is the residual SD under either link. There the package defaults are read in

units of the IPD outcome SD, sd(y): normal(0, 10 * sd(y)) for the intercepts, normal(0, 2.5 * sd(y)) for the coefficients (identity link) and a half-normal with scale 2.5 * sd(y) for the residual SD (either link), and autoscale = TRUE gives a coefficient scale of sd * sd(y) / sd(x). A prior written out by the user is used as given. prior_summary() prints the scales the model used. Run prior_sensitivity() for the relaxed model, whose beta_comparator is identified only by the aggregate likelihood.

References

Gelman, A., Jakulin, A., Pittau, M. G., & Su, Y.-S. (2008). A weakly informative default prior distribution for logistic and other regression models. *Annals of Applied Statistics*, 2(4), 1360-1383.

Vehtari, A. et al. Prior Choice Recommendations (Stan wiki): https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations.

Examples

```
# Default weakly-very-weak intercept prior
prior_normal(mean = 0, sd = 10)

# Package starting value for regression coefficients
prior_normal(mean = 0, sd = 2.5)

# Autoscaled coefficient prior (dividing 2.5 by each covariate's SD)
prior_normal(mean = 0, sd = 2.5, autoscale = TRUE)
```

---

prior_sensitivity *Prior sensitivity analysis for an ML-UMR fit*

Description

Refit an mlumr() model across a grid of prior_beta scales (keeping the family, mean, and df fixed) and summarize how the posterior for the marginal treatment effects (delta_index, delta_comparator) moves. This is the workflow recommended by Vehtari et al.'s prior-choice wiki for judging how much of the posterior is driven by the data versus the prior.

Usage

```
prior_sensitivity(
  fit,
  prior_beta_scales = c(0.5, 1, 2.5, 5, 10),
  prior_beta_comparator_scales = NULL,
  probs = c(0.025, 0.5, 0.975),
  verbose = TRUE,
  ...
)
```

Arguments

`fit` A fitted `mlumr_fit` object to re-fit under alternative priors.

`prior_beta_scales` Numeric vector of scales for `prior_beta`. Default `c(0.5, 1, 2.5, 5, 10)`. For a normal identity-link fit whose `prior_beta` is the package default or autoscaled, each scale is in units of the IPD outcome SD, as the fit's own prior is (see `prior_normal()`). The intercept and `sigma` priors are held at the fit's own.

`prior_beta_comparator_scales` (Relaxed fits only.) Numeric vector of scales for `prior_beta_comparator`, paired elementwise with `prior_beta_scales`. `NULL` (default) sweeps the comparator prior in parallel with `prior_beta_scales`; the scale used is reported in the `scale_comparator` column. Ignored, with a warning, for SPFA fits.

`probs` Quantiles for summarizing each posterior (default `c(0.025, 0.5, 0.975)`).

`verbose` Logical; if `FALSE`, suppresses progress messages and final printed summary table.

`...` Additional arguments forwarded to `mlumr()` on each refit (e.g. `chains`, `iter`, `refresh`). Sampling defaults otherwise inherit from the original fit.

Details

The design-matrix controls (`center`, `qr`) are taken from the original fit and replayed, so a refit reproduces the original parameterization instead of reverting to the defaults. A fit made with `center = FALSE` or `qr = TRUE` is a different parameterization, and replaying the defaults would vary the model as well as the prior.

Only the scale of `prior_beta` is varied; its family and mean, and every other prior and setting, come from the original fit. Each scale is applied to every coefficient, so the sweep reflects one level of prior informativeness per refit; an exponential `prior_beta` is swapped for `prior_normal(0, scale)`. For a relaxed fit the comparator prior is swept alongside, because the index-population estimand is driven by the comparator coefficients; holding their prior fixed would report a flat curve for exactly the quantity most exposed to the prior.

Value

A data frame with one row per (prior scale, summarized parameter) pair, and columns `scale`, `scale_comparator` (dropped when the model has no comparator coefficient prior), `parameter`, `effect`, `at_time` (present only when the summarized effect has an evaluation time, so absent for every non-survival family and for survival scalars that carry none), `mean`, `sd`, and one column per requested quantile, named `q` followed by the percentage (the default `probs` give `q2.5`, `q50`, `q97.5`), matching `marginal_effects()`. Quantiles are columns, not a row dimension. Side effect: prints a summary table at the end when `verbose = TRUE`, which is the default; `verbose = FALSE` returns the same data frame and prints nothing.

See Also

`prior_summary()` for a one-shot description of the priors on a fit; `marginal_effects()` for the posterior summary quantities this sweep tracks.

Examples

```
## Not run:
sens <- prior_sensitivity(fit_spfa, prior_beta_scales = c(1, 2.5, 5))

## End(Not run)
```

`prior_student_t` *Specify a Student-t prior*

Description

Heavier-tailed alternative to `prior_normal()`. A Student-t with moderate degrees of freedom can be a robust weakly informative starting family, but its scale still requires calibration to the link, outcome, and predictor scaling.

Usage

```
prior_student_t(df = 5, mean = 0, sd = 2.5, autoscale = FALSE)
```

Arguments

`df`	Degrees of freedom (must be positive).
`mean`	Prior location (default 0).
`sd`	Prior scale (default 2.5).
`autoscale`	See `prior_normal()`. Default `FALSE`.

Value

A list with components `distribution = "student_t"`, `df`, `mean`, `sd`, `autoscale`.

Examples

```
# A moderately heavy-tailed coefficient prior
prior_student_t(df = 5, mean = 0, sd = 2.5)
```

`prior_summary` *Summary of priors used by a fitted ML-UMR model*

Description

Print a human-readable summary of every prior that was used to fit an `mlumr()` model, including the effective per-coefficient scales after autoscaling. Mirrors the spirit of `rstanarm::prior_summary()`.

Usage

```
prior_summary(object, ...)

## Default S3 method:
prior_summary(object, ...)

## S3 method for class 'mlumr_fit'
prior_summary(object, digits = 3, ...)
```

Arguments

`object`	An `mlumr_fit` object.
`...`	Unused.
`digits`	Number of significant digits for numeric values (default 3).

Value

Invisibly returns a list describing the priors; the side effect is printing a formatted summary.

See Also

`prior_sensitivity()` to quantify how much the posterior moves under alternative `prior_beta` scales; `prior_normal()`, `prior_student_t()`, `prior_cauchy()`, `prior_exponential()` for the prior constructors themselves.

Examples

```
## Not run:
fit <- mlumr(dat)
prior_summary(fit)

## End(Not run)
```

psoriasis_agd — *Plaque psoriasis: comparator aggregate data (binary)*

Description

Aggregate (arm-level) summary for the comparator arm (FIXTURE, secukinumab 300 mg), the AgD comparator in the binary ML-UMR example; pair with psoriasis_ipd.

Usage

```
psoriasis_agd
```

Format

A data frame with 1 row and 11 columns: study, treatment; response counts pasi75_r / pasi75_n; and covariate summaries age_mean / age_sd, bsa_mean / bsa_sd, weight_mean / weight_sd, and prevsys_prop (a proportion).

Source

The FIXTURE secukinumab-300 arm of multinma::plaque_psoriasis_agd (GPL-3; Phillippo, *multinma*, 2024), subset to the columns used in the binary vignette. See psoriasis_ipd for the original trial sources.

psoriasis_ipd — *Plaque psoriasis: index individual patient data (binary)*

Description

Individual patient data for the index arm (UNCOVER-2, ixekizumab Q4W), the binary (PASI 75 response) ML-UMR worked example; pair with psoriasis_agd.

Usage

```
psoriasis_ipd
```

Format

A data frame with 347 rows and 8 columns:

study, treatment study and treatment labels

subject row identifier

pasi75 binary PASI 75 response indicator

age, bsa, weight, prevsys patient covariates (age in years, body-surface area in percent, weight in kg, and previous systemic treatment as a 1/0 indicator)

weight is missing for 2 of the 347 rows, as it is in the multinma source. set_ipd drops incomplete rows with a warning, so a fit adjusting for weight uses 345 of them.

Details

The two source trials are not genuinely disconnected: UNCOVER-2 and FIXTURE both include placebo and etanercept arms. Those arms are omitted here deliberately, so that this pair of datasets poses the unanchored problem ML-UMR exists for.

Source

The UNCOVER-2 ixekizumab-Q4W arm of `multinma::plaque_psoriasis_ipd` (GPL-3; Phillippo, *multinma*, 2024, doi:10.5281/zenodo.3904454), subset to the columns used in the binary vignette. multinma provides *simulated* IPD resembling the UNCOVER-2 / UNCOVER-3 (ixekizumab; Griffiths et al. 2015, doi:10.1016/s01406736(15)601258) and secukinumab (Langley et al. 2014, doi:10.1056/nejmoa1314258) plaque-psoriasis trials. See `data-raw/prepare_multinma_subsets.R`.

`qbern` *Bernoulli quantile function*

Description

Bernoulli quantile function

Usage

```
qbern(p, prob, lower.tail = TRUE, log.p = FALSE)
```

Arguments

`p`	Vector of probabilities
`prob`	Success probability
`lower.tail`	Logical; if TRUE, probabilities are P(X <= x)
`log.p`	Logical; if TRUE, probabilities are given as log(p)

Value

Integer vector of 0s and 1s

set_agd *Set up aggregate data (AgD)*

Description

Prepare AgD from the comparator treatment for an unanchored indirect comparison.

Usage

```
set_agd(
  data,
  treatment,
  family = c("binomial", "normal", "poisson"),
  outcome_n = NULL,
  outcome_r = NULL,
  outcome_mean = NULL,
  outcome_se = NULL,
  outcome_E = NULL,
  cov_means,
  cov_sds = NULL,
  cov_types = NULL,
  study = NULL
)
```

Arguments

data	Data frame containing AgD summary statistics
treatment	Column name for treatment variable
family	Outcome family: "binomial", "normal", or "poisson". Time-to-event comparator data go to set_agd_surv() instead, which takes reconstructed pseudo-IPD rather than a scalar outcome summary.
outcome_n	Column name for sample size. Required for binomial. For normal, required when there is more than one aggregate row, because the comparator-population estimand is the size-weighted mixture of those rows and they cannot be combined without knowing how large each is; optional for a single row, where the weighting is irrelevant.
outcome_r	Column name for number of events (required for binomial and poisson)
outcome_mean	Column name for mean outcome (required for normal)
outcome_se	Column name for standard error of outcome (required for normal)
outcome_E	Column name for total exposure (required for poisson)
cov_means	Character vector of column names for covariate means/proportions
cov_sds	Character vector of column names for covariate SDs (NA for binary covariates)
cov_types	Character vector specifying "continuous" or "binary" for each covariate. If NULL, inferred from presence of SD.
study	Column name for study identifier (optional)

Details

Rows must partition the aggregate sample. Each row contributes its own likelihood factor, as if the rows were disjoint sets of patients. One arm, or one set of mutually exclusive subgroup cells, is right; several overlapping subgroup tables of the same participants count every patient once per table and overstate the precision. Nothing in the data reveals the overlap, so it is not checked. See `vignette("subgroup-identification", "mlumr")` for how many rows the relaxed model needs.

Scales. For `family = "normal"`, `outcome_mean` and `outcome_se` are on the arithmetic scale under both links; a geometric mean or a log-scale summary is a different quantity and cannot be converted by the delta method. For `family = "poisson"`, `outcome_r` is the total count and `outcome_E` the total person-time, and the covariate distribution the rate is averaged over has to describe the covariates weighted by exposure; person-level moments stand in for that only when exposure carries no information about the rate within the row. For `family = "binomial"`, `outcome_r` and `outcome_n` are counts of events and trials.

Value

An object of class `mlumr_agd`. As for `set_ipd()`, the internal column names cannot be used as column names in `data`.

Examples

```
## Not run:
# Binary outcome
agd <- set_agd(
  data = trial_b,
  treatment = "trt",
  outcome_n = "n_total",
  outcome_r = "n_events",
  cov_means = c("age_mean", "sex_prop"),
  cov_sds = c("age_sd", NA),
  cov_types = c("continuous", "binary")
)

# Continuous outcome
agd <- set_agd(
  data = trial_b,
  treatment = "trt",
  family = "normal",
  outcome_mean = "mean_score",
  outcome_se = "se_score",
  outcome_n = "n_total",
  cov_means = c("age_mean", "sex_prop")
)

# Count outcome
agd <- set_agd(
  data = trial_b,
  treatment = "trt",
  family = "poisson",
  outcome_r = "n_events",
  outcome_E = "person_years",
```

```
  cov_means = c("age_mean", "sex_prop")
)

## End(Not run)
```

---

set_agd_surv *Set up aggregate survival data (reconstructed pseudo-IPD)*

Description

Prepare comparator aggregate survival data for an unanchored indirect comparison. The comparator arm is supplied as **reconstructed pseudo-IPD** (event/censoring times digitized from a published Kaplan-Meier curve, e.g. via the Guyot algorithm) together with summary covariate moments (means/SDs). The Stan model integrates the comparator likelihood over the covariate distribution implied by those moments.

Usage

```
set_agd_surv(
  data,
  treatment,
  Surv = NULL,
  time = NULL,
  status = NULL,
  entry_time = NULL,
  cov_means,
  cov_sds = NULL,
  cov_types = NULL,
  study = NULL,
  arm = NULL
)
```

Arguments

`data`	Data frame of reconstructed pseudo-IPD (one row per pseudo-individual).
`treatment`	Column name for the (single) comparator treatment.
`Surv`	Optional `survival::Surv()` object describing the outcome. Use this for left/interval censoring or delayed entry.
`time`, `status`, `entry_time`	Character column names as an alternative to `Surv` (right-censoring with status `0`/`1`, plus optional delayed entry).
`cov_means`	Character vector of covariate mean/proportion column names (constant within each arm). Suffixes `_mean`/`_prop` are stripped to match the IPD covariate names.
`cov_sds`	Character vector of covariate SD column names (`NA` for binary covariates). `NULL` treats all covariates as binary.

cov_types	Character vector of "continuous"/"binary" per covariate. If NULL, inferred from the presence of an SD column.
study	Optional study identifier column.
arm	Optional arm identifier column. Only a single comparator arm is supported; if supplied, it must have one unique value. Multi-arm reconstructed survival comparators are rejected until a weighting estimand is implemented. Defaults to a single arm.

Details

Under delayed entry the comparator likelihood conditions each integration point on survival to its entry time and then averages, so cov_means, cov_sds and the distributions add_integration() builds must describe the population observed at entry (those in the risk set), not a baseline population before selection. With varying entry times that population can differ by entry time while the model has one distribution per arm; pooled summaries are right only under a common entry time or when the covariate distribution among those observed at entry is the same at every entry time. Neither condition is checkable from the summaries supplied. Delayed entry in the individual arm is unaffected.

Value

An object of class mlumr_agd_surv (also inheriting mlumr_agd). The internal column names cannot be used as column names in data.

Reconstruction uncertainty is not propagated

The pseudo-individual records enter the likelihood as observed data, so credible intervals are conditional on this one reconstruction and narrower than the evidence supports. Treat the reconstruction as an analysis choice: digitize the curve more than once or perturb the points within their reading error, refit, and report the spread across refits beside the within-fit interval.

See Also

set_agd() for non-survival aggregate data; multinma::set_agd_surv() is the ML-NMR equivalent.

Examples

```
## Not run:
agd <- set_agd_surv(
  data = comparator_km,
  treatment = "trt",
  time = "time", status = "status",
  cov_means = c("age_mean", "male_prop"),
  cov_sds = c("age_sd", NA),
  cov_types = c("continuous", "binary")
)

## End(Not run)
```

set_ipd *Set up individual patient data (IPD)*

Description

Prepare IPD from the index treatment for an unanchored indirect comparison.

Usage

```
set_ipd(
  data,
  treatment,
  outcome = NULL,
  covariates,
  family = c("binomial", "normal", "poisson", "survival"),
  exposure = NULL,
  study = NULL,
  Surv = NULL,
  time = NULL,
  status = NULL,
  entry_time = NULL
)
```

Arguments

data	Data frame containing IPD
treatment	Column name for treatment variable
outcome	Column name for outcome variable. For `family = "binomial"`, must be binary (0/1). For `family = "normal"`, any numeric. For `family = "poisson"`, non-negative integer counts. Not used (leave `NULL`) for `family = "survival"`, which uses `Surv`/`time`/`status` instead.
covariates	Character vector of covariate column names
family	Outcome family: `"binomial"`, `"normal"`, `"poisson"`, or `"survival"` (time-to-event)
exposure	Column name for exposure/time-at-risk (required when `family = "poisson"`)
study	Column name for study identifier (optional)
Surv	For `family = "survival"`, an optional `survival::Surv()` object describing the outcome (use for left/interval censoring or delayed entry).
time, status, entry_time	For `family = "survival"`, character column names as an alternative to `Surv` (right-censoring with status `0`/`1`, plus optional delayed entry).

Value

An object of class `mlumr_ipd`. Its `$data` holds the treatment, study, outcome and covariate columns under internal names, which cannot be used as column names in `data`.

Examples

```
## Not run:
# Binary outcome
ipd <- set_ipd(
  data = trial_a,
  treatment = "trt",
  outcome = "response",
  covariates = c("age", "sex")
)

# Continuous outcome
ipd <- set_ipd(
  data = trial_a,
  treatment = "trt",
  outcome = "score",
  covariates = c("age", "sex"),
  family = "normal"
)

# Count outcome with exposure
ipd <- set_ipd(
  data = trial_a,
  treatment = "trt",
  outcome = "events",
  covariates = c("age", "sex"),
  family = "poisson",
  exposure = "person_years"
)

## End(Not run)
```

---

shoulder_agd *Shoulder pain: simulated comparator aggregate data (continuous)*

Description

Aggregate summary of the comparator treatment (exercise therapy, ET) for the continuous ML-UMR example; pair with `shoulder_ipd`. The two arms are treated as separate single-arm sources to illustrate an unanchored comparison. Not real patient data.

Usage

```
shoulder_agd
```

Format

A data frame with 1 row and 10 columns: `study`, `treatment`, `n`, outcome summary (`y_mean`, `y_se`) and covariate summaries (`age_mean`/`age_sd`, `sex_prop` (a proportion), `baseline_vas_mean`/`baseline_vas_sd`).

Source

Simulated; see shoulder_ipd for the modeling basis.

---

shoulder_ipd *Shoulder pain: simulated index IPD (continuous)*

Description

Simulated individual patient data for the index treatment (arthroscopic subacromial decompression, ASD) in a continuous-outcome ML-UMR example. Pair with shoulder_agd (the comparator). Not real patient data.

Usage

```
shoulder_ipd
```

Format

A data frame with 147 rows and 7 columns:

study study label ("FIMPACT")

treatment index treatment label ("ASD")

subject row identifier

age age in years

sex 1 = male, 0 = female

baseline_vas baseline shoulder pain on activity (VAS 0-100)

pain_vas_activity shoulder pain on activity at 24 months (VAS 0-100)

Details

The index and comparator arms come from the same trial, so both carry the study label "FIMPACT". Splitting one randomized trial into a single-arm IPD source and a single-arm aggregate source is what makes this an unanchored example that still has a full-data comparison to be checked against: fitting the two arms together on the complete synthetic records gives the quantity the unanchored methods are trying to recover once one arm has been reduced to summaries. That is a data-reduction benchmark. It is not a randomized reference and not a known population causal effect. The generator draws treatment first and then synthesizes the covariates conditional on it, so the synthetic arms are not a fresh randomized assignment independent of the generated baseline variables; and the full-data estimate carries its own sampling error. Validating an estimator against a causal target would need a specified covariate distribution, potential-outcome mechanism, assignment rule and estimand, which these records do not provide. Because the label is shared, combine_data warns that IPD and AgD come from the same study; that warning is expected here and is the honest reading of the data. It does not fire for psoriasis_ipd or ndmm_ipd, whose arms really do come from different trials.

Source

Simulated (not real patient data), generated with the **synthpop** package (sequential CART; Nowok, Raab and Dibben 2016, doi:10.18637/jss.v074.i11) from the FIMPACT 10-year trial (BMJ 2025;391:e086201; dataset CC BY 4.0, University of Helsinki / Finnish Ministry of Education open-data portal, doi:10.23729/fdd323a34bf6983bc6b38d9c93aeadbe74), preserving its covariate and covariate-outcome relationships; fidelity validated with the **syntheticdata** package. See `data-raw/simulate_external_data.R`.

`stc` *Simulated treatment comparison via G-computation*

Description

Perform an unanchored simulated treatment comparison (STC) with marginal standardization. Fits an outcome regression to the single IPD treatment arm, standardizes its predictions over the comparator-population covariate distribution, and contrasts that outcome with the reported comparator-arm outcome in the same population. It does not standardize either treatment to the index population.

Usage

```
stc(
  data,
  link = NULL,
  conf_level = 0.95,
  distribution = "weibull",
  n_boot = 200L,
  seed = NULL,
  rmst_horizon = NULL
)
```

Arguments

`data`	An `mlumr_data` object from `combine_data()`. Integration points from `add_integration()` are required whenever the outcome model uses a nonlinear link (binomial, Poisson, normal-log, or survival). Substitution of aggregate means is exact only for a normal identity-link model.
`link`	Link function. For binomial: `"logit"` (default), `"probit"`, or `"cloglog"`. For normal: `"identity"` (default) or `"log"`. For poisson: `"log"` (default). Ignored for survival. If `NULL`, uses the canonical default.
`conf_level`	Confidence level for the interval (default 0.95)
`distribution`	For `family = "survival"`: the parametric distribution of the survival G-computation (default `"weibull"`), fitted with **flexsurv**; the estimand is the restricted mean survival time difference. `"mspline"` and `"pexp"` have no parametric analogue and fall back to a Weibull fit with a warning, recorded as `approximated = TRUE`. Survival STC supports right-censored data without delayed entry.

`n_boot` For `family = "survival"` only: number of bootstrap resamples for the RMST-difference standard error (default `200`; 0 gives a point estimate with no interval). Other families use the delta method.

`seed` For `family = "survival"` only: optional integer seed for the bootstrap, making the standard error reproducible. The global random number stream is restored on exit. Ignored for other families.

`rmst_horizon` For `family = "survival"` only: the restriction time of the RMST difference. Defaults to the largest observed time across both arms; set it explicitly to match an `mlumr()` fit, whose default can differ. A horizon beyond the observed range extrapolates and warns.

Details

For binomial outcomes the result carries the link-scale effect, the standardized and observed event probabilities, the risk difference and the log risk ratio, with delta-method Wald intervals; the observed comparator proportion gets the exact Clopper-Pearson interval, as in `naive()`. An observed comparator arm with zero or all events uses the pseudo-count `(r + 0.5) / (n + 1)` in transformed measures. For Poisson outcomes `$rd` is a rate difference per unit exposure and `$estimate` the log rate ratio, with a 0.5 continuity correction for a zero comparator count.

Scale note: `$estimate` (and the binomial `$log_rr`) is on the link / log scale, where the null is 0. To compare against the natural-scale risk ratio or rate ratio from `marginal_effects()` (where the null is 1), exponentiate it (e.g. `exp(result$estimate)`).

Normal-family weighting note: across multiple AgD rows, the normal STC comparator-population prediction and observed mean use sample-size (`outcome_n`) weights, matching the Bayesian ML-UMR comparator-population estimand. `outcome_n` is required when there is more than one row; a single row has weight one. The observed comparator-mean variance combines independent, mutually exclusive strata as `sum(w^2 * se^2)` using normalized population weights.

A GLM is fitted to the IPD, its predictions are averaged over the comparator covariate distribution (the integration points, or the AgD means for the identity-link normal case) on the response scale, and the average is contrasted with the reported comparator outcome, as in Ren et al.'s unanchored STC. Standard errors are first-order delta-method values, conditional on the integration grid and the reported comparator summaries. The estimand is `E_B[m_A(X)] - E_B[Y_B]` in the comparator population (`$rd` for binomial, `$md` for normal); `$estimate` is the link-scale contrast of the two standardized quantities. Survival STC contrasts the index RMST standardized to the comparator covariates with the RMST of an intercept-only `flexsurv::flexsurvreg()` fit to the pseudo-IPD. The effect is not transported to the index population; `mlumr(model = "relaxed")` estimates comparator-specific covariate effects for any other target.

Value

An object of class `mlumr_stc`. Its `separation` component records the outcome of the binomial separation check: `status` is `"not_separated"` when the exact check ran and found none, `"unknown"` when only the fitted-value screen ran (it cannot see quasi-complete separation), and `"not_applicable"` for other outcome models. A separated fit, and a Poisson fit with no events, are refused rather than returned.

References

Ren S, Ren S, Welton NJ, Strong M (2024). Advancing unanchored simulated treatment comparisons: A novel implementation and simulation study. *Research Synthesis Methods*, 15(4), 657-670. doi:10.1002/jrsm.1718

Remiro-Azocar A, Heath A, Baio G (2022). Parametric G-computation for compatible indirect treatment comparisons with limited individual patient data. *Research Synthesis Methods*, 13(6), 716-744. doi:10.1002/jrsm.1565

Examples

```
## Not run:
result <- stc(dat)
print(result)

## End(Not run)
```

`unnest_integration` *Expand integration points into a long-format data frame*

Description

Expand integration points into a long-format data frame

Usage

```
unnest_integration(data)
```

Arguments

`data`	An `mlumr_data` object with integration points

Value

A data frame with columns for each covariate plus `.int_id` and `.agd_row`

Index

∗ **datasets**
caries_agd, 8
caries_ipd, 8
ndmm_agd, 33
ndmm_agd_covs, 34
ndmm_ipd, 35
psoriasis_agd, 48
psoriasis_ipd, 48
shoulder_agd, 55
shoulder_ipd, 56

add_integration, 3
add_integration(), *12*, *19*, *20*, *22*, *26*, *53*, *57*

calculate_dic, 5
calculate_loo, 5
calculate_loo(), *5*, *7*, *13*
calculate_waic, 7
calculate_waic(), *5*, *13*
caries_agd, 8, *8*
caries_ipd, 8, *8*
check_diagnostics, 10
check_identification, 10
check_identification(), *29*
check_integration, 11
check_integration(), *4*
combine_data, *9*, 12, *56*
combine_data(), *4*, *20*, *23*, *32*, *57*
compare_models, 13
conditional_effects, 14
conditional_effects(), *16*, *25*, *37*, *41*
conditional_predict, 16
conditional_predict(), *15*, *41*

dbern, 17
default_prior_aux *(default_priors)*, 17
default_prior_aux(), *27*
default_prior_beta *(default_priors)*, 17
default_prior_beta(), *27*
default_prior_intercept *(default_priors)*, 17
default_prior_intercept(), *27*
default_prior_sigma *(default_priors)*, 17
default_prior_sigma(), *27*
default_prior_smooth *(default_priors)*, 17
default_prior_smooth(), *27*
default_priors, 17
dgamma *(GammaDist)*, 19
distr, 18
distr(), *4*, *19*, *20*, *22*
dlogitnorm *(logitNormal)*, 21

flexsurv::flexsurvreg(), *58*

GammaDist, 19, *22*
geom_km, 20
geom_km(), *38*
ggplot2::geom_step(), *21*
ggplot2::labs(), *31*

logitNormal, 21
loo::loo(), *6*
loo::loo_compare(), *13*, *14*
loo::loo_moment_match(), *6*
loo::waic(), *7*

make_knots, 22
make_knots(), *27*
marginal_effects, 23
marginal_effects(), *14*, *15*, *28*, *32*, *33*, *37*, *41*, *45*, *58*
mlumr, 26
mlumr(), *10*, *11*, *17*, *23*, *33*, *39*, *43–45*, *47*, *58*
mlumr_engine, 30
mlumr_engine(), *28*
mlumr_forest, 31

naive, 32
naive(), *25*, *32*, *58*

ndmm_agd, 33, *35*
ndmm_agd_covs, *33*, 34, *35*
ndmm_ipd, *9*, *33, 34*, 35, *56*

pbern, 36
pgamma *(*GammaDist*)*, 19
plogitnorm *(*logitNormal*)*, 21
plot.mlumr_conditional_effects, 36
plot.mlumr_marginal_effects, 37
plot.mlumr_prediction, 38
plot.mlumr_prediction(), *21*
plot_prior_posterior, 39
predict.mlumr_fit, 40
predict.mlumr_fit(), *15, 16*, *25*, *38*
prior_cauchy, 42
prior_cauchy(), *47*
prior_exponential, 42
prior_exponential(), *27*, *47*
prior_normal, 43
prior_normal(), *17, 18*, *27*, *42*, *45–47*
prior_sensitivity, 44
prior_sensitivity(), *6*, *10, 11*, *25*, *29*, *44*, *47*
prior_sigma, *42*
prior_student_t, 46
prior_student_t(), *42*, *47*
prior_summary, 47
prior_summary(), *17*, *29*, *39*, *44, 45*
psoriasis_agd, 48, *48*
psoriasis_ipd, *9*, 48, *48*, *56*

qbern, 49
qbern(), *20*
qgamma *(*GammaDist*)*, 19
qlogitnorm *(*logitNormal*)*, 21

rstan::sampling(), *29*

set_agd, 50
set_agd(), *13*, *29*, *53*
set_agd_surv, 52
set_agd_surv(), *50*
set_ipd, *9*, *48*, 54
set_ipd(), *13*, *51*
shoulder_agd, 55, *56*
shoulder_ipd, *55*, 56, *56*
stats::dnorm(), *22*
stats::GammaDist, *19*
stats::pnorm(), *22*
stats::qnorm(), *22*
stc, 57
stc(), *25*, *32, 33*
survival::Surv(), *52*, *54*

unnest_integration, 59