Package 'poissonsuperlearner'

May 9, 2026

Title Poisson Super Learner

Version 0.1.1

Description Provides tools for fitting piece-wise constant hazard models for survival and competing risks data, including ensemble hazard estimation via the Super Learner framework. The package supports estimation of survival functions and absolute risk predictions from fitted cause-specific hazard models. For the Super Learner framework see van der Laan, Polley and Hubbard (2007) <doi:10.2202/1544-6115.1309>.

License GPL (>= 2)

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.3.3

Depends data.table, sampling, riskRegression

Imports Rcpp, methods, lava, Matrix, glmnet, mgcv

Suggests knitr, rmarkdown, survival, prodlim, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author Gabriele Pittarello [aut, cre],
Helene Rytgaard [aut],
Thomas Gerds [aut]

Maintainer Gabriele Pittarello <gabriele.pittarello@sund.ku.dk>

Repository CRAN

Date/Publication 2026-04-04 10:00:02 UTC

Contents

`coef.base_learner` *Extract coefficients from a fitted base learner*

Description

Convenience method to extract (cause-specific) model coefficients from a fitted `base_learner` returned by `fit_learner()`.

Usage

```
## S3 method for class 'base_learner'
coef(object, cause = NULL, ...)
```

Arguments

`object`	`base_learner`. A fitted object returned by `fit_learner()`.
`cause`	`numeric(1)` or `NULL`. Which cause to extract coefficients for. If `NULL`, coefficients are returned for all causes. Causes are indexed `1, 2, ..., object$data_info$n_crisks` (with `0` reserved for censoring).
`...`	Passed to the underlying `coef()` method of the fitted learner object (learner-dependent; e.g., `s` for `glmnet`).

Details

For competing risks, `fit_learner()` fits one model per cause, stored in `object$learner_fit[[k]]` for `k = 1, 2, ..., K`. This method simply dispatches to the underlying model's `coef()` method for each fitted object.

Learner-dependent output. The returned coefficient object depends on the base learner used (e.g. a numeric vector, a sparse matrix, a list, etc.). This method does not post-process or rename coefficients; it returns the output of `coef(object$learner_fit[[k]], ...)` unchanged.

Value

If cause is a single integer, returns the coefficient object produced by `coef()` for that cause-specific fitted model.

If `cause = NULL`, returns a `list` of length `object$data_info$n_crisks`, where element `[[k]]` contains coefficients for cause `k`.

If no fitted model is present (`object$learner_fit` is `NULL`), signals a message and returns `invisible(object)`.

Examples

```
d <- simulateStenoT1(50, competing_risks = TRUE)
lrn <- Learner_glmnet(covariates = c("age", "value_LDL"),
                      lambda = 0, cross_validation = FALSE)
bl <- fit_learner(d, learner = lrn, id = "id",
                  status = "status_cvd", event_time = "time_cvd",
                  number_of_nodes = 4)

# coefficients for cause 1
coef(bl, cause = 1)

# coefficients for all causes (list)
coef(bl)
```

---

coef.poisson_superlearner

Extract stacking (meta-learner) coefficients from a fitted Poisson Super Learner

---

Description

Extracts the **meta-learner coefficients** (stacking weights) from a fitted `poisson_superlearner` object returned by `Superlearner()`.

Usage

```
## S3 method for class 'poisson_superlearner'
coef(object, cause = NULL, model = "sl", ...)
```

Arguments

`object`	`poisson_superlearner`. A fitted ensemble returned by `Superlearner()`.
`cause`	`numeric(1)` or `NULL`. Which cause to extract meta-learner coefficients for. If `NULL`, coefficients are returned for all causes. Causes are indexed `1, 2, ..., object$data_info$n_cris`
`model`	Scalar model selector. Default is `"sl"` for the stacked super learner. Other allowed values are:
	`0` **or** `"sl"` Use the super learner prediction.

learner label Use one stored base learner by its label in `object$data_info$learners_labels`.

`"learner_j"` Use the `j`-th stored learner.

integer `j >= 1` Use the `j`-th stored learner.

`...` Passed to the underlying `coef()` method of the fitted meta-learner (learner-dependent; e.g., `s` for `glmnet`).

Details

For each cause `k`, the ensemble stores a fitted meta-learner in `object$superlearner[[k]]$meta_learner_fit`. This method dispatches to the underlying `coef()` method for that fitted meta-learner.

What coefficients represent. These coefficients correspond to the meta-learner regression of the outcome on the cross-validated base-learner predictions (`Z1`, `Z2`, ...). Under the default meta-learner, they are the stacking weights (on the scale defined by the meta-learner).

Learner-dependent output. The returned coefficient object depends on the meta-learner implementation (by default a `glmnet` fit, often returning a sparse matrix). This method does not rename `Z*` terms or post-process coefficients; it returns the output of `coef(object$superlearner[[k]]$meta_learner_fit, ...)` unchanged.

Single-learner special case. If the ensemble was fit with only one base learner, no meta-learner is fit and `meta_learner_fit` is `NULL`. In that case, `coef()` for the `poisson_superlearner` does not have meta-learner coefficients to return.

Value

If `cause` is a single integer, returns the coefficient object produced by `coef()` for the cause-specific fitted meta-learner.

If `cause = NULL`, returns a `list` of length `object$data_info$n_crisks`, where element `[[k]]` contains meta-learner coefficients for cause `k`.

If no fitted ensemble is present (`object$superlearner` is `NULL`), signals a message and returns `invisible(object)`.

Examples

```
d <- simulateStenoT1(50, competing_risks = TRUE)
learners <- list(
 glm = Learner_glmnet(covariates = c("age", "value_LDL"), lambda = 0, cross_validation = FALSE),
  gam = Learner_gam(covariates = c("age", "value_LDL"))
)
fit <- Superlearner(d, id="id", status="status_cvd", event_time="time_cvd",
                    learners=learners, number_of_nodes=4, nfold=2)

# meta-learner coefficients (cause 1)
coef(fit, cause = 1)

# meta-learner coefficients for all causes (list)
coef(fit)
```

`fit_learner` *Fit a single base learner*

Description

Pre-processes subject-level time-to-event data into a long Poisson format on a piecewise-constant time grid, then fits **one** initialized learner object. For competing risks, a separate model is fit for each event type (cause) using the standard cause-specific Poisson likelihood on the long data.

Usage

```
fit_learner(
  data,
  learner,
  id = "id",
  stratified_k_fold = FALSE,
  status = "status",
  event_time = NULL,
  number_of_nodes = NULL,
  nodes = NULL,
  variable_transformation = NULL,
  ...
)
```

Arguments

`data`	`data.frame`. Subject-level input data (one row per subject).
`learner`	Reference-class learner object (e.g. from `Learner_glmnet()`, `Learner_hal()` or `Learner_gam()`). Must implement a `$private_fit(dt_long)` method that fits the learner on long Poisson data for one cause.
`id`	`character(1)`. Name of the subject identifier column. If not found in `data`, an `id` column is created automatically.
`stratified_k_fold`	`logical(1)`. Reserved argument for future fold strategy. Currently ignored.
`status`	`character(1)`. Name of the event-status column. Must be coded with `0` = censoring and `1,2,...,K` for event types (causes). If there is no `0` in `status`, the data are treated as uncensored.
`event_time`	`character(1)`. Name of the event/censoring time column. Must be present in `data`.
`number_of_nodes`	`numeric(1)` or `NULL`. If not `NULL`, constructs a quantile-based node grid with `number_of_nodes + 1` cut points (including endpoints), then adds `0` if missing.
`nodes`	`numeric` or `NULL`. Explicit time-node grid (cut points). If supplied, `number_of_nodes` is ignored. `0` is added if missing. Nodes beyond `max(event_time)` are dropped.

variable_transformation
: list/character/formula or NULL. Optional transformations applied to the internally created long Poisson data before fitting (via apply_transformations()).

...
: Additional arguments currently ignored.

Value

An object of class base_learner, i.e. a named list with:

model The **learner object** that was fit (the input learner), stored for later prediction. This contains the learner specification (e.g., covariates, tuning parameters).

learner_fit A list of fitted model objects, **one per cause**. Its length equals data_info$n_crisks. The list is created by splitting the internally pre-processed long data by cause indicator k and calling model$private_fit() on each split.

- Names typically correspond to the cause labels "1", "2", ..., "K".
- Each element is **learner-dependent**: e.g. for Learner_glmnet it may be a "glmnet" (often wrapped, e.g. "fishnet") fit; for other learners it will be whatever $private_fit() returns.
- Each fitted object is trained on long Poisson data representing the piecewise-constant hazard for that cause across the node intervals.

data_info A list of bookkeeping information needed for prediction and interpretation:

id Identifier column name used.

status Status column name used.

event_time Event/censoring time column name used.

nodes Numeric vector of node cut points used for the piecewise grid (includes 0 and is sorted). These are the interval boundaries used in the long Poisson representation.

maximum_followup max(data[[event_time]]).

n_crisks Number of event types (causes) detected. If censoring is present (0 in status), then n_crisks = #unique(status) - 1; otherwise n_crisks = #unique(status).

variable_transformation The transformation specification passed in variable_transformation (or NULL).

Examples

```
d <- simulateStenoT1(50, competing_risks = TRUE)
lrn <- Learner_glmnet(covariates = c("age", "value_LDL"),
                      lambda = 0,
                      cross_validation = FALSE)
bl <- fit_learner(d,
                  learner = lrn,
                  id = "id",
                  status = "status_cvd",
                  event_time = "time_cvd",
                  number_of_nodes = 4)
```

Learner_gam-class *GAM learner via* mgcv::bam

Description

Learner_gam is a Reference Class implementing the learner interface used by Superlearner() and fit_learner().

Arguments

covariates	character. Right-hand-side terms, including mgcv smooths (e.g. "s(age)") and/or linear terms (e.g. "value_LDL").
cross_validation	logical. Included for compatibility with the learner interface; smoothing selection is controlled by mgcv and arguments in ....

Details

User-facing API: users should **only initialize** the learner and pass it to Superlearner() / fit_learner(). The methods private_fit() and private_predictor() are part of the internal learner interface and are **not meant to be called directly by users**.

Wrapper role: this class wraps mgcv::bam in a piecewise-constant hazard workflow. The package-specific contribution is to provide a convenient interface for the long-format Poisson likelihood with offsets for time at risk, and optional node terms encoding the baseline hazard, while forwarding standard mgcv::bam arguments supplied via ....

Model

Let $0 = t_0 < t_1 < \cdots < t_m$ denote time knots and define interval indicators $I_k(t) = 1\{t \in (t_k, t_{k+1}]\}$. The piecewise-constant hazard model with an additive predictor is

$$\lambda(t \mid x) = \sum_{k=0}^{m} I_k(t) \exp\{\eta(x) + \gamma_k\}.$$

The additive predictor $\eta(x)$ is constructed from covariates (smooth terms such as s(age) and/or linear terms) and estimated by mgcv.

Fields

covariates (character) Terms used to build the additive predictor (may include s() terms).

cross_validation (logical) Workflow flag; see Details.

intercept (logical) Whether to include an intercept.

add_nodes (logical) If TRUE, include interval ("node") effects encoding the baseline hazard.

formula (character) Formula string passed to mgcv::bam.

learner (function) Backend fitter (mgcv::bam).

fit_arguments (list) Additional arguments forwarded to mgcv::bam.

Methods (internal learner interface)

`initialize(...)` Construct and configure the learner. This is the only method users should call.

`private_fit(data, ...)` Internal. Fits a Poisson GAM with offset `log(tij)` on long-format data.

`private_predictor(model, newdata, ...)` Internal. Predicts hazards on the response scale.

Examples

```
lrn <- Learner_gam(covariates = c("s(age)", "value_LDL"))
```

Learner_glmnet-class *Penalized Poisson learner via* `glmnet`

Description

`Learner_glmnet` is a Reference Class implementing the learner interface used by `Superlearner()` and `fit_learner()`.

Details

User-facing API: users are expected to **initialize** the learner (i.e., call `Learner_glmnet(...)`) and pass the resulting object to `Superlearner()` or `fit_learner()`. The remaining methods documented below (e.g., `private_fit()`, `private_predictor()`) are part of the internal learner interface and are **not meant to be called directly by users**.

Wrapper role: this class is a user-friendly wrapper around the existing `glmnet` implementation. The package-specific contribution is to provide a piecewise-constant hazard workflow: create the long-format Poisson data with offsets for time at risk, add the interval ("node") structure for the baseline hazard when requested, and forward standard `glmnet` arguments supplied at initialization to the backend fitter.

Model

Let $0 = t_0 < t_1 < \cdots < t_m$ denote time knots and define interval indicators $I_k(t) = 1\{t \in (t_k, t_{k+1}]\}$. The piecewise-constant hazard model is

$$\lambda(t \mid x) = \sum_{k=0}^{m} I_k(t)\, \lambda_k(x), \qquad \lambda_k(x) = \exp(\beta^\top x + \gamma_k).$$

Penalization is applied to the regression coefficients through the `glmnet` elastic-net penalty. If you want node (baseline) terms to be unpenalized, use `penalty.factor` via `...` (and set it consistently with how your design matrix encodes nodes).

Fields

`covariates` (`character`) Names of covariate columns used in the model.

`cross_validation` (`logical`) If `TRUE`, chooses `lambda` by `glmnet::cv.glmnet`.

`intercept` (`logical`) Whether to include an intercept in the backend fit.

`add_nodes` (`logical`) If `TRUE`, include interval ("node") effects encoding the baseline hazard.

`lambda` (`numeric`) If `cross_validation=FALSE`, the `lambda` used in the final fit.

`formula` (`character`) Formula string used to create the design matrix in long format.

`learner` (`function`) Backend fitter (`glmnet::glmnet` or `glmnet::cv.glmnet`).

`fit_arguments` (`list`) Additional arguments forwarded to the backend fitter.

Methods (internal learner interface)

`initialize(...)` Construct and configure the learner. This is the only method users should call.

`private_fit(data, ...)` Internal. Fits a Poisson model with offset `log(tij)` on long-format data.

`private_predictor(model, newdata, ...)` Internal. Predicts hazards on the response scale for long-format `newdata`.

Examples

```
lrn <- Learner_glmnet(covariates = c("age", "sex"), alpha = 1, cross_validation = TRUE)
```

Learner_hal-class *HAL learner for piecewise Poisson hazards*

Description

`Learner_hal` is a Reference Class implementing the learner interface used by `Superlearner()` and `fit_learner()`.

Details

User-facing API: users should **only initialize** the learner and pass it to `Superlearner()` / `fit_learner()`. The methods `private_fit()` and `private_predictor()` (and any basis-construction helpers) are part of the internal learner interface and are **not meant to be called directly by users**.

Wrapper role: this class provides a piecewise-constant hazard wrapper around a HAL-style indicator-basis construction, estimated by L1-penalized Poisson regression using a `glmnet` backend. The package-specific contribution is to (i) construct the long-format Poisson representation with offsets for time at risk, (ii) generate indicator bases compatible with piecewise hazards, and (iii) forward backend fitting arguments supplied via ....

Model

Let $0 = t_0 < t_1 < \cdots < t_m$ denote time knots and define interval indicators $I_k(t) = 1\{t \in (t_k, t_{k+1}]\}$. The HAL piecewise-constant hazard model is

$$\lambda(t \mid x) = \sum_{k=0}^{m} I_k(t) \exp\{f(t, x)\},$$

where $f(t, x)$ is approximated by a finite linear combination of indicator basis functions.

Two-covariate illustration

Let $x = (x_1, x_2)$ be two covariates and let $t_0 < t_1 < \cdots < t_R$ be time grid points used to create step functions in time. Choose covariate cutpoints $c_{1,1}, \ldots, c_{1,K_1}$ for x_1 and $c_{2,1}, \ldots, c_{2,K_2}$ for x_2.

Define indicator bases:

$$B_r(t) = 1\{t_r \le t\}$$
$$B_{1,p}(x) = 1\{c_{1,p} \le x_1\}$$
$$B_{2,q}(x) = 1\{c_{2,q} \le x_2\}$$

A main-effects HAL approximation on the log-hazard scale can be written as:

$$f_\beta(t, x) = \beta_0 + \sum_{r=1}^{R} \beta_r B_r(t) + \sum_{r=1}^{R} \sum_{p=1}^{K_1} \beta_{r,1,p} B_r(t) B_{1,p}(x) + \sum_{r=1}^{R} \sum_{q=1}^{K_2} \beta_{r,2,q} B_r(t) B_{2,q}(x).$$

If `max_degree >= 2`, the learner additionally includes interaction bases such as

$$\sum_{r=1}^{R} \sum_{p=1}^{K_1} \sum_{q=1}^{K_2} \beta_{r,12,pq} B_r(t) B_{1,p}(x) B_{2,q}(x).$$

How reference class parameters map to the model

`covariates` Covariate columns used to build covariate indicator bases.

`num_knots` Controls the number of cutpoints per covariate used for indicator bases.

`max_degree` Maximum interaction order included in the basis expansion.

`add_nodes` If `TRUE`, includes interval ("node") structure for the baseline hazard.

`intercept` Whether the backend penalized regression includes an intercept term.

`cross_validation` If `TRUE`, selects the penalty level using `glmnet::cv.glmnet`.

`fit_arguments` Additional arguments forwarded to the `glmnet` backend (e.g. `nfolds`).

Fields

`covariates` (`character`) Names of covariate columns used in the basis.

`cross_validation` (`logical`) Whether to use `cv.glmnet` to select the penalty.

`intercept` (`logical`) Backend intercept flag.

`add_nodes` (`logical`) Whether node (time-interval) effects are included.

`max_degree` (`integer`) Maximum interaction order.

`num_knots` (`numeric`) Knots used for basis construction.

`lambda_opt` (`numeric`) Selected penalty level when using cross-validation.

`fit_arguments` (`list`) Extra backend arguments forwarded to `glmnet`.

Methods (internal learner interface)

`initialize(...)` Construct and configure the learner. This is the only method users should call.

`private_fit(data, ...)` Internal. Builds bases and fits the penalized Poisson model with offset `log(tij)`.

`private_predictor(model, newdata, ...)` Internal. Evaluates the fitted approximation and returns hazards on the response scale.

Examples

```
lrn <- Learner_hal(covariates = c("age", "sex"), max_degree = 2L, num_knots = c(10L, 5L))
```

---

pch_absolute_risk *Absolute risk (cumulative incidence) for a cause under piecewise-constant hazards*

Description

Computes, per row, the cumulative incidence function at the end of each interval, grouped by `id`. The number of causes is inferred from the number of columns in `haz`.

Usage

```
pch_absolute_risk(id, dt, haz, cause_idx, one_based = TRUE, na_is_zero = FALSE)
```

Arguments

`id`	Integer vector. Sorted by `id` then time.
`dt`	Numeric vector of interval lengths.
`haz`	Numeric matrix (n x C) of cause-specific hazards per interval. Columns correspond to causes 1..C.
`cause_idx`	Integer. Index of the cause of interest (1-based by default).
`one_based`	Logical. If `TRUE`, `cause_idx` is 1-based. If `FALSE`, 0-based.
`na_is_zero`	Logical. If `TRUE`, treat NA/Inf hazards as zero.

Value

Numeric vector of cumulative incidence values at the end of each interval.

Examples

```
id <- c(1L, 1L, 2L, 2L)
dt <- c(1, 1, 1, 1)
haz <- rbind(
  c(0.10, 0.05),
  c(0.20, 0.10),
  c(0.05, 0.02),
  c(0.10, 0.03)
)
pch_absolute_risk(id = id, dt = dt, haz = haz, cause_idx = 1)
```

`pch_absolute_risk_euler` *Absolute risk (Euler approximation) for a cause under piecewise-constant hazards*

Description

Computes the cumulative incidence function using the first-order Euler (discrete) approximation:

$$F_j(t) \approx \sum S(t_{k-1})\lambda_{j,k}\Delta t_k$$

Grouped by `id`, this returns the cumulative incidence at the end of each interval.

Usage

```
pch_absolute_risk_euler(
  id,
  dt,
  haz,
  cause_idx,
  one_based = TRUE,
  na_is_zero = FALSE
)
```

Arguments

`id`	Integer vector. Sorted by `id` then time.
`dt`	Numeric vector of interval lengths.
`haz`	Numeric matrix (n x C) of cause-specific hazards per interval.
`cause_idx`	Integer. Index of the cause of interest (1-based by default).
`one_based`	Logical. If `TRUE`, `cause_idx` is 1-based. If `FALSE`, 0-based.
`na_is_zero`	Logical. If `TRUE`, treat NA/Inf hazards as zero.

Value

Numeric vector of cumulative incidence values (Euler approximation) at the end of each interval.

Examples

```
id <- c(1L, 1L, 2L, 2L)
dt <- c(1, 1, 1, 1)
haz <- rbind(
  c(0.10, 0.05),
  c(0.20, 0.10),
  c(0.05, 0.02),
  c(0.10, 0.03)
)
pch_absolute_risk_euler(id = id, dt = dt, haz = haz, cause_idx = 1)
```

---

pch_survival *Piecewise-constant hazards survival function*

Description

Computes survival at the end of each interval for competing risks with piecewise constant hazards.

Usage

```
pch_survival(id, dt, haz, na_is_zero = FALSE)
```

Arguments

id	Integer vector of subject IDs, sorted by id then time.
dt	Numeric vector of interval lengths.
haz	Numeric matrix (n x C) of cause-specific hazards.
na_is_zero	Logical. If TRUE, treat NA hazards as zero.

Value

Numeric vector of survival probabilities at the end of each interval.

Examples

```
id <- c(1L, 1L, 2L, 2L)
dt <- c(1, 1, 1, 1)
haz <- rbind(
  c(0.10, 0.05),
  c(0.20, 0.10),
  c(0.05, 0.02),
  c(0.10, 0.03)
)
pch_survival(id = id, dt = dt, haz = haz)
```

predict.base_learner *Predict hazards, survival and absolute risk from a fitted base learner*

Description

Computes **cause-specific piecewise-constant hazards** (pwch_k), the corresponding **survival function**, and **absolute risk** for a given cause, at user-supplied prediction horizons `times`, using a fitted `base_learner` object (single learner; no stacking).

Usage

```
## S3 method for class 'base_learner'
predict(object, newdata, times, cause = 1, ...)
```

Arguments

`object`	`base_learner`. A fitted object returned by `fit_learner()`. It contains the learner specification in `object$model` and cause-specific fitted models in `object$learner_fit`.
`newdata`	`data.frame`/`data.table`. New covariate data (one row per subject). If `newdata` contains the original `event_time`, `status`, or `id` columns used for fitting, they are ignored for prediction.
`times`	`numeric`. Prediction horizon(s). May include `0`. Times larger than `object$data_info$maximum_follow` are not supported: if **all** requested times exceed the maximum follow-up, a warning is issued and `NULL` is returned; if only **some** exceed, output rows for those times are returned with `NA` predictions.
`cause`	`numeric(1)`. Cause index (1, 2, ...) used for the `absolute_risk` calculation.
`...`	Additional arguments (currently ignored).

Details

Internally, `newdata` is expanded to a Cartesian product with `times`, converted to long Poisson format on `object$data_info$nodes`, and the fitted learner for each cause in `object$learner_fit` is used to predict the cause-specific hazards. Survival and absolute risk are then computed from the predicted hazards.

Special case `times = 0`**:** when `0` is included in `times`, the returned rows have `survival_function = 1`, `absolute_risk = 0`, and all `pwch_k = 0` at time 0.

Identifiers in the output: if `newdata` contains the `id` column, it is carried into the output. If `newdata` does not contain an id column, an internal id is created for computation, but it is not guaranteed to appear in the returned table unless it was present in `newdata`.

Value

A `data.table` with one row per (`row in newdata, time in times`) and columns:

(original columns) All columns from `newdata` (excluding ignored event columns).

time column A column with name `object$data_info$event_time` holding the requested horizon.

pwch_1, pwch_2, ... Predicted cause-specific piecewise hazards at the horizon.

survival_function Predicted survival probability at the horizon.

absolute_risk Predicted cumulative incidence (absolute risk) for `cause` at the horizon.

Examples

```
d <- simulateStenoT1(120, competing_risks = TRUE)
lrn <- Learner_glmnet(covariates = c("age", "value_LDL"), lambda = 0, cross_validation = FALSE)
bl <- fit_learner(d, learner = lrn, id="id", status="status_cvd", event_time="time_cvd",
                  number_of_nodes=8)
p <- predict(bl, newdata = d[1:5], times = c(0, 2, 5), cause = 1)
head(p)
```

---

predict.poisson_superlearner

Predict hazards, survival and absolute risk from a fitted Poisson Super Learner

---

Description

Computes **cause-specific piecewise-constant hazards** (`pwch_k`), the corresponding **survival function**, and **absolute risk** for a given cause, at user-supplied prediction horizons `times`, for each row in `newdata`.

Usage

```
## S3 method for class 'poisson_superlearner'
predict(object, newdata, times, cause = 1, model = "sl", ...)
```

Arguments

`object`	`poisson_superlearner`. A fitted ensemble from `Superlearner()`.
`newdata`	`data.frame`/`data.table`. New covariate data (one row per subject). If `newdata` contains the original `event_time`, `status`, or `id` columns used for fitting, they are ignored for prediction.
`times`	`numeric`. Prediction horizon(s). May include `0`. Times larger than `object$data_info$maximum_follow` are not supported: if **all** requested times exceed the maximum follow-up, a warning is issued and `NULL` is returned; if only **some** exceed, output rows for those times are returned with `NA` predictions.
`cause`	`numeric(1)`. Cause index (1, 2, ...) used for the `absolute_risk` calculation.
`model`	Scalar model selector. Default is `"sl"` for the stacked super learner. Other allowed values are:
	`0` **or** `"sl"` Use the super learner prediction.

learner label Use one stored base learner by its label in `object$data_info$learners_labels`.

`"learner_j"` Use the `j`-th stored learner.

integer `j >= 1` Use the `j`-th stored learner.

Numeric positions refer to the learners actually stored in the fitted object.

`...` Additional arguments (currently ignored).

Details

Internally, `newdata` is expanded to a Cartesian product with the requested `times`, converted to long Poisson format on `object$data_info$nodes`, and hazards are predicted either from the stacked super learner (`model = "sl"`) or from one selected fitted base learner. Survival and absolute risk are then computed from the predicted hazards.

Special case `times = 0`**:** when `0` is included in `times`, the returned rows have `survival_function = 1`, `absolute_risk = 0`, and all `pwch_k = 0` at time 0.

Identifiers in the output: if `newdata` contains the `id` column, it is carried into the output. If `newdata` does not contain an id column, an internal id is created for computation, but it is not guaranteed to appear in the returned table unless it was present in `newdata`.

Value

A `data.table` with one row per (`row in newdata, time in times`) and columns:

(original columns) All columns from `newdata` (excluding ignored event columns).

time column A column with name `object$data_info$event_time` holding the requested horizon.

pwch_1, pwch_2, ... Predicted cause-specific piecewise hazards at the horizon.

survival_function Predicted survival probability at the horizon.

absolute_risk Predicted cumulative incidence (absolute risk) for `cause` at the horizon.

Examples

```
d <- simulateStenoT1(30, competing_risks = TRUE)

learners <- list(
  lasso = Learner_glmnet(
    covariates = "sex",
    alpha = 1,
    lambda = 0.01,
    cross_validation = FALSE
  ),
  ridge = Learner_glmnet(
    covariates = c("sex", "value_LDL"),
    alpha = 0,
    lambda = 0.01,
    cross_validation = FALSE
  )
)
```

```
fit <- Superlearner(
  data = d,
  id = "id",
  status = "status_cvd",
  event_time = "time_cvd",
  learners = learners,
  number_of_nodes = 3,
  nfold = 2
)
p <- predict(fit, newdata = d[1:3], times = c(0, 2), cause = 1)
p[, .(id, time_cvd, absolute_risk)]
```

`predictRisk.base_learner` *Absolute-risk matrix predictions for a fitted base learner*

Description

Absolute-risk matrix predictions for a fitted base learner

Usage

```
## S3 method for class 'base_learner'
predictRisk(object, newdata, times, cause = 1, ...)
```

Arguments

`object`	`base_learner`. Fitted object from `fit_learner()`.
`newdata`	`data.frame`. New covariate data.
`times`	`numeric`. Prediction times.
`cause`	`numeric(1)`. Cause index.
`...`	Unused.

Value

`numeric` matrix with `nrow(newdata)` rows and `length(times)` columns.

Examples

```
d <- simulateStenoT1(30, competing_risks = TRUE)
lrn <- Learner_glmnet(
  covariates = c("sex", "value_LDL"),
  lambda = 0.01,
  cross_validation = FALSE
)
bl <- fit_learner(
```

```
  d,
  learner = lrn,
  id = "id",
  status = "status_cvd",
  event_time = "time_cvd",
  number_of_nodes = 3
)

if (requireNamespace("riskRegression", quietly = TRUE)) {
  riskRegression::predictRisk(bl, newdata = d[1:3], times = c(1, 3), cause = 1)
}
```

`predictRisk.poisson_superlearner` *Absolute-risk matrix predictions for a fitted Poisson Super Learner*

Description

S3 method compatible with `riskRegression::predictRisk` returning one column per requested time.

Usage

```
## S3 method for class 'poisson_superlearner'
predictRisk(object, newdata, times, cause = 1, model = "sl", ...)
```

Arguments

`object`	`poisson_superlearner`. Fitted object.
`newdata`	`data.frame`. New covariate data.
`times`	`numeric`. Prediction times.
`cause`	`numeric(1)`. Cause index.
`model`	Scalar model selector. Default is `"sl"`. Allowed values are the same as in `predict.poisson_superlearner()`.
`...`	Unused.

Value

`numeric` matrix with `nrow(newdata)` rows and `length(times)` columns.

Examples

```
d <- simulateStenoT1(30, competing_risks = TRUE)

learners <- list(
  lasso = Learner_glmnet(
    covariates = "sex",
    alpha = 1,
    lambda = 0.01,
    cross_validation = FALSE
  ),
  ridge = Learner_glmnet(
    covariates = c("sex", "value_LDL"),
    alpha = 0,
    lambda = 0.01,
    cross_validation = FALSE
  )
)

fit <- Superlearner(
  data = d,
  id = "id",
  status = "status_cvd",
  event_time = "time_cvd",
  learners = learners,
  number_of_nodes = 3,
  nfold = 2
)

if (requireNamespace("riskRegression", quietly = TRUE)) {
  riskRegression::predictRisk(fit, newdata = d[1:3], times = c(1, 3), cause = 1)
}
```

`print.base_learner` *Print method for* `base_learner`

Description

Prints a compact description of the fitted base learner, including the learner type, the time-grid used, and (optionally) the fitted model object for a given cause.

Usage

```
## S3 method for class 'base_learner'
print(x, cause = 1, ...)
```

Arguments

x	base_learner object returned by fit_learner().
cause	numeric(1) or NULL. Which cause to print the fitted model for. If NULL, prints one line per cause (classes only) instead of printing the full fitted objects.
...	Passed to the underlying fitted object print() method when cause is a single integer.

Value

Invisibly returns x.

Examples

```
d <- simulateStenoT1(30, competing_risks = TRUE)
lrn <- Learner_glmnet(
  covariates = c("sex", "value_LDL"),
  lambda = 0.01,
  cross_validation = FALSE
)
bl <- fit_learner(
  d,
  learner = lrn,
  id = "id",
  status = "status_cvd",
  event_time = "time_cvd",
  number_of_nodes = 3
)
print(bl, cause = NULL)
```

---

print.poisson_superlearner

Print method for poisson_superlearner

Description

Prints a compact description of the fitted Poisson Super Learner, including the number of base learners, the meta-learner, the time-grid used, and competing-risk structure. Optionally prints the fitted meta-learner for a given cause.

Usage

```
## S3 method for class 'poisson_superlearner'
print(x, cause = 1, model = "sl", ...)
```

Arguments

x	poisson_superlearner object returned by Superlearner().
cause	numeric(1) or NULL. Which cause's meta-learner fit to print. If NULL, prints one line per cause (classes only) instead of printing the full fitted objects.
model	Scalar model selector. Default is "sl" for the stacked super learner. Other allowed values are: 0 **or** "sl" Use the super learner prediction. **learner label** Use one stored base learner by its label in object$data_info$learners_labels. "learner_j" Use the j-th stored learner. **integer** j >= 1 Use the j-th stored learner.
...	Passed to the underlying fitted meta-learner print() method when cause is a single integer.

Value

Invisibly returns x.

Examples

```
d <- simulateStenoT1(30, competing_risks = TRUE)

learners <- list(
  lasso = Learner_glmnet(
    covariates = "sex",
    alpha = 1,
    lambda = 0.01,
    cross_validation = FALSE
  ),
  ridge = Learner_glmnet(
    covariates = c("sex", "value_LDL"),
    alpha = 0,
    lambda = 0.01,
    cross_validation = FALSE
  )
)

fit <- Superlearner(
  data = d,
  id = "id",
  status = "status_cvd",
  event_time = "time_cvd",
  learners = learners,
  number_of_nodes = 3,
  nfold = 2
)

print(fit, cause = NULL)
```

`simulateStenoT1` *Simulate time-to-event data for hypothetical type-1 diabetes patients*

Description

Simulate synthetic data inspired by the Steno Type-1 risk engine

Usage

```
simulateStenoT1(
  n,
  coefficient_age = 0.05,
  coefficient_LDL = 0.1,
  value_diabetis = 0.02,
  keep = NULL,
  scenario = c("alpha", "beta"),
  competing_risks = FALSE
)
```

Arguments

`n`	`numeric(1)`. Number of subjects to simulate.
`coefficient_age`	`numeric(1)`. Log-hazard coefficient for age in the CVD model (`time.event.1`).
`coefficient_LDL`	`numeric(1)`. Log-hazard coefficient for LDL in the CVD model (`time.event.1`).
`value_diabetis`	`numeric(1)`. Log-hazard coefficient for diabetes duration in the CVD model (`time.event.1`).
`keep`	`character` or `NULL`. Optional subset of columns to retain. If supplied, only those columns are returned.
`scenario`	`character(1)`. One of `"alpha"` or `"beta"`. Scenario `"beta"` modifies the CVD hazard by adding nonlinear hinge-squared terms in age and LDL.
`competing_risks`	`logical(1)`. If `TRUE` and `scenario = "alpha"`, simulates two event causes (CVD and death without CVD). Otherwise simulates CVD vs censoring.

Details

Generates baseline covariates and event times for CVD and censoring, with an optional competing-risks setting, for examples, benchmarks and tests.

The simulator uses a structural equation model (via `lava::lvm`) to generate realistic correlations between covariates. Event times are then generated from cause-specific Weibull proportional hazards models, where the linear predictor depends on the simulated covariates (and scenario).

The following baseline covariates are generated (column name, type, interpretation):

sex `factor`. Binary sex indicator (generated Bernoulli, then stored as factor).

age `numeric`. Age at baseline (years).

diabetes_duration `numeric`. Duration of diabetes at baseline (years).

value_SBP `numeric`. Systolic blood pressure (SBP).

value_LDL `numeric`. LDL cholesterol.

value_HBA1C `numeric`. HbA1c.

value_Albuminuria `factor` with levels `Normal`, `Micro`, `Macro`. Albuminuria category.

eGFR `numeric`. Estimated glomerular filtration rate, constructed from latent age-dependent log2 eGFR components (higher values indicate better kidney function).

value_Smoking `factor`. Smoking indicator (generated from a logistic model, then stored as factor).

value_Motion `factor`. Physical activity indicator (generated from a logistic model, then stored as factor).

Event time variables are generated from latent Weibull PH models: `time.event.1` (CVD), `time.event.0` (censoring), and in scenario `"alpha"` also `time.event.2` (death without prior CVD). These latent variables are used to construct the observed outcome variables returned by the function (see below).

Value

A `data.table` with at least the following columns:

id `integer`. Subject identifier (1, ..., n).

time_cvd `numeric`. Observed follow-up time (minimum of event and censoring times; also includes competing risk time if `competing_risks = TRUE` in scenario `"alpha"`).

status_cvd `integer`. Observed event status: `0` = censored, `1` = CVD, and if `competing_risks = TRUE` in scenario `"alpha"`, `2` = death without prior CVD.

time `numeric`. Alias of `time_cvd` (kept for convenience).

event `integer`. Alias of `status_cvd` (kept for convenience).

uncensored_time_cvd `numeric`. Event time ignoring censoring (minimum of event causes only).

uncensored_status_cvd `integer`. Event cause ignoring censoring. In scenario `"alpha"` this is `1` (CVD) or `2` (death without CVD); in scenario `"beta"` this is always `1`.

uncensored_time `numeric`. Alias of `uncensored_time_cvd`.

uncensored_event `integer`. Alias of `uncensored_status_cvd`.

In addition, the returned table contains all baseline covariates listed in **Details**. Internal latent variables used only for simulation are removed before returning (e.g., log2 eGFR components and, in scenario `"beta"`, the hinge-squared features).

Author(s)

Thomas A. Gerds tag@biostat.ku.dk

Examples

```
simulateStenoT1(n = 20, scenario = "alpha", competing_risks = TRUE)
```

`summary.base_learner` *Summarize a fitted base learner object*

Description

Dispatches to the underlying fitted model's `summary()` method for the selected cause, or returns a list of summaries for all causes.

Usage

```
## S3 method for class 'base_learner'
summary(object, cause = 1, ...)
```

Arguments

`object`	`base_learner` returned by `fit_learner()`.
`cause`	`numeric(1)` or `NULL`. Which cause to summarize. If `NULL`, returns one summary per cause.
`...`	Passed to the underlying `summary()` method (learner-dependent).

Value

If `cause` is a single integer, returns the underlying model summary for that cause. If `cause = NULL`, returns a list of summaries (one per cause).

Examples

```
d <- simulateStenoT1(30, competing_risks = TRUE)
lrn <- Learner_glmnet(
  covariates = c("sex", "value_LDL"),
  lambda = 0.01,
  cross_validation = FALSE
)
bl <- fit_learner(
  d,
  learner = lrn,
  id = "id",
  status = "status_cvd",
  event_time = "time_cvd",
  number_of_nodes = 3
)
out <- summary(bl, cause = 1)
```

summary.poisson_superlearner *Summarize a fitted Poisson Super Learner object*

Description

Prints:

1. a compact description of the fitted ensemble,
2. cross-validated deviances for base learners (when available),
3. cause-specific meta-learner coefficients (stacking weights).

Usage

```
## S3 method for class 'poisson_superlearner'
summary(object, cause = NULL, model = "sl", ...)
```

Arguments

object	poisson_superlearner returned by Superlearner().
cause	numeric(1) or NULL. Which cause's meta-learner fit to print. If NULL, prints one line per cause (classes only) instead of printing the full fitted objects.
model	Scalar model selector. Default is "sl" for the stacked super learner. Other allowed values are: 0 **or** "sl" Use the super learner prediction. **learner label** Use one stored base learner by its label in object$data_info$learners_labels. "learner_j" Use the j-th stored learner. **integer** j >= 1 Use the j-th stored learner.
...	Passed to the underlying coef() method for the fitted meta-learner (learner-dependent; e.g. s for glmnet).

Value

Invisibly returns a list with elements:

cross_validation_deviance data.table (or NULL).

meta_coefficients List of length n_crisks with cause-specific coefficient objects (or NULL).

Examples

```
d <- simulateStenoT1(30, competing_risks = TRUE)

learners <- list(
  lasso = Learner_glmnet(
    covariates = "sex",
    alpha = 1,
```

```
    lambda = 0.01,
    cross_validation = FALSE
  ),
  ridge = Learner_glmnet(
    covariates = c("sex", "value_LDL"),
    alpha = 0,
    lambda = 0.01,
    cross_validation = FALSE
  )
)

fit <- Superlearner(
  data = d,
  id = "id",
  status = "status_cvd",
  event_time = "time_cvd",
  learners = learners,
  number_of_nodes = 3,
  nfold = 2
)

s <- summary(fit, cause = 1)
names(s)
```

`Superlearner` *Fit a Poisson Super Learner ensemble*

Description

Fits an ensemble of cause-specific piecewise-constant hazard models using a long-format Poisson representation and combines them through a meta-learner (stacking).

Usage

```
Superlearner(
  data,
  id = "id",
  status = "status",
  event_time = NULL,
  learners,
  number_of_nodes = NULL,
  nodes = NULL,
  variable_transformation = NULL,
  nfold = 3,
  ...
)
```

Arguments

data	data.frame. Subject-level input data, one row per subject.
id	character(1). Name of the subject identifier column. If missing, an id column is created automatically.
status	character(1). Name of the event-status column. It must be coded with 0 for censoring and 1, 2, ..., K for event types. If there is no 0 in status, the data are treated as uncensored.
event_time	character(1). Name of the event or censoring time column.
learners	list. List of initialized learner reference-class objects, for example Learner_glmnet(), Learner_hal(), or Learner_gam(). If unnamed, learners are named "learner_1", "learner_2", and so on. Each learner must implement $private_fit(dt_long) and $private_predictor(model, newdata).
number_of_nodes	numeric(1) or NULL. If not NULL, constructs a quantile-based node grid with number_of_nodes + 1 cut points. Ignored when nodes is supplied.
nodes	numeric or NULL. Explicit time-node grid. If supplied, number_of_nodes is ignored. 0 is added if missing, and nodes larger than max(event_time) are dropped.
variable_transformation	Optional transformation specification passed to apply_transformations() on the internally created long-format data.
nfold	numeric(1). Number of folds for cross-validation stacking.
...	Additional arguments currently ignored.

Details

Internally, the function:

1. builds a time grid (nodes) and converts the subject-level data to a long Poisson format;
2. fits each base learner once on the full long data for each cause;
3. removes learners that already fail on the full data;
4. uses nfold cross-validation to obtain out-of-sample base-learner predictions (Z1, Z2, ...) for stacking;
5. removes learners whose cross-validated prediction column is entirely missing for at least one cause;
6. fits a cause-specific meta-learner on the retained stacked predictions.

If all learners fail on the full data, the function stops with an error. If only one learner remains after the full-data screening step or after the cross-validation screening step, no meta-learner is fit. In that case, metalearner is NULL, each superlearner[[k]]$meta_learner_fit is NULL, and prediction is based directly on the stored fitted base learner.Numeric learner positions always refer to the learners actually retained in the fitted object.

Value

An object of class `poisson_superlearner`, stored as a named `list` with the following components:

`learners`: the retained base learner objects.

`metalearner`: the meta-learner object used for stacking. If no stacking is performed because only one learner remains, `metalearner` is `NULL`.

`superlearner`: a `list` of length `data_info$n_crisks`, one entry per cause. For cause `k`, `superlearner[[k]]` is a `list` with two elements:

- `learners_fit`: the fitted base learner object or objects for cause `k`. If more than one learner is retained, this is a `list` with one fitted object per retained learner. If only one learner remains, this is the single fitted learner object itself.
- `meta_learner_fit`: the fitted cause-specific meta-learner for cause `k`. If no stacking is performed, this is `NULL`.

`cross_validation_deviance`: a `data.table` with columns `learner` and `deviance`, giving the mean cross-validated Poisson deviance for each retained base learner. This component is present when cross-validated model comparison is available.

`data_info`: a `list` of bookkeeping information used for prediction and interpretation, containing:

- `id`: identifier column name used.
- `status`: status column name used.
- `event_time`: event-time column name used.
- `nodes`: numeric vector of node cut points used for the piecewise grid.
- `nfold`: number of folds used for stacking.
- `maximum_followup`: maximum observed follow-up time.
- `n_crisks`: number of event types detected.
- `learners_labels`: character vector of retained learner labels.
- `variable_transformation`: the transformation specification passed in `variable_transformation`, or `NULL`.

Examples

```
data <- simulateStenoT1(50, competing_risks = TRUE)

learners <- list(
  glm = Learner_glmnet(
    covariates = c("sex", "value_LDL"),
    lambda = 0,
    cross_validation = FALSE
  ),
  ridge = Learner_glmnet(
    covariates = c("sex", "value_LDL"),
    alpha = 0,
    lambda = 0.01,
    cross_validation = FALSE
```

```
  )
)

fit <- Superlearner(
  data = data,
  id = "id",
  status = "status_cvd",
  event_time = "time_cvd",
  learners = learners,
  number_of_nodes = 3,
  nfold = 2
)
```

Index