Package 'orthoMTL'

August 23, 2026

Type Package

Title Multi-Task Learning with Orthogonal Constraints

Version 0.1.0

Description Fits regularised multi-task learning models where relationships between tasks are controlled via orthogonality or disjoint-support constraints. Supports regression, binary classification, and censored survival data. In survival mode, time-to-event outcomes are converted into binary labels at user-defined thresholds, enabling the discovery of features with time-varying effects that standard proportional-hazards models cannot detect. Implements the penalty described in Vervier et al. (2014) <https://hal.science/hal-00985654>.

License GPL-3

Encoding UTF-8

Imports parallel, doParallel, foreach, ggplot2, rlang, stats

VignetteBuilder knitr

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

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

NeedsCompilation no

Author Kevin Vervier [aut, cre], Novartis Pharma AG [cph, fnd]

Maintainer Kevin Vervier <kevin.vervier@novartis.com>

Repository CRAN

Date/Publication 2026-08-23 10:10:02 UTC

Contents

`accuracy_mtl` *Classification Accuracy for Multi-Task Predictions*

Description

Pooled fraction of correctly classified non-NA cells, for `orthoMTL` fits in classification mode (`logistic = TRUE`). A cell is positive when its true label is `> 0` and predicted positive when its score is `> 0`. This works for both the $\{-1, +1\}$ encoding the solver optimises and a $\{0, 1\}$ encoding, and for raw `"link"` scores or `"response"` probabilities (threshold 0.5 corresponds to a link of 0... see note).

Usage

```
accuracy_mtl(true.label.mat, pred.label.mat, threshold = 0)
```

Arguments

`true.label.mat`	A numeric matrix of true responses (`n x numTasks`). NA cells are ignored.
`pred.label.mat`	A numeric matrix of predictions of the same dimensions (as produced by `predict.orthoMTL`).
`threshold`	Decision threshold applied to `pred.label.mat`. Default `0` matches raw link scores; pass `0.5` for probabilities from `predict(..., type = "response")`.

Value

A single numeric value in `[0, 1]` (higher is better).

See Also

auc_mtl, cindex_mtl

Examples

```
set.seed(1)
Y <- matrix(sample(c(-1, 1), 30, replace = TRUE), 10, 3)
P <- Y * abs(matrix(rnorm(30), 10, 3))   # mostly correct signs
accuracy_mtl(Y, P)
```

---

auc_mtl *Area Under the ROC Curve for Multi-Task Predictions*

Description

Pooled AUC over every non-NA cell, computed via the Mann-Whitney U statistic (with 0.5 credit for ties), for orthoMTL fits in classification mode. The positive class is defined by true label > 0. AUC is invariant to monotone transforms, so raw "link" scores and "response" probabilities give identical results.

Usage

```
auc_mtl(true.label.mat, pred.label.mat)
```

Arguments

true.label.mat	A numeric matrix of true responses (n x numTasks). NA cells are ignored.
pred.label.mat	A numeric matrix of predictions of the same dimensions (as produced by predict.orthoMTL).

Value

A single numeric value in [0, 1] (higher is better; 0.5 = random).

See Also

accuracy_mtl, cindex_mtl

Examples

```
set.seed(1)
Y <- matrix(sample(c(-1, 1), 30, replace = TRUE), 10, 3)
P <- Y + matrix(rnorm(30), 10, 3)
auc_mtl(Y, P)
```

`bootstrap_orthoMTL` *Bootstrap Inference for orthoMTL Coefficients*

Description

Estimates coefficient variability and statistical relevance by comparing bootstrapped models (real signal) against null models (permuted outcomes). This two-pronged approach answers: (1) how stable is each coefficient across resamples, and (2) is each coefficient distinguishable from what would be obtained by chance.

Usage

```
bootstrap_orthoMTL(
  X,
  Y,
  lambda = 1,
  alpha = 0,
  step_size = 0.1,
  K = NULL,
  disjoint = FALSE,
  schedule = c("sqrt", "log", "const", "linear"),
  survival = FALSE,
  censored.mat = NULL,
  n_repeats = 100,
  n_cores = 2,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`X`	A numeric matrix of predictor variables with dimensions `n` x `p`.
`Y`	A numeric matrix of response labels with dimensions `n` x `numTasks`. May contain `NA` for censored observations.
`lambda`	Regularisation parameter for the orthogonal penalty.
`alpha`	Elastic-net mixing parameter in $[0, 1]$, passed through to `orthoMTL`. Default: `0`.
`step_size`	Step size for gradient descent. Default: `0.1`.
`K`	Constraint matrix of dimensions `numTasks` x `numTasks`. Default: `NULL` (identity).
`disjoint`	Logical. Enforce disjoint supports? Default: `FALSE`.
`schedule`	Character; the gradient-step decay schedule passed to `orthoMTL` for every resample/permutation fit. One of `"sqrt"` (default), `"log"`, `"const"`, `"linear"`. Because bootstrap refits the model many times, switching to `"log"` or `"const"` – which reach the same optimum in far fewer iterations – can substantially cut total runtime; see the `schedule` argument of `orthoMTL` for details.

survival	Logical. Use censored survival loss? Default: FALSE.
censored.mat	A numeric indicator matrix of dimensions n x numTasks. Required when survival = TRUE.
n_repeats	Number of bootstrap/permutation repeats. Default: 100.
n_cores	Number of cores for parallel execution. Default: 2.
seed	Optional base random seed. Default: NULL (no seed is set; repeats vary via the ambient RNG state and the run is not reproducible). If supplied, repeat i uses set.seed(seed + i) for i in 1:n_repeats, so the whole run becomes reproducible while still varying across repeats.
verbose	Logical. Print progress information? Default: TRUE.

Details

Real bootstrap: For each repeat, rows of X, Y, and censored.mat are resampled *with replacement*. The model is refit with identical hyperparameters. This produces a distribution of coefficient values reflecting estimation variability.

Null permutation: For each repeat, rows of Y and censored.mat are permuted *without replacement* while X remains fixed. This breaks the association between features and outcomes, producing a null distribution of coefficients.

Comparing real vs null distributions for each feature and task indicates whether observed coefficients are distinguishable from noise. Visualise with plot_bootstrap.

Value

An object of class "bootstrap_orthoMTL" containing:

results A tidy data.frame with columns id (feature name), time (task/threshold), coeff (coefficient value), group ("real" or "null"), and repeat_id (integer).

coefficients_real A list of n_repeats coefficient matrices (each p x numTasks).

coefficients_null A list of n_repeats coefficient matrices from permuted outcomes.

obj_real Numeric vector of final objective values for real bootstrap models.

obj_null Numeric vector of final objective values for null permutation models.

n_repeats Number of repeats.

n_features Number of features.

n_tasks Number of tasks.

feature_names Character vector of feature names.

task_names Character vector of task names.

call The matched function call.

See Also

orthoMTL, plot_bootstrap

Examples

```
set.seed(42)
n <- 30; p <- 5; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("V", seq_len(p))
SurvTime <- rexp(n, rate = 0.1)
Event <- rbinom(n, 1, 0.7)
thresholds <- c(4, 6, 10)

Y <- create_longitudinal_labels(SurvTime, Event, thresholds)
W <- create_indicator_matrix(Y)
K <- create_constraint_matrix(n_tasks)

boot_res <- bootstrap_orthoMTL(
  X = X, Y = Y, lambda = 1e-3, step_size = 0.1,
  K = K, survival = TRUE, censored.mat = W,
  n_repeats = 5, n_cores = 1, verbose = FALSE
)

print(boot_res)
head(boot_res$results)
```

---

cindex_mtl *Concordance Index for Multi-Task Predictions*

Description

Computes a concordance index (C-index) adapted for the multi-task survival framework. Predictions across tasks are aggregated by row-sum to produce a single score per patient, then concordance is evaluated over pairs where at least one member has a fully observed outcome.

Usage

```
cindex_mtl(true.label.mat, pred.label.mat)
```

Arguments

`true.label.mat`	A numeric matrix of true response labels with dimensions `n` x `numTasks`. May contain `NA` for censored observations (as produced by `create_longitudinal_labels`).
`pred.label.mat`	A numeric matrix of predicted response values with dimensions `n` x `numTasks` (as produced by `predict.orthoMTL`).

Details

The effective survival time for each patient is derived as the highest task index where the true label is positive (i.e., the last threshold at which the patient was progression-free).

A patient is considered "uncensored" only if all task labels are non-`NA`. Concordant pairs require both a correct ordering of effective survival times and a matching ordering of predicted scores.

Value

A numeric value between 0 and 1 (higher is better). A value of 0.5 indicates random concordance.

Known limitations

The following limitations are documented and flagged for future investigation:

- No credit for tied predictions or tied survival times
- Strict censoring: uncensored requires all tasks observed
- Equal weighting of all tasks in the row-sum aggregation

See Also

`predict.orthoMTL` for generating the prediction matrix.

Examples

```
# Simulate a small multi-task prediction scenario
set.seed(42)
n <- 30; n_tasks <- 4
SurvTime <- rexp(n, rate = 0.1)
Event <- rbinom(n, 1, 0.7)
thresholds <- c(4, 6, 10, 15)

Y <- create_longitudinal_labels(SurvTime, Event, thresholds)

# Simulate imperfect predictions (add noise to true labels)
pred <- Y
pred[is.na(pred)] <- 0.5
pred <- pred + matrix(rnorm(n * n_tasks, sd = 0.3), n, n_tasks)

cindex_mtl(Y, pred)
```

`coef.orthoMTL` *Extract Coefficients from an orthoMTL Model*

Description

Returns the coefficient matrix from a fitted `orthoMTL` model.

Usage

```
## S3 method for class 'orthoMTL'
coef(object, ...)
```

Arguments

`object`	A fitted model object of class `"orthoMTL"`.
`...`	Additional arguments (currently ignored).

Value

A numeric matrix of regression coefficients with dimensions `n_features` x `n_tasks`. Row names correspond to feature names and column names to task names, if available.

Examples

```
set.seed(42)
n <- 100; p <- 10; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("V", seq_len(p))
Y <- X %*% matrix(rnorm(p * n_tasks), p) + matrix(rnorm(n * n_tasks), n) * 0.1
K <- matrix(1, n_tasks, n_tasks); diag(K) <- 0.5
fit <- orthoMTL(X, Y, lambda = 1e-3, K = K)
coef(fit)
```

---

create_constraint_matrix

Create Diffusion Constraint Matrix

Description

Builds a square constraint matrix `K` encoding the prior belief that temporally distant tasks should have more orthogonal coefficients. Off-diagonal entries accumulate weight proportional to the distance between task indices, implementing a diffusion-like pattern.

Usage

```
create_constraint_matrix(numTasks, diag_val = 0.5)
```

Arguments

`numTasks`	An integer specifying the number of tasks.
`diag_val`	A numeric value for the diagonal of `K`. Default: `0.5`. This value is typically overridden during cross-validation (see `cv_orthoMTL`).

Details

This matrix is passed to `orthoMTL` via the `K` argument to control the orthogonality penalty between tasks.

The construction rule: for each distance level `h` from 1 to `numTasks`, add 1 to all entries where `|row - col| > h`. This produces a matrix where nearby tasks (adjacent thresholds) share more support, while distant tasks are pushed toward orthogonality.

For survival analysis with time thresholds, this encodes the assumption that the set of predictive features changes gradually over time rather than abruptly.

Value

A numeric square matrix of dimensions `numTasks` x `numTasks`. Off-diagonal entry `K[i,j]` is larger when tasks `i` and `j` are further apart. Diagonal entries are set to `diag_val`.

See Also

`cv_orthoMTL`

Examples

```
# 5-task constraint matrix
K <- create_constraint_matrix(5)
K

# Override diagonal for a specific penalty balance
K <- create_constraint_matrix(7, diag_val = 6)
K
```

`create_indicator_matrix`

Create Censoring Indicator Matrix

Description

Converts `NA` entries in a longitudinal label matrix (as produced by `create_longitudinal_labels`) into a binary indicator matrix. This indicator is used by `orthoMTL` to mask censored observations in the loss computation.

Usage

```
create_indicator_matrix(Y)
```

Arguments

`Y`	A numeric matrix of dimensions `n` x `numTasks`, typically produced by `create_longitudinal_labels`. May contain `NA` values for censored observations.

Value

A numeric matrix of the same dimensions as `Y`. `1` = label is observed (known), `0` = label is censored (unknown).

See Also

`create_longitudinal_labels`

Examples

```
set.seed(42)
SurvTime <- rexp(10, rate = 0.1)
Event <- rbinom(10, 1, 0.7)
Y <- create_longitudinal_labels(SurvTime, Event, c(4, 6, 10))
W <- create_indicator_matrix(Y)
W   # 1 = observed, 0 = censored
```

`create_longitudinal_labels` *Convert Survival Data to Longitudinal Binary Labels*

Description

Transforms time-to-event survival data into a matrix of binary labels at user-defined time thresholds. This is the core data transformation that enables survival analysis within the multi-task learning framework.

Usage

```
create_longitudinal_labels(SurvTime, Event, thresholds = c(4, 6))
```

Arguments

`SurvTime`	A numeric vector of length `n` containing observed survival times. Must be non-negative.
`Event`	A numeric vector of length `n` encoding censoring status. `1` = event observed, `0` = censored.
`thresholds`	A numeric vector of length `numTasks` representing the time thresholds at which to evaluate survival status. Default: `c(4, 6)`.

Details

The encoding logic for each patient at each threshold is:

- `1` — patient is progression-free at this threshold (survival time exceeds threshold, regardless of event status)
- `0` — patient experienced an event before this threshold (survival time < threshold AND event observed)
- `NA` — patient was censored before this threshold (survival time < threshold AND no event observed); label is unknown

Value

A numeric matrix of dimensions `n` x `numTasks`. Column names are set to the threshold values. Contains `1`, `0`, and `NA` values as described above.

See Also

create_indicator_matrix to convert NA values into a binary censoring indicator matrix.

Examples

```
# Simulate 10 patients
set.seed(42)
SurvTime <- rexp(10, rate = 0.1)
Event <- rbinom(10, 1, 0.7)
thresholds <- c(4, 6, 10, 15)

Y <- create_longitudinal_labels(SurvTime, Event, thresholds)
Y   # 1 = progression-free, 0 = event, NA = censored
```

---

cv_orthoMTL *Cross-Validation for orthoMTL Hyperparameter Selection*

Description

Performs a parallelised grid search over hyperparameters for orthoMTL, evaluating each configuration via cross-validated concordance index. Returns the best configuration without retraining a final model (that is the caller's responsibility).

Usage

```
cv_orthoMTL(
  X.train,
  Y.train,
  W.train = NULL,
  K = NULL,
  lambdas = c(0.001, 0.01),
  alphas = 0,
  stepsizes = c(0.1, 0.5),
  diag_vals = c(0.5, 1),
  survival = TRUE,
  logistic = FALSE,
  metric = NULL,
  disjoint = FALSE,
  schedule = c("sqrt", "log", "const", "linear"),
  folds = NULL,
  n_cores = 2,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

X.train	A numeric matrix of training features with dimensions n x p.
Y.train	A numeric matrix of training labels with dimensions n x numTasks. May contain NA for censored observations.
W.train	A numeric indicator matrix of dimensions n x numTasks, where 1 = observed and 0 = censored. Required when survival = TRUE.
K	A square constraint matrix of dimensions numTasks x numTasks. The diagonal will be overridden by values in diag_vals during the search.
lambdas	A numeric vector of regularisation parameters to search.
alphas	A numeric vector of elastic-net mixing parameters in $[0, 1]$ to search. Default: 0 (pure orthogonality penalty, no sparsity).
stepsizes	A numeric vector of gradient descent step sizes to search.
diag_vals	A numeric vector of diagonal values for the constraint matrix K to search.
survival	Logical. Use censored survival loss? Default: TRUE.
logistic	Logical. Fit logistic (classification) models? Passed through to orthoMTL. Default: FALSE.
metric	Character; the scoring metric maximised/minimised over the grid, or NULL (default) to choose automatically by mode: "cindex" when survival = TRUE, "auc" when logistic = TRUE, otherwise "rmse". Supported values: "cindex", "auc", "accuracy" (higher is better), "rmse" (lower is better), "r2" (higher is better).
disjoint	Logical. Enforce disjoint supports? Default: FALSE.
schedule	Character; the gradient-step decay schedule passed to orthoMTL for every fit. One of "sqrt" (default), "log", "const", "linear". "log" or "const" typically reach the same optimum in far fewer iterations than the default "sqrt", which can noticeably speed up the grid search; see the schedule argument of orthoMTL for the trade-offs. Applied uniformly to all configurations (it is not part of the tuning grid).
folds	An integer vector of length n assigning each training observation to a fold. If NULL, a 5-fold assignment is generated with a warning.
n_cores	Integer. Number of cores for parallel execution. Default: 2.
seed	Optional integer random seed for reproducibility. Default: NULL (no seed is set; the ambient RNG state is used as-is). Used both for auto-generated fold assignment and for each orthoMTL fit in the grid.
verbose	Logical. Print progress information? Default: TRUE.

Details

The grid is constructed as the full Cartesian product of lambdas, alphas, stepsizes, and diag_vals. Each configuration is evaluated independently in parallel across cores. Within each configuration, folds are evaluated sequentially and the per-fold C-indices are averaged.

The best configuration is selected by joint maximisation of the mean CV C-index over the entire flattened grid (not greedy sequential search).

This function does *not* retrain a final model. Use the returned hyperparameters to train via orthoMTL.

Value

An object of class "cv_orthoMTL" containing:

best A list with the best hyperparameters: lambda, alpha, stepsize, diag_val, and the corresponding cv_score.

results A data.frame of all configurations with their mean CV C-index, sorted descending by cv_score.

folds The fold assignment vector used.

n_configs Total number of configurations tested.

n_folds Number of unique folds.

call The matched function call.

See Also

orthoMTL, cindex_mtl

Examples

```
set.seed(42)
n <- 50; p <- 5; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("V", seq_len(p))
SurvTime <- rexp(n, rate = 0.1)
Event <- rbinom(n, 1, 0.7)
thresholds <- c(4, 6, 10)

Y <- create_longitudinal_labels(SurvTime, Event, thresholds)
W <- create_indicator_matrix(Y)
K <- create_constraint_matrix(n_tasks)

folds <- rep(1:2, length.out = n)

cv_res <- cv_orthoMTL(
  X.train = X, Y.train = Y, W.train = W, K = K,
  lambdas = c(1e-3, 1e-2), alphas = 0,
  stepsizes = c(0.1), diag_vals = c(0.5, 1),
  survival = TRUE, disjoint = FALSE,
  folds = folds, n_cores = 1, seed = 42, verbose = FALSE
)

print(cv_res)
cv_res$best
```

nnmaxheap_C *Project a Vector onto Non-Negative Non-Increasing Space*

Description

Applies an isotonic regression-style projection to enforce that the output vector is non-negative and non-increasing. Used internally by `predict.orthoMTL` to ensure survival predictions are monotonically decreasing across time thresholds.

Usage

```
nnmaxheap_C(m)
```

Arguments

`m`	A numeric vector to project.

Details

This function implements a pool-adjacent-violators style algorithm. It replaces the external `Iso` package dependency used in the predecessor `orthopen` package.

Value

A numeric vector of the same length as `m`, projected onto the non-negative non-increasing constraint space.

Examples

```
# Project a vector onto the non-negative non-increasing space
nnmaxheap_C(c(3, 1, 2, -1))

# Already valid input: returned unchanged
nnmaxheap_C(c(5, 3, 3, 1))
```

orthoMTL *Multi-task learning with orthogonal constraints*

Description

This function solves a multi-task problem where relationships between tasks can be complex

Usage

```
orthoMTL(
  X,
  Y,
  lambda = 1,
  step_size = 0.1,
  tol = 1e-05,
  stop_no_improve = 100,
  max_iter = 1e+06,
  W_0 = NULL,
  seed = NULL,
  K = NULL,
  disjoint = FALSE,
  logistic = FALSE,
  alpha = 0,
  schedule = c("sqrt", "log", "const", "linear"),
  survival = FALSE,
  censored.mat = NULL,
  verbose = 0
)
```

Arguments

X	a matrix of predictor variables with dimensions n x p
Y	a matrix of response variables with dimensions n x numTasks, where numTasks is the number of response variables. NAs can be used for censored data.
lambda	the regularization parameter for the OrthoPen penalty, default is 1
step_size	the step size for updating the regression coefficients in gradient descent, default is 0.1
tol	Convergence tolerance. The algorithm stops when the improvement in the objective function is less than `tol`. Default: `1e-5`.
stop_no_improve	the number of iterations without improvement in the objective function to trigger convergence, default is 100
max_iter	the maximum number of iterations, default is 1e+06
W_0	a matrix of initial values for the regression coefficients, default is NULL, as not provided and will be randomly attributed
seed	an optional random seed for reproducibility, default is `NULL` (no seed is set; the ambient RNG state is used as-is). Only relevant when `W_0` is not provided and `disjoint = TRUE`, since that is the only case where `orthoMTL` draws random numbers. Pass an integer to make that random initialisation reproducible.
K	a constraint matrix of weights to adjust the OrthoPen penalty, default is an identity matrix with dimensions numTasks x numTasks
disjoint	a logical value indicating whether the response variables should have disjoint supports, default is FALSE

`logistic` a logical value indicating whether logistic regression should be used instead of linear regression, default is FALSE

`alpha` the elastic-net mixing parameter in $[0, 1]$, default is 0. The penalty is $\lambda[(1-\alpha)/2\,\Omega_K(W)^2 + \alpha\|W\|_1]$; `alpha = 0` gives the pure orthogonality penalty and `alpha = 1` gives a pure Lasso.

`schedule` Character; the gradient-step decay schedule – how the per-iteration scale grows with the iteration index i. The coefficient update is $W \leftarrow W - \texttt{step\_size}\,\nabla/(\texttt{scale}\cdot\|\nabla\|)$, so a faster-growing `scale` means a faster-decaying effective step. One of:

`"sqrt"` (default) `scale` $= \sqrt{i}$, effective step $\propto 1/\sqrt{i}$ – the classic subgradient schedule and the historical hardcoded behaviour.

`"log"` `scale` $= \log(i) + 1$ (slow decay).

`"const"` `scale` $= 1$ (no decay).

`"linear"` `scale` $= i$, effective step $\propto 1/i$ (Robbins-Monro).

The default `"sqrt"` is retained for backward compatibility: it reproduces previously published results exactly. An A/B study shipped with the package (`validation/ab_s02_gradient_` found that `"sqrt"` reaches the *correct* optimum but converges to it roughly 12–17x slower than `"log"` and `"const"` (which reach the same objective in a small fraction of the iterations), while `"linear"` decays too aggressively and can stall short of the optimum. Prefer `"log"` or `"const"` when convergence speed matters; note that changing the schedule changes the optimisation path and therefore the exact coefficients, so it should not be altered when reproducing published runs.

`survival` a logical value indicating whether survival analysis should be performed, default is FALSE

`censored.mat` a matrix indicating whether observations are censored, used only if survival=TRUE

`verbose` the level of verbosity, default is 0 (no messages)

Value

a list containing the following elements:

`B` a matrix of regression coefficients with dimensions p x numTasks

`obj` the final objective function when algorithms stops

`imax` the number of iterations

References

Kevin Vervier, Pierre Mahé, Alexandre d'Aspremont, Jean-Baptiste Veyrieras, Jean-Philippe Vert (2014). On learning matrices with orthogonal columns or disjoint supports. https://hal.science/hal-00985654/file/learningDisjointSupports.pdf

Examples

```
# Regression with orthogonal columns
set.seed(42)
n <- 100; p <- 10; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
```

```
W_true <- qr.Q(qr(matrix(rnorm(p * n_tasks), p, n_tasks)))
Y <- X %*% W_true + matrix(rnorm(n * n_tasks), n) * 0.1
K <- matrix(1, n_tasks, n_tasks)
diag(K) <- 0.5
fit <- orthoMTL(X, Y, lambda = 1e-3, K = K, disjoint = FALSE)
fit$B           # coefficient matrix
fit$converged   # did optimisation converge?
```

`plot_bootstrap` *Bootstrap Coefficient Comparison Plot*

Description

Displays faceted line plots comparing real (bootstrapped) vs null (permuted) coefficient trajectories across tasks for selected or all features. Each panel shows the mean and standard error of the coefficient at each task/threshold.

Usage

```
plot_bootstrap(x, features = NULL, batch_size = 9)
```

Arguments

`x`	Either a `"bootstrap_orthoMTL"` object (as returned by `bootstrap_orthoMTL`) or a `data.frame` with columns `id`, `time`, `coeff`, and `group`.
`features`	A character vector of feature names to plot. If `NULL` (default), all features are plotted.
`batch_size`	Number of features per facet page. Default: `9`.

Details

Blue lines show coefficients from models fitted on bootstrapped (resampled) data — reflecting estimation variability under real signal. Grey lines show coefficients from models fitted on permuted outcomes — reflecting the null distribution.

When the blue (real) band is clearly separated from the grey (null) band, the feature's coefficient is distinguishable from noise at that task/threshold.

Value

A list of `ggplot2` objects, one per batch/page.

See Also

`bootstrap_orthoMTL`

Examples

```
set.seed(42)
n <- 30; p <- 5; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("V", seq_len(p))
SurvTime <- rexp(n, rate = 0.1)
Event <- rbinom(n, 1, 0.7)
thresholds <- c(4, 6, 10)
Y <- create_longitudinal_labels(SurvTime, Event, thresholds)
W <- create_indicator_matrix(Y)
K <- create_constraint_matrix(n_tasks)
boot_res <- bootstrap_orthoMTL(
  X = X, Y = Y, lambda = 1e-3, step_size = 0.1,
  K = K, survival = TRUE, censored.mat = W,
  n_repeats = 5, n_cores = 1, verbose = FALSE
)
plots <- plot_bootstrap(boot_res)
plots[[1]]
```

---

plot_correlation *Task Correlation Map for orthoMTL*

Description

Displays pairwise distances between task coefficient vectors as a heatmap. Distance is computed as `1 - cosine_similarity`. Values near 0 indicate similar coefficient profiles; values near 1 indicate orthogonal profiles.

Usage

```
plot_correlation(x, midpoint = 0.5, limits = c(0, 1))
```

Arguments

`x`	Either a fitted `"orthoMTL"` object or a numeric coefficient matrix with dimensions `p` x `numTasks`.
`midpoint`	Midpoint for the colour scale. Default: `0.5`.
`limits`	Numeric vector of length 2 for the colour scale limits. Default: `c(0, 1)`.

Value

A `ggplot2` object.

See Also

`plot_heatmap`

Examples

```
set.seed(42)
n <- 100; p <- 10; n_tasks <- 4
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("V", seq_len(p))
Y <- X %*% matrix(rnorm(p * n_tasks), p) + matrix(rnorm(n * n_tasks), n) * 0.1
colnames(Y) <- c("T1", "T2", "T3", "T4")
K <- matrix(1, n_tasks, n_tasks); diag(K) <- 0.5
fit <- orthoMTL(X, Y, lambda = 1e-3, K = K)

plot_correlation(fit)
```

---

plot_heatmap *Coefficient Heatmap for orthoMTL*

Description

Displays the coefficient matrix as a heatmap. Features (rows) can optionally be reordered by their mean coefficient across tasks.

Usage

```
plot_heatmap(x, reorder = TRUE)
```

Arguments

x	Either a fitted "orthoMTL" object or a numeric coefficient matrix with dimensions p x numTasks. If an "orthoMTL" object, coefficients are extracted via coef().
reorder	Logical. If TRUE (default), rows are sorted by mean coefficient across tasks (ascending).

Value

A ggplot2 object.

See Also

coef.orthoMTL, plot_correlation

Examples

```
set.seed(42)
n <- 100; p <- 10; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("V", seq_len(p))
Y <- X %*% matrix(rnorm(p * n_tasks), p) + matrix(rnorm(n * n_tasks), n) * 0.1
colnames(Y) <- c("T1", "T2", "T3")
```

```
K <- matrix(1, n_tasks, n_tasks); diag(K) <- 0.5
fit <- orthoMTL(X, Y, lambda = 1e-3, K = K)

# From fitted object
plot_heatmap(fit)

# From raw matrix
plot_heatmap(coef(fit), reorder = FALSE)
```

`plot_prediction` *Prediction Swimmer Plot for orthoMTL*

Description

Displays a prediction matrix as a heatmap (swimmer plot), where rows are patients and columns are tasks/thresholds.

Usage

```
plot_prediction(x)
```

Arguments

`x`	A numeric prediction matrix with dimensions `n` x `numTasks`, typically produced by `predict.orthoMTL`.

Value

A `ggplot2` object.

See Also

`predict.orthoMTL`

Examples

```
set.seed(42)
n <- 50; p <- 10; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("V", seq_len(p))
Y <- X %*% matrix(rnorm(p * n_tasks), p) + matrix(rnorm(n * n_tasks), n) * 0.1
colnames(Y) <- c("T1", "T2", "T3")
K <- matrix(1, n_tasks, n_tasks); diag(K) <- 0.5
fit <- orthoMTL(X, Y, lambda = 1e-3, K = K)

preds <- predict(fit, newdata = X)
plot_prediction(preds)
```

predict.orthoMTL *Predict from an orthoMTL Model*

Description

Generate predictions from a fitted orthoMTL model for new observations. In survival mode, predictions are projected onto the non-negative non-increasing space to ensure monotonicity across tasks.

Usage

```
## S3 method for class 'orthoMTL'
predict(object, newdata, type = c("link", "response", "class"), ...)
```

Arguments

object	A fitted model object of class "orthoMTL".
newdata	A numeric matrix of new observations with dimensions n_new x p. Column names are used for feature alignment if available in the fitted object.
type	Character; scale of the returned predictions. One of: "link" (default) the raw linear predictor XB (after the survival monotonicity projection, if applicable). Preserves the historical behaviour. "response" for logistic fits, the sigmoid $1/(1+e^{-XB})$ giving $P(Y=+1)$; for non-logistic fits identical to "link". "class" for logistic fits, the predicted class label in $\{-1,+1\}$ (sign(XB), with 0 mapped to +1). Errors for non-logistic fits.
...	Additional arguments (currently ignored).

Details

If the fitted object contains feature_names (i.e., the training matrix X had column names), newdata columns are aligned to match. Missing features cause an error. Extra features trigger a warning and are dropped.

In survival mode (object$hyperparameters$survival == TRUE), each row of the raw prediction matrix is projected via nnmaxheap_C() to enforce non-negative, non-increasing values across tasks (time thresholds).

Value

A numeric matrix of predictions with dimensions n_new x n_tasks. Column names correspond to task names if available.

References

Vervier, K., Mahe, P., d'Aspremont, A., Veyrieras, J.-B., and Vert, J.-P. (2014). On Learning Matrices with Orthogonal Columns or Disjoint Supports. *ECML-PKDD 2014*. https://hal.science/hal-00985654

Examples

```
set.seed(42)
n <- 100; p <- 10; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("V", seq_len(p))
W_true <- qr.Q(qr(matrix(rnorm(p * n_tasks), p, n_tasks)))
Y <- X %*% W_true + matrix(rnorm(n * n_tasks), n) * 0.1
K <- matrix(1, n_tasks, n_tasks); diag(K) <- 0.5
fit <- orthoMTL(X, Y, lambda = 1e-3, K = K, disjoint = FALSE)

X_new <- matrix(rnorm(20 * p), 20, p)
colnames(X_new) <- paste0("V", seq_len(p))
preds <- predict(fit, newdata = X_new)
dim(preds)
```

---

print.bootstrap_orthoMTL

Print Bootstrap Inference Results

Description

Displays a summary of bootstrap inference results from bootstrap_orthoMTL.

Usage

```
## S3 method for class 'bootstrap_orthoMTL'
print(x, ...)
```

Arguments

x	An object of class "bootstrap_orthoMTL".
...	Additional arguments (currently ignored).

Value

Invisibly returns x.

Examples

```
# See ?bootstrap_orthoMTL for a full example
```

print.cv_orthoMTL *Print Cross-Validation Results*

Description

Displays a summary of cross-validation results from cv_orthoMTL.

Usage

```
## S3 method for class 'cv_orthoMTL'
print(x, n_top = 5, ...)
```

Arguments

x	An object of class "cv_orthoMTL".
n_top	Number of top configurations to display. Default: 5.
...	Additional arguments (currently ignored).

Value

Invisibly returns x.

Examples

```
# See ?cv_orthoMTL for a full example
```

print.orthoMTL *Print an orthoMTL Object*

Description

Displays a compact summary of a fitted orthoMTL model.

Usage

```
## S3 method for class 'orthoMTL'
print(x, ...)
```

Arguments

x	A fitted model object of class "orthoMTL".
...	Additional arguments (currently ignored).

Value

Invisibly returns x.

Examples

```
set.seed(42)
n <- 100; p <- 10; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
Y <- X %*% matrix(rnorm(p * n_tasks), p) + matrix(rnorm(n * n_tasks), n) * 0.1
fit <- orthoMTL(X, Y, lambda = 1e-3)
print(fit)
```

---

print.simulated_mtl *Print Simulated Multi-Task Data*

Description

Displays a summary of a simulated dataset from simulate_mtl.

Usage

```
## S3 method for class 'simulated_mtl'
print(x, ...)
```

Arguments

x	An object of class "simulated_mtl".
...	Additional arguments (currently ignored).

Value

Invisibly returns x.

Examples

```
sim <- simulate_mtl(n = 100, p = 20, n_signals = 4)
print(sim)
```

---

r2_mtl *Coefficient of Determination (R-squared) for Multi-Task Regression*

Description

Pooled $R^2 = 1 - SS_{res}/SS_{tot}$ over every non-NA cell, for orthoMTL fits in regression mode. SS_{tot} uses the global mean of the pooled true values.

Usage

```
r2_mtl(true.label.mat, pred.label.mat)
```

Arguments

`true.label.mat`	A numeric matrix of true responses (`n` x `numTasks`). `NA` cells are ignored.
`pred.label.mat`	A numeric matrix of predictions of the same dimensions (as produced by `predict.orthoMTL`).

Value

A single numeric value (higher is better; 1 = perfect). Can be negative when predictions are worse than the mean.

See Also

`rmse_mtl`, `cindex_mtl`

Examples

```
set.seed(1)
Y <- matrix(rnorm(30), 10, 3)
P <- Y + matrix(rnorm(30, sd = 0.1), 10, 3)
r2_mtl(Y, P)
```

`rmse_mtl` *Root Mean Squared Error for Multi-Task Regression*

Description

Pooled RMSE over every non-`NA` cell of the true/predicted matrices, for `orthoMTL` fits in regression mode (`logistic` = `FALSE`, `survival` = `FALSE`).

Usage

```
rmse_mtl(true.label.mat, pred.label.mat)
```

Arguments

`true.label.mat`	A numeric matrix of true responses (`n` x `numTasks`). `NA` cells are ignored.
`pred.label.mat`	A numeric matrix of predictions of the same dimensions (as produced by `predict.orthoMTL`).

Value

A single non-negative numeric value (lower is better).

See Also

`r2_mtl`, `cindex_mtl`

Examples

```
set.seed(1)
Y <- matrix(rnorm(30), 10, 3)
P <- Y + matrix(rnorm(30, sd = 0.1), 10, 3)
rmse_mtl(Y, P)
```

`simulate_mtl` *Simulate Multi-Task Data with Time-Varying Effects*

Description

Generates a realistic simulated dataset with binary and continuous features and a known ground-truth coefficient structure. The `mode` argument selects the response type: piecewise-exponential **survival** times (default), multi-task **regression** targets, or multi-task binary **classification** labels. Designed for demonstrating and testing `orthoMTL`.

Usage

```
simulate_mtl(
  n = 200,
  p = 30,
  n_signals = 5,
  n_continuous = 1,
  thresholds = c(4, 6, 10, 15),
  mode = c("survival", "regression", "classification"),
  noise_sd = 1,
  censoring_max = 25,
  baseline_hazard = 0.05,
  effect_strength = 0.8,
  treatment_effect = -0.3,
  seed = NULL
)
```

Arguments

`n`	Number of patients. Default: `200`.
`p`	Number of features excluding the treatment column. Default: `30`.
`n_signals`	Number of features with true non-zero effects. Default: `5`. Must be <= `p`.
`n_continuous`	Number of continuous features (placed first in the feature matrix). Default: `1`. Remaining features are binary.
`thresholds`	Numeric vector of time thresholds defining the task structure. Default: `c(4, 6, 10, 15)`.
`mode`	Character; the response type to generate. One of `"survival"` (default), `"regression"`, or `"classification"`. The feature matrix and ground-truth coefficients are generated identically across modes; only the response differs:

survival piecewise-exponential `SurvTime`/`Event` (the historical behaviour).

regression `Y = X %*% beta + N(0, noise_sd^2)`, returned as the `n x length(thresholds)` matrix `Y`.

classification binary labels in $\{-1, +1\}$ sampled from $P(Y = +1) = 1/(1 + e^{-Xbeta})$, returned as `Y`. This matches the label encoding `orthoMTL` optimises with `logistic = TRUE`.

`noise_sd` Standard deviation of the Gaussian noise added in `mode = "regression"`. Ignored otherwise. Default: `1`.

`censoring_max` Maximum censoring time. Censoring times are drawn from `Unif(0, censoring_max)`. Default: `25`.

`baseline_hazard`
Baseline hazard rate per interval. Default: `0.05`.

`effect_strength`
Multiplier controlling the magnitude of feature effects. Default: `0.5`.

`treatment_effect`
Effect of treatment on the log-hazard (negative = protective). Default: `-0.3`.

`seed` Optional random seed for reproducibility. Default: `NULL` (no seed is set; the ambient RNG state is used as-is). Pass an integer to make the simulated data reproducible.

Details

Feature structure:

- Continuous features are drawn from `N(0, 1)`.
- Binary features have prevalences drawn from `Beta(2, 10)`, producing a realistic range (~5-30%).
- Treatment is balanced 1:1 via random assignment.

Effect templates: Each signal feature is assigned one of five temporal patterns:

- `"early"`: strong effect at early thresholds, fading to zero at late.
- `"late"`: zero at early thresholds, emerging at late.
- `"constant"`: equal effect across all thresholds (detectable by standard Cox models).
- `"increasing"`: effect grows over time.
- `"decreasing"`: effect shrinks over time.

Templates are assigned cyclically across signal features with random sign (risk-increasing or protective).

Survival time generation: Uses a piecewise-exponential model where the hazard in each interval is `h_k(i) = baseline_hazard * exp(X[i, ] %*% beta[, k])`. Patients progress through intervals sequentially; an event occurs when the simulated time within an interval is shorter than the interval width.

Censoring: Independent of event times. `C ~ Unif(0, censoring_max)`. Observed time = `min(T, C)`, event indicator = `T <= C`.

Value

An object of class `"simulated_mtl"` containing:

mode The response type generated.

Y For `mode = "regression" / "classification"`, the `n x length(thresholds)` response matrix. `NULL` in survival mode (use `SurvTime/Event` instead).

SurvTime, Event Survival mode only (`NULL` otherwise): observed times and event indicators.

X Numeric matrix of dimensions `n x (p + 1)`. Columns include `n_continuous` continuous features (standard normal), `p - n_continuous` binary features (prevalences drawn from `Beta(2, 10)`), and a `treatment` column (balanced 1:1).

SurvTime Numeric vector of observed survival times.

Event Binary vector: `1` = event observed, `0` = censored.

treatment Binary vector (also present as last column of `X`).

feature_names Character vector of column names of `X`.

ground_truth A list containing:

- **coefficients** Matrix of true coefficients with dimensions `(p + 1) x length(thresholds)`.
- **signal_features** Names of features with non-zero effects.
- **null_features** Names of features with zero effects.
- **effect_types** Named character vector mapping signal features to their temporal effect template.
- **thresholds** The thresholds used.
- **baseline_hazard** The baseline hazard used.

n, p, n_signals, n_continuous, thresholds, seed Input parameters stored for reference.

call The matched function call.

See Also

`create_longitudinal_labels`, `orthoMTL`

Examples

```
# Generate simulated data
sim <- simulate_mtl(n = 100, p = 20, n_signals = 4, seed = 42)
sim

# Inspect ground truth
sim$ground_truth$signal_features
sim$ground_truth$effect_types
sim$ground_truth$coefficients[sim$ground_truth$signal_features, ]

# Use in orthoMTL workflow

thresholds <- c(4, 6, 10, 15)
Y <- create_longitudinal_labels(sim$SurvTime, sim$Event, thresholds)
W <- create_indicator_matrix(Y)
K <- create_constraint_matrix(length(thresholds))
```

```
fit <- orthoMTL(sim$X, Y, lambda = 1e-3, K = K,
                survival = TRUE, censored.mat = W)
summary(fit)
```

`summary.orthoMTL` *Summarise an orthoMTL Object*

Description

Displays a detailed summary of a fitted `orthoMTL` model, including hyperparameters, coefficient matrix statistics, and top features by mean absolute coefficient.

Usage

```
## S3 method for class 'orthoMTL'
summary(object, n_top = 5, ...)
```

Arguments

`object`	A fitted model object of class `"orthoMTL"`.
`n_top`	Number of top features to display. Default: 5.
`...`	Additional arguments (currently ignored).

Value

Invisibly returns `object`.

Examples

```
set.seed(42)
n <- 100; p <- 10; n_tasks <- 3
X <- matrix(rnorm(n * p), n, p)
colnames(X) <- paste0("V", seq_len(p))
Y <- X %*% matrix(rnorm(p * n_tasks), p) + matrix(rnorm(n * n_tasks), n) * 0.1
fit <- orthoMTL(X, Y, lambda = 1e-3)
summary(fit)
```

Index