

Package ‘fdb’

October 10, 2026

Type Package

Title Frequentist Dynamic Borrowing for Hybrid-Control Survival Trials

Version 0.2.2

Description Implements a class of likelihood-informed frequentist dynamic borrowing methods for hybrid-control survival trials based on penalized Cox partial likelihood estimation. Implements four likelihood-informed penalty structures (precision-weighted L1, smoothed integrated-gate, information-adaptive minimax concave penalty (MCP), and likelihood-ratio-weighted L1), together with the adaptive lasso borrowing approach of Li et al. (2023, <doi:10.1002/bimj.202100406>). Provides conditional model-based standard errors and local plug-in sandwich variance approximations, with smoothed penalties. Tools for design-stage lambda calibration via simulation, including a two-stage coarse-fine grid search, drift-level early stopping, and per-method tuning under both inference types, are also provided. A simulation harness for evaluating type I error and statistical power across population drift scenarios is included.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.6.0)

Imports survival, stats, parallel, utils

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

URL <https://github.com/yamagubed/fdb>

BugReports <https://github.com/yamagubed/fdb/issues>

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

RoxygenNote 7.3.3

NeedsCompilation no

Author Yusuke Yamaguchi [aut, cre]

Maintainer Yusuke Yamaguchi <yamagubed@gmail.com>
Repository CRAN
Date/Publication 2026-10-10 15:50:08 UTC

Contents

fdb-package	2
add_ess_to_simulation_result	4
calibrate_all_lambdas	4
calibrate_censor_rate	6
calibrate_lambda_grid	7
calibrate_lambda_grid_two_stage	9
compute_ess_from_raw	11
compute_sandwich_se	12
drift_set	13
fit_all_methods	13
fit_internal_only	15
fit_li_adaptive_lasso	15
fit_naive_pooled	16
fit_one_penalized_method	17
fit_P1_precision_L1	18
fit_P2_gated_L1	19
fit_P3_info_MCP	20
fit_P4_LRweighted_L1	21
lambdas_default	22
method_labels	22
run_drift_curve	23
run_fdb_study	24
run_simulation	26
scenario_S1	27
simulate_hybrid_cox	27
Index	30

fdb-package	<i>fdb: Frequentist Dynamic Borrowing for Hybrid-Control Survival Trials</i>
-------------	--

Description

Implements likelihood-informed frequentist dynamic borrowing methods for hybrid-control survival trials based on penalized Cox partial likelihood estimation, together with design-stage lambda calibration and a simulation harness for evaluating operating characteristics.

Main user-facing functions

- `simulate_hybrid_cox`: Generate a hybrid-control Cox proportional hazards dataset.
- `fit_all_methods`: Fit all borrowing methods on a single dataset, returning both model-based and sandwich-based inference quantities.
- `fit_one_penalized_method`: Fit a single penalized borrowing method.
- `run_simulation`: Run a Monte Carlo simulation under a fixed scenario, comparing all methods.
- `calibrate_lambda_grid`, `calibrate_lambda_grid_two_stage`, `calibrate_all_lambdas`: Design-stage lambda calibration utilities.
- `run_fdb_study`: One-stop wrapper that performs lambda calibration and then evaluates type I error and power curves across population drift scenarios.

Penalty methods

The package implements four likelihood-informed penalties together with the adaptive lasso comparator of Li et al. (2023):

- **Adaptive lasso**: adaptive lasso (Li et al. 2023).
- **Precision-weighted L1**: precision-weighted L1 penalty.
- **Integrated-gate**: smoothed integrated-gate penalty.
- **Information-adaptive MCP**: information-adaptive minimax concave penalty (MCP).
- **LR-weighted L1**: likelihood-ratio-weighted L1 penalty.

References

- Li, R., Lin, R., Huang, J., Tian, L., and Zhu, J. (2023). A frequentist approach to dynamic borrowing. *Biometrical Journal* 65(7), 2100406.
- Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics* 38(2), 894-942.
- Andersen, P. K. and Gill, R. D. (1982). Cox's regression model for counting processes: A large sample study. *The Annals of Statistics* 10(4), 1100-1120.

Inference and calibration

Model-based standard errors from profiled fits treat the fitted drift as fixed. Sandwich standard errors are local plug-in approximations holding first-stage weights fixed; they do not guarantee nominal coverage. Curvature clipping modifies the variance calculation, not the fitted coefficients. Calibration targets a prespecified drift grid and threshold, with Monte Carlo error. It does not establish control between grid points or outside the grid. Effective sample size is a variance-equivalent gain and can be negative; it does not account for bias.

Author(s)

Maintainer: Yusuke Yamaguchi <yamagubed@gmail.com>

See Also

Useful links:

- <https://github.com/yamagubed/fdb>
- Report bugs at <https://github.com/yamagubed/fdb/issues>

`add_ess_to_simulation_result`

Append ESS to a simulation summary

Description

Append ESS to a simulation summary

Usage

```
add_ess_to_simulation_result(simres, NS, ref_method = "Internal-only")
```

Arguments

<code>simres</code>	Output of run_simulation .
<code>NS</code>	Reference sample-size scale. The study wrappers use the total internal randomized sample size, $nI1 + nI0$.
<code>ref_method</code>	Reference method (default "Internal-only").

Value

The same `simres` list with an ESS column merged into its summary component.

`calibrate_all_lambdas` *Calibrate lambda for all borrowing methods*

Description

Runs lambda calibration for all five penalized borrowing methods (Li adaptive lasso plus P1-P4) and returns a unified set of calibrated lambdas under both model-based and sandwich inference.

Usage

```

calibrate_all_lambdas(
  lambda_grid,
  scenario_base,
  drift_set,
  drift_set_cal = NULL,
  drift_set_confirm = NULL,
  nsim_cal = 500,
  nsim_confirm = nsim_cal,
  alpha = 0.025,
  alpha_cal = alpha,
  seed = 1,
  parallel = FALSE,
  ncores = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS,
  gamma_li = 1,
  gate_c = 1.64,
  gate_tau = 0.25,
  gamma_mcp = 3,
  delta_bounds = DEFAULT_DELTA_BOUNDS,
  n_grid_opt = DEFAULT_N_GRID_OPT,
  two_stage = TRUE,
  n_fine = 6,
  primary_inference = c("sandwich", "model_based"),
  confirm_full_drift = TRUE,
  early_stop_drift = TRUE,
  stop_rule = c("point", "upper95"),
  select_rule = c("point", "upper95"),
  rho_mcp = DEFAULT_RHO_MCP
)

```

Arguments

<code>lambda_grid</code>	Initial (coarse) lambda grid used for all methods.
<code>scenario_base</code>	A scenario list.
<code>drift_set</code>	Default calibration drift grid when <code>drift_set_cal</code> is omitted.
<code>drift_set_cal</code>	Reduced drift set for the calibration stages.
<code>drift_set_confirm</code>	Drift set for the optional confirmation stage; NULL uses the calibration grid.
<code>nsim_cal, nsim_confirm</code>	Replicates per drift value in calibration and confirmation stages.
<code>alpha, alpha_cal, seed, parallel, ncores, robust, eps, gamma_li</code>	As in calibrate_lambda_grid_two_stage .
<code>gate_c, gate_tau, gamma_mcp, delta_bounds, n_grid_opt, n_fine</code>	As in calibrate_lambda_grid_two_stage .

two_stage Logical; use two-stage calibration if TRUE, otherwise single-stage.
 primary_inference, confirm_full_drift, early_stop_drift
 As in [calibrate_lambda_grid_two_stage](#).
 stop_rule, select_rule
 As in [calibrate_lambda_grid_two_stage](#).
 rho_mcp MCP transition fraction in (0, 1), default 0.1.

Value

A list with calibration_table (long form), calibration_summary (per-method, per-lambda worst case), lambda_star (5 x 2 matrix indexed by method and inference), lambda_star_table (long form), and method_calibrations (per-method full output).

calibrate_censor_rate *Calculate a fixed exponential censoring rate*

Description

Solves for the rate giving the target marginal censoring probability among internal subjects, weighted by their treatment allocation. Uses numerical integration of the Weibull event-time distribution; normally distributed covariate effects are integrated by 64-point Gaussian quadrature. No random draws or realized trial data are used. The rate does not depend on external drift or external sample size. Calculate separate rates under the null and alternative if the same internal censoring target is desired under both.

Usage

```
calibrate_censor_rate(
  nI1 = 150,
  nI0 = 150,
  theta0 = 0,
  p = 5,
  beta = NULL,
  rho = 0,
  shape = 1.2,
  lambda = 0.02,
  target_cens = 0.2
)
```

Arguments

nI1, nI0 Internal treated and control sample sizes.
 theta0 True treatment log hazard ratio.
 p Number of normally distributed baseline covariates.
 beta Covariate coefficients; NULL uses rep(0.2, p).

rho	Equicorrelation of internal covariates.
shape, lambda	Baseline Weibull cumulative hazard is $\lambda * t^{\text{shape}}$.
target_cens	Target internal censoring proportion in $[0, 1)$.

Value

A nonnegative exponential rate. Zero means no censoring.

Examples

```
calibrate_censor_rate(300, 150, p = 0, target_cens = 0.4)
```

calibrate_lambda_grid *Single-stage lambda calibration for one borrowing method*

Description

For each lambda in lambda_grid, simulates nsim replicates under each drift value in drift_set (with $\theta_0 = 0$), records the model-based and sandwich rejection rates, and selects the largest lambda whose worst-case rejection rate over the drift set does not exceed alpha_cal. Missing statistics are excluded from descriptive rejection rates, but a candidate with incomplete results is ineligible for the affected inference type. Fitting failures generate a diagnostic warning; if both inference types have no usable results at a drift, calibration stops.

Usage

```
calibrate_lambda_grid(
  method = c("Li", "P1", "P2", "P3", "P4"),
  lambda_grid,
  scenario_base,
  drift_set,
  nsim = 300,
  alpha = 0.025,
  alpha_cal = alpha,
  seed = 1,
  parallel = FALSE,
  ncores = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS,
  gamma_li = 1,
  gate_c = 1.64,
  gate_tau = 0.25,
  gamma_mcp = 3,
  delta_bounds = DEFAULT_DELTA_BOUNDS,
  n_grid_opt = DEFAULT_N_GRID_OPT,
  early_stop_drift = FALSE,
  stop_rule = c("point", "upper95"),
```

```

    select_rule = c("point", "upper95"),
    rho_mcp = DEFAULT_RHO_MCP
  )

```

Arguments

method	One of "Li", "P1", "P2", "P3", "P4".
lambda_grid	Numeric vector of candidate lambda values.
scenario_base	A scenario list as accepted by run_simulation . Its theta0 and delta0 are overwritten internally.
drift_set	Numeric vector of drift values (log HR) on which to evaluate type I error.
nsim	Number of replicates per drift value.
alpha	Nominal level used for rejection decisions.
alpha_cal	Calibration threshold (worst-case rejection rate must not exceed this).
seed	RNG seed.
parallel	Logical; enable parallel evaluation across replicates.
ncores	Number of workers; NULL uses two. Checks use at most two.
robust	Use robust (Lin-Wei) Cox SEs.
eps	Smoothing parameter.
gamma_li	Adaptive lasso exponent.
gate_c, gate_tau	P2 gate parameters.
gamma_mcp	MCP shape parameter for P3.
delta_bounds	Optimization interval for delta.
n_grid_opt	Coarse-grid points for non-convex objectives.
early_stop_drift	Logical; if TRUE, drop lambda values that have already failed the calibration constraint after evaluating a subset of drift values.
stop_rule	Either "point" (use point estimate of worst-case rejection rate for early stopping) or "upper95" (use the Monte Carlo 95% upper confidence bound).
select_rule	Selection rule for the final calibrated lambda (same options as stop_rule).
rho_mcp	MCP transition fraction in (0, 1), default 0.1.

Value

A list with elements method, details (per-lambda x drift), summary (per-lambda worst-case), calibration_table (long format), and lambda_star (selected lambdas for both inference types).

Examples

```
# Tiny execution example; use substantially more replicates for calibration.
cal <- calibrate_lambda_grid(method = "P1",
                             lambda_grid = c(0.05, 0.2),
                             scenario_base = scenario_S1,
                             drift_set = make_drift_set_from_values(
                               c(1.0, 1.1)),
                             nsim = 2, seed = 1)

cal$lambda_star
```

```
calibrate_lambda_grid_two_stage
```

Two-stage lambda calibration for one borrowing method

Description

Performs a coarse-grid calibration on a reduced drift set (`drift_set_cal`), refines the grid around the calibrated lambda, repeats the fine-grid calibration on `drift_set_cal`, and optionally confirms candidates on `drift_set_confirm`. When this is NULL, confirmation uses `drift_set_cal`. An explicit confirmation grid is honored and may be wider than the calibration grid. The fine-stage grid is the union of the original coarse grid and the refined points. Smaller candidates are retained for confirmation, even if they failed an earlier stage. Selection uses the final stage's results; a passing coarse-stage result is never substituted for confirmation.

Usage

```
calibrate_lambda_grid_two_stage(
  method = c("Li", "P1", "P2", "P3", "P4"),
  lambda_grid_coarse,
  scenario_base,
  drift_set_cal,
  drift_set_confirm = NULL,
  nsim_cal = 300,
  nsim_confirm = nsim_cal,
  alpha = 0.025,
  alpha_cal = alpha,
  seed = 1,
  parallel = FALSE,
  ncores = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS,
  gamma_li = 1,
  gate_c = 1.64,
  gate_tau = 0.25,
  gamma_mcp = 3,
  delta_bounds = DEFAULT_DELTA_BOUNDS,
```

```

n_grid_opt = DEFAULT_N_GRID_OPT,
n_fine = 6,
primary_inference = c("sandwich", "model_based"),
confirm_full_drift = TRUE,
early_stop_drift = TRUE,
stop_rule = c("point", "upper95"),
select_rule = c("point", "upper95"),
rho_mcp = DEFAULT_RHO_MCP
)

```

Arguments

method	One of "Li", "P1", "P2", "P3", "P4".
lambda_grid_coarse	Initial coarse lambda grid.
scenario_base	A scenario list.
drift_set_cal	Drift set used for coarse and fine calibration.
drift_set_confirm	Confirmation drift grid; NULL uses drift_set_cal.
nsim_cal	Replicates per drift value in the calibration stages.
nsim_confirm	Replicates per drift value in the confirmation stage.
alpha, alpha_cal	Nominal level and calibration threshold.
seed	RNG seed.
parallel, ncores	Parallelization controls.
robust, eps, gamma_li, gate_c, gate_tau, gamma_mcp, delta_bounds, n_grid_opt	Same as in calibrate_lambda_grid .
n_fine	Number of refined points, before adding the original coarse grid.
primary_inference	Which inference type drives refinement.
confirm_full_drift	Logical; if TRUE, perform a confirmation stage using drift_set_confirm, or drift_set_cal if omitted.
early_stop_drift, stop_rule, select_rule	As in calibrate_lambda_grid .
rho_mcp	MCP transition fraction in (0, 1), default 0.1.

Value

A list with components method, coarse, fine, confirm, calibration_summary, calibration_table, lambda_star, and final_stage.

compute_ess_from_raw	<i>Compute effective sample size from raw simulation output</i>
----------------------	---

Description

Computes a variance-ratio effective sample size for each method relative to a reference method (typically internal-only), based on the Monte Carlo variance of the treatment effect estimator across replicates: $ESS_m = NS \cdot (Var_{ref}/Var_m - 1)$.

Usage

```
compute_ess_from_raw(
  raw_df,
  NS,
  ref_method = "Internal-only",
  methods_exclude = NULL
)
```

Arguments

raw_df	A data frame with at least method and theta_hat columns.
NS	Reference sample-size scale. The study wrappers use the total internal randomized sample size, nI1 + nI0.
ref_method	Name of the reference method (default "Internal-only").
methods_exclude	Optional character vector of methods to exclude from the output.

Value

A data frame with method and ESS columns. ESS is a variance-equivalent gain, can be negative, and does not measure bias.

Examples

```
sim_out <- run_simulation(nsim = 2, scenario = scenario_S1,
                        lambdas = lambdas_default, alpha = 0.025, seed = 1)
compute_ess_from_raw(sim_out$raw, NS = 300)
```

compute_sandwich_se	<i>Sandwich standard error for a penalized Cox borrowing estimator</i>
---------------------	--

Description

Computes the plug-in sandwich variance for the smoothed penalized Cox M-estimator. The Cox information and score residuals are evaluated at the penalized parameter estimate, and the penalty curvature is added to the (delta, delta) element of the bread. First-stage estimates are held fixed; this does not establish conditional or unconditional variance validity. Clipped curvature modifies the variance calculation without changing the fitted coefficients.

Usage

```
compute_sandwich_se(dat, delta_hat, theta_hat, beta_hat, pen_curv, xnames)
```

Arguments

dat	A data frame with columns time, status, T, Z, and covariates named in xnames.
delta_hat	Penalized estimate of the drift parameter.
theta_hat	Penalized estimate of the treatment effect.
beta_hat	Penalized estimate of the covariate coefficient vector.
pen_curv	Penalty curvature $p''_{\lambda, \varepsilon}(\hat{\delta})$.
xnames	Character vector of covariate names.

Value

A list with components:

se_sand	Sandwich standard error for theta_hat.
Var_sand	Full sandwich variance matrix.
coef_names, theta_idx, delta_idx	Coefficient bookkeeping.
pen_curv	Penalty curvature used in the bread.
note	Status note ("OK" on success).

drift_set	<i>Drift set utilities</i>
-----------	----------------------------

Description

`make_drift_set` builds a regular grid of drift values from a range and step in HR units, optionally including the null drift (HR = 1). `make_drift_set_from_values` converts a user-supplied set of drift HR values to log-HR.

Usage

```
make_drift_set(drift_hr_range = c(0.8, 1.2), by_hr = 0.05, include_zero = TRUE)
```

```
make_drift_set_from_values(drift_hr_values, include_zero = TRUE)
```

Arguments

`drift_hr_range` Numeric vector of length 2 giving the inclusive range of drift HR values (e.g. `c(0.8, 1.2)`).

`by_hr` Step size in HR units.

`include_zero` If TRUE, ensure the null drift ($\delta = 0$) is included.

`drift_hr_values` Numeric vector of drift HR values.

Value

A numeric vector of drift values on the log-HR scale.

fit_all_methods	<i>Fit all borrowing methods on a single dataset</i>
-----------------	--

Description

Convenience wrapper that fits internal-only, naive pooled, Li adaptive lasso, and all four likelihood-informed penalties (P1-P4) on a single hybrid-control dataset and returns a tidy data frame of results. Both model-based and sandwich standard errors are reported.

Usage

```
fit_all_methods(
  dat,
  lambda_li = 0.2,
  gamma_li = 1,
  lambda_p1 = 0.2,
  lambda_p2 = 0.2,
```

```

    gate_c = 1.64,
    gate_tau = 0.25,
    lambda_p3 = 0.2,
    gamma_mcp = 3,
    lambda_p4 = 0.2,
    delta_bounds = DEFAULT_DELTA_BOUNDS,
    robust = FALSE,
    eps = SMOOTH_EPS,
    n_grid_opt = DEFAULT_N_GRID_OPT,
    rho_mcp = DEFAULT_RHO_MCP
  )

```

Arguments

<code>dat</code>	A data frame conforming to the column conventions of <code>simulate_hybrid_cox</code> .
<code>lambda_li, gamma_li</code>	Tuning for the adaptive lasso method.
<code>lambda_p1</code>	Tuning for the precision-weighted L1 (P1).
<code>lambda_p2, gate_c, gate_tau</code>	Tuning for the smoothed integrated-gate L1 (P2).
<code>lambda_p3, gamma_mcp</code>	Tuning for the information-adaptive MCP (P3).
<code>lambda_p4</code>	Tuning for the likelihood-ratio-weighted L1 (P4).
<code>delta_bounds</code>	Optimization interval for δ .
<code>robust</code>	Use robust (Lin-Wei) Cox standard errors.
<code>eps</code>	Smoothing parameter for $ \delta _\varepsilon$.
<code>n_grid_opt</code>	Number of grid points for the coarse search (used by P2 and P3).
<code>rho_mcp</code>	MCP transition fraction in (0, 1), default 0.1.

Value

A data frame with one row per method, columns `method`, `theta_hat`, `se_theta` (model-based), `se_sand` (sandwich), `z`, `z_sand`, `delta_hat`, `pen_curv`, `pen_curv_raw`.

Examples

```

set.seed(1)
sim <- simulate_hybrid_cox(nI1 = 100, nI0 = 100, nE = 200,
                          theta0 = log(0.8), delta0 = 0)
fit_all_methods(sim$data)

```

fit_internal_only	<i>Internal-control-only Cox analysis</i>
-------------------	---

Description

Fits a Cox proportional hazards model on the internal ($Z = 0$) subjects only, without any borrowing from external controls.

Usage

```
fit_internal_only(dat, xnames = NULL, robust = FALSE)
```

Arguments

dat	A data frame from simulate_hybrid_cox or conforming to its column conventions.
xnames	Character vector of covariate names. If NULL, automatically detected as columns matching " <code>^X\d+\$</code> ".
robust	Use robust (Lin-Wei) Cox standard errors.

Value

A list of estimates and inference quantities.

fit_li_adaptive_lasso	<i>Adaptive lasso borrowing of Li et al. (2023)</i>
-----------------------	---

Description

Implements the adaptive lasso borrowing penalty $p_\lambda(\delta) = \lambda |\hat{\delta}_0|^{-\gamma} |\delta|$, where $\hat{\delta}_0$ is the unpenalized maximum partial likelihood estimator of the drift parameter.

Usage

```
fit_li_adaptive_lasso(
  dat,
  xnames = NULL,
  lambda,
  gamma = 1,
  delta_bounds = DEFAULT_DELTA_BOUNDS,
  full_fit = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS
)
```

Arguments

dat	A data frame from simulate_hybrid_cox or conforming to its column conventions.
xnames	Character vector of covariate names. If NULL, automatically detected.
lambda	Penalty strength ($\lambda \geq 0$).
gamma	Adaptive-weight exponent (typically 1).
delta_bounds	Numeric vector of length 2 giving the optimization interval for δ .
full_fit	Optional pre-computed full Cox fit (output of an internal helper). If NULL, the fit is computed.
robust	Use robust (Lin-Wei) Cox standard errors.
eps	Smoothing parameter for $ \delta _\epsilon$.

Value

A list of estimates and inference quantities, including the effective penalty weight w , the initial estimator delta0_hat , and its standard error se_delta .

References

Li, R., Lin, R., Huang, J., Tian, L., and Zhu, J. (2023). A frequentist approach to dynamic borrowing. *Biometrical Journal* 65(7), 2100406.

fit_naive_pooled	<i>Naive pooled Cox analysis</i>
------------------	----------------------------------

Description

Fits a Cox proportional hazards model treating concurrent and external controls as exchangeable (full pooling), without any indicator for external-control membership.

Usage

```
fit_naive_pooled(dat, xnames = NULL, robust = FALSE)
```

Arguments

dat	A data frame from simulate_hybrid_cox or conforming to its column conventions.
xnames	Character vector of covariate names. If NULL, automatically detected as columns matching " $^X\d+\$$ ".
robust	Use robust (Lin-Wei) Cox standard errors.

Value

A list of estimates and inference quantities.

fit_one_penalized_method

Fit a single penalized borrowing method

Description

Dispatches to the appropriate user-facing fit function for one of the implemented penalized borrowing methods.

Usage

```
fit_one_penalized_method(
  dat,
  method = c("Li", "P1", "P2", "P3", "P4"),
  lambda,
  gamma_li = 1,
  gate_c = 1.64,
  gate_tau = 0.25,
  gamma_mcp = 3,
  delta_bounds = DEFAULT_DELTA_BOUNDS,
  robust = FALSE,
  eps = SMOOTH_EPS,
  n_grid_opt = DEFAULT_N_GRID_OPT,
  rho_mcp = DEFAULT_RHO_MCP
)
```

Arguments

dat	A data frame conforming to the column conventions of simulate_hybrid_cox .
method	One of "Li", "P1", "P2", "P3", "P4".
lambda	Penalty strength.
gamma_li	Exponent for the adaptive lasso weight (Li method).
gate_c, gate_tau	Threshold and smoothness for the P2 gate.
gamma_mcp	MCP shape parameter for P3.
delta_bounds	Optimization interval for delta.
robust	Use robust (Lin-Wei) Cox standard errors.
eps	Smoothing parameter for $ \delta _\varepsilon$.
n_grid_opt	Number of grid points for the coarse search (used by P2 and P3).
rho_mcp	MCP transition fraction in (0, 1), default 0.1.

Value

A list of estimates and inference quantities (same format as the underlying fit functions).

Examples

```
set.seed(1)
sim <- simulate_hybrid_cox(nI1 = 100, nI0 = 100, nE = 200,
  theta0 = log(0.8), delta0 = 0)
fit_one_penalized_method(sim$data, method = "P1", lambda = 0.2)
```

fit_P1_precision_L1	<i>Precision-weighted L1 penalty (P1)</i>
---------------------	---

Description

Implements $p_\lambda(\delta) = \lambda|\delta|/\widehat{SE}(\hat{\delta}_0)$, where $\widehat{SE}(\hat{\delta}_0)$ is the standard error of the unpenalized drift estimator.

Usage

```
fit_P1_precision_L1(
  dat,
  xnames = NULL,
  lambda,
  delta_bounds = DEFAULT_DELTA_BOUNDS,
  full_fit = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS
)
```

Arguments

dat	A data frame from simulate_hybrid_cox or conforming to its column conventions.
xnames	Character vector of covariate names. If NULL, automatically detected.
lambda	Penalty strength ($\lambda \geq 0$).
delta_bounds	Numeric vector of length 2 giving the optimization interval for δ .
full_fit	Optional pre-computed full Cox fit (output of an internal helper). If NULL, the fit is computed.
robust	Use robust (Lin-Wei) Cox standard errors.
eps	Smoothing parameter for $ \delta _\epsilon$.

Value

A list of estimates and inference quantities, including the effective penalty weight w and the first-stage standard error `se_delta`.

fit_P2_gated_L1	<i>Smoothed integrated-gate penalty (P2)</i>
-----------------	--

Description

Implements

$$p_{\lambda,\varepsilon}(\delta) = \lambda \int_{\varepsilon}^{\sqrt{\delta^2 + \varepsilon^2}} [1 + \exp\{(u/\widehat{SE}(\hat{\delta}_0) - c)/\tau\}]^{-1} du,$$

which provides a continuous relaxation of test-then-pool procedures driven by standardized evidence against commensurability.

Usage

```
fit_P2_gated_L1(
  dat,
  xnames = NULL,
  lambda,
  c = 1.64,
  tau = 0.25,
  delta_bounds = DEFAULT_DELTA_BOUNDS,
  full_fit = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS,
  n_grid_opt = DEFAULT_N_GRID_OPT
)
```

Arguments

dat	A data frame from simulate_hybrid_cox or conforming to its column conventions.
xnames	Character vector of covariate names. If NULL, automatically detected.
lambda	Penalty strength ($\lambda \geq 0$).
c	Evidence threshold (default 1.64, one-sided 5% gate).
tau	Gate smoothness parameter (positive).
delta_bounds	Numeric vector of length 2 giving the optimization interval for δ .
full_fit	Optional pre-computed full Cox fit (output of an internal helper). If NULL, the fit is computed.
robust	Use robust (Lin-Wei) Cox standard errors.
eps	Smoothing parameter for $ \delta _{\varepsilon}$.
n_grid_opt	Number of grid points for the coarse search in the non-convex objective.

Value

A list of estimates and inference quantities, including the evidence-gate value `g_hat` and the raw (unstabilized) penalty curvature `pen_curv_raw`.

fit_P3_info_MCP

*Information-adaptive minimax concave penalty (P3)***Description**

Implements

$$p_{\lambda}(\delta) = MCP(\delta / \widehat{SE}(\hat{\delta}_0); \lambda, \gamma_{MCP}),$$

where MCP is the minimax concave penalty of Zhang (2010). Reduces bias when moderate population drift is present while retaining strong shrinkage near $\delta = 0$.

Usage

```
fit_P3_info_MCP(
  dat,
  xnames = NULL,
  lambda,
  gamma_mcp = 3,
  delta_bounds = DEFAULT_DELTA_BOUNDS,
  full_fit = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS,
  n_grid_opt = DEFAULT_N_GRID_OPT,
  rho_mcp = DEFAULT_RHO_MCP
)
```

Arguments

dat	A data frame from simulate_hybrid_cox or conforming to its column conventions.
xnames	Character vector of covariate names. If NULL, automatically detected.
lambda	Penalty strength ($\lambda \geq 0$).
gamma_mcp	MCP shape parameter (> 1).
delta_bounds	Numeric vector of length 2 giving the optimization interval for δ .
full_fit	Optional pre-computed full Cox fit (output of an internal helper). If NULL, the fit is computed.
robust	Use robust (Lin-Wei) Cox standard errors.
eps	Smoothing parameter for $ \delta _{\varepsilon}$.
n_grid_opt	Number of grid points for the coarse search in the non-convex objective.
rho_mcp	MCP transition fraction in $(0, 1)$; $h = \text{rho_mcp} * \text{gamma_mcp} * \text{lambda}$, in standard-error units. The software default 0.1 is not calibrated.

Details

MCP is smoothed at both the origin and the flat-tail transition by integrating the continuously differentiable slope described in the manuscript. The origin smoothing parameter ϵ remains on the log-hazard-ratio scale. The transition half-width h is on the standardized-drift scale. First-stage quantities are held fixed. This replaces the pre-0.2.2 MCP parameterization and requires new calibration. Grid-assisted optimization includes the origin, initial drift estimate, and both transition boundaries on either side, and refines every detected local minimum.

Value

A list of estimates and inference quantities, including the standardized λ , origin slope $\lambda_{\text{eff}} = \lambda / \text{se_delta}$, standardized transition width h , p_3 parameterization, and the raw curvature pen_curv_raw .

References

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics* 38(2), 894-942.

fit_P4_LRweighted_L1	<i>Likelihood-ratio-weighted L1 penalty (P4)</i>
----------------------	--

Description

Implements $p_\lambda(\delta) = \lambda|\delta| \exp\{-\frac{1}{2}LR(\delta = 0)\}$, where $LR(\delta = 0)$ is the likelihood ratio statistic for testing the null of no population drift. Borrowing is downweighted when external data conflict strongly with internal controls.

Usage

```
fit_P4_LRweighted_L1(
  dat,
  xnames = NULL,
  lambda,
  delta_bounds = DEFAULT_DELTA_BOUNDS,
  full_fit = NULL,
  node_lta_fit = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS
)
```

Arguments

<code>dat</code>	A data frame from simulate_hybrid_cox or conforming to its column conventions.
<code>xnames</code>	Character vector of covariate names. If <code>NULL</code> , automatically detected.

lambda	Penalty strength ($\lambda \geq 0$).
delta_bounds	Numeric vector of length 2 giving the optimization interval for δ .
full_fit	Optional pre-computed full Cox fit (output of an internal helper). If NULL, the fit is computed.
nodelta_fit	Optional pre-computed restricted (no-Z) Cox fit.
robust	Use robust (Lin-Wei) Cox standard errors.
eps	Smoothing parameter for $ \delta _\epsilon$.

Value

A list of estimates and inference quantities, including the likelihood ratio statistic LR0 and effective weight w.

lambdas_default	<i>Default tuning parameters</i>
-----------------	----------------------------------

Description

A reference tuning list for the borrowing methods, used when no calibrated tuning is supplied. The default penalty strengths ($\lambda = 0.20$) are reasonable starting values for the default scenario but should generally be replaced by calibrated values for confirmatory use.

Usage

lambdas_default

Format

A list with elements lambda_li, gamma_li, lambda_p1, lambda_p2, gate_c, gate_tau, lambda_p3, gamma_mcp, rho_mcp, lambda_p4, and delta_bounds.

method_labels	<i>Manuscript method labels</i>
---------------	---------------------------------

Description

Converts short calibration keys or legacy result tags to manuscript labels. Unknown labels are retained. With no argument, returns the seven labels in manuscript order. Calibration API keys Li and P1-P4 remain unchanged.

Usage

method_labels(x = NULL)

Arguments

x Character vector of keys or labels, or NULL for all labels.

Value

A character vector of display labels.

run_drift_curve	<i>Evaluate operating characteristics across a set of drift values</i>
-----------------	--

Description

For each drift value in `drift_set`, runs a simulation under the supplied `theta0` (e.g. 0 for type I error, $\log(0.8)$ for power) and the given tuning parameters, then returns a stacked summary across drift values. The reference internal randomized sample size used for ESS is taken as `nI1 + nI0` from `scenario_base`.

Usage

```
run_drift_curve(
  theta0,
  drift_set,
  scenario_base,
  lambdas,
  nsim = 1000,
  alpha = 0.025,
  seed = 1,
  parallel = FALSE,
  ncores = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS,
  n_grid_opt = DEFAULT_N_GRID_OPT,
  keep_raw = FALSE
)
```

Arguments

<code>theta0</code>	True treatment effect (log HR) used in the simulations.
<code>drift_set</code>	Numeric vector of drift values (log HR).
<code>scenario_base</code>	A scenario list. <code>theta0</code> and <code>delta0</code> are overwritten internally for each drift value.
<code>lambdas</code>	Tuning list (as in run_simulation).
<code>nsim</code>	Number of replicates per drift value.
<code>alpha</code>	Nominal level.
<code>seed</code>	RNG seed (per-drift offsets are added internally).

parallel, ncores, robust, eps, n_grid_opt
 As in [run_simulation](#).
 keep_raw Retain replicate estimates. Defaults to FALSE.

Value

By default a data frame of per-drift summaries. With keep_raw = TRUE, a list with summary and raw. Replicate IDs are unique within each drift_index and method.

run_fdb_study	<i>One-stop wrapper: calibrate lambda, then evaluate type I and power</i>
---------------	---

Description

Optionally runs design-stage lambda calibration for all five borrowing methods, then evaluates type I error and power curves across the supplied drift set using the calibrated lambdas. If do_calibration = FALSE, default lambdas are used.

Usage

```
run_fdb_study(
  scenario_base = scenario_S1,
  drift_hr_range = c(0.8, 1.2),
  drift_by_hr = 0.05,
  drift_hr_values_cal = c(0.8, 0.9, 1, 1.1, 1.2),
  alpha = 0.025,
  alt_hr = 0.8,
  do_calibration = TRUE,
  lambda_grid = exp(seq(log(0.02), log(2), length.out = 6)),
  lambda_grid_fine_length = 6,
  nsim_cal = 500,
  nsim_confirm = nsim_cal,
  nsim_curve = 1000,
  alpha_cal = alpha,
  two_stage_cal = TRUE,
  confirm_full_drift = TRUE,
  primary_calibration_inference = c("sandwich", "model_based"),
  early_stop_cal = TRUE,
  cal_stop_rule = c("point", "upper95"),
  cal_select_rule = c("point", "upper95"),
  parallel = FALSE,
  ncores = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS,
  n_grid_opt = DEFAULT_N_GRID_OPT,
  seed = 1,
  export_dir = NULL,
```

```

    rho_mcp = DEFAULT_RHO_MCP,
    keep_raw = FALSE
)

```

Arguments

scenario_base A scenario list. Defaults to [scenario_S1](#).
drift_hr_range Numeric vector of length 2 giving the inclusive range of drift HR values for the final curves.
drift_by_hr Step in HR units for the final drift grid.
drift_hr_values_cal Numeric vector of drift HR values used for the (reduced) calibration drift grid. Defaults to a five-point grid.
alpha Nominal one-sided level for the final analysis.
alt_hr Alternative-hypothesis HR used in the power curve (e.g. 0.8).
do_calibration Logical; if TRUE, run lambda calibration before the final curves.
lambda_grid Initial coarse lambda grid for calibration.
lambda_grid_fine_length Number of points in the fine refinement grid.
nsim_cal, nsim_confirm Calibration replicates per drift value.
nsim_curve Replicates per drift value for the final curves.
alpha_cal Calibration threshold (defaults to alpha).
two_stage_cal Use two-stage calibration.
confirm_full_drift Run independent confirmation on the calibration drift grid. The wider evaluation grid does not change the calibration target.
primary_calibration_inference Which inference type is used for tuning during calibration ("sandwich" or "model_based").
early_stop_cal, cal_stop_rule, cal_select_rule Drift-level early stopping and selection rules.
parallel, ncores Parallelization controls.
robust, eps, n_grid_opt Estimation controls.
seed RNG seed.
export_dir If non-NULL, write CSV outputs and an RDS of metadata to this directory.
rho_mcp MCP transition fraction in (0, 1), default 0.1.
keep_raw Retain replicate estimates in raw_type1 and raw_power, and export an RDS file when export_dir is set.

Value

A list with scenario_base, drift_hr_range, drift_set, drift_set_cal, alpha, alpha_cal, alt_hr, lambdas (final tuning), calibration (calibration output, or NULL), type1_curve, power_curve, raw_type1, and raw_power. The last two are NULL unless keep_raw is true.

run_simulation	<i>Run a Monte Carlo simulation under a fixed scenario</i>
----------------	--

Description

For each of `nsim` replicates, simulates a hybrid-control Cox dataset and fits all borrowing methods. Returns per-replicate raw results and aggregated summaries under both model-based and sandwich inference.

Usage

```
run_simulation(
  nsim = 200,
  scenario,
  lambdas,
  alpha = 0.025,
  one_sided = TRUE,
  seed = 1,
  parallel = FALSE,
  ncores = NULL,
  robust = FALSE,
  eps = SMOOTH_EPS,
  n_grid_opt = DEFAULT_N_GRID_OPT
)
```

Arguments

<code>nsim</code>	Number of Monte Carlo replicates, at least two.
<code>scenario</code>	A list of data-generating parameters with elements <code>nI1</code> , <code>nI0</code> , <code>nE</code> , <code>theta0</code> , <code>delta0</code> , <code>p</code> , <code>beta</code> , <code>rho</code> , <code>cov_shift</code> , <code>shape</code> , <code>lambda</code> , and <code>target_cens</code> . An optional <code>censor_rate</code> fixes the rate; otherwise it is calculated before generating replicates from the internal population. See scenario_S1 for an example.
<code>lambdas</code>	A list of tuning parameters with elements <code>lambda_li</code> , <code>gamma_li</code> , <code>lambda_p1</code> , <code>lambda_p2</code> , <code>gate_c</code> , <code>gate_tau</code> , <code>lambda_p3</code> , <code>gamma_mcp</code> , <code>rho_mcp</code> (optional, default 0.1), <code>lambda_p4</code> , and <code>delta_bounds</code> . See lambdas_default for an example.
<code>alpha</code>	Nominal one-sided (or two-sided) significance level.
<code>one_sided</code>	Logical; if TRUE, rejection corresponds to a hazard ratio less than one.
<code>seed</code>	RNG seed.
<code>parallel</code>	Logical; if TRUE, parallelize replicates using <code>parallel::parLapply</code> .
<code>ncores</code>	Number of workers; NULL uses two. Checks use at most two.
<code>robust</code>	Use robust (Lin-Wei) Cox standard errors.
<code>eps</code>	Smoothing parameter for $ \delta _\varepsilon$.
<code>n_grid_opt</code>	Number of grid points for the coarse search in non-convex objectives.

Value

A list with:

`raw` Per-replicate results data frame.

`summary` Aggregated summaries for both inference types (valid-result counts, Monte Carlo standard errors, rejection rate, bias, RMSE, empirical SE, average SE, 95% coverage), with an inference column.

`scenario, lambdas, settings` The inputs and run metadata.

Examples

```
sim_out <- run_simulation(nsim = 2, scenario = scenario_S1,
                        lambdas = lambdas_default, alpha = 0.025, seed = 1)
subset(sim_out$summary, inference == "sandwich")
```

scenario_S1	<i>Example simulation scenario</i>
-------------	------------------------------------

Description

A reference scenario used in the examples and unit tests. Provides 150 internal treated, 150 internal concurrent control, and 300 external control subjects, with 5 correlated covariates and a Weibull baseline.

Usage

```
scenario_S1
```

Format

A list with elements `nI1`, `nI0`, `nE`, `theta0`, `delta0`, `p`, `beta`, `rho`, `cov_shift`, `shape`, `lambda`, and `target_cens`.

simulate_hybrid_cox	<i>Simulate a hybrid-control Cox proportional hazards dataset</i>
---------------------	---

Description

Generates a randomized trial augmented with an external control arm, under a Cox proportional hazards model

$$h(t \mid T, Z, X) = h_0(t) \exp(\theta T + \delta Z + \beta^\top X),$$

where T is the treatment indicator, Z is the external-control indicator, X is a vector of baseline covariates, θ is the treatment effect, and δ is the population drift between concurrent and external controls.

Usage

```
simulate_hybrid_cox(
  nI1 = 150,
  nI0 = 150,
  nE = 300,
  theta0 = 0,
  delta0 = 0,
  p = 5,
  beta = NULL,
  rho = 0,
  cov_shift = rep(0, p),
  shape = 1.2,
  lambda = 0.02,
  target_cens = 0.2,
  censor_rate = NULL
)
```

Arguments

nI1	Number of internal randomized treated subjects.
nI0	Number of internal randomized concurrent control subjects.
nE	Number of external control subjects.
theta0	True treatment effect (log hazard ratio).
delta0	True population drift (log hazard ratio for external vs. internal controls).
p	Number of covariates; use 0 for the no-covariate setting.
beta	Numeric vector of covariate coefficients (length p). Defaults to <code>rep(0.2, p)</code> .
rho	Equicorrelation of covariates.
cov_shift	Covariate mean shift in the external control arm (numeric vector of length p).
shape	Weibull shape parameter for event-time generation.
lambda	Baseline scale parameter for event-time generation.
target_cens	Target censoring proportion among internal subjects.
censor_rate	Fixed exponential censoring rate. If NULL, calculated deterministically from the internal population distribution using calibrate_censor_rate . Zero gives no censoring. An explicit rate overrides the target; realized censoring is random.

Value

A list with components:

- `data` A data frame with time, status, T, Z, and covariates X1, ..., Xp.
- `truth` The true parameters used to generate the data.
- `settings` The simulation settings, including the calibrated censoring rate.

Examples

```
set.seed(1)
sim <- simulate_hybrid_cox(nI1 = 100, nI0 = 100, nE = 200,
                           theta0 = log(0.8), delta0 = 0)
head(sim$data)
```

Index

* datasets

`lambdas_default`, [22](#)

`scenario_S1`, [27](#)

`add_ess_to_simulation_result`, [4](#)

`calibrate_all_lambdas`, [3](#), [4](#)

`calibrate_censor_rate`, [6](#), [28](#)

`calibrate_lambda_grid`, [3](#), [7](#), [10](#)

`calibrate_lambda_grid_two_stage`, [3](#), [5](#), [6](#),
 [9](#)

`compute_ess_from_raw`, [11](#)

`compute_sandwich_se`, [12](#)

`drift_set`, [13](#)

`fdb` (`fdb-package`), [2](#)

`fdb-package`, [2](#)

`fit_all_methods`, [3](#), [13](#)

`fit_internal_only`, [15](#)

`fit_li_adaptive_lasso`, [15](#)

`fit_naive_pooled`, [16](#)

`fit_one_penalized_method`, [3](#), [17](#)

`fit_P1_precision_L1`, [18](#)

`fit_P2_gated_L1`, [19](#)

`fit_P3_info_MCP`, [20](#)

`fit_P4_LRweighted_L1`, [21](#)

`lambdas_default`, [22](#), [26](#)

`make_drift_set` (`drift_set`), [13](#)

`make_drift_set_from_values` (`drift_set`),
 [13](#)

`method_labels`, [22](#)

`run_drift_curve`, [23](#)

`run_fdb_study`, [3](#), [24](#)

`run_simulation`, [3](#), [4](#), [8](#), [23](#), [24](#), [26](#)

`scenario_S1`, [25](#), [26](#), [27](#)

`simulate_hybrid_cox`, [3](#), [14–21](#), [27](#)