Package: selSurv (via r-universe)

October 31, 2026

Title Inference after Cox-Lasso Variable Selection

Version 0.1.1

Description Methods for inference after Cox-Lasso variable selection in right-censored survival data. The package provides Cox-Lasso fitting, support-restricted Cox refitting, sample splitting, approximately debiased coefficient estimates, score-level bootstrap intervals, and model-based bootstrap intervals with fixed-support or repeated-selection procedures. The methods differ in their inferential targets and theoretical guarantees. Relevant methodological investigations are presented in Schemet and Friedrich-Welz (2026) <doi:10.1186/s12874-026-02887-0>, Schemet, Groll and Friedrich-Welz (2026) <doi:10.48550/arXiv.2608.18893>, and Schemet and Friedrich-Welz (2026) <doi:10.48550/arXiv.2608.24230>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports stats, utils, survival, glmnet, MASS

Suggests knitr, Matrix, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Lena Schemet [aut, cre], Sarah Friedrich-Welz [aut], Andreas Groll [aut]

Maintainer Lena Schemet <lena.schemet@uni-a.de>

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

Date/Publication 2026-10-31 17:50:21 UTC

RemoteUrl https://github.com/cran/selSurv

RemoteRef HEAD

RemoteSha 427eadb58bc391954fcd9cab44bdf23f2e6e19f1

Contents

selSurv-package . . . 2
compare_selection_diagnostics . . . 2
coxlasso_boot . . . 3
coxlasso_boot_parametric . . . 5
coxlasso_boot_reselect . . . 6
coxlasso_debias . . . 8
coxlasso_efron_events . . . 9
coxlasso_fit . . . 10
coxlasso_refit . . . 11
coxlasso_split . . . 12
coxlasso_wildboot . . . 13
plot_selection_diagnostics . . . 14
print-methods . . . 15
selSurv_methods . . . 16
selSurv_targets . . . 16
selSurv_workflows . . . 17

Index **19**

`selSurv-package` *Selective inference workflows for penalized Cox models*

Description

Research workflows for Cox-Lasso fitting, refitting, debiasing, bootstrap intervals, split-sample inference, and procedure-level re-selection diagnostics.

Details

The package distinguishes full-model/debiased, fixed-support, and procedure-level zero-padded coefficient targets.

`compare_selection_diagnostics`
Compare selection diagnostics across bootstrap workflows

Description

Combines the `diagnostics` tables returned by `coxlasso_boot_parametric()` for fixed-support, fixed-lambda re-selection, and retuned re-selection workflows.

Usage

```
compare_selection_diagnostics(
  fixed = NULL,
  reselect_fixed_lambda = NULL,
  reselect_retuned = NULL,
  digits = 2
)
```

Arguments

`fixed`	Optional result from `coxlasso_boot_parametric(..., workflow = "fixed_support")`.
`reselect_fixed_lambda`	Optional result from `coxlasso_boot_parametric(..., workflow = "reselect_fixed_lambda")`.
`reselect_retuned`	Optional result from `coxlasso_boot_parametric(..., workflow = "reselect_retuned")`.
`digits`	Number of digits used for rounding. Use `NULL` for no rounding.

Value

A data frame with one row per term and workflow-specific re-selection and zero-frequency columns.

`coxlasso_boot` *Unified bootstrap inference for Cox-Lasso models*

Description

Computes bootstrap intervals for either an approximately debiased coefficient vector or a fixed-support Cox refit. Select the procedure with `inference` or `type`; the two routes have different inferential interpretations.

Usage

```
coxlasso_boot(
  object,
  inference = NULL,
  type = NULL,
  B = 500L,
  conf_level = 0.95,
  seed = NULL,
  ...
)
```

Arguments

`object`	A fitted Cox-Lasso object or an object returned by `coxlasso_debias()`.
`inference`	Optional inferential family. One of `"debiased"` or `"conditional"`.
`type`	Bootstrap type: `"wild"` or `"efron"` for debiasing; `"model_based"` for fixed-support resampling. Compatible types may be combined. With an explicit `inference`, `"all"` selects every available type in that family.
`B`	Number of bootstrap repetitions. Must be a positive integer.
`conf_level`	Confidence level strictly between zero and one.
`seed`	Optional non-negative integer seed. When multiple bootstrap types are requested, consecutive seeds are used in the supplied order.
`...`	Additional arguments forwarded to the selected backend engine.

Details

Supplying `type = "wild"` or `"efron"` automatically selects `inference = "debiased"`. Supplying `type = "model_based"` automatically selects `inference = "conditional"`.

Conversely, `inference = "debiased"` defaults to `type = "wild"`, while `inference = "conditional"` defaults to `type = "model_based"`.

For debiased bootstrap inference, a fitted object is automatically passed through `coxlasso_debias()` before the bootstrap engine is run. Debiasing remains independently available without bootstrap by calling `coxlasso_debias(fit)`.

Under `inference = "conditional"`, the support selected in the observed data is held fixed during bootstrap refitting. This does not condition on the complete selection event.

For `inference = "debiased"`, the score bootstrap quantifies variation around the implemented one-step correction; it does not independently establish high-dimensional coverage validity.

Value

An object of class `selsurv_boot`. Its `results` element contains one `selsurv_boot_method` object per requested bootstrap type.

See Also

`selSurv_workflows`

Examples

```
set.seed(2026)
n <- 120L
X <- matrix(stats::rnorm(n * 4L), nrow = n)
colnames(X) <- paste0("x", seq_len(ncol(X)))
event_time <- stats::rexp(n, rate = 0.12 * exp(X[, 1] - 0.7 * X[, 2]))
censor_time <- stats::rexp(n, rate = 0.06)
time <- pmin(event_time, censor_time)
status <- as.integer(event_time <= censor_time)
fit <- coxlasso_fit(X, time, status, nfolds = 3, lambda_choice = "min")
```

```
# Illustrative B only; use many more draws for inference.
db <- coxlasso_debias(fit)
boot_score <- coxlasso_boot(fit, type = "wild", B = 8, seed = 2026)
boot_fixed <- coxlasso_boot(
  fit, inference = "conditional", B = 8, seed = 2026
)
summary(boot_fixed)
```

`coxlasso_boot_parametric` *Model-based bootstrap for Cox-Lasso post-selection workflows*

Description

Runs a model-based Cox bootstrap, either keeping the observed support fixed or repeating selection within each resample. Under re-selection, unselected coefficients are stored as zero. The resulting distribution describes the fitted procedure rather than an exact selection-event-conditional distribution.

Usage

```
coxlasso_boot_parametric(
  fit,
  B = 200,
  workflow = c(
    "fixed_support", "reselect_fixed_lambda", "reselect_retuned"
  ),
  lambda_choice = fit$lambda_choice,
  nfolds = 10,
  alpha = 0.1,
  ci_type = c("percentile", "basic"),
  seed = NULL
)
```

Arguments

`fit`	A `selSurv_fit` object.
`B`	Number of bootstrap samples.
`workflow`	Bootstrap workflow: fixed support, fixed-penalty re-selection, or retuned re-selection.
`lambda_choice`	Tuning rule for `"reselect_retuned"`.
`nfolds`	Number of folds for retuning.
`alpha`	Interval significance level, so `1 - alpha` is the nominal confidence level. This is *not* the mixing parameter.

ci_type	"percentile" or "basic".
seed	Optional random seed.

Value

A selSurv_boot object.

Choosing between bootstrap interfaces

For fixed-support bootstrap, use coxlasso_boot(). Set inference = "conditional".

To repeat selection, use coxlasso_boot_reselect(). This interface has its own tuning and return conventions. Fits with a non-Lasso mixing parameter can be processed, but a warning is issued because these settings have not been assessed separately for inference. Here alpha is the significance level, not the penalty mixing parameter.

See Also

selSurv_workflows

---

coxlasso_boot_reselect

Model-based Cox bootstrap with re-selection

Description

Uses a model-based Cox bootstrap to compare fixed-support refitting with procedures that repeat variable selection. An unselected coefficient is recorded as zero; failed Cox refits are recorded as missing, not as zero.

Usage

```
coxlasso_boot_reselect(
  x, time, status,
  B = 500L,
  methods = c(
    "fixed_support", "reselect_fixed_lambda", "reselect_cv_min"
  ),
  conf_level = 0.95,
  seed = NULL,
  nfolds = 10L,
  original = NULL,
  original_support = NULL,
  original_lambda = NULL,
  standardize = TRUE,
  alpha = 1,
  type_measure = "deviance",
  ties = "efron",
```

```
  censoring = "reverse_km",
  rng_mode = c("independent_streams", "legacy_nki70"),
  glmnet_args = list(),
  progress = interactive()
)
```

Arguments

x	Numeric matrix or data frame of covariates. In data frames, numeric and integer columns remain numeric, while logical, factor, ordered-factor, and character columns are expanded using treatment coding with the first level as the reference. Ordered factors are treated as ordinary categorical predictors. Categorical predictors contribute $k-1$ separately penalized design-matrix coordinates; they are not selected as groups. To control the reference category, supply an explicit factor level order or use `relevel()` before fitting.
time	Follow-up times.
status	Event indicator coded 0/1.
B	Number of bootstrap replicates.
methods	Character vector of bootstrap procedures. Supported names are listed in `selSurv_workflows`.
conf_level	Confidence level.
seed	Optional non-negative integer seed. If `NULL`, one random root seed is generated and stored in the returned object; all internal bootstrap and CV streams are then derived deterministically from it.
nfolds	Number of cross-validation folds.
original	Optional precomputed observed-data selection and refit object. This is an advanced reproducibility interface; most users should leave it as `NULL` and let the function fit the observed-data workflow.
original_support	Observed-data support; provide it together with `original_lambda`.
original_lambda	Optional observed-data Cox-Lasso penalty. Must be supplied together with `original_support`.
standardize	Passed to glmnet. The default is `TRUE`; set to `FALSE` when `x` has already been standardized or when reproducing an analysis that used externally standardized covariates.
alpha	Elastic-net mixing parameter for selection. Lasso (`alpha = 1`) is the primary evaluation setting. Other values are accepted with an inferential-scope warning.
type_measure	Cross-validation loss passed to `cv.glmnet()`.
ties	Ties method for the unpenalized Cox refits. The normal default is `"efron"`; exact `legacy_nki70` reproduction forces Breslow.
censoring	Bootstrap censoring model; currently `"reverse_km"`.
rng_mode	`"independent_streams"` for method-order-invariant random streams, or `"legacy_nki70"` only when reproducing the archived NKI70 run.
glmnet_args	Additional arguments forwarded to glmnet fits.
progress	Show bootstrap progress.

Details

Re-selection characterizes variability in the selection-and-refit procedure. It does not condition on the complete selection event, and its intervals should not automatically be interpreted as confidence intervals for a single fixed coefficient.

The default RNG mode uses separate deterministic streams for bootstrap-data generation and CV folds so that results for a named method do not depend on which other methods are requested. If `seed = NULL`, a root seed is drawn once at the start of the call and stored in the returned object, after which the same deterministic stream construction is used. `rng_mode = "legacy_nki70"` is retained solely for reproducibility of the archived NKI70 B=500 analysis.

Value

An object of class `selsurv_reselection` and `cox_reselection_bootstrap`.

See Also

`selSurv_workflows`

Examples

```
set.seed(2026)
n <- 120L
X <- matrix(stats::rnorm(n * 4L), nrow = n)
colnames(X) <- paste0("x", seq_len(ncol(X)))
event_time <- stats::rexp(n, rate = 0.12 * exp(X[, 1] - 0.7 * X[, 2]))
censor_time <- stats::rexp(n, rate = 0.06)
time <- pmin(event_time, censor_time)
status <- as.integer(event_time <= censor_time)
# Illustrative B only; increase it before interpreting intervals.
rs <- coxlasso_boot_reselect(
  x = X, time = time, status = status,
  B = 8, nfolds = 3,
  methods = c(
    "fixed_support", "reselect_fixed_lambda",
    "reselect_cv_min"
  ),
  conf_level = 0.95, seed = 2026, progress = FALSE
)
summary(rs)$diagnostics
```

`coxlasso_debias` *Approximate debiasing for a Cox-Lasso fit*

Description

Applies a one-step Cox score correction using a ridge-stabilized inverse information matrix. This research implementation does not supply a general high-dimensional validity guarantee. It requires a Lasso pilot (`fit$alpha = 1`); other penalties are rejected.

Usage

```
coxlasso_debias(fit, ridge = 1e-04)
```

Arguments

fit	A selSurv_fit object.
ridge	Ridge-stabilization term added to the information matrix before inversion.

Details

The score calculation warrants additional scrutiny for tied event times. Results from such data should be treated cautiously.

Value

A selSurv_debias object.

See Also

selSurv_workflows

---

coxlasso_efron_events *Event-level bootstrap for debiased Cox-Lasso score contributions*

Description

Event-level bootstrap for debiased Cox-Lasso score contributions

Usage

```
coxlasso_efron_events(db, B = 500, alpha = 0.1, seed = NULL)
```

Arguments

db	A selSurv_debias object.
B	Number of bootstrap draws.
alpha	Significance level.
seed	Optional random seed.

Details

The debiased estimator must originate from a Cox-Lasso fit with penalty mixing parameter alpha = 1.

Value

A selSurv_boot object.

`coxlasso_fit` *Fit a Cox-Lasso workflow*

Description

Fits a penalized Cox model with cross-validated tuning and stores the selected model. The default mixing parameter (`alpha = 1`) yields the Lasso.

Usage

```
coxlasso_fit(
  X,
  time,
  status,
  lambda_choice = c("min", "1se"),
  nfolds = 10,
  alpha = 1,
  standardize = TRUE,
  ...
)
```

Arguments

`X`	Numeric matrix or data frame of covariates. In data frames, numeric and integer columns remain numeric, while logical, factor, ordered-factor, and character columns are expanded using treatment coding with the first level as the reference. Ordered factors are treated as ordinary categorical predictors. Categorical predictors contribute $k-1$ separately penalized design-matrix coordinates; they are not selected as groups. To control the reference category, supply an explicit factor level order or use `relevel()` before fitting.
`time`	Follow-up time.
`status`	Event indicator coded 0/1.
`lambda_choice`	Either `"min"` or `"1se"`.
`nfolds`	Number of cross-validation folds.
`alpha`	Elastic-net mixing parameter passed to `glmnet`. `alpha = 1` gives the Lasso. The downstream inference workflows have been evaluated primarily for this setting.
`standardize`	Passed to `glmnet`.
`...`	Additional arguments passed to `glmnet::cv.glmnet()`.

Value

A `selSurv_fit` object.

Examples

```
set.seed(2026)
n <- 120L
X <- matrix(stats::rnorm(n * 4L), nrow = n)
colnames(X) <- paste0("x", seq_len(ncol(X)))
event_time <- stats::rexp(n, rate = 0.12 * exp(X[, 1] - 0.7 * X[, 2]))
censor_time <- stats::rexp(n, rate = 0.06)
time <- pmin(event_time, censor_time)
status <- as.integer(event_time <= censor_time)
fit <- coxlasso_fit(X, time, status, nfolds = 3, lambda_choice = "min")
fit$selected
```

`coxlasso_refit` *Refit Cox model on selected support*

Description

Refit Cox model on selected support

Usage

```
coxlasso_refit(
  fit,
  support = fit$support,
  X = fit$X,
  time = fit$time,
  status = fit$status
)
```

Arguments

`fit`	A `selSurv_fit` object.
`support`	Optional integer support. Defaults to `fit$support`.
`X`	Optional numeric matrix or data frame. After input normalization, its design-matrix columns must match `fit$X` exactly in names and order.
`time`	Optional time vector for bootstrap data.
`status`	Optional status vector for bootstrap data.

Value

A list with support-specific and zero-padded estimates.

coxlasso_split *Split-sample inference after Cox-Lasso selection*

Description

Uses one subsample to choose predictors and a disjoint subsample to estimate coefficients and form Wald intervals.

Usage

```
coxlasso_split(
  X,
  time,
  status,
  split_fraction = 0.5,
  seed = NULL,
  lambda_choice = c("min", "1se"),
  nfolds = 10,
  conf_level = 0.95,
  penalty_alpha = 1,
  ...
)
```

Arguments

X	Numeric matrix or data frame of covariates. In data frames, numeric and integer columns remain numeric, while logical, factor, ordered-factor, and character columns are expanded using treatment coding with the first level as the reference. Ordered factors are treated as ordinary categorical predictors. Categorical predictors contribute $k-1$ separately penalized design-matrix coordinates; they are not selected as groups. To control the reference category, supply an explicit factor level order or use `relevel()` before fitting.
time	Follow-up time.
status	Event indicator coded 0/1.
split_fraction	Fraction of observations used for selection.
seed	Optional random seed.
lambda_choice	Cox-Lasso tuning rule used in the selection split.
nfolds	Number of folds for the selection split.
conf_level	Confidence level for the split-sample Wald intervals. Must be strictly between zero and one.
penalty_alpha	Elastic-net mixing parameter used for penalized Cox screening in the selection split. Passed as `alpha` to `coxlasso_fit()`. Values other than 1 are computationally supported but trigger a warning because other penalties have not been assessed separately in the inference workflows.

...	Additional arguments passed to `coxlasso_fit()`. For backward compatibility, a named `alpha` argument is interpreted as the deprecated confidence-interval significance level; use `conf_level` instead.

Value

A `selSurv_split` object.

`coxlasso_wildboot` *Multiplier wild bootstrap for debiased Cox-Lasso estimator*

Description

Multiplier wild bootstrap for debiased Cox-Lasso estimator

Usage

```
coxlasso_wildboot(
  db,
  B = 500,
  alpha = 0.1,
  type = c("percentile"),
  seed = NULL
)
```

Arguments

db	A `selSurv_debias` object.
B	Number of bootstrap draws.
alpha	Significance level.
type	Interval type, currently percentile on the bootstrap debiased scale.
seed	Optional random seed.

Details

The debiased estimator must originate from a Cox-Lasso fit with penalty mixing parameter `alpha = 1`.

Value

A `selSurv_boot` object.

`plot_selection_diagnostics` *Plot selection diagnostics across bootstrap workflows*

Description

Displays re-selection or zero frequencies as dot plots or grouped bars.

Supply a combined selection-diagnostics table (see `compare_selection_diagnostics()`).

Usage

```
plot_selection_diagnostics(
  diagnostics,
  metric = c("reselect", "zero"),
  plot_type = c("dot", "bar"),
  workflows = NULL,
  sort_by = NULL,
  decreasing = TRUE,
  palette = c("color", "grey"),
  ylim = c(0, 1),
  xlim = c(0, 1),
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  legend = TRUE,
  legend_position = c(
    "top_panel", "inside", "none", "side"
  ),
  colors = NULL,
  pch = c(16, 17, 15, 18),
  cex_points = 1.1,
  cex_names = 0.85,
  cex_legend = 0.85,
  las = 2,
  ...
)
```

Arguments

`diagnostics`	A data frame returned by `compare_selection_diagnostics()`.
`metric`	Which quantity to plot: `"reselect"` or `"zero"`.
`plot_type`	Plot type. `"dot"` gives a horizontal dot plot; `"bar"` gives grouped bars.
`workflows`	Character vector selecting workflows to show.
`sort_by`	Optional column name used to order terms.
`decreasing`	Logical; should sorting be decreasing?

palette	Color palette. "color" gives a colorblind-friendly palette; "grey" gives neutral greys.
ylim	Optional y-axis limits for bar plots.
xlim	Optional x-axis limits for dot plots.
main	Optional plot title.
xlab	Optional x-axis label for dot plots.
ylab	Optional y-axis label for bar plots.
legend	Logical; add a legend?
legend_position	Legend placement. "top_panel" uses a separate legend panel above the plot and is robust in the RStudio zoom window.
colors	Optional vector of colors. Overrides palette.
pch	Point symbols for dot plots.
cex_points	Point size for dot plots.
cex_names	Axis-label size for term labels.
cex_legend	Legend text size.
las	Axis label orientation for bar plots.
...	Additional graphical arguments passed to plotting functions.

Value

Invisibly returns the matrix used for plotting.

---

print-methods *Print methods for selSurv objects*

Description

Print methods for objects returned by coxlasso_fit(), coxlasso_boot_parametric(), coxlasso_debias(), and coxlasso_split().

Usage

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

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

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

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

Arguments

x	An object returned by a selSurv workflow function.
...	Additional arguments, currently ignored.

Value

The input object x, invisibly.

---

selSurv_methods *List implemented package methods*

Description

List implemented package methods

Usage

```
selSurv_methods()
```

Value

A data frame with methods, targets, and status.

---

selSurv_targets *Explain the implemented inferential targets*

Description

Explain the implemented inferential targets

Usage

```
selSurv_targets()
```

Value

A data frame describing the package workflows and targets.

`selSurv_workflows` *Choosing an inference method in selSurv*

Description

A guide to inference after Cox-Lasso variable selection.

Start with `coxlasso_fit()`. Subsequent methods differ in the coefficient or procedure they describe and in whether variable selection is repeated during resampling.

Details

These methods address different inferential questions. An interval's interpretation depends on the target and the procedure used to construct it.

Which function should I use?

- **Approximate full-model-coordinate inference.** Use `coxlasso_debias()` for a ridge-stabilized one-step correction. Optionally use `coxlasso_boot()` with `inference = "debiased"` and either `type = "wild"` or `type = "efron"` for a score-level bootstrap.

  This is an approximate, research-stage method. Neither the one-step correction nor the bootstrap alone establishes general high-dimensional confidence-interval validity.
- **Fixed-support model-based bootstrap.** Use `coxlasso_boot()` with `inference = "conditional"`. The default bootstrap type is `"model_based"`. The original selected support is held fixed throughout bootstrap refitting.

  In this interface, "conditional" means that the *observed* support is kept fixed. No conditioning on the complete selection event is performed.
- **Procedure-level re-selection.** Use `coxlasso_boot_reselect()` to compare fixed-support refitting with repeated Cox-Lasso selection. Available procedures include:
  - `"fixed_support"`: retain the original support;
  - `"reselect_fixed_lambda"`: reselect with a fixed penalty;
  - `"reselect_cv_min"`: reselect using CV-min;
  - `"reselect_cv_1se"`: reselect using the one-SE rule;
  - `"reselect_aic"`: reselect by AIC.

  The reported bootstrap vectors assign zero to unselected coordinates; failures remain missing. Selection frequencies and model overlap describe stability. Resulting intervals need not have standard fixed-parameter coverage interpretations.

Relation to the separate parametric interface

`coxlasso_boot_parametric()` provides a separate model-based interface, including an option to retune the penalty. Its arguments, interval construction, and results differ from those of the recommended interfaces above. Use it when its specific procedure is needed; the functions are not aliases.

Meaning of alpha and confidence-level arguments

- In `coxlasso_fit()`, `alpha` is the elastic-net mixing parameter. `alpha = 1` corresponds to Lasso.
- In `coxlasso_boot_reselect()`, `alpha` is also the mixing parameter.
- In `coxlasso_split()`, `penalty_alpha` is the mixing parameter; `conf_level` specifies the confidence level.
- In `coxlasso_boot_parametric()`, `alpha` denotes the significance level, rather than the mixing parameter.
- The lower-level score bootstrap functions also use `alpha` for the significance level.

Use named arguments when moving between these interfaces.

Scope and tied event times

The methods have been evaluated primarily for Lasso selection (`alpha = 1`). `coxlasso_fit()` permits other mixing parameters. The debiasing and score-bootstrap procedures require a Lasso pilot. Other inference routes may run with non-Lasso penalties, but those settings have not been evaluated separately.

Selection and unpenalized refitting can use different methods for handling tied event times. In particular, the current one-step score calculation has not been established to agree with the intended Cox score when failures are tied. Interpret debiased results for tied times with caution.

The Efron *event-score bootstrap* resamples event contributions; it should not be confused with the Efron partial-likelihood approximation for tied event times.

See Also

- `coxlasso_fit()`
- `coxlasso_debias()`
- `coxlasso_boot()`
- `coxlasso_boot_reselect()`
- `coxlasso_boot_parametric()`
- `coxlasso_split()`

Index

∗ **package**
selSurv_workflows, 17

compare_selection_diagnostics, 2
coxlasso_boot, 3
coxlasso_boot(), *6*, *17, 18*
coxlasso_boot_parametric, 5
coxlasso_boot_parametric(), *17, 18*
coxlasso_boot_reselect, 6
coxlasso_boot_reselect(), *6*, *17, 18*
coxlasso_debias, 8
coxlasso_debias(), *4*, *17, 18*
coxlasso_efron_events, 9
coxlasso_fit, 10
coxlasso_fit(), *17, 18*
coxlasso_refit, 11
coxlasso_split, 12
coxlasso_split(), *18*
coxlasso_wildboot, 13

plot_selection_diagnostics, 14
print-methods, 15
print.selSurv_boot *(*print-methods*)*, 15
print.selSurv_debias *(*print-methods*)*, 15
print.selSurv_fit *(*print-methods*)*, 15
print.selSurv_split *(*print-methods*)*, 15

selSurv-package, 2
selSurv_methods, 16
selSurv_targets, 16
selSurv_workflows, *4*, *6–9*, 17