Package: ssurvdr (via r-universe)

October 22, 2026

Title Sparse Survival Dimension Reduction for High-Dimensional Censored Data

Version 0.1.0

Description Implements sparse survival sufficient dimension reduction methods for high-dimensional censored data, including covariance-stabilised SAVE, Maximum Entropy Covariance estimation, information-complexity-guided variable weighting, dual-space high-dimensional SAVE computation, row-wise post-estimation sparsification paths, and Cox proportional hazards model selection.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Depends R (>= 4.1.0)

Imports stats, MASS, survival

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/ilovemaths/ssurvdr

BugReports https://github.com/ilovemaths/ssurvdr/issues

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

Date/Publication 2026-10-11 19:04:00 UTC

RemoteUrl https://github.com/ilovemaths/ssurvdr

RemoteRef HEAD

RemoteSha bc2d82a71e0c356c758a040888927aa0865b2827

Contents

compute_c1f_weights 2
compute_icomp 3
evaluate_ssurvdr 4
extract_sdr_basis 5
fit_reduced_cox 6
fit_ssurvdr 7
fit_ssurvdr_grid 8
fit_ssurvdr_sparse_path 9
mec_covariance 11
run_ssurvdr_simulation 12
save_kernel 13
screen_survival_variables 14
select_variables 15
simulate_ssurvdr 15
sparse_projection 17
sparsify_sdr_basis 18
summarise_ssurvdr 19
survival_slicing 19

Index **21**

`compute_c1f_weights` *Compute basis-adaptive variable weights*

Description

Computes adaptive variable weights using row-wise initial SDR signal strength and covariance-complexity adjustment. The function is intended for weighted sparse projection in high-dimensional sufficient dimension reduction.

Usage

```
compute_c1f_weights(
  basis_fit,
  matrix = NULL,
  alpha = 1,
  gamma = 1,
  eps = 1e-08,
  normalise = TRUE,
  cap = NULL
)
```

Arguments

basis_fit	Object returned by extract_sdr_basis(), an ssurvdr_sparse_projection object, or a numeric basis matrix.
matrix	Optional covariance, SAVE, or information matrix used to compute eigenvalue complexity.
alpha	Non-negative complexity adjustment parameter.
gamma	Positive exponent applied to the inverse initial signal.
eps	Small positive stabilisation constant.
normalise	Logical. If TRUE, weights are normalised to have mean 1.
cap	Optional positive upper cap for weights.

Details

The returned c1f field is retained for compatibility. Its implemented formula is (p / 2) * log(arithmetic_mean / geometric_mean) of the stabilised eigenvalues, the same C1 expression used by compute_icomp().
The variable weights use row norms and a separate loading-weighted eigenvalue-dispersion score; the scalar c1f is a diagnostic and does not directly enter the weights. With matrix = NULL, the complexity scores are zero and c1f is missing. Finite weights are strictly positive, with no universal upper bound when cap = NULL. Normalisation precedes optional capping.

Value

A list containing adaptive weights, initial signal scores, complexity scores, and diagnostics.

Examples

```
basis <- matrix(c(1, 2, 4), ncol = 1)
weights <- compute_c1f_weights(basis, matrix = diag(c(1, 2, 4)))
weights$weights
weights$c1f
```

---

compute_icomp *Compute information criteria for reduced Cox model*

Description

Computes AIC, BIC, CAIC, ICOMP(IFIM), and ICOMP(misspec) for a reduced Cox proportional hazards model fitted after sufficient dimension reduction.

Usage

```
compute_icomp(cox_fit, eps = 1e-08)
```

Arguments

`cox_fit`	Object returned by `fit_reduced_cox()`.
`eps`	Small positive stabilisation constant for eigenvalues.

Value

A list containing information criteria, covariance-complexity diagnostics, and likelihood quantities.

`evaluate_ssurvdr` *Evaluate ssurvdr fitted model against known simulation truth*

Description

Evaluates selected variables, subspace recovery, and survival prediction performance when the true active set and true SDR basis are known.

Usage

```
evaluate_ssurvdr(
  fit,
  simulation,
  variable_selection = NULL,
  top = NULL,
  threshold = 1e-06,
  screen_indices = NULL
)
```

Arguments

`fit`	Object returned by `fit_ssurvdr()` or `fit_ssurvdr_grid()`.
`simulation`	Object returned by `simulate_ssurvdr()`.
`variable_selection`	Optional object returned by `select_variables()`. If NULL, variable selection is computed from the fitted basis.
`top`	Optional number of variables to select by row-norm ranking.
`threshold`	Numeric threshold for row-norm selection when top is NULL.
`screen_indices`	Optional integer vector giving the original predictor indices retained after screening. If supplied, the estimated basis is expanded back to the original predictor dimension before projection distance is computed. If `variable_selection` is supplied, its selected indices must already refer to the original predictor matrix.

Value

A list containing variable-selection metrics, projection distance, C-index, selected variables, and diagnostics.

extract_sdr_basis *Extract sufficient dimension reduction basis*

Description

Extracts leading sufficient dimension reduction directions from an object returned by `save_kernel()`. The function reuses the eigensystem already computed by save_kernel() and therefore avoids a redundant eigendecomposition.

Usage

```
extract_sdr_basis(
  save_fit,
  d = 1,
  standardise = TRUE,
  sort_eigenvalues = TRUE,
  store_projection = NULL
)
```

Arguments

`save_fit`	Object returned by `save_kernel()`.
`d`	Structural dimension to extract.
`standardise`	Logical. If TRUE, normalise basis vectors to unit norm.
`sort_eigenvalues`	Logical. If TRUE, sort eigenvalues in decreasing order.
`store_projection`	Logical or NULL. If TRUE, store B B'. If NULL, dual fits never store the dense p x p projection matrix; primal fits store it only when p <= 2000. This prevents dual-space fits from reintroducing a dense high-dimensional allocation after basis extraction.

Details

For a dual-space SAVE fit, estimation is restricted to the observed data span. This removes non-identifiable null-space directions that arise when p > n.

Value

A list containing SDR basis vectors, eigenvalues, optional projection matrix, preprocessing metadata, and diagnostics.

`fit_reduced_cox` *Fit reduced Cox proportional hazards model*

Description

Fits a Cox proportional hazards model using sufficient dimension reduction scores obtained from SDR basis vectors.

Usage

```
fit_reduced_cox(
  X,
  time,
  status,
  basis_fit,
  ties = "efron",
  x = TRUE,
  y = TRUE,
  model = TRUE
)
```

Arguments

`X`	Numeric predictor matrix with observations in rows.
`time`	Numeric vector of observed survival times.
`status`	Numeric or integer censoring indicator vector.
`basis_fit`	Object returned by `extract_sdr_basis()`.
`ties`	Character string specifying ties handling method for coxph.
`x`	Logical. Passed to survival::coxph().
`y`	Logical. Passed to survival::coxph().
`model`	Logical. Passed to survival::coxph().

Value

A list containing the reduced predictors, fitted Cox model, coefficients, linear predictors, likelihood statistics, and diagnostics.

fit_ssurvdr — *Fit sparse survival dimension reduction model*

Description

Fits the current end-to-end ssurvdr workflow: survival-informed slicing, covariance-stabilised survival SAVE kernel construction, SDR basis extraction, reduced Cox proportional hazards modelling, and information-criterion scoring.

Usage

```
fit_ssurvdr(
  X,
  time,
  status,
  d = 1,
  n_slices = 5,
  min_slice_size = 2,
  covariance = c("mec", "sample"),
  rho = 0.5,
  alpha = 0.05,
  eps = 1e-08,
  centre = TRUE,
  scale = FALSE,
  ties = "efron",
  backend = c("auto", "primal", "dual")
)
```

Arguments

`X`	Numeric predictor matrix with observations in rows.
`time`	Numeric vector of observed survival times.
`status`	Numeric or integer censoring indicator vector.
`d`	Structural dimension.
`n_slices`	Number of survival-informed slices within each censoring group.
`min_slice_size`	Minimum number of observations allowed in a slice.
`covariance`	Character string specifying covariance estimator. Currently supports "sample" and "mec".
`rho`	MEC convex mixing weight.
`alpha`	MEC diagonal stabilisation weight.
`eps`	Small positive stabilisation constant.
`centre`	Logical. If TRUE, centre predictors in SAVE construction.
`scale`	Logical. If TRUE, scale predictors in SAVE construction.
`ties`	Character string specifying ties handling method for Cox model.
`backend`	Computational backend passed to `save_kernel()`.

Value

A list containing slicing output, SAVE kernel, SDR basis, reduced Cox model, information criteria, and diagnostics.

Examples

```
dat <- simulate_ssurvdr(n = 100, p = 12, active = 1:3, seed = 2026)
fit <- fit_ssurvdr(dat$X, dat$time, dat$status, d = 2,
                   n_slices = 3, covariance = "mec", backend = "primal")
fit$criteria
```

---

fit_ssurvdr_grid *Fit ssurvdr model over a tuning grid*

Description

Fits the ssurvdr workflow over candidate covariance estimators and structural dimensions, then returns model-comparison results using AIC, BIC, CAIC, ICOMP(IFIM), ICOMP(misspec), C-index, runtime, and condition number.

Usage

```
fit_ssurvdr_grid(
  X,
  time,
  status,
  d_grid = 1:3,
  covariance_grid = c("mec", "sample"),
  n_slices = 5,
  min_slice_size = 2,
  rho = 0.5,
  alpha = 0.05,
  eps = 1e-08,
  centre = TRUE,
  scale = FALSE,
  ties = "efron",
  selector = c("ICOMP_IFIM", "ICOMP_MISSPEC", "AIC", "BIC", "CAIC"),
  continue_on_error = TRUE,
  backend = c("auto", "primal", "dual"),
  reuse_kernel = TRUE
)
```

Arguments

X	Numeric predictor matrix with observations in rows.
time	Numeric vector of observed survival times.

status	Numeric or integer censoring indicator vector.
d_grid	Integer vector of candidate structural dimensions.
covariance_grid	Character vector of covariance estimators. Currently supports "mec" and "sample".
n_slices	Number of survival-informed slices within each censoring group.
min_slice_size	Minimum number of observations allowed in a slice.
rho	MEC convex mixing weight.
alpha	MEC diagonal stabilisation weight.
eps	Small positive stabilisation constant.
centre	Logical. If TRUE, centre predictors in SAVE construction.
scale	Logical. If TRUE, scale predictors in SAVE construction.
ties	Character string specifying ties handling method for Cox model.
selector	Criterion used to rank models. One of "ICOMP_IFIM", "ICOMP_MISSPEC", "AIC", "BIC", or "CAIC".
continue_on_error	Logical. If TRUE, failed candidate models are recorded and the grid search continues.
backend	Computational backend passed to `save_kernel()`.
reuse_kernel	Logical. If TRUE, compute the SAVE operator once per covariance estimator and reuse it across d. FALSE retains the legacy fit-by-fit behaviour and is mainly intended for regression testing.

Details

The survival slices and SAVE operator are computed once per covariance estimator and reused across structural dimensions, avoiding redundant kernel construction and eigendecomposition.

Value

A list containing all fitted models, a model-comparison table, the best model, cached SAVE operators, and diagnostics.

`fit_ssurvdr_sparse_path`

Fit an ICOMP-guided row-sparsity path

Description

Evaluates a sequence of row-sparse SDR bases obtained from a previously fitted dense SDR basis. Predictor rows are ranked by their Euclidean row norms, the leading k rows are retained for each candidate support size, and the retained basis is re-orthonormalised before refitting the reduced Cox model.

Usage

```
fit_ssurvdr_sparse_path(
  X,
  time,
  status,
  dense_fit,
  support_grid = NULL,
  selector = c("ICOMP_IFIM", "ICOMP_MISSPEC", "AIC", "BIC", "CAIC"),
  ties = "efron",
  eps = 1e-08,
  variable_names = NULL,
  rank_tol = 1e-10,
  continue_on_error = TRUE,
  store_fits = FALSE
)
```

Arguments

`X`	Numeric predictor matrix with observations in rows.
`time`	Numeric vector of observed survival times.
`status`	Numeric or integer censoring indicator vector.
`dense_fit`	Either an `ssurvdr_fit` object or an `ssurvdr_basis` object representing the dense SDR solution whose row norms define the support path.
`support_grid`	Integer vector of candidate support sizes. If NULL, a deterministic default grid spanning from d to p is used.
`selector`	Criterion used to rank the support path. One of `ICOMP_IFIM`, `ICOMP_MISSPEC`, `AIC`, `BIC`, or `CAIC`.
`ties`	Character string specifying ties handling for the Cox model.
`eps`	Small positive stabilisation constant passed to `compute_icomp()`.
`variable_names`	Optional character vector of predictor names.
`rank_tol`	Positive numerical rank tolerance passed to `sparsify_sdr_basis()`.
`continue_on_error`	Logical. If TRUE, invalid support sizes are recorded and the remaining path continues.
`store_fits`	Logical. If TRUE, retain all successful sparse Cox fits. Otherwise only the selected fit and path table are returned.

Details

Model comparison uses the existing Cox-model criteria returned by `compute_icomp()`. In particular, `ICOMP_IFIM` and `ICOMP_MISSPEC` remain the package's established information-complexity criteria. No additional support- size penalty is silently added. Consequently, the returned path should be inspected for boundary selection and compared with the dense-support candidate.

This procedure is a post-estimation row-sparsification path. It is not a group-lasso, Fantope, or penalised SAVE estimator.

Value

An object of class `ssurvdr_sparse_path` containing the support-path results, selected support, best sparse support excluding the fully dense candidate, and the selected fitted model.

---

mec_covariance *Maximum Entropy Covariance Estimator*

Description

Computes the Maximum Entropy Covariance Estimator (MEC) from a predictor matrix and survival-informed slice membership. The estimator combines a pooled within-slice covariance matrix with the entropy-maximising slice covariance matrix, with optional diagonal stabilisation.

Usage

```
mec_covariance(
  X,
  slices,
  rho = 0.5,
  alpha = 0.05,
  eps = 1e-08,
  centre = TRUE,
  scale = FALSE
)
```

Arguments

`X`	Numeric matrix or data frame of predictors with observations in rows.
`slices`	Either an object returned by `survival_slicing()` or an integer vector of slice memberships.
`rho`	Convex mixing weight for the entropy-selected covariance matrix. Must lie between 0 and 1, inclusive.
`alpha`	Optional diagonal stabilisation weight. Must lie between 0 and 1, inclusive.
`eps`	Small positive constant used for eigenvalue and logarithmic stabilisation.
`centre`	Logical. If TRUE, centre columns of X before covariance estimation.
`scale`	Logical. If TRUE, scale columns of X before covariance estimation.

Value

A list containing the MEC covariance matrix, pooled covariance matrix, entropy-selected covariance matrix, entropy scores, selected slice, and diagnostic quantities.

run_ssurvdr_simulation

Run Monte Carlo simulation for ssurvdr

Description

Runs repeated simulation experiments for sparse survival dimension reduction under low-dimensional, borderline-dimensional, high-dimensional, and ultra-high-dimensional settings.

Usage

```
run_ssurvdr_simulation(
  n = 120,
  p = 30,
  active = 1:5,
  covariance = c("identity", "ar1", "compound", "block"),
  rho = 0.5,
  signal_strength = 1.2,
  censoring_rate = 0.08,
  n_rep = 50,
  screening = TRUE,
  screen_top = NULL,
  covariance_methods = c("mec", "sample"),
  d_grid = 1:3,
  n_slices = 4,
  selector = c("ICOMP_IFIM", "ICOMP_MISSPEC", "AIC", "BIC", "CAIC"),
  variable_top = NULL,
  seed = NULL,
  continue_on_error = TRUE
)
```

Arguments

`n`	Sample size.
`p`	Number of predictors.
`active`	Integer vector of active predictor indices.
`covariance`	Data-generating covariance structure.
`rho`	Correlation parameter.
`signal_strength`	Survival signal strength.
`censoring_rate`	Exponential censoring rate.
`n_rep`	Number of Monte Carlo replications.
`screening`	Logical. If TRUE, univariate Cox screening is performed.
`screen_top`	Number of variables retained after screening.

`covariance_methods`
: Covariance methods used in fit_ssurvdr_grid().

`d_grid` Candidate structural dimensions.

`n_slices` Number of survival-informed slices.

`selector` Model selection criterion.

`variable_top` Number of variables selected for evaluation.

`seed` Optional random seed.

`continue_on_error`
: Logical. If TRUE, failed replications are recorded.

Value

A list containing replication-level results, summary results, and simulation settings.

`save_kernel` *Compute survival SAVE kernel matrix*

Description

Computes the covariance-stabilised survival SAVE operator for sufficient dimension reduction in censored survival data. For very high-dimensional problems, a compact dual-space backend avoids materialising dense p x p covariance and SAVE matrices and restricts estimation to the observed data span.

Usage

```
save_kernel(
  X,
  slices,
  covariance = c("sample", "mec"),
  rho = 0.5,
  alpha = 0.05,
  eps = 1e-08,
  centre = TRUE,
  scale = FALSE,
  backend = c("auto", "primal", "dual")
)
```

Arguments

`X` Numeric predictor matrix with observations in rows.

`slices` Either an object returned by `survival_slicing()` or an integer vector of slice memberships.

`covariance` Character string specifying covariance estimator. Currently supports "sample" and "mec".

rho	MEC shrinkage parameter.
alpha	MEC diagonal stabilisation parameter.
eps	Small positive stabilisation constant.
centre	Logical. If TRUE, centre predictors.
scale	Logical. If TRUE, scale predictors.
backend	Computational backend. "primal" reproduces the dense p x p implementation. "dual" uses a compact row-space representation. "auto" uses the dual backend when p > max(n, 1000) and the primal backend otherwise.

Value

A list containing the SAVE representation, eigensystem, preprocessing metadata, and diagnostics.

---

screen_survival_variables

Screen variables using univariate Cox models

Description

Performs preliminary survival variable screening using univariate Cox proportional hazards models. This function is intended as a high-dimensional screening layer before survival SAVE, MEC covariance estimation, and sparse SDR modelling.

Usage

```
screen_survival_variables(
  X,
  time,
  status,
  top = NULL,
  p_value = NULL,
  min_events = 5,
  ties = "efron"
)
```

Arguments

X	Numeric predictor matrix with observations in rows.
time	Numeric vector of observed survival times.
status	Numeric or integer censoring indicator vector.
top	Optional number of top variables to retain.
p_value	Optional p-value threshold for screening.
min_events	Minimum number of observed events required.
ties	Ties handling method passed to survival::coxph().

Value

A list containing screening scores, selected variables, selected indices, and diagnostics.

---

select_variables *Select variables from SDR basis*

Description

Selects variables using row norms of the estimated sufficient dimension reduction basis matrix.

Usage

```
select_variables(
  basis_fit,
  threshold = 1e-06,
  top = NULL,
  variable_names = NULL
)
```

Arguments

basis_fit	Object returned by extract_sdr_basis() or a numeric basis matrix.
threshold	Numeric threshold for variable selection.
top	Optional integer. If supplied, returns the top variables by row norm.
variable_names	Optional character vector of predictor names.

Value

A list containing variable scores, selected variables, selected indices, and selection diagnostics.

---

simulate_ssurvdr *Simulate high-dimensional censored survival data for ssurvdr*

Description

Simulates predictors, sparse true SDR direction, event times, censoring times, observed survival times, and censoring indicators for testing sparse survival dimension reduction methods.

Usage

```
simulate_ssurvdr(
  n = 100,
  p = 50,
  active = 1:5,
  beta = NULL,
  covariance = c("identity", "ar1", "compound", "block"),
  rho = 0.5,
  block_size = 10,
  signal_strength = 1,
  baseline_hazard = 0.1,
  censoring_rate = 0.1,
  standardise_X = TRUE,
  seed = NULL
)
```

Arguments

`n`	Sample size.
`p`	Number of predictors.
`active`	Integer vector of active predictor indices.
`beta`	Optional numeric vector of active coefficients. If NULL, equal coefficients are used.
`covariance`	Covariance structure. One of "identity", "ar1", "compound", or "block".
`rho`	Correlation parameter for AR(1), compound, or block covariance.
`block_size`	Block size for block covariance.
`signal_strength`	Multiplicative signal strength.
`baseline_hazard`	Exponential baseline hazard rate.
`censoring_rate`	Rate parameter for exponential censoring distribution.
`standardise_X`	Logical. If TRUE, predictors are centred and scaled.
`seed`	Optional random seed.

Value

A list containing simulated predictors, survival outcome, true basis, true active set, covariance matrix, and simulation metadata.

sparse_projection *Sparse projection from SAVE kernel*

Description

Computes sparse sufficient dimension reduction directions from a SAVE kernel using weighted soft-thresholding followed by eigendecomposition.

Usage

```
sparse_projection(
  save_fit,
  d = 1,
  lambda = 0.01,
  weights = NULL,
  threshold = 1e-06,
  standardise = TRUE
)
```

Arguments

save_fit	Object returned by save_kernel().
d	Structural dimension.
lambda	Non-negative sparsity parameter.
weights	Optional non-negative variable weights of length p.
threshold	Final row-norm threshold for exact zeroing of weak rows.
standardise	Logical. If TRUE, normalise sparse directions to unit norm.

Details

This function thresholds entries of a primal SAVE kernel before extracting directions. It requires a dense primal kernel. For a row-support path using a fitted primal or dual basis, use fit_ssurvdr_sparse_path().

Value

A list containing sparse basis, sparse projection matrix, selected variables, thresholded kernel, and diagnostics.

sparsify_sdr_basis *Row-sparsify an SDR basis*

Description

Constructs a row-sparse sufficient dimension reduction basis by retaining the predictors with the largest row norms from an existing SDR basis. The retained columns are re-orthonormalised without altering the exact zero pattern of the excluded predictor rows.

Usage

```
sparsify_sdr_basis(
  basis_fit,
  support_size,
  variable_names = NULL,
  rank_tol = 1e-10
)
```

Arguments

basis_fit	Object returned by extract_sdr_basis() or another object inheriting from ssurvdr_basis.
support_size	Positive integer number of predictor rows to retain. Must be at least the structural dimension and no larger than p.
variable_names	Optional character vector of predictor names. If omitted, row names of the basis are used, followed by V1, ..., Vp if needed.
rank_tol	Positive numerical tolerance used to verify that the retained rows still span the requested structural dimension.

Details

This function is deliberately a post-estimation row-sparsification operator. It does not claim to solve a penalised SAVE optimisation problem.

Value

An object inheriting from ssurvdr_sparse_basis and ssurvdr_basis. It contains the row-sparse orthonormal basis, selected predictor indices, row-norm ranking, retained basis energy, and subspace diagnostics.

summarise_ssurvdr *Summarise ssurvdr model output*

Description

Produces clean summary tables from objects returned by fit_ssurvdr(), fit_ssurvdr_grid(), and select_variables().

Usage

```
summarise_ssurvdr(object, top = 10)
```

Arguments

object	An object returned by fit_ssurvdr(), fit_ssurvdr_grid(), or select_variables().
top	Number of top variables to return when summarising a variable selection object.

Value

A list of summary tables.

survival_slicing *Construct survival-informed slices for censored survival data*

Description

This function partitions survival observations by censoring status and then slices observed survival times within each censoring group. It is intended for sufficient dimension reduction methods for censored survival data.

Usage

```
survival_slicing(
  time,
  status,
  n_slices = 5,
  min_slice_size = 2,
  drop_empty = TRUE
)
```

Arguments

`time`	A numeric vector of observed survival times.
`status`	A numeric, integer, or logical vector of censoring indicators. Conventionally, 1 indicates observed event and 0 indicates censoring.
`n_slices`	Number of time slices to construct within each censoring group.
`min_slice_size`	Minimum number of observations allowed in a slice.
`drop_empty`	Logical. If TRUE, empty slices are dropped.

Value

A list with the slice membership, slice labels, slice table, and basic metadata.

Index

compute_c1f_weights, 2
compute_icomp, 3
compute_icomp(), *3*, *10*

evaluate_ssurvdr, 4
extract_sdr_basis, 5
extract_sdr_basis(), *3*, *6*, *15*, *18*

fit_reduced_cox, 6
fit_reduced_cox(), *4*
fit_ssurvdr, 7
fit_ssurvdr(), *4*
fit_ssurvdr_grid, 8
fit_ssurvdr_grid(), *4*
fit_ssurvdr_sparse_path, 9
fit_ssurvdr_sparse_path(), *17*

mec_covariance, 11

run_ssurvdr_simulation, 12

save_kernel, 13
save_kernel(), *5*, *7*, *9*, *17*
screen_survival_variables, 14
select_variables, 15
select_variables(), *4*
simulate_ssurvdr, 15
simulate_ssurvdr(), *4*
sparse_projection, 17
sparsify_sdr_basis, 18
summarise_ssurvdr, 19
survival_slicing, 19
survival_slicing(), *11*, *13*