

Package ‘risdr’

October 12, 2026

Title Regularised and Information-Theoretic Sufficient Dimension Reduction

Version 0.4.0

Description Implements covariance-stabilised sufficient dimension reduction for continuous responses, with an additional sparse engine for continuous, categorical, and right-censored survival responses and information-theoretic structural dimension selection. Supported methods include sliced inverse regression, sliced average variance estimation, directional regression, and principal Hessian directions. Sample, ridge, Oracle Approximating Shrinkage, Ledoit-Wolf, and Maximum Entropy Covariance (MEC) estimators are provided alongside prediction, resampling, simulation, and diagnostic utilities. The sufficient dimension reduction methods build on Li (1991) <[doi:10.1080/01621459.1991.10475035](https://doi.org/10.1080/01621459.1991.10475035)>, Li (1992) <[doi:10.1080/01621459.1992.10476258](https://doi.org/10.1080/01621459.1992.10476258)>, and Li and Wang (2007) <[doi:10.1198/016214507000000536](https://doi.org/10.1198/016214507000000536)>. Covariance shrinkage follows Olorede and Yahya (2019) <[doi:10.48550/arXiv.1909.13017](https://doi.org/10.48550/arXiv.1909.13017)>, Ledoit and Wolf (2004) <[doi:10.1016/S0047-259X\(03\)00096-4](https://doi.org/10.1016/S0047-259X(03)00096-4)> and Chen et al. (2010) <[doi:10.1109/TSP.2010.2053029](https://doi.org/10.1109/TSP.2010.2053029)>.

License GPL (>= 3)

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

Imports stats, graphics, utils, MASS, Matrix, survival, nnet

Suggests dplyr, knitr, readr, rmarkdown, testthat (>= 3.0.0), VIM, withr

Config/testthat/edition 3

Config/Needs/coverage covr, xml2

Config/Needs/website r-lib/pkgdown

VignetteBuilder knitr, rmarkdown

URL <https://github.com/ilovemaths/risdr>,
<https://ilovemaths.github.io/risdr/>

BugReports <https://github.com/ilovemaths/risdr/issues>

Config/roxygen2/version 8.1.0

RoxygenNote 7.3.3

NeedsCompilation no

Author Kabir Olorede [aut, cre]

Maintainer Kabir Olorede <kabirolorede@gmail.com>

Repository CRAN

Date/Publication 2026-10-12 19:10:16 UTC

Contents

risdr-package	4
adjusted_r_squared	5
arl_covariance	5
C1F	6
choose_dimension	6
compute_dr	7
compute_ic	8
compute_ic_values	9
compute_information_criteria	10
compute_penalty_weights	10
compute_phd	11
compute_save	12
compute_scores	13
compute_sdr	13
compute_sir	14
cov_complexity_C1	15
cov_complexity_C1F	16
cov_condition_number	16
cov_diagnostics	17
cov_effective_rank	17
cov_lw	18
cov_mec	18
cov_oas	19
cov_ridge	20
cov_sample	21
criterion_weights	21
dual_pca_scores	22
estimate_cov	23
evaluate_prediction	24
evaluate_prediction_cv	24
extract_loadings	26
filter_low_variance	26
fit_downstream_lm	27
fit_risdr	27

fit_risdr_dual	29
fit_risdr_realdata	31
fit_risdr_sparse	32
fit_risdr_sparse_realdata	34
fit_sparse_grid	35
mae	36
make_cv_folds	37
make_slices	37
mape	38
plot_cv_dimension	38
plot_dimension_selection	39
plot_ladle	39
plot_loadings	40
plot_prediction	40
plot_residuals	41
plot_scee	41
plot_sufficient	42
plot_sufficient2d	42
predict.risdr	43
predict.risdr_sparse	44
prediction_correlation	45
predict_downstream_lm	45
prepare_survival_response	46
print.risdr	46
print.risdr_dual	47
print.risdr_realdata	47
print.risdr_sparse	48
print.risdr_sparse_realdata	48
print.summary.risdr	49
print.summary.risdr_sparse	49
projection_matrix	50
rmse	50
run_one_simulation	51
run_risdr_simulation	52
r_squared	52
select_dimension	53
select_dimension_cv	54
select_dimension_cv_icom	55
select_dimension_ladle	56
simulate_risdr_data	57
slice_covariances	59
slice_means	60
slice_proportions	60
slice_response	61
slice_summary	61
stabilize_cov	62
stabilize_eigenfloor	63
stabilize_nearest_pd	63

stabilize_ridge	64
subspace_distance	64
summarise_simulation	65
summarise_sparsity	65
summary.risdr	66
summary.risdr_dual	66
summary.risdr_sparse	67
weighted_soft_threshold	67
weighted_sparsify_directions	68

Index	69
--------------	-----------

risdr-package	<i>risdr: Regularised and Information-Theoretic Sufficient Dimension Reduction</i>
---------------	--

Description

The package implements SIR, SAVE, DR, and pHd for continuous responses, together with covariance regularisation, information-theoretic structural dimension selection, prediction, resampling, simulation, and diagnostic utilities.

Main interface

Use `fit_risdr()` to estimate an SDR model and `predict.risdr()` to obtain predictions for new observations. Use `select_dimension_cv()`, `select_dimension_cv_icoomp()`, or `select_dimension_ladle()` for complementary structural-dimension diagnostics.

Scope

`fit_risdr()` retains its continuous-response scope. Use `fit_risdr_sparse()` for joint dimension and sparsity selection with continuous, categorical, and right-censored survival responses, or `fit_risdr_dual()` for sample-space PCA reduction when predictors outnumber observations. `fit_risdr_sparse_realddata()` adds variance screening. See the sparse-survival vignette for the frozen thesis conventions.

Author(s)

Maintainer: Kabir Olorede <kabirolorede@gmail.com>

See Also

Useful links:

- <https://github.com/ilovemaths/risdr>
- <https://ilovemaths.github.io/risdr/>
- Report bugs at <https://github.com/ilovemaths/risdr/issues>

adjusted_r_squared	<i>Adjusted coefficient of determination</i>
--------------------	--

Description

Adjusted coefficient of determination

Usage

adjusted_r_squared(y_true, y_pred, d)

Arguments

- y_true Observed response values.
- y_pred Predicted response values.
- d Number of predictors in the downstream model.

Value

Numeric adjusted R-squared.

Examples

```
adjusted_r_squared(  
  c(2, 4, 6, 8, 10),  
  c(2.2, 3.8, 5.7, 8.1, 9.8),  
  d = 1  
)
```

ar1_covariance	<i>AR(1) covariance matrix</i>
----------------	--------------------------------

Description

AR(1) covariance matrix

Usage

ar1_covariance(p, rho)

Arguments

- p Dimension.
- rho Correlation parameter.

Value

Covariance matrix.

Examples

```
ar1_covariance(p = 4, rho = 0.5)
```

C1F	<i>C1F complexity and induced adaptive weights</i>
-----	--

Description

Computes cumulative scale-invariant covariance complexity from leading eigenvalues and converts the complexity profile to relative weights.

Usage

```
C1F(evals, positive_only = TRUE, cumulative = TRUE, tol = 1e-10)
```

Arguments

- evals Numeric vector of eigenvalues.
- positive_only Logical; retain only positive eigenvalues.
- cumulative Logical; compute cumulative C1F values.
- tol Numerical tolerance.

Value

A list with C1F values, induced weights, and processed eigenvalues.

choose_dimension	<i>Choose dimension from criteria table</i>
------------------	---

Description

Choose dimension from criteria table

Usage

```
choose_dimension(  
  d_table,  
  selector = c("cicomp", "icomp", "bic", "caic", "aic")  
)
```

Arguments

d_table Data frame returned by select_dimension().
 selector Criterion used for selection.

Value

Selected structural dimension.

Examples

```
scores <- as.matrix(mtcars[, c("wt", "hp", "disp")])
dimension_table <- select_dimension(scores, mtcars$mpg, d_max = 2)
choose_dimension(dimension_table, selector = "bic")
```

compute_dr	<i>Compute Directional Regression</i>
------------	---------------------------------------

Description

Computes Directional Regression using the canonical pairwise-slice formulation.

Usage

```
compute_dr(
  X,
  y,
  Sigma = NULL,
  nslices = 6,
  slice_type = c("quantile", "equal_width"),
  stabilize_slices = TRUE,
  stabilization = c("eigenfloor", "ridge", "nearest_pd"),
  eps = 1e-08
)
```

Arguments

X Numeric predictor matrix.
 y Numeric response vector.
 Sigma Covariance matrix used for standardisation.
 nslices Number of response slices.
 slice_type Slicing strategy.
 stabilize_slices Logical. Stabilise slice covariance matrices.
 stabilization Stabilisation method.
 eps Eigenvalue floor.

Details

Directional Regression is computed using the pairwise-slice formulation.

$$M_{DR} = \sum_h \sum_k p_h p_k (2I_p - A_{hk})(2I_p - A_{hk})^\top$$

where

$$A_{hk} = \Sigma_h + \Sigma_k + (\mu_h - \mu_k)(\mu_h - \mu_k)^\top$$

Value

A list containing DR kernel, eigenvalues, directions, scores, and slices.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
fit <- compute_dr(X, y = mtcars$mpg, nslices = 4)
head(fit$eigenvalues)
```

compute_ic

Compute one reduced-model information criterion

Description

Compute one reduced-model information criterion

Usage

```
compute_ic(
  y,
  Z,
  criterion = c("cicomp", "icomp", "aic", "caic", "bic"),
  response_type = c("continuous", "categorical", "survival"),
  delta = NULL,
  n_active = NULL,
  support_penalty = FALSE,
  tol = 1e-10
)
```

Arguments

y	Response vector.
Z	Reduced predictors.
criterion	Criterion name.
response_type	Response type.

delta	Optional event indicator.
n_active	Optional active-support size.
support_penalty	Include active-support size in effective df.
tol	Numerical tolerance.

Value

Scalar criterion value.

compute_ic_values	<i>Compute all reduced-model information criteria</i>
-------------------	---

Description

Compute all reduced-model information criteria

Usage

```
compute_ic_values(
  y,
  Z,
  response_type = c("continuous", "categorical", "survival"),
  delta = NULL,
  n_active = NULL,
  support_penalty = FALSE,
  tol = 1e-10
)
```

Arguments

y	Response vector.
Z	Reduced predictors.
response_type	Response type.
delta	Optional event indicator.
n_active	Optional active-support size.
support_penalty	Include active-support size in effective df.
tol	Numerical tolerance.

Value

Named numeric vector containing AIC, BIC, CAIC, ICOMP, CICOMP, log-likelihood, effective df, and C1F.

```
compute_information_criteria
```

Compute model selection criteria

Description

Computes AIC, BIC, CAIC, ICOMP, and CICOMP for a fitted linear model.

Usage

```
compute_information_criteria(fit, complexity = c("C1", "C1F"), eps = 1e-10)
```

Arguments

<code>fit</code>	A fitted lm object.
<code>complexity</code>	Character string. Complexity measure, either "C1" or "C1F".
<code>eps</code>	Eigenvalue floor.

Value

Named numeric vector of criteria.

Examples

```
fit <- stats::lm(mpg ~ wt + hp, data = mtcars)
compute_information_criteria(fit)
```

```
compute_penalty_weights
```

Compute adaptive penalty weights from C1F variable contributions

Description

For mode = "c1f", the regularised covariance matrix is decomposed into original-coordinate C1F contributions. Variables with larger contribution receive smaller soft-threshold weights, matching the direction of the historical C1F weighting rule while avoiding the invalid mapping of eigenvalue indices directly to predictor indices. `inverse_c1f` reverses that weighting. `uniform` returns equal weights.

Usage

```
compute_penalty_weights(
  Sigma,
  mode = c("c1f", "inverse_c1f", "uniform"),
  tol = 1e-10
)
```

Arguments

Sigma	Regularised covariance matrix.
mode	One of "c1f", "inverse_c1f", or "uniform".
tol	Numerical tolerance.

Value

Numeric vector of variable-level penalty weights with mean one.

compute_phd	<i>Compute Principal Hessian Directions</i>
-------------	---

Description

Compute Principal Hessian Directions

Usage

```
compute_phd(X, y, Sigma = NULL, eps = 1e-08)
```

Arguments

X	Numeric predictor matrix.
y	Numeric response vector.
Sigma	Covariance matrix used for standardisation.
eps	Eigenvalue floor.

Value

A list containing pHd kernel, eigenvalues, directions, scores.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
fit <- compute_phd(X, y = mtcars$mpg)
head(fit$eigenvalues)
```

compute_save

Compute Sliced Average Variance Estimation

Description

Compute Sliced Average Variance Estimation

Usage

```
compute_save(
  X,
  y,
  Sigma = NULL,
  nslices = 6,
  slice_type = c("quantile", "equal_width"),
  stabilize_slices = TRUE,
  stabilization = c("eigenfloor", "ridge", "nearest_pd"),
  eps = 1e-08
)
```

Arguments

X	Numeric predictor matrix.
y	Numeric response vector.
Sigma	Covariance matrix used for standardisation.
nslices	Number of response slices.
slice_type	Slicing strategy.
stabilize_slices	Logical. Stabilise slice covariance matrices.
stabilization	Stabilisation method.
eps	Eigenvalue floor.

Value

A list containing SAVE kernel, eigenvalues, directions, scores, and slices.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
fit <- compute_save(X, y = mtcars$mpg, nslices = 4)
head(fit$eigenvalues)
```

compute_scores	<i>Compute SDR scores for new data</i>
----------------	--

Description

Projects new predictor observations onto estimated SDR directions.

Usage

```
compute_scores(newX, directions)
```

Arguments

newX	New predictor matrix or data frame.
directions	Matrix of SDR directions.

Value

Matrix of SDR scores.

Examples

```
X <- as.matrix(mtcars[1:5, c("wt", "hp", "disp")])
directions <- diag(3)[, 1:2, drop = FALSE]
compute_scores(X, directions)
```

compute_sdr	<i>General SDR kernel dispatcher</i>
-------------	--------------------------------------

Description

General SDR kernel dispatcher

Usage

```
compute_sdr(
  X,
  y,
  method = c("dr", "sir", "save", "phd"),
  Sigma = NULL,
  nslices = 6,
  ...
)
```

Arguments

<code>X</code>	Numeric predictor matrix.
<code>y</code>	Numeric response vector.
<code>method</code>	SDR method.
<code>Sigma</code>	Covariance matrix.
<code>nslices</code>	Number of slices.
<code>...</code>	Additional arguments passed to internal methods.

Value

SDR fit components.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
Sigma <- cov_oas(X)
fit <- compute_sdr(
  X,
  y = mtcars$mpg,
  method = "sir",
  Sigma = Sigma,
  nslices = 4
)
dim(fit$scores)
```

compute_sir

Compute Sliced Inverse Regression

Description

Compute Sliced Inverse Regression

Usage

```
compute_sir(
  X,
  y,
  Sigma = NULL,
  nslices = 6,
  slice_type = c("quantile", "equal_width"),
  eps = 1e-08
)
```

Arguments

<code>X</code>	Numeric predictor matrix.
<code>y</code>	Numeric response vector.
<code>Sigma</code>	Covariance matrix used for standardisation.
<code>nslices</code>	Number of response slices.
<code>slice_type</code>	Slicing strategy.
<code>eps</code>	Eigenvalue floor.

Value

A list containing SIR kernel, eigenvalues, directions, scores, and slices.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
fit <- compute_sir(X, y = mtcars$mpg, nslices = 4)
head(fit$eigenvalues)
```

cov_complexity_C1	<i>Covariance complexity C1</i>
-------------------	---------------------------------

Description

Computes Bozdogan-type maximal covariance complexity.

Usage

```
cov_complexity_C1(Sigma, eps = 1e-10)
```

Arguments

<code>Sigma</code>	A covariance matrix.
<code>eps</code>	Eigenvalue floor.

Value

Numeric complexity value.

cov_complexity_C1F	<i>Scale-invariant covariance complexity C1F</i>
--------------------	--

Description

Computes the scale-invariant C1F covariance complexity measure.

Usage

```
cov_complexity_C1F(Sigma, eps = 1e-10)
```

Arguments

Sigma	A covariance matrix.
eps	Eigenvalue floor.

Value

Numeric complexity value.

cov_condition_number	<i>Covariance condition number</i>
----------------------	------------------------------------

Description

Computes the spectral condition number of a covariance matrix.

Usage

```
cov_condition_number(Sigma, eps = 1e-12)
```

Arguments

Sigma	A covariance matrix.
eps	Small positive value used to avoid division by zero.

Value

Numeric condition number.

Examples

```
Sigma <- cov(mtcars[, c("dis", "hp", "wt")])  
cov_condition_number(Sigma)
```

cov_diagnostics	<i>Covariance eigenvalue summary</i>
-----------------	--------------------------------------

Description

Provides diagnostic eigenvalue summaries for a covariance matrix.

Usage

```
cov_diagnostics(Sigma, tol = 1e-06)
```

Arguments

Sigma	A covariance matrix.
tol	Threshold for effective rank.

Value

A list of covariance diagnostics.

Examples

```
Sigma <- cov(mtcars[, c("disp", "hp", "wt")])  
cov_diagnostics(Sigma)
```

cov_effective_rank	<i>Effective covariance rank</i>
--------------------	----------------------------------

Description

Computes the number of eigenvalues exceeding a threshold.

Usage

```
cov_effective_rank(Sigma, tol = 1e-06)
```

Arguments

Sigma	A covariance matrix.
tol	Positive threshold.

Value

Effective rank.

Examples

```
Sigma <- cov(mtcars[, c("disp", "hp", "wt")])
cov_effective_rank(Sigma)
```

 cov_lw

Ledoit-Wolf type covariance estimator

Description

Computes a practical Ledoit-Wolf type shrinkage estimator toward a scaled identity target.

Usage

```
cov_lw(X)
```

Arguments

X Numeric matrix or data frame.

Value

A covariance matrix.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
Sigma <- cov_lw(X)
attr(Sigma, "shrinkage")
```

 cov_mec

Maximum Entropy Covariance estimator

Description

Computes a maximum-entropy motivated covariance estimator for SDR.

Usage

```
cov_mec(
  X,
  y,
  nslices = 6,
  response_type = c("continuous", "categorical", "survival"),
  delta = NULL,
  alpha = NULL,
  eps = 1e-06,
  ...
)
```

Arguments

<code>X</code>	Numeric matrix or data frame.
<code>y</code>	Numeric response vector.
<code>nslices</code>	Number of response slices.
<code>response_type</code>	Response type. One of "continuous", "categorical", or "survival".
<code>delta</code>	Optional event indicator for survival data.
<code>alpha</code>	Convex weight assigned to the entropy-maximising slice covariance. If NULL, alpha is chosen adaptively from the condition number of the pooled covariance.
<code>eps</code>	Small positive eigenvalue floor used for log-determinant stability.
<code>...</code>	Additional arguments passed to internal methods.

Details

The estimator uses response slicing to obtain slice-specific covariance matrices, selects the covariance matrix with maximum log-determinant entropy, and combines it with the pooled covariance matrix through convex shrinkage.

For censored survival responses, the function accepts `delta` and applies a survival-aware response construction before slicing.

Value

A covariance matrix with attributes containing the shrinkage weight and entropy diagnostics.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
Sigma <- cov_mec(X, y = mtcars$mpg, nslices = 4)
attr(Sigma, "alpha")
attr(Sigma, "entropy_slice")
```

 cov_oas

Oracle Approximating Shrinkage covariance estimator

Description

Computes a simplified Oracle Approximating Shrinkage (OAS) covariance estimator for stabilising the sample covariance matrix.

Usage

```
cov_oas(X)
```

Arguments

<code>X</code>	Numeric matrix or data frame.
----------------	-------------------------------

Value

A covariance matrix.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
Sigma <- cov_oas(X)
attr(Sigma, "shrinkage")
```

 cov_ridge

Ridge-type covariance estimator

Description

Computes a convex shrinkage covariance estimator of the form

$$\Sigma_\lambda = (1 - \lambda)S + \lambda T,$$

where S is the sample covariance matrix and T is a scaled identity target.

Usage

```
cov_ridge(X, lambda = 0.1)
```

Arguments

<code>X</code>	Numeric matrix or data frame.
<code>lambda</code>	Shrinkage intensity in $[0, 1]$.

Value

A covariance matrix.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
cov_ridge(X, lambda = 0.2)
```

cov_sample	<i>Sample covariance matrix</i>
------------	---------------------------------

Description

Computes the usual unbiased sample covariance matrix.

Usage

```
cov_sample(X)
```

Arguments

X Numeric matrix or data frame.

Value

A covariance matrix.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
cov_sample(X)
```

criterion_weights	<i>Information criterion weights</i>
-------------------	--------------------------------------

Description

Computes relative weights from any information criterion column.

Usage

```
criterion_weights(
  d_table,
  criterion = c("AIC", "BIC", "CAIC", "ICOMP", "CICOMP")
)
```

Arguments

d_table Dimension selection table.
criterion Criterion column name.

Value

Data frame with criterion differences and weights.

Examples

```
scores <- as.matrix(mtcars[, c("wt", "hp", "disp")])
dimension_table <- select_dimension(scores, mtcars$mpg, d_max = 2)
criterion_weights(dimension_table, criterion = "BIC")
```

dual_pca_scores

Dual PCA score construction for p much larger than n **Description**

Constructs low-rank sample-space scores without forming a p by p covariance matrix. This is useful when the number of predictors is very large.

Usage

```
dual_pca_scores(
  X,
  dual_rank = NULL,
  variance_explained = NULL,
  center = TRUE,
  scale_X = TRUE,
  eps = 1e-10
)
```

Arguments

<code>X</code>	Numeric predictor matrix with observations in rows.
<code>dual_rank</code>	Number of dual components to retain.
<code>variance_explained</code>	Optional cumulative variance threshold.
<code>center</code>	Logical; if TRUE, centre predictors.
<code>scale_X</code>	Logical; if TRUE, scale predictors.
<code>eps</code>	Numerical tolerance.

Value

A list containing scores, loadings, singular values, centre, scale, and retained rank.

estimate_cov

*General covariance estimator dispatcher***Description**

Dispatches to the requested covariance estimator. For MEC, the function supports continuous, categorical, and censored survival responses by passing `response_type` and `delta` to `cov_mec()`.

Usage

```
estimate_cov(
  X,
  y = NULL,
  method = c("sample", "ridge", "oas", "lw", "mec"),
  nslices = 6,
  response_type = c("continuous", "categorical", "survival"),
  delta = NULL,
  ...
)
```

Arguments

<code>X</code>	Numeric matrix or data frame.
<code>y</code>	Optional response vector required for MEC.
<code>method</code>	Covariance estimator.
<code>nslices</code>	Number of slices for MEC.
<code>response_type</code>	Response type. One of "continuous", "categorical", or "survival".
<code>delta</code>	Optional event indicator for survival data.
<code>...</code>	Additional arguments passed to internal covariance estimators.

Value

A covariance matrix.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])

estimate_cov(X, method = "oas")
estimate_cov(
  X,
  y = mtcars$mpg,
  method = "mec",
  nslices = 4
)
```

evaluate_prediction	<i>Evaluate predictions</i>
---------------------	-----------------------------

Description

Computes common prediction metrics.

Usage

```
evaluate_prediction(y_true, y_pred, d = NULL)
```

Arguments

y_true	Observed response values.
y_pred	Predicted response values.
d	Optional number of reduced predictors.

Value

A data frame of prediction metrics.

Examples

```
evaluate_prediction(  
  y_true = c(2, 4, 6, 8, 10),  
  y_pred = c(2.2, 3.8, 5.7, 8.1, 9.8),  
  d = 1  
)
```

evaluate_prediction_cv

Repeated cross-validation for predictive assessment

Description

Repeats V-fold cross-validation for a fixed SDR and covariance-estimator combination. Failed folds are retained with diagnostic messages.

Usage

```

evaluate_prediction_cv(
  X,
  y,
  sdr_method = c("dr", "sir", "save", "phd"),
  cov_method = c("sample", "ridge", "oas", "lw", "mec"),
  d,
  v = 5,
  repeats = 10,
  nslices = 6,
  standardize = TRUE,
  stabilize = TRUE,
  stabilization = c("eigenfloor", "ridge", "nearest_pd"),
  seed = NULL,
  cov_args = list(),
  stabilization_args = list(),
  sdr_args = list(),
  ...
)

```

Arguments

X	Numeric predictor matrix or data frame.
y	Numeric continuous response vector.
sdr_method	SDR method.
cov_method	Covariance estimator.
d	Fixed structural dimension.
v	Number of folds.
repeats	Number of cross-validation repetitions.
nslices	Number of response slices.
standardize	Logical. Standardise within each training fold.
stabilize	Logical. Stabilise the estimated covariance matrix.
stabilization	Covariance stabilisation method.
seed	Optional random seed.
cov_args	Named list of covariance-estimator arguments.
stabilization_args	Named list of stabilisation arguments.
sdr_args	Named list of SDR-kernel arguments.
...	Backward-compatible component arguments.

Value

A list containing a one-row summary and fold-level results.

<code>extract_loadings</code>	<i>Extract SDR loadings</i>
-------------------------------	-----------------------------

Description

Extract SDR loadings

Usage

```
extract_loadings(directions, variables = NULL)
```

Arguments

- `directions` Matrix of SDR directions.
- `variables` Variable names.

Value

A data frame of loadings.

<code>filter_low_variance</code>	<i>Filter low-variance predictors</i>
----------------------------------	---------------------------------------

Description

Removes predictors with variance below a specified quantile.

Usage

```
filter_low_variance(X, variance_quantile = 0.25)
```

Arguments

- `X` Predictor matrix.
- `variance_quantile` Quantile threshold.

Value

Filtered matrix.

fit_downstream_lm	<i>Fit downstream regression model on SDR scores</i>
-------------------	--

Description

Fits a linear regression model using the first d SDR scores.

Usage

```
fit_downstream_lm(scores, y, d)
```

Arguments

scores	Matrix of SDR scores.
y	Numeric response vector.
d	Structural dimension.

Value

A fitted lm object.

Examples

```
scores <- as.matrix(mtcars[, c("wt", "hp")])
fit_downstream_lm(scores, y = mtcars$mpg, d = 2)
```

fit_risdr	<i>Fit regularised and information-theoretic SDR model</i>
-----------	--

Description

Fits sufficient dimension reduction models with optional covariance regularisation and information-theoretic structural dimension selection.

Usage

```
fit_risdr(
  X,
  y,
  sdr_method = c("dr", "sir", "save", "phd"),
  cov_method = c("sample", "ridge", "oas", "lw", "mec"),
  stabilize = TRUE,
  stabilization = c("eigenfloor", "ridge", "nearest_pd"),
  nslices = 6,
  d = NULL,
```

```

    d_max = 10,
    selector = c("cicomp", "icomp", "bic", "caic", "aic"),
    standardize = TRUE,
    complexity = c("C1", "C1F"),
    cov_args = list(),
    stabilization_args = list(),
    sdr_args = list(),
    ...
  )

```

Arguments

<code>X</code>	Numeric predictor matrix or data frame.
<code>y</code>	Numeric continuous response vector.
<code>sdr_method</code>	SDR method: "dr", "sir", "save", or "phd".
<code>cov_method</code>	Covariance estimator: "sample", "ridge", "oas", "lw", or "mec".
<code>stabilize</code>	Logical. If TRUE, stabilises the estimated covariance matrix.
<code>stabilization</code>	Stabilisation method.
<code>nslices</code>	Number of slices for inverse regression methods.
<code>d</code>	Optional structural dimension. If NULL, selected by criterion.
<code>d_max</code>	Maximum candidate structural dimension.
<code>selector</code>	Criterion for selecting <code>d</code> .
<code>standardize</code>	Logical. If TRUE, column-standardises <code>X</code> before covariance estimation.
<code>complexity</code>	Complexity measure for ICOMP: "C1" or "C1F".
<code>cov_args</code>	Named list of arguments for the selected covariance estimator.
<code>stabilization_args</code>	Named list of arguments for covariance stabilisation.
<code>sdr_args</code>	Named list of arguments for the selected SDR kernel.
<code>...</code>	Backward-compatible component arguments. New code should use the three explicit argument lists.

Value

An object of class "risdr".

Examples

```

simulated <- simulate_risdr_data(
  n = 60,
  p = 6,
  d = 2,
  seed = 2026
)

fit <- fit_risdr(

```

```

    X = simulated$X,
    y = simulated$y,
    sdr_method = "sir",
    cov_method = "oas",
    nslices = 4,
    d = 1,
    d_max = 3
  )

  fit

```

fit_risdr_dual

Fit dual high-dimensional RISDR

Description

Constructs a low-rank PCA score representation, fits the unified RISDR core in that score space, maps the dense and selected sparse SDR directions back to the original predictor coordinates, and ranks variables using the Euclidean loading norm across only the selected sparse directions.

Usage

```

fit_risdr_dual(
  X,
  y,
  delta = NULL,
  response_type = c("continuous", "categorical", "survival"),
  dual_rank = 100L,
  variance_explained = NULL,
  d = NULL,
  d_max = 10L,
  nslices = 4L,
  sdr_method = c("dr", "sir", "save", "phd"),
  cov_method = c("mec", "oas", "lw", "sample", "ridge"),
  selector = c("cicomp", "icomp", "aic", "caic", "bic"),
  lambda_grid = seq(0, 0.3, length.out = 10L),
  adapt_weights = TRUE,
  weight_mode = c("c1f", "inverse_c1f", "uniform"),
  support_penalty = FALSE,
  standardize_dual = TRUE,
  gene_select_top = NULL,
  gene_select_quantile = 0.99,
  center = TRUE,
  scale_X = TRUE,
  verbose = TRUE,
  ...
)

```

Arguments

<code>X</code>	Numeric high-dimensional predictor matrix.
<code>y</code>	Response vector. For survival data, observed time.
<code>delta</code>	Optional 0/1 event indicator.
<code>response_type</code>	One of "continuous", "categorical", or "survival".
<code>dual_rank</code>	Maximum retained PCA rank.
<code>variance_explained</code>	Optional cumulative variance threshold.
<code>d</code>	Optional fixed structural dimension. If NULL, selected by selector.
<code>d_max</code>	Maximum candidate structural dimension.
<code>nslices</code>	Number of slices, or slices per censoring-status group.
<code>sdr_method</code>	SDR method.
<code>cov_method</code>	Covariance method.
<code>selector</code>	Information criterion used to select (<code>d</code> , <code>lambda</code>).
<code>lambda_grid</code>	Sparsity tuning grid.
<code>adapt_weights</code>	Logical. Use adaptive penalty weights.
<code>weight_mode</code>	Adaptive weighting mode.
<code>support_penalty</code>	Logical. Include active support in criterion df.
<code>standardize_dual</code>	Logical. Standardise retained PCA scores inside <code>fit_risdr_sparse()</code> .
<code>gene_select_top</code>	Optional number of top-ranked variables.
<code>gene_select_quantile</code>	Quantile used when <code>gene_select_top</code> is NULL.
<code>center</code>	Logical. Centre predictors before dual PCA.
<code>scale_X</code>	Logical. Scale predictors before dual PCA.
<code>verbose</code>	Logical. Emit progress messages.
<code>...</code>	Additional arguments passed to <code>fit_risdr_sparse()</code> .

Details

The inner RISDR standardisation scale is explicitly undone before mapping directions through the PCA loadings. The returned reduced predictors are taken directly from the fitted core and are therefore exactly aligned with the model used for information-criterion selection.

This entry point uses the frozen sparse thesis engine. The existing `fit_risdr()` and `fit_risdr_realddata()` interfaces retain their continuous-response behaviour from version 0.3.1. Survival fits use Efron Cox partial likelihood. Apparent fit statistics and loading ranks are exploratory and do not establish external predictive validity.

Value

An object of class `risdr_dual`.

Examples

```

set.seed(2026)
X <- matrix(rnorm(80 * 6), 80)
colnames(X) <- paste0("G", 1:6)
time <- rexp(80, rate = exp(0.5 * X[, 1]))
delta <- rep(c(0L, 1L), 40)
fit <- fit_risdr_dual(
  X, time, delta = delta, response_type = "survival",
  sdr_method = "sir", cov_method = "oas", nslices = 4,
  dual_rank = 4, verbose = FALSE, d_max = 2, lambda_grid = c(0, 0.05)
)
fit$d
predict(fit, X[1:2, , drop = FALSE], type = "risk")

```

fit_risdr_realdata	<i>Fit RISDR to real high-dimensional data</i>
--------------------	--

Description

Fit RISDR to real high-dimensional data

Usage

```

fit_risdr_realdata(
  X,
  y,
  delta = NULL,
  response_type = c("continuous", "binary", "multiclass", "survival"),
  variance_quantile = 0.25,
  d = NULL,
  ...
)

```

Arguments

X	Predictor matrix.
y	Response vector.
delta	Optional survival censoring indicator.
response_type	Response type.
variance_quantile	Variance filtering threshold.
d	Optional fixed structural dimension passed to <code>fit_risdr()</code> .
...	Additional arguments passed to <code>fit_risdr()</code> .

Value

Fitted RISDR workflow object.

fit_risdr_sparse	<i>Fit regularised and information-theoretic sufficient dimension reduction</i>
------------------	---

Description

Fits an SDR model using a regularised covariance estimator, a selected inverse-regression kernel, adaptive sparse direction estimation, and information-criterion profiling jointly over structural dimension and sparsity tuning parameter.

Usage

```
fit_risdr_sparse(
  X,
  y,
  delta = NULL,
  response_type = c("continuous", "categorical", "survival"),
  sdr_method = c("dr", "sir", "save", "phd"),
  cov_method = c("mec", "oas", "lw", "sample", "ridge", "user"),
  stabilize = TRUE,
  stabilization = c("eigenfloor", "ridge", "nearest_pd"),
  nslices = 6L,
  d = NULL,
  d_max = 10L,
  selector = c("cicomp", "icomp", "bic", "caic", "aic"),
  standardize = TRUE,
  complexity = "C1F",
  lambda_grid = seq(0, 0.3, length.out = 10L),
  adapt_weights = TRUE,
  weight_mode = c("c1f", "inverse_c1f", "uniform"),
  penalty_weights = NULL,
  support_penalty = FALSE,
  cov_fun = NULL,
  shrinkage = 1e-04,
  tol = 1e-10,
  ...
)
```

Arguments

X	Numeric predictor matrix or data frame.
y	Response vector. For survival data, observed survival time.
delta	Optional 0/1 event indicator for survival data.
response_type	One of "continuous", "categorical", or "survival".
sdr_method	One of "dr", "sir", "save", or "phd".
cov_method	One of "mec", "oas", "lw", "sample", "ridge", or "user".

stabilize	Logical. Stage-3 thesis fits require covariance stabilisation and therefore currently require TRUE.
stabilization	Stabilisation method passed to <code>estimate_cov()</code> .
nslices	Number of response slices, or slices per censoring-status group for survival data.
d	Optional fixed structural dimension. If NULL, selected by selector from candidates 1 through d_max.
d_max	Maximum candidate structural dimension.
selector	One of "cicomp", "icom", "bic", "caic", or "aic".
standardize	Logical. Standardise predictor columns before fitting.
complexity	Retained for compatibility. The Stage-3 thesis engine uses the scale-invariant C1F complexity in ICOMP/CICOMP.
lambda_grid	Non-negative sparsity tuning grid.
adapt_weights	Logical. Use covariance-derived adaptive weights.
weight_mode	One of "c1f", "inverse_c1f", or "uniform".
penalty_weights	Optional user-supplied variable-level weights.
support_penalty	Logical. If TRUE, active-support size contributes to the effective parameter count in the information criteria.
cov_fun	Optional user-defined covariance estimator.
shrinkage	Ridge shrinkage parameter used by <code>estimate_cov()</code> .
tol	Numerical tolerance.
...	Additional arguments passed only to a user covariance estimator.

Details

For `d = NULL`, every candidate dimension from 1 through `d_max` is evaluated over the complete `lambda_grid`. The requested selector then chooses the final `(d, lambda)` pair. For explicit `d`, the same audit grid is retained but the final dimension is fixed and only `lambda` is selected within that dimension.

This entry point uses the frozen sparse thesis engine. The existing `fit_risdr()` and `fit_risdr_realddata()` interfaces retain their continuous-response behaviour from version 0.3.1. Survival fits use Efron Cox partial likelihood. Apparent fit statistics and loading ranks are exploratory and do not establish external predictive validity.

Value

An object of class `risdr` containing the complete `(d, lambda)` selection grid, profiled dimension table, selected sparse directions, reduced predictors, covariance diagnostics, and compatibility fields.

Examples

```

set.seed(2026)
X <- matrix(rnorm(80 * 6), 80)
colnames(X) <- paste0("G", 1:6)
time <- rexp(80, rate = exp(0.5 * X[, 1]))
delta <- rep(c(0L, 1L), 40)
fit <- fit_risdr_sparse(
  X, time, delta = delta, response_type = "survival",
  sdr_method = "sir", cov_method = "oas", nslices = 4,
  d_max = 2, lambda_grid = c(0, 0.05)
)
fit$d
predict(fit, X[1:2, , drop = FALSE], type = "risk")

```

fit_risdr_sparse_realdata

Fit RISDR to real high-dimensional data

Description

Fit RISDR to real high-dimensional data

Usage

```

fit_risdr_sparse_realdata(
  X,
  y,
  delta = NULL,
  response_type = c("continuous", "binary", "multiclass", "categorical", "survival"),
  variance_quantile = 0.25,
  ...
)

```

Arguments

X	Predictor matrix.
y	Response vector.
delta	Optional survival censoring indicator.
response_type	Response type.
variance_quantile	Variance filtering threshold.
...	Additional arguments passed to fit_risdr_sparse().

Details

This entry point uses the frozen sparse thesis engine. The existing `fit_risdr()` and `fit_risdr_realdata()` interfaces retain their continuous-response behaviour from version 0.3.1. Survival fits use Efron Cox partial likelihood. Apparent fit statistics and loading ranks are exploratory and do not establish external predictive validity.

Value

Fitted RISDR workflow object. Fit RISDR to a real-data predictor matrix

Applies outcome cleaning, low-variance screening, the unified RISDR core, and descriptive loading-based variable ranking. The wrapper returns the reduced predictors computed by the core itself so that centring and standardisation are not silently bypassed.

"multiclass", "categorical", or "survival". low-variance screen.

An object of class `risdr_realdata`.

<code>fit_sparse_grid</code>	<i>Fit sparse SDR directions over a lambda grid</i>
------------------------------	---

Description

Accepts either an explicit dense direction matrix or a historical kernel object containing directions or vectors.

Usage

```
fit_sparse_grid(
  kernel_obj = NULL,
  directions = NULL,
  d = 2L,
  lambda_grid = seq(0, 0.5, length.out = 25L),
  penalty_weights = NULL,
  adapt_weights = FALSE,
  Sigma = NULL,
  weight_mode = c("c1f", "inverse_c1f", "uniform"),
  tol = 1e-10
)
```

Arguments

<code>kernel_obj</code>	Optional historical kernel object.
<code>directions</code>	Optional dense direction matrix.
<code>d</code>	Structural dimension.
<code>lambda_grid</code>	Numeric vector of tuning values.
<code>penalty_weights</code>	Optional user-supplied penalty weights.

- adapt_weights Logical; derive weights from Sigma when TRUE.
- Sigma Optional covariance matrix.
- weight_mode Adaptive weighting mode.
- tol Numerical tolerance.

Value

Sparse-grid object.

mae	<i>Mean absolute error</i>
-----	----------------------------

Description

Mean absolute error

Usage

mae(y_true, y_pred)

Arguments

- y_true Observed response values.
- y_pred Predicted response values.

Value

Numeric MAE.

Examples

mae(c(2, 4, 6, 8), c(2.2, 3.8, 5.7, 8.1))

make_cv_folds	Create cross-validation folds
---------------	-------------------------------

Description

Create cross-validation folds

Usage

```
make_cv_folds(n, v = 5, seed = NULL)
```

Arguments

n	Sample size.
v	Number of folds.
seed	Optional random seed.

Value

Integer vector of fold memberships.

make_slices	Create response slices
-------------	------------------------

Description

Constructs response slices for inverse regression-based sufficient dimension reduction methods.

Usage

```
make_slices(y, nslices = 6, type = c("quantile", "equal_width"))
```

Arguments

y	Numeric continuous response vector.
nslices	Number of slices.
type	Slicing strategy. Currently supports "quantile" and "equal_width".

Value

Integer vector of slice memberships.

Examples

```
slices <- make_slices(mtcars$mpg, nslices = 4)
table(slices)
```

mape	<i>Mean absolute percentage error</i>
------	---------------------------------------

Description

Mean absolute percentage error

Usage

```
mape(y_true, y_pred)
```

Arguments

y_true	Observed response values.
y_pred	Predicted response values.

Value

Numeric MAPE.

Examples

```
mape(c(2, 4, 6, 8), c(2.2, 3.8, 5.7, 8.1))
```

plot_cv_dimension	<i>Plot cross-validation dimension selection result</i>
-------------------	---

Description

Plot cross-validation dimension selection result

Usage

```
plot_cv_dimension(cv, metric = NULL, ...)
```

Arguments

cv	Object returned by select_dimension_cv().
metric	Metric to plot.
...	Additional arguments passed to internal methods.

Value

Invisibly returns the CV table.

plot_dimension_selection	<i>Plot dimension selection criteria</i>
--------------------------	--

Description

Plot dimension selection criteria

Usage

```
plot_dimension_selection(
  fit,
  criteria = c("AIC", "BIC", "CAIC", "ICOMP", "CICOMP"),
  ...
)
```

Arguments

fit	Object of class "risdr".
criteria	Character vector of criteria to plot.
...	Additional graphical arguments passed to graphics::matplot() .

Value

Invisibly returns dimension table.

plot_ladle	<i>Plot ladle dimension diagnostic</i>
------------	--

Description

Plot ladle dimension diagnostic

Usage

```
plot_ladle(ladle, ...)
```

Arguments

ladle	Object returned by select_dimension_ladle().
...	Additional graphical arguments passed to graphics::matplot() .

Value

Invisibly returns ladle table.

plot_loadings	<i>Plot SDR loadings</i>
---------------	--------------------------

Description

Displays the largest absolute loadings for a selected SDR direction.

Usage

```
plot_loadings(fit, direction = 1, top = 15, ...)
```

Arguments

fit	Object of class "risdr".
direction	Direction index.
top	Number of variables to display.
...	Additional graphical arguments passed to graphics::barplot() .

Value

Invisibly returns loading table.

plot_prediction	<i>Prediction plot</i>
-----------------	------------------------

Description

Plots observed versus predicted values.

Usage

```
plot_prediction(y_true, y_pred, ...)
```

Arguments

y_true	Observed response values.
y_pred	Predicted response values.
...	Additional graphical arguments passed to graphics::plot() .

Value

Invisibly returns plotted data.

plot_residuals	<i>Residual plot</i>
----------------	----------------------

Description

Plots residuals against predicted values.

Usage

```
plot_residuals(y_true, y_pred, ...)
```

Arguments

y_true	Observed response values.
y_pred	Predicted response values.
...	Additional graphical arguments passed to graphics::plot() .

Value

Invisibly returns plotted data.

plot_scee	<i>Scree plot of SDR eigenvalues</i>
-----------	--------------------------------------

Description

Scree plot of SDR eigenvalues

Usage

```
plot_scee(fit, n_eigen = 20, type = "b", ...)
```

Arguments

fit	Object of class "risdr".
n_eigen	Number of eigenvalues to display.
type	Plot type passed to base R plot().
...	Additional graphical arguments passed to graphics::plot() .

Value

Invisibly returns eigenvalues plotted.

plot_sufficient	<i>Sufficient summary plot</i>
-----------------	--------------------------------

Description

Plots the response against a selected SDR direction.

Usage

```
plot_sufficient(fit, direction = 1, ...)
```

Arguments

fit	Object of class "risdr".
direction	SDR direction to plot.
...	Additional graphical arguments passed to graphics::plot() .

Value

Invisibly returns plotted data.

plot_sufficient2d	<i>Two-direction sufficient summary plot</i>
-------------------	--

Description

Plots the first two selected SDR scores, optionally coloured by response.

Usage

```
plot_sufficient2d(
  fit,
  directions = c(1, 2),
  colour_by_y = TRUE,
  groups = 4,
  ...
)
```

Arguments

fit	Object of class "risdr".
directions	Integer vector of length 2.
colour_by_y	Logical. If TRUE, colours points by response quantile group.
groups	Number of response colour groups.
...	Additional graphical arguments passed to graphics::plot() .

Value

Invisibly returns plotted data.

predict.risdr	<i>Predict method for risdr objects</i>
---------------	---

Description

Predict method for risdr objects

Usage

```
## S3 method for class 'risdr'
predict(object, newX, d = NULL, ...)
```

Arguments

object	Object of class "risdr".
newX	New predictor matrix or data frame.
d	Optional structural dimension.
...	Additional arguments passed to internal methods.

Value

Numeric vector of predictions.

Examples

```
simulated <- simulate_risdr_data(
  n = 60,
  p = 6,
  d = 2,
  seed = 2026
)

fit <- fit_risdr(
  X = simulated$X,
  y = simulated$y,
  sdr_method = "sir",
  cov_method = "oas",
  nslices = 4,
  d = 1,
  d_max = 3
)

predict(fit, newX = simulated$X[1:5, , drop = FALSE])
```

predict.risdr_sparse *Predict from the sparse or dual SDR engine*

Description

Projects new observations using the training centring, scaling, and selected directions. Survival predictions are Cox linear predictors or relative risks; they are not predicted survival times.

Usage

```
## S3 method for class 'risdr_sparse'
predict(
  object,
  newX,
  type = c("response", "scores", "link", "risk", "class", "probs"),
  ...
)

## S3 method for class 'risdr_dual'
predict(
  object,
  newX,
  type = c("response", "scores", "link", "risk", "class", "probs"),
  ...
)
```

Arguments

object	A sparse or dual RISDR fit.
newX	Numeric predictor matrix or data frame, with named columns.
type	Prediction type. "response" returns continuous predictions, binary probabilities, multiclass labels, or survival relative risks. "scores" returns the selected SDR coordinates. "link" and "risk" are available for survival outcomes. "class" and "probs" are available for categorical outcomes.
...	Additional arguments passed to the downstream prediction method.

Value

Predictions or a score matrix, according to type.

`prediction_correlation`*Prediction correlation*

Description

Prediction correlation

Usage

```
prediction_correlation(y_true, y_pred)
```

Arguments

<code>y_true</code>	Observed response values.
<code>y_pred</code>	Predicted response values.

Value

Pearson correlation.

Examples

```
prediction_correlation(  
  c(2, 4, 6, 8),  
  c(2.2, 3.8, 5.7, 8.1)  
)
```

`predict_downstream_lm` *Predict from downstream SDR regression model*

Description

Predict from downstream SDR regression model

Usage

```
predict_downstream_lm(fit_lm, scores_new, d)
```

Arguments

<code>fit_lm</code>	Fitted linear model from <code>fit_downstream_lm()</code> .
<code>scores_new</code>	Matrix of new SDR scores.
<code>d</code>	Structural dimension.

Value

Numeric vector of predictions.

```
prepare_survival_response
```

Prepare survival response

Description

Prepare survival response

Usage

```
prepare_survival_response(time, delta)
```

Arguments

time	Survival time vector.
delta	Event indicator vector.

Value

Cleaned survival response list.

```
print.risdr
```

Print risdr object

Description

Print risdr object

Usage

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

Arguments

x	Object of class "risdr".
...	Additional arguments passed to internal methods.

Value

Invisibly returns x.

print.risdr_dual	<i>Print method for dual RISDR objects</i>
------------------	--

Description

Print method for dual RISDR objects

Usage

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

Arguments

x	Object of class risdr_dual.
...	Additional arguments passed to internal methods.

print.risdr_realdata	<i>Print real-data RISDR workflow</i>
----------------------	---------------------------------------

Description

Print real-data RISDR workflow

Usage

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

Arguments

x	Object of class risdr_realdata.
...	Additional arguments.

Value

Invisibly returns x.

print.risdr_sparse	<i>Print risdr object</i>
--------------------	---------------------------

Description

Print risdr object

Usage

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

Arguments

x	Object of class "risdr".
...	Additional arguments passed to internal methods.

Value

Invisibly returns x.

print.risdr_sparse_realdata	<i>Print real-data RISDR workflow</i>
-----------------------------	---------------------------------------

Description

Print real-data RISDR workflow

Usage

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

Arguments

x	Object of class risdr_realdata.
...	Additional arguments.

print.summary.risdr	<i>Print summary of risdr object</i>
---------------------	--------------------------------------

Description

Print summary of risdr object

Usage

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

Arguments

x	Object of class "summary.risdr".
...	Additional arguments passed to internal methods.

Value

Invisibly returns x.

print.summary.risdr_sparse	<i>Print summary of risdr object</i>
----------------------------	--------------------------------------

Description

Print summary of risdr object

Usage

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

Arguments

x	Object of class "summary.risdr_sparse".
...	Additional arguments passed to internal methods.

Value

Invisibly returns x.

projection_matrix	<i>Projection matrix</i>
-------------------	--------------------------

Description

Projection matrix

Usage

```
projection_matrix(B)
```

Arguments

B Basis matrix.

Value

Projection matrix.

Examples

```
B <- matrix(c(1, 0, 0, 0, 1, 0), nrow = 3)
projection_matrix(B)
```

rmse	<i>Root mean squared error</i>
------	--------------------------------

Description

Root mean squared error

Usage

```
rmse(y_true, y_pred)
```

Arguments

y_true Observed response values.
y_pred Predicted response values.

Value

Numeric RMSE.

Examples

```
rmse(c(2, 4, 6, 8), c(2.2, 3.8, 5.7, 8.1))
```

run_one_simulation	<i>Run one SDR simulation replication</i>
--------------------	---

Description

Run one SDR simulation replication

Usage

```
run_one_simulation(  
  n = 200,  
  p = 50,  
  d = 2,  
  rho = 0.6,  
  sigma = 1,  
  model = "linear_quadratic",  
  beta_type = "coordinate",  
  sdr_method = "dr",  
  cov_method = "oas",  
  nslices = 6,  
  selector = "cicomp",  
  d_max = 10,  
  seed = NULL  
)
```

Arguments

n	Sample size.
p	Number of predictors.
d	Structural dimension.
rho	Correlation parameter.
sigma	Error standard deviation.
model	Data-generating model.
beta_type	Type of true central subspace basis.
sdr_method	SDR method.
cov_method	Covariance method.
nslices	Number of slices.
selector	Dimension selection criterion.
d_max	Maximum candidate dimension.
seed	Optional random seed. The test sample uses seed + 1.

Value

A data frame with simulation results.

run_risdr_simulation	<i>Run SDR simulation study</i>
----------------------	---------------------------------

Description

Run SDR simulation study

Usage

```
run_risdr_simulation(  
  R = 200,  
  rho_values = c(0.3, 0.6, 0.9),  
  methods = c("sir", "save", "dr", "phd"),  
  cov_methods = c("sample", "oas", "mec"),  
  seed = NULL,  
  ...  
)
```

Arguments

- R Number of replications.
- rho_values Correlation values.
- methods SDR methods.
- cov_methods Covariance methods.
- seed Optional starting seed for reproducible replications.
- ... Additional arguments passed to [run_one_simulation\(\)](#).

Value

Data frame of simulation results.

r_squared	<i>Coefficient of determination</i>
-----------	-------------------------------------

Description

Coefficient of determination

Usage

```
r_squared(y_true, y_pred)
```

Arguments

y_true	Observed response values.
y_pred	Predicted response values.

Value

Numeric R-squared.

Examples

```
r_squared(c(2, 4, 6, 8), c(2.2, 3.8, 5.7, 8.1))
```

select_dimension	<i>Select structural dimension</i>
------------------	------------------------------------

Description

Fits linear models on SDR scores for candidate dimensions and computes information criteria.

Usage

```
select_dimension(  
  scores,  
  y,  
  d_max = 10,  
  complexity = c("C1", "C1F"),  
  eps = 1e-10  
)
```

Arguments

scores	Matrix of SDR scores.
y	Numeric response vector.
d_max	Maximum candidate dimension.
complexity	Complexity measure for ICOMP.
eps	Eigenvalue floor.

Value

A data frame of criteria by candidate dimension.

Examples

```

scores <- as.matrix(mtcars[, c("wt", "hp", "disp")])
dimension_table <- select_dimension(
  scores = scores,
  y = mtcars$mpg,
  d_max = 2
)
dimension_table

```

select_dimension_cv	<i>Cross-validation dimension selection for SDR</i>
---------------------	---

Description

Selects structural dimension by V-fold cross-validation. For each candidate dimension, SDR is fitted on the training folds, a downstream linear regression is fitted using the reduced predictors, and prediction error is evaluated on the validation fold.

Usage

```

select_dimension_cv(
  X,
  y,
  sdr_method = c("dr", "sir", "save", "phd"),
  cov_method = c("sample", "ridge", "oas", "lw", "mec"),
  d_max = 10,
  v = 5,
  nslices = 6,
  standardize = TRUE,
  stabilize = TRUE,
  stabilization = c("eigenfloor", "ridge", "nearest_pd"),
  metric = c("RMSE", "MAE"),
  seed = NULL,
  cov_args = list(),
  stabilization_args = list(),
  sdr_args = list(),
  ...
)

```

Arguments

X	Numeric predictor matrix or data frame.
y	Numeric continuous response vector.
sdr_method	SDR method.
cov_method	Covariance estimator.
d_max	Maximum candidate structural dimension.

v	Number of folds.
nslices	Number of slices.
standardize	Logical. If TRUE, standardises predictors inside each training fold.
stabilize	Logical. If TRUE, stabilises covariance matrix.
stabilization	Stabilisation method.
metric	Prediction metric used for selection.
seed	Optional random seed.
cov_args	Named list of arguments for the covariance estimator.
stabilization_args	Named list of arguments for covariance stabilisation.
sdr_args	Named list of arguments for the SDR kernel.
...	Backward-compatible component arguments.

Value

A list containing the CV table, selected dimension, and fold-level results.

select_dimension_cv_icom

Complexity-aware cross-validation for structural dimension selection

Description

Combines out-of-fold prediction error with rescaled BIC, CAIC, or CCOMP penalties. The tuning constant lambda controls the contribution of the information-criterion component.

Usage

```
select_dimension_cv_icom(
  X,
  y,
  sdr_method = c("dr", "sir", "save", "phd"),
  cov_method = c("sample", "ridge", "oas", "lw", "mec"),
  d_max = 10,
  v = 5,
  nslices = 6,
  standardize = TRUE,
  stabilize = TRUE,
  stabilization = c("eigenfloor", "ridge", "nearest_pd"),
  complexity = c("C1", "C1F"),
  lambda = 1,
  seed = NULL,
  cov_args = list(),
  stabilization_args = list(),
  sdr_args = list(),
  ...
)
```

Arguments

<code>X</code>	Numeric predictor matrix or data frame.
<code>y</code>	Numeric continuous response vector.
<code>sdr_method</code>	SDR method.
<code>cov_method</code>	Covariance estimator.
<code>d_max</code>	Maximum candidate structural dimension.
<code>v</code>	Number of cross-validation folds.
<code>nslices</code>	Number of response slices.
<code>standardize</code>	Logical. Standardise within each training fold.
<code>stabilize</code>	Logical. Stabilise the estimated covariance matrix.
<code>stabilization</code>	Covariance stabilisation method.
<code>complexity</code>	Covariance complexity measure.
<code>lambda</code>	Non-negative information-criterion weight.
<code>seed</code>	Optional random seed.
<code>cov_args</code>	Named list of covariance-estimator arguments.
<code>stabilization_args</code>	Named list of stabilisation arguments.
<code>sdr_args</code>	Named list of SDR-kernel arguments.
<code>...</code>	Backward-compatible component arguments.

Value

A list containing selected dimensions, the aggregated cross-validation table, and fold-level results.

select_dimension_ladle

Ladle-type structural dimension diagnostic

Description

Computes a simple ladle-type diagnostic by combining normalised eigenvalue information with a bootstrap-based subspace instability measure.

Usage

```
select_dimension_ladle(
  X,
  y,
  sdr_method = c("dr", "sir", "save", "phd"),
  cov_method = c("sample", "ridge", "oas", "lw", "mec"),
  d_max = 10,
  B = 100,
```

```

    nslices = 6,
    standardize = TRUE,
    stabilize = TRUE,
    stabilization = c("eigenfloor", "ridge", "nearest_pd"),
    seed = NULL,
    cov_args = list(),
    stabilization_args = list(),
    sdr_args = list(),
    ...
)

```

Arguments

<code>X</code>	Numeric predictor matrix or data frame.
<code>y</code>	Numeric continuous response vector.
<code>sdr_method</code>	SDR method.
<code>cov_method</code>	Covariance estimator.
<code>d_max</code>	Maximum candidate structural dimension.
<code>B</code>	Number of bootstrap replications.
<code>nslices</code>	Number of slices.
<code>standardize</code>	Logical. If TRUE, standardises predictors.
<code>stabilize</code>	Logical. If TRUE, stabilises covariance matrix.
<code>stabilization</code>	Stabilisation method.
<code>seed</code>	Optional random seed.
<code>cov_args</code>	Named list of covariance-estimator arguments.
<code>stabilization_args</code>	Named list of stabilisation arguments.
<code>sdr_args</code>	Named list of SDR-kernel arguments.
<code>...</code>	Backward-compatible component arguments.

Value

A list containing ladle table, selected dimension, and bootstrap distances.

<code>simulate_risdr_data</code>	<i>Simulate SDR data</i>
----------------------------------	--------------------------

Description

Simulates data from a sufficient dimension reduction model with an autoregressive covariance structure. The function returns the true orthonormal central subspace basis, which allows simulation studies to evaluate both prediction accuracy and subspace recovery.

Usage

```
simulate_risdr_data(
  n = 200,
  p = 50,
  d = 2,
  rho = 0.6,
  sigma = 1,
  signal_strength = 1,
  model = c("linear_quadratic", "symmetric", "interaction", "linear"),
  beta_type = c("coordinate", "random_sparse", "random_dense"),
  beta = NULL,
  seed = NULL
)
```

Arguments

n	Sample size.
p	Number of predictors.
d	Structural dimension.
rho	AR(1) correlation parameter.
sigma	Error standard deviation.
signal_strength	Multiplicative strength of the regression signal.
model	Data-generating model.
beta_type	Type of true central subspace basis.
beta	Optional user-supplied p by d central subspace basis. When supplied, its orthonormal basis is used and beta_type is ignored.
seed	Optional random seed.

Value

A list containing the simulated predictors, response, true basis, latent sufficient predictors, population covariance matrix, signal, error, and simulation settings.

Examples

```
simulated <- simulate_risdr_data(
  n = 60,
  p = 6,
  d = 2,
  rho = 0.4,
  seed = 2026
)

dim(simulated$X)
head(simulated$y)
crossprod(simulated$beta)
```

slice_covariances	<i>Compute slice covariance matrices</i>
-------------------	--

Description

Computes slice-specific covariance matrices.

Usage

```
slice_covariances(  
  X,  
  slices,  
  stabilize = TRUE,  
  stabilization = c("eigenfloor", "ridge", "nearest_pd"),  
  eps = 1e-06  
)
```

Arguments

X	Numeric matrix.
slices	Integer slice memberships.
stabilize	Logical. If TRUE, stabilises each slice covariance matrix.
stabilization	Stabilisation method.
eps	Eigenvalue floor.

Value

A list of covariance matrices.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])  
slices <- make_slices(mtcars$mpg, nslices = 4)  
covariances <- slice_covariances(X, slices)  
names(covariances)
```

slice_means	<i>Compute slice means</i>
-------------	----------------------------

Description

Computes slice-specific means of predictors.

Usage

```
slice_means(X, slices)
```

Arguments

X	Numeric matrix.
slices	Integer slice memberships.

Value

Matrix of slice means with rows corresponding to slices.

Examples

```
X <- as.matrix(mtcars[, c("disp", "hp", "wt")])
slices <- make_slices(mtcars$mpg, nslices = 4)
slice_means(X, slices)
```

slice_proportions	<i>Compute slice proportions</i>
-------------------	----------------------------------

Description

Computes empirical slice probabilities.

Usage

```
slice_proportions(slices)
```

Arguments

slices	Integer slice memberships.
--------	----------------------------

Value

Numeric vector of slice proportions.

Examples

```
slices <- make_slices(mtcars$mpg, nslices = 4)
slice_proportions(slices)
```

slice_response	<i>Slice response for inverse regression</i>
----------------	--

Description

Constructs slice labels for continuous, categorical, or censored survival responses. For censored survival data, double slicing is used by first splitting on censoring status and then slicing observed time within each status group.

Usage

```
slice_response(  
  y,  
  nslices = 5L,  
  response_type = c("continuous", "categorical", "survival"),  
  delta = NULL  
)
```

Arguments

y	Response vector. For survival data, observed time.
nslices	Number of slices for continuous data or per censoring-status group for survival data.
response_type	One of "continuous", "categorical", or "survival".
delta	Optional 0/1 event indicator for survival data.

Value

A list containing integer slice labels and metadata.

slice_summary	<i>Summarise response slices</i>
---------------	----------------------------------

Description

Produces a summary table of slice frequencies and response ranges.

Usage

```
slice_summary(y, slices)
```

Arguments

y	Numeric response vector.
slices	Integer slice memberships.

Value

A data frame summarising slices.

Examples

```
slices <- make_slices(mtcars$mpg, nslices = 4)
slice_summary(mtcars$mpg, slices)
```

stabilize_cov

General covariance stabilisation dispatcher

Description

Stabilises a covariance matrix using one of several available methods.

Usage

```
stabilize_cov(
  Sigma,
  method = c("eigenfloor", "ridge", "nearest_pd"),
  eps = 1e-06,
  lambda = 1e-04,
  keep_diag = TRUE
)
```

Arguments

Sigma	A square symmetric covariance matrix.
method	Stabilisation method.
eps	Positive eigenvalue floor for eigenfloor method.
lambda	Ridge parameter for ridge method.
keep_diag	Logical. Used by nearest positive definite method.

Value

A stabilised covariance matrix.

Examples

```
Sigma <- matrix(c(1, 1, 1, 1), nrow = 2)
stabilize_cov(Sigma, method = "eigenfloor")
```

stabilize_eigenfloor	<i>Eigenvalue floor stabilisation</i>
----------------------	---------------------------------------

Description

Stabilises a symmetric covariance matrix by replacing eigenvalues smaller than a positive threshold with that threshold.

Usage

```
stabilize_eigenfloor(Sigma, eps = 1e-06)
```

Arguments

Sigma	A square symmetric covariance matrix.
eps	Positive eigenvalue floor.

Value

A symmetric positive definite covariance matrix.

Examples

```
Sigma <- matrix(c(1, 1, 1, 1), nrow = 2)
eigen(stabilize_eigenfloor(Sigma))$values
```

stabilize_nearest_pd	<i>Nearest positive definite stabilisation</i>
----------------------	--

Description

Stabilises a covariance matrix by projecting it to the nearest positive definite matrix using `Matrix::nearPD()`.

Usage

```
stabilize_nearest_pd(Sigma, keep_diag = TRUE)
```

Arguments

Sigma	A square symmetric covariance matrix.
keep_diag	Logical. If TRUE, keeps original diagonal where possible.

Value

A symmetric positive definite covariance matrix.

Examples

```
Sigma <- matrix(c(1, 1, 1, 1), nrow = 2)
eigen(stabilize_nearest_pd(Sigma))$values
```

stabilize_ridge	<i>Ridge stabilisation of covariance matrix</i>
-----------------	---

Description

Stabilises a covariance matrix by adding a positive multiple of the identity matrix.

Usage

```
stabilize_ridge(Sigma, lambda = 1e-04)
```

Arguments

Sigma	A square symmetric covariance matrix.
lambda	Ridge stabilisation parameter.

Value

A symmetric positive definite covariance matrix.

Examples

```
Sigma <- matrix(c(1, 1, 1, 1), nrow = 2)
eigen(stabilize_ridge(Sigma))$values
```

subspace_distance	<i>Subspace distance</i>
-------------------	--------------------------

Description

Computes Frobenius distance between projection matrices.

Usage

```
subspace_distance(B_hat, B)
```

Arguments

B_hat	Estimated basis matrix.
B	True basis matrix.

Value

Numeric subspace distance.

Examples

```
B <- matrix(c(1, 0, 0, 0, 1, 0), nrow = 3)
transformed <- B %*% diag(c(2, 3))
subspace_distance(transformed, B)
```

summarise_simulation	<i>Summarise simulation results</i>
----------------------	-------------------------------------

Description

Summarise simulation results

Usage

```
summarise_simulation(sim_results)
```

Arguments

sim_results Data frame from run_risdr_simulation().

Value

Summary data frame.

summarise_sparsity	<i>Summarise sparsity pattern</i>
--------------------	-----------------------------------

Description

Summarise sparsity pattern

Usage

```
summarise_sparsity(V, tol = 1e-08)
```

Arguments

V Direction matrix.
tol Threshold for active support.

Value

Sparsity summary.

summary.risdr	<i>Summarise risdr object</i>
---------------	-------------------------------

Description

Summarise risdr object

Usage

```
## S3 method for class 'risdr'
summary(object, ...)
```

Arguments

object	Object of class "risdr".
...	Additional arguments passed to internal methods.

Value

A list summary.

summary.risdr_dual	<i>Summary method for dual RISDR objects</i>
--------------------	--

Description

Summary method for dual RISDR objects

Usage

```
## S3 method for class 'risdr_dual'
summary(object, top_n = 20L, ...)
```

Arguments

object	Object of class risdr_dual.
top_n	Number of selected variables to print.
...	Additional arguments passed to internal methods.

summary.risdr_sparse	<i>Summarise risdr object</i>
----------------------	-------------------------------

Description

Summarise risdr object

Usage

```
## S3 method for class 'risdr_sparse'  
summary(object, ...)
```

Arguments

object	Object of class "risdr".
...	Additional arguments passed to internal methods.

Value

A list summary.

weighted_soft_threshold	<i>Weighted soft-threshold operator</i>
-------------------------	---

Description

Weighted soft-threshold operator

Usage

```
weighted_soft_threshold(z, lambda, weights = NULL, tol = 1e-10)
```

Arguments

z	Numeric vector.
lambda	Non-negative tuning parameter.
weights	Penalty weights.
tol	Numerical tolerance.

Value

Thresholded vector.

`weighted_sparsify_directions`*Weighted sparse directions from dense SDR directions*

Description

Weighted sparse directions from dense SDR directions

Usage

```
weighted_sparsify_directions(V, lambda, weights = NULL, tol = 1e-10)
```

Arguments

<code>V</code>	Dense direction matrix.
<code>lambda</code>	Non-negative tuning parameter.
<code>weights</code>	Variable-level penalty weights.
<code>tol</code>	Numerical tolerance.

Value

Sparse direction matrix.

Index

adjusted_r_squared, 5
ar1_covariance, 5

C1F, 6
choose_dimension, 6
compute_dr, 7
compute_ic, 8
compute_ic_values, 9
compute_information_criteria, 10
compute_penalty_weights, 10
compute_phd, 11
compute_save, 12
compute_scores, 13
compute_sdr, 13
compute_sir, 14
cov_complexity_C1, 15
cov_complexity_C1F, 16
cov_condition_number, 16
cov_diagnostics, 17
cov_effective_rank, 17
cov_lw, 18
cov_mec, 18
cov_oas, 19
cov_ridge, 20
cov_sample, 21
criterion_weights, 21

dual_pca_scores, 22

estimate_cov, 23
estimate_cov(), 33
evaluate_prediction, 24
evaluate_prediction_cv, 24
extract_loadings, 26

filter_low_variance, 26
fit_downstream_lm, 27
fit_risdr, 27
fit_risdr(), 4, 30, 31, 33, 35
fit_risdr_dual, 29

fit_risdr_dual(), 4
fit_risdr_realdata, 31
fit_risdr_realdata(), 30, 33, 35
fit_risdr_sparse, 32
fit_risdr_sparse(), 4, 30
fit_risdr_sparse_realdata, 34
fit_risdr_sparse_realdata(), 4
fit_sparse_grid, 35

graphics::barplot(), 40
graphics::matplot(), 39
graphics::plot(), 40–42

mae, 36
make_cv_folds, 37
make_slices, 37
mape, 38

plot_cv_dimension, 38
plot_dimension_selection, 39
plot_ladle, 39
plot_loadings, 40
plot_prediction, 40
plot_residuals, 41
plot_scee, 41
plot_sufficient, 42
plot_sufficient2d, 42
predict.risdr, 43
predict.risdr(), 4
predict.risdr_dual
 (predict.risdr_sparse), 44
predict.risdr_sparse, 44
predict_downstream_lm, 45
prediction_correlation, 45
prepare_survival_response, 46
print.risdr, 46
print.risdr_dual, 47
print.risdr_realdata, 47
print.risdr_sparse, 48
print.risdr_sparse_realdata, 48

`print.summary.risdr`, 49
`print.summary.risdr_sparse`, 49
`projection_matrix`, 50

`r_squared`, 52
`risdr` (`risdr`-package), 4
`risdr`-package, 4
`rmse`, 50
`run_one_simulation`, 51
`run_one_simulation()`, 52
`run_risdr_simulation`, 52

`select_dimension`, 53
`select_dimension_cv`, 54
`select_dimension_cv()`, 4
`select_dimension_cv_icomp`, 55
`select_dimension_cv_icomp()`, 4
`select_dimension_ladle`, 56
`select_dimension_ladle()`, 4
`simulate_risdr_data`, 57
`slice_covariances`, 59
`slice_means`, 60
`slice_proportions`, 60
`slice_response`, 61
`slice_summary`, 61
`stabilize_cov`, 62
`stabilize_eigenfloor`, 63
`stabilize_nearest_pd`, 63
`stabilize_ridge`, 64
`subspace_distance`, 64
`summarise_simulation`, 65
`summarise_sparsity`, 65
`summary.risdr`, 66
`summary.risdr_dual`, 66
`summary.risdr_sparse`, 67

`weighted_soft_threshold`, 67
`weighted_sparsify_directions`, 68