Package 'risdr'

July 28, 2026

Title Regularised and Information-Theoretic Sufficient Dimension Reduction

Version 0.3.1

Description Implements covariance-stabilised sufficient dimension reduction for continuous responses with 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>, Li (1992) <doi:10.1080/01621459.1992.10476258>, and Li and Wang (2007) <doi:10.1198/016214507000000536>. Covariance shrinkage follows Olorede and Yahya (2019) <doi:10.48550/arXiv.1909.13017>, Ledoit and Wolf (2004) <doi:10.1016/S0047-259X(03)00096-4> and Chen et al. (2010) <doi:10.1109/TSP.2010.2053029>.

License GPL (>= 3)

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

RoxygenNote 7.3.3

Imports stats, graphics, utils, MASS, Matrix

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

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

NeedsCompilation no

Author Kabir Olorede [aut, cre]

Maintainer Kabir Olorede <kabirolorede@gmail.com>

Repository CRAN

Date/Publication 2026-07-28 16:40:02 UTC

Contents

`risdr-package` *risdr: Regularised and Information-Theoretic Sufficient Dimension Reduction*

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_icomp()`, or `select_dimension_ladle()` for complementary structural-dimension diagnostics.

Scope

The verified modelling workflow supports continuous responses. Binary, multiclass, and censored survival extensions remain outside the supported modelling interface.

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` *Adjusted coefficient of determination*

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` *AR(1) covariance matrix*

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)
```

`choose_dimension` *Choose dimension from criteria table*

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 *Compute Directional Regression*

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_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

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

Value

Named numeric vector of criteria.

Examples

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

compute_phd *Compute Principal Hessian Directions*

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 *Compute SDR scores for new data*

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 *General SDR kernel dispatcher*

Description

General SDR kernel dispatcher

Usage

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

Arguments

X	Numeric predictor matrix.
y	Numeric response vector.
method	SDR method.
Sigma	Covariance matrix.
nslices	Number of slices.
...	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

`X`	Numeric predictor matrix.
`y`	Numeric response vector.
`Sigma`	Covariance matrix used for standardisation.
`nslices`	Number of response slices.
`slice_type`	Slicing strategy.
`eps`	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 *Covariance complexity C1*

Description

Computes Bozdogan-type maximal covariance complexity.

Usage

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

Arguments

Sigma	A covariance matrix.
eps	Eigenvalue floor.

Value

Numeric complexity value.

cov_complexity_C1F *Scale-invariant covariance complexity C1F*

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 *Covariance condition number*

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("disp", "hp", "wt")])
cov_condition_number(Sigma)
```

cov_diagnostics *Covariance eigenvalue summary*

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` *Effective covariance rank*

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

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

X	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

X	Numeric matrix or data frame.
lambda	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 *Sample covariance matrix*

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` *Information criterion weights*

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")
```

`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

X	Numeric matrix or data frame.
y	Optional response vector required for MEC.
method	Covariance estimator.
nslices	Number of slices for MEC.
response_type	Response type. One of "continuous", "categorical", or "survival".
delta	Optional event indicator for survival data.
...	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 *Evaluate predictions*

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.

---

extract_loadings *Extract SDR loadings*

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.

filter_low_variance *Filter low-variance predictors*

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 *Fit downstream regression model on SDR scores*

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` *Fit regularised and information-theoretic SDR model*

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

`X`	Numeric predictor matrix or data frame.
`y`	Numeric continuous response vector.
`sdr_method`	SDR method: "dr", "sir", "save", or "phd".
`cov_method`	Covariance estimator: "sample", "ridge", "oas", "lw", or "mec".
`stabilize`	Logical. If TRUE, stabilises the estimated covariance matrix.
`stabilization`	Stabilisation method.
`nslices`	Number of slices for inverse regression methods.
`d`	Optional structural dimension. If NULL, selected by criterion.
`d_max`	Maximum candidate structural dimension.
`selector`	Criterion for selecting d.
`standardize`	Logical. If TRUE, column-standardises X before covariance estimation.
`complexity`	Complexity measure for ICOMP: "C1" or "C1F".

cov_args	Named list of arguments for the selected covariance estimator.
stabilization_args	Named list of arguments for covariance stabilisation.
sdr_args	Named list of arguments for the selected SDR kernel.
...	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_realdata *Fit RISDR to real high-dimensional data*

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 fit_risdr().
...	Additional arguments passed to fit_risdr().

Value

Fitted RISDR workflow object.

---

mae *Mean absolute error*

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 *Mean absolute percentage error*

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 *Plot cross-validation dimension selection result*

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 *Plot dimension selection criteria*

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 *Plot ladle dimension diagnostic*

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 *Plot SDR loadings*

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 *Prediction plot*

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 *Residual plot*

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_scree *Scree plot of SDR eigenvalues*

Description

Scree plot of SDR eigenvalues

Usage

```
plot_scree(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 *Sufficient summary plot*

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 *Two-direction sufficient summary plot*

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` *Predict method for risdr objects*

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])
```

prediction_correlation *Prediction correlation*

Description

Prediction correlation

Usage

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

Arguments

y_true	Observed response values.
y_pred	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

fit_lm	Fitted linear model from fit_downstream_lm().
scores_new	Matrix of new SDR scores.
d	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_realdata *Print real-data RISDR workflow*

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.summary.risdr *Print summary of risdr object*

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.

projection_matrix *Projection matrix*

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 *Root mean squared error*

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` *Run one SDR simulation replication*

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 *Run SDR simulation study*

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 *Coefficient of determination*

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 *Select structural dimension*

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` *Cross-validation dimension selection for SDR*

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_icomp`

Complexity-aware cross-validation for structural dimension selection

---

Description

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

Usage

```
select_dimension_cv_icomp(
  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

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 cross-validation folds.
nslices	Number of response slices.
standardize	Logical. Standardise within each training fold.
stabilize	Logical. Stabilise the estimated covariance matrix.
stabilization	Covariance stabilisation method.
complexity	Covariance complexity measure.
lambda	Non-negative information-criterion weight.
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 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

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.
B	Number of bootstrap replications.
nslices	Number of slices.
standardize	Logical. If TRUE, standardises predictors.
stabilize	Logical. If TRUE, stabilises covariance matrix.
stabilization	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 ladle table, selected dimension, and bootstrap distances.

---

simulate_risdr_data *Simulate SDR data*

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` *Compute slice covariance matrices*

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 *Compute slice means*

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 *Compute slice proportions*

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_summary *Summarise response slices*

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 *Eigenvalue floor stabilisation*

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 *Nearest positive definite stabilisation*

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 *Ridge stabilisation of covariance matrix*

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 *Subspace distance*

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 *Summarise simulation results*

Description

Summarise simulation results

Usage

```
summarise_simulation(sim_results)
```

Arguments

sim_results	Data frame from run_risdr_simulation().

Value

Summary data frame.

`summary.risdr` *Summarise risdr object*

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.

Index