

Package: orthoDr (via r-universe)

October 22, 2026

Type Package

Title Semi-Parametric Dimension Reduction Models Using Orthogonality
Constrained Optimization

Version 2.0.0

Date 2026-05-23

Description Utilize an orthogonality constrained optimization algorithm of Wen & Yin (2013) <[DOI:10.1007/s10107-012-0584-1](https://doi.org/10.1007/s10107-012-0584-1)> to solve a variety of dimension reduction problems in the semiparametric framework, such as Ma & Zhu (2012) <[DOI:10.1080/01621459.2011.646925](https://doi.org/10.1080/01621459.2011.646925)>, Ma & Zhu (2013) <[DOI:10.1214/12-AOS1072](https://doi.org/10.1214/12-AOS1072)>, Sun, Zhu, Wang & Zeng (2017) <[doi:10.48550/arXiv.1704.05046](https://doi.org/10.48550/arXiv.1704.05046)> and Zhou & Zhu (2018+) <[doi:10.48550/arXiv.1802.06156](https://doi.org/10.48550/arXiv.1802.06156)>. It also serves as a general purpose optimization solver for problems with orthogonality constraints. Parallel computing for approximating the gradient is enabled through 'OpenMP'.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

NeedsCompilation yes

Imports Rcpp (>= 1.0.0), survival, dr, pracma, plot3D, rgl, MASS

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

LinkingTo Rcpp, RcppArmadillo

URL <https://github.com/rqzhu-aide/orthoDr>

BugReports <https://github.com/rqzhu-aide/orthoDr/issues>

RoxygenNote 8.0.0

Config/pak/sysreqs cmake libfreetype6-dev libglu1-mesa-dev make texlive libpng-dev libuv1-dev libgl1-mesa-dev zlib1g-dev

Repository <https://teazrq.r-universe.dev>
Date/Publication 2026-06-01 13:16:21 UTC
RemoteUrl <https://github.com/teazrq/orthodr>
RemoteRef HEAD
RemoteSha 16fc0f11015f62aa9111ce237269ee2a48fed587

Contents

CP_SIR	2
dist_cross	3
distance	4
hMave	5
kernel_weight	6
ortho_optim	7
orthoDr_pdose	9
orthoDr_reg	11
orthoDr_surv	14
predict.orthoDr	16
pSAVE	17
silverman	18
skcm.clinical	19
skcm.melgene	19
view_dr_surv	20
Index	22

CP_SIR	<i>Counting-Process Based Sliced Inverse Regression</i>
--------	---

Description

Fits the CP-SIR model for right-censored survival outcomes. This model is correct only under strong assumptions, but since it requires only an SVD, its solution is commonly used as the initial value in [orthoDr_surv](#) optimization.

Usage

```
CP_SIR(x, y, censor, bw = silverman(1, length(y)))
```

Arguments

- x A matrix (n x p) of features (continuous only).
- y A numeric vector of observed survival times.
- censor A numeric vector of censoring indicators (1 = event, 0 = censored).
- bw Numeric; kernel bandwidth for nonparametric estimation (one-dimensional). Default: Silverman’s rule via [silverman](#).

Value

A list with components:

values Eigenvalues of the estimation matrix.

vectors Estimated directions (columns), ordered by eigenvalues.

References

Sun, Q., Zhu, R., Wang, T. and Zeng, D. (2017). Counting process based dimension reduction method for censored outcomes. *arXiv preprint* doi:10.48550/arXiv.1704.05046.

See Also

[orthoDr_surv](#), [silverman](#)

Examples

```
# This is setting 1 in Sun et. al. (2017) with reduced sample size
library(MASS)
set.seed(1)
N = 200; P = 6
V=0.5^abs(outer(1:P, 1:P, "-"))
dataX = as.matrix(mvrnorm(N, mu=rep(0,P), Sigma=V))
failedEDR = as.matrix(c(1, 0.5, 0, 0, 0, rep(0, P-5)))
censorEDR = as.matrix(c(0, 0, 0, 1, 1, rep(0, P-5)))
T = rexp(N, exp(dataX %%% failedEDR))
C = rexp(N, exp(dataX %%% censorEDR - 1))
ndr = 1
Y = pmin(T, C)
Censor = (T < C)

# fit the model
cpsir.fit = CP_SIR(dataX, Y, Censor)
distance(failedEDR, cpsir.fit$vectors[, 1:ndr, drop = FALSE], "dist")
```

dist_cross

dist_cross

Description

Cross distance matrix. An extension to the dist() function. Calculate the Gaussian kernel distance between rows of X1 and rows of X2

Usage

```
dist_cross(x1, x2)
```

Arguments

x1	first data matrix
x2	second data matrix

Value

A distance matrix, with its (i, j)th element being the Gaussian kernel distance between ith row of X1 and jth row of X2.

Examples

```
# two matrices
set.seed(1)
x1 = matrix(rnorm(10), 5, 2)
x2 = matrix(rnorm(6), 3, 2)
dist_cross(x1, x2)
```

distance	<i>Distance Between Two Linear Subspaces</i>
----------	--

Description

Calculate a distance metric between two linear subspaces spanned by the columns of s1 and s2. Several metrics are supported.

Usage

```
distance(s1, s2, type = "dist", x = NULL)
```

Arguments

s1	A matrix whose columns span the first subspace.
s2	A matrix whose columns span the second subspace.
type	Character; the distance measure to use: "dist" Frobenius norm of the difference between projection matrices (default). "trace" Trace correlation (1 = identical). "canonical" Mean canonical correlation; requires x. "sine" Sine metric between subspaces.
x	A matrix of covariate values, required only when type = "canonical".

Value

A numeric scalar: the distance (or correlation) between the two subspaces. Interpretation depends on type.

Examples

```
# two spaces
failedR = as.matrix(cbind(c(1, 1, 0, 0, 0, 0),
                           c(0, 0, 1, -1, 0, 0)))
B = as.matrix(cbind(c(0.1, 1.1, 0, 0, 0, 0),
                    c(0, 0, 1.1, -0.9, 0, 0)))

distance(failedR, B, "dist")
distance(failedR, B, "trace")

N=300
P=6
dataX = matrix(rnorm(N*P), N, P)
distance(failedR, B, "canonical", dataX)
```

hMave

Hazard MAVE for Censored Survival Data

Description

An almost direct R translation of Xia, Zhang & Xu's (2010) hMAVE Matlab code, with additional options for supplying a custom initial value. This algorithm does *not* use the orthogonality constrained optimization framework of [ortho_optim](#).

Usage

```
hMave(x, y, censor, m0, B0 = NULL)
```

Arguments

x	A matrix (n x p) of features (continuous only).
y	A numeric vector of observed survival times.
censor	A numeric vector of censoring indicators (1 = event, 0 = censored).
m0	Integer; number of structural dimensions to estimate.
B0	An optional initial B matrix (p x m0). If NULL (default), the algorithm initializes from the eigenstructure of x.

Value

A list with components:

B The estimated direction matrix (p x m0).
 cv Leave-one-out cross-validation error.

References

Xia, Y., Zhang, D., & Xu, J. (2010). Dimension reduction and semiparametric estimation of survival models. *Journal of the American Statistical Association*, 105(489), 278–290. doi:10.1198/jasa.2009.tm09372.

Examples

```
# generate some survival data
set.seed(1)
P = 7
N = 150
dataX = matrix(runif(N*P), N, P)
failedR = as.matrix(cbind(c(1, 1.3, -1.3, 1, -0.5, 0.5, -0.5, rep(0, P-7))))
T = exp(dataX %*% failedR + rnorm(N))
C = runif(N, 0, 15)
Y = pmin(T, C)
Censor = (T < C)

# fit the model
hMave.fit = hMave(dataX, Y, Censor, 1)
```

kernel_weight	<i>kernel_weight</i>
---------------	----------------------

Description

Calculate the Gaussian kernel weights between rows of X1 and rows of X2

Usage

```
kernel_weight(x1, x2, kernel = "gaussian", dist = "euclidean")
```

Arguments

x1	first data matrix
x2	second data matrix
kernel	the kernel function, currently only using Gaussian kernel
dist	the distance metric, currently only using the Euclidean distance

Value

A distance matrix, with its (i, j)th element being the kernel weights for the ith row of X1 jth row of X2.

Examples

```
# two matrices
set.seed(1)
x1 = matrix(rnorm(10), 5, 2)
x2 = matrix(rnorm(6), 3, 2)
kernel_weight(x1, x2)
```

ortho_optim

*Orthogonality Constrained Optimization***Description**

A general-purpose optimization solver for problems with orthogonality constraints on the columns of B . Implements the curvilinear search algorithm of Wen & Yin (2013), which ensures that B remains on the Stiefel manifold $B^T B = I$ throughout optimization. If no analytical gradient is provided, a numerical approximation is used.

Usage

```
ortho_optim(
  B,
  fn,
  grad = NULL,
  ...,
  maximize = FALSE,
  control = list(),
  maxitr = 500,
  verbose = 0
)
```

Arguments

B	Initial B values. A matrix whose columns are subject to the orthogonality constraint. If the columns are not orthogonal, they will be orthogonalized via Gram-Schmidt before optimization begins.
fn	Objective function. The first argument must be B (a matrix). Should return a single numeric value.
grad	Gradient function. The first argument must be B . Should return a matrix of the same dimension as B . If <code>NULL</code> (default), the gradient is approximated numerically via finite differences.
...	Additional named arguments passed to <code>fn</code> and <code>grad</code> .
maximize	Logical; if <code>TRUE</code> , the solver maximizes <code>fn</code> instead of minimizing it. Default is <code>FALSE</code> .
control	A list of tuning parameters for the optimizer: rho Controls the linear approximation quality in the curvilinear search. Default: <code>1e-4</code>.

	eta	Factor for decreasing the step size in backtracking line search. Default: 0.2.
	gamma	Nonmonotone search parameter from Zhang & Hager (2004). Default: 0.85.
	tau	Initial step size for updating B. Default: 1e-3.
	epsilon	Step size for finite-difference gradient approximation (only used when grad = NULL). Default: 1e-6.
	btol	Tolerance for orthogonality deviation of B. Convergence is reached when $\ B^T B - I\ _F < \text{btol}$. Default: 1e-6.
	ftol	Tolerance for relative change in the objective value. Default: 1e-6.
	gtol	Tolerance for the gradient (projected onto the manifold). Default: 1e-6.
maxitr		Maximum number of iterations. Default: 500.
verbose		Integer; if > 0, prints iteration progress. Default: 0.

Value

An object of class `c("orthoDr", "fit", "optim")`, which is a list containing:

- `B` The optimized matrix (columns are orthonormal).
- `fn` The final objective function value.
- `fn_Seq` Numeric vector of objective values at each iteration (length `maxitr`, padded with zeros after convergence).
- `itr` Number of iterations performed.
- `converge` Convergence code.
- `method` "true gradient" or "approx. gradient".

References

Wen, Z. and Yin, W. (2013). A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2), 397–434. doi:[10.1007/s1010701205841](https://doi.org/10.1007/s1010701205841).

Zhang, H. and Hager, W. W. (2004). A nonmonotone line search technique and its application to unconstrained optimization. *SIAM Journal on Optimization*, 14(4), 1043–1056. doi:[10.1137/S1052623403426556](https://doi.org/10.1137/S1052623403426556).

Examples

```
# Eigenvalue problem: minimize -0.5 * tr(B'A B) s.t. B'B = I
library(pracma)
set.seed(1)
n <- 100; k <- 6
A <- matrix(rnorm(n * n), n, n)
A <- t(A) %*% A
B <- gramSchmidt(matrix(rnorm(n * k), n, k))$Q

fx <- function(B, A) -0.5 * sum(diag(t(B) %*% A %*% B))
gx <- function(B, A) -A %*% B
fit <- ortho_optim(B, fx, gx, A = A)
```

```

fx(fit$B, A)

# Compare with the analytical solution from eigen()
sol <- eigen(A)$vectors[, 1:k]
fx(sol, A)

```

orthoDr_pdose

*Personalized Dose Estimation via Dimension Reduction***Description**

Fits a personalized dose model using the direct learning or pseudo-direct learning method of Zhou & Zhu (2021). The direction matrix B identifies the subspace of covariates that informs optimal dosing.

Usage

```

orthoDr_pdose(
  x,
  a,
  r,
  ndr = 2,
  B.initial = NULL,
  bw = NULL,
  lambda = 0.1,
  K = sqrt(length(r)),
  method = c("direct", "pseudo_direct"),
  keep.data = FALSE,
  control = list(),
  maxitr = 500,
  verbose = 0,
  ncore = 0
)

```

Arguments

x	A matrix or data.frame of features (continuous only).
a	A numeric vector of observed dose levels.
r	A numeric vector of observed rewards (outcomes).
ndr	Number of directions to estimate.
B.initial	Initial B matrix. If NULL (default), the partial SAVE model (pSAVE) is used. If specified, must have <code>nrow(B.initial) == ncol(x)</code> and <code>ncol(B.initial) == ndr</code> . Will be orthogonalized via Gram-Schmidt.
bw	Kernel bandwidth, assuming each variable has unit variance. If NULL (default), Silverman's rule is used.

lambda	Penalty level for kernel ridge regression. If a vector of values is provided, GCV is used to select the best value. Default: 0.1.
K	Number of grid points for dose levels. Default: $\sqrt{\text{length}(r)}$.
method	The method: "direct" or "pseudo_direct". Default: "direct".
keep.data	Logical; if TRUE, the original data is stored in the returned object for prediction. Default: FALSE.
control	A list of optimizer tuning parameters (see ortho_optim for details): rho, eta, gamma, tau, epsilon, btol, ftol, gtol.
maxitr	Maximum number of iterations. Default: 500.
verbose	Integer; if > 0, prints iteration progress. Default: 0.
ncore	Integer; number of cores for parallel computation via OpenMP. 0 = auto-detect. Default: 0.

Value

An object of class `c("orthoDr", "fit", "pdose")`, which is a list containing:

`B` The estimated direction matrix (columns are orthonormal).

`fn` The final objective function value.

`itr` Number of iterations performed.

`converge` Convergence code.

`method` The method used ("direct" or "pseudo_direct").

`keep.data` Whether original data was retained.

References

Zhou, W. and Zhu, R. (2021). A parsimonious personalized dose model via dimension reduction. *Biometrika*, 108(3), 643–659. doi:[10.1093/biomet/asaa094](https://doi.org/10.1093/biomet/asaa094).

Examples

```
# generate personalized dose data
exampleset <- function(size, ncov) {
  X <- matrix(runif(size * ncov, -1, 1), ncol = ncov)
  A <- runif(size, 0, 2)
  Edr <- as.matrix(c(0.5, -0.5))
  D_opt <- X %*% Edr + 1
  mu <- 2 + 0.5 * (X %*% Edr) - 7 * abs(D_opt - A)
  R <- rnorm(length(mu), mu, 1)
  R <- R - min(R)
  list(X = X, A = A, R = R, D_opt = D_opt, mu = mu)
}

set.seed(123)
n <- 150; p <- 2; ndr <- 1
train <- exampleset(n, p)
test <- exampleset(500, p)
```

```
# direct learning method
orthofit <- orthoDr_pdose(train$X, train$A, train$R,
  ndr = ndr, lambda = 0.1, method = "direct",
  K = sqrt(n), keep.data = TRUE,
  maxitr = 150, verbose = 0, ncore = 2)

dose <- predict(orthofit, test$X)
mean((test$D_opt - dose$pred)^2)

# pseudo-direct learning method
orthofit <- orthoDr_pdose(train$X, train$A, train$R,
  ndr = ndr, lambda = seq(0.1, 0.2, 0.01),
  method = "pseudo_direct", K = as.integer(sqrt(n)),
  keep.data = TRUE, maxitr = 150, ncore = 2)

dose <- predict(orthofit, test$X)
mean((test$D_opt - dose$pred)^2)
```

orthoDr_reg

Semiparametric Dimension Reduction for Regression

Description

Fits a semiparametric sufficient dimension reduction model for continuous outcomes using the methods of Ma & Zhu (2012, 2013). The direction matrix B is estimated via orthogonality-constrained optimization on the Stiefel manifold, ensuring $B^\top B = I$.

Kernel density estimation operates on the scaled projections $\beta^\top x$, so the procedure is robust to the scale of x . However, raw feature values enter the Nadaraya–Watson estimator unscaled — users with features of very different variance should standardise x beforehand. The outcome y is internally scaled.

Usage

```
orthoDr_reg(
  x,
  y,
  method = c("sir", "save", "phd", "local", "seff"),
  ndr = 2,
  B.initial = NULL,
  bw = NULL,
  keep.data = FALSE,
  control = list(),
  maxitr = 500,
  verbose = 0,
  ncore = 0
)
```

Arguments

<code>x</code>	A numeric matrix of predictors (n observations $\times$ p features). Not automatically scaled.
<code>y</code>	A numeric vector of continuous outcomes (length n).
<code>method</code>	The dimension reduction method. One of "sir", "save", "phd", "local", or "seff". See Details for method-specific behaviour.
<code>ndr</code>	Integer; number of directions to estimate (default 2). Must be ≥ 1 and $\leq \text{ncol}(x)$. Values > 4 produce a warning.
<code>B.initial</code>	Initial B matrix ($p \times ndr$), or NULL (default). When NULL, <code>initB()</code> selects the best among SIR, SAVE, and PHD starters by minimising the squared norm of the estimating equation. Exception: for <code>method = "local"</code> with <code>ndr = 1</code> , a multi-start strategy runs LOCAL from all three inits and selects the result that maximises the nonparametric R^2 of the Nadaraya–Watson regression of Y on $B^T X$.
<code>bw</code>	X-kernel bandwidth. NULL (default) uses Silverman’s rule of thumb. Internal Y-kernel bandwidths are derived from the sample size and structural dimension — they are not user-tunable.
<code>keep.data</code>	Logical; if TRUE, the original data is stored in the returned object for use by <code>predict</code> . Default FALSE.
<code>control</code>	A named list of optimizer tolerances. Recognised entries and their defaults: <code>rho</code> 1e-4 — augmented Lagrangian penalty <code>eta</code> 0.2 — step-size decay factor <code>gamma</code> 0.85 — linesearch contraction <code>tau</code> 1e-3 — initial step-size adjustment <code>epsilon</code> 1e-6 — finite-difference perturbation <code>btol</code> 1e-6 — parameter-change tolerance <code>ftol</code> 1e-6 — objective-change tolerance <code>gtol</code> 1e-6 — gradient-norm tolerance
<code>maxitr</code>	Maximum number of iterations (default 500). Clamped to ≥ 5 internally.
<code>verbose</code>	Integer; set > 0 to print iteration progress and total elapsed time. Default 0.
<code>ncore</code>	Integer; number of CPU threads for parallel gradient approximation. 0 (default) uses all available threads.

Details**Methods:**

SIR Semiparametric sliced inverse regression (Ma & Zhu 2012).

SAVE Semiparametric sliced average variance estimation.

PHD Semiparametric principal Hessian directions.

Local Locally efficient estimator (Ma & Zhu 2013, Section~3.1) with a normal posited model for the conditional density. This yields a mean-regression estimator — it captures dependence of Y on $\beta^\top x$ through the conditional mean only, not higher moments. For $d = 1$ with default initialisation, a multi-start strategy runs LOCAL from SIR, SAVE, and PHD starters and selects the result that maximises the nonparametric R^2 of the Nadaraya–Watson regression of Y on $\beta^\top x$.

SEFF Semiparametric efficient estimator (Ma & Zhu 2013, Section 3.2). Uses a two-step procedure: (1) obtain a root- n consistent initial $\tilde{\beta}$ via SAVE; (2) estimate the nonparametric nuisance components (conditional density $\hat{\eta}_2$ and its derivative, plus $\hat{E}[x | \beta^\top x]$) once at $\tilde{\beta}$; (3) optimise the efficient score with those estimates held fixed.

Value

An object of class `c("orthoDr", "fit", "reg")`, a list with components:

`B` Estimated $p \times ndr$ direction matrix with orthonormal columns.

`fn` Final objective value.

`itr` Number of iterations performed.

`converge` Convergence code (0 = success).

`method` Method used (e.g., "sir").

`keep.data` Whether original data was stored.

If `keep.data = TRUE`, the list also contains `x`, `y`, and `bw` for prediction.

References

Ma, Y. and Zhu, L. (2012). A semiparametric approach to dimension reduction. *Journal of the American Statistical Association*, 107(497), 168–179. doi:[10.1080/01621459.2011.646925](https://doi.org/10.1080/01621459.2011.646925).

Ma, Y. and Zhu, L. (2013). Efficient estimation in sufficient dimension reduction. *Annals of Statistics*, 41(1), 250–268. doi:[10.1214/12AOS1072](https://doi.org/10.1214/12AOS1072).

See Also

[predict.orthoDr](#), [ortho_optim](#)

Examples

```
set.seed(1)
N <- 100; P <- 4
X <- matrix(rnorm(N * P), N, P)

# Mean model – SIR
Y <- -1 + X[, 1] + rnorm(N)
fit_sir <- orthoDr_reg(X, Y, ndr = 1, method = "sir")
fit_sir$B

# Variance model – PHD
Y <- -1 + X[, 1]^2 + rnorm(N)
fit_phd <- orthoDr_reg(X, Y, ndr = 1, method = "phd")
fit_phd$B

# Efficient estimator (SEFF) – works on both mean and variance models
fit_seff <- orthoDr_reg(X, Y, ndr = 1, method = "seff")
fit_seff$B

# Custom initial + prediction
```

```
B0 <- matrix(c(1, 0, 0, 0), 4, 1)
fit <- orthoDr_reg(X, Y, ndr = 1, B.initial = B0, keep.data = TRUE)
predict(fit, X[1:5, ])
```

orthoDr_surv	<i>Semiparametric Dimension Reduction for Censored Survival Outcomes</i>
--------------	--

Description

Fits a counting-process-based semiparametric dimension reduction (IR-CP) model for right-censored survival outcomes (Sun, Zhu, Wang & Zeng, 2017). Three estimating equations are available: "forward", "dn", and "dm".

Usage

```
orthoDr_surv(
  x,
  y,
  censor,
  method = "dm",
  ndr = ifelse(method == "forward", 1, 2),
  B.initial = NULL,
  bw = NULL,
  keep.data = FALSE,
  control = list(),
  maxitr = 500,
  verbose = 0,
  ncore = 0
)
```

Arguments

x	A matrix or data.frame of features. Columns are not automatically scaled to unit variance.
y	A numeric vector of observed survival times.
censor	A numeric vector of censoring indicators (1 = event, 0 = censored).
method	The estimating equation: "forward" (1-d model), "dn" (counting process), or "dm" (martingale). Default: "dm".
ndr	Number of directions to estimate. Default: 2 (automatically set to 1 for method = "forward").
B.initial	Initial B matrix. If NULL (default), the counting-process-based SIR model (CP_SIR) is used. If specified, must have <code>nrow(B.initial) == ncol(x)</code> and <code>ncol(B.initial) == ndr</code> . Will be orthogonalized via Gram-Schmidt.

bw	Kernel bandwidth, assuming each variable has unit variance. If NULL (default), Silverman's rule is used.
keep.data	Logical; if TRUE, the original data is stored in the returned object for prediction. Default: FALSE.
control	A list of optimizer tuning parameters (see ortho_optim for details): rho, eta, gamma, tau, epsilon, btol, ftol, gtol.
maxitr	Maximum number of iterations. Default: 500.
verbose	Integer; if > 0, prints iteration progress. Default: 0.
ncore	Integer; number of cores for parallel computation via OpenMP. 0 = auto-detect. Default: 0.

Value

An object of class `c("orthoDr", "fit", "surv")`, which is a list containing:

`B` The estimated direction matrix (columns are orthonormal).

`fn` The final objective function value.

`itr` Number of iterations performed.

`converge` Convergence code.

`method` The method used (e.g., "dm").

`keep.data` Whether original data was retained.

References

Sun, Q., Zhu, R., Wang, T. and Zeng, D. (2017). Counting process based dimension reduction method for censored outcomes. *arXiv preprint* [doi:10.48550/arXiv.1704.05046](#).

See Also

[orthoDr_pdose](#), [CP_SIR](#), [view_dr_surv](#)

Examples

```
# Setting 1 from Sun et al. (2017), reduced sample size
library(MASS)
set.seed(1)
N <- 200; P <- 6
V <- 0.5^abs(outer(1:P, 1:P, "-"))
dataX <- as.matrix(mvrnorm(N, mu = rep(0, P), Sigma = V))
failedEDR <- as.matrix(c(1, 0.5, 0, 0, 0, rep(0, P - 5)))
censorEDR <- as.matrix(c(0, 0, 0, 1, 1, rep(0, P - 5)))
T <- rexp(N, exp(dataX %*% failedEDR))
C <- rexp(N, exp(dataX %*% censorEDR - 1))
ndr <- 1
Y <- pmin(T, C)
Censor <- (T < C)

# forward model (1-d only)
```

```

forward.fit <- orthoDr_surv(dataX, Y, Censor, method = "forward")
distance(failedR, forward.fit$B, "dist")

# counting process model
dn.fit <- orthoDr_surv(dataX, Y, Censor, method = "dn", ndr = ndr)
distance(failedR, dn.fit$B, "dist")

# martingale model
dm.fit <- orthoDr_surv(dataX, Y, Censor, method = "dm", ndr = ndr)
distance(failedR, dm.fit$B, "dist")

```

predict.orthoDr	<i>predict.orthoDr</i>
-----------------	------------------------

Description

The prediction function for orthoDr fitted models

Usage

```

## S3 method for class 'orthoDr'
predict(object, testx, ...)

```

Arguments

object	A fitted orthoDr object
testx	Testing data
...	...

Value

The predicted object

Examples

```

# generate some survival data
N = 100; P = 4; dataX = matrix(rnorm(N*P), N, P)
Y = exp(-1 + dataX[,1] + rnorm(N))
Censor = rbinom(N, 1, 0.8)

# fit the model with keep.data = TRUE
orthoDr.fit = orthoDr_surv(dataX, Y, Censor, ndr = 1,
                          method = "dm", keep.data = TRUE)

#predict 10 new observations
predict(orthoDr.fit, matrix(rnorm(10*P), 10, P))

# generate some personalized dose scenario

```

```

exampleset <- function(size,ncov){

  X = matrix(runif(size*ncov,-1,1),ncol=ncov)
  A = runif(size,0,2)

  Edr = as.matrix(c(0.5,-0.5))

  D_opt = X %%% Edr + 1

  mu = 2 + 0.5*(X %%% Edr) - 7*abs(D_opt-A)

  R = rnorm(length(mu),mu,1)

  R = R - min(R)

  datainfo = list(X=X,A=A,R=R,D_opt=D_opt,mu=mu)
  return(datainfo)
}

# generate data

set.seed(123)
n = 150
p = 2
ndr =1
train = exampleset(n,p)
test = exampleset(500,p)

# the direct learning method
orthofit = orthoDr_pdose(train$X, train$A, train$R, ndr = ndr, lambda = 0.1,
                        method = "direct", K = as.integer(sqrt(n)), keep.data = TRUE,
                        maxitr = 150, verbose = FALSE, ncore = 2)

predict(orthofit,test$X)

# the pseudo direct learning method
orthofit = orthoDr_pdose(train$X, train$A, train$R, ndr = ndr, lambda = seq(0.1,0.2,0.01),
                        method = "pseudo_direct", K = as.integer(sqrt(n)), keep.data = TRUE,
                        maxitr = 150, verbose = FALSE, ncore = 2)

predict(orthofit,test$X)

```

Description

Fits the partial SAVE model for dose-response settings. This model relies on strong assumptions; its solution is primarily used as an initial value in [orthoDr_pdose](#) optimization.

Usage

```
pSAVE(x, a, r, ndr = 2, nslices0 = 2)
```

Arguments

<code>x</code>	A matrix (n x p) of features (continuous only).
<code>a</code>	A numeric vector of observed dose levels.
<code>r</code>	A numeric vector of rewards (outcomes).
<code>ndr</code>	Integer; number of structural dimensions to estimate. Default: 2.
<code>nslices0</code>	Integer; number of slices used for SAVE. Default: 2.

Value

A matrix whose columns are the estimated basis vectors of the central subspace, ordered by eigenvalues.

References

Feng, Z., Wen, M.X., Yu, Z. and Zhu, L. (2013). On partial sufficient dimension reduction with applications to partially linear multi-index models. *Journal of the American Statistical Association*, 108(501), 237–256. doi:[10.1080/01621459.2013.849167](https://doi.org/10.1080/01621459.2013.849167).

See Also

[orthoDr_pdose](#)

Examples

```
set.seed(1)
N <- 200; P <- 4
X <- matrix(rnorm(N * P), N, P)
dose <- runif(N, 0, 2)
reward <- X[, 1] + rnorm(N)
pSAVE(X, dose, reward, ndr = 1)
```

silverman

A simple Silverman bandwidth formula

Description

Compute the Silverman rule-of-thumb bandwidth for kernel density estimation.

Usage

```
silverman(d, n)
```

Arguments

d	Integer; number of dimensions.
n	Integer; number of observations.

Value

A numeric scalar: the Silverman bandwidth.

Examples

```
silverman(1, 300)
```

skcm.clinical	<i>skcm.clinical</i>
---------------	----------------------

Description

The clinical variables of the SKCM dataset. The original data was obtained from The Cancer Genome Atlas (TCGA).

Usage

```
skcm.clinical
```

Format

Contains 469 subjects with 156 failures. Each row contains one subject, subject ID is indicated by row name. Variables include Time, Censor, Gender and Age. Age has 8 missing values.

References

<https://www.cancer.gov/ccg/research/genome-sequencing/tcga>

skcm.melgene	<i>skcm.melgene</i>
--------------	---------------------

Description

The expression of top 20 genes of cutaneous melanoma literature based on the MelGene Database.

Usage

```
skcm.melgene
```

Format

Each row contains one subject, subject ID is indicated by row name. Gene names in the columns. The columns are scaled.

References

Chatzinasiou, Foteini, Christina M. Lill, Katerina Kypreou, Irene Stefanaki, Vasiliki Nicolaou, George Spyrou, Evangelos Evangelou et al. "Comprehensive field synopsis and systematic meta-analyses of genetic association studies in cutaneous melanoma." Journal of the National Cancer Institute 103, no. 16 (2011): 1227-1235.

<https://www.cancer.gov/ccg/research/genome-sequencing/tcga>

view_dr_surv	<i>2D or 3D view of survival data on reduced dimension</i>
--------------	--

Description

Produce 2D or 3D plots of right censored survival data based on a given dimension reduction space

Usage

```
view_dr_surv(
  x,
  y,
  censor,
  B = NULL,
  bw = NULL,
  FUN = "log",
  type = "2D",
  legend.add = TRUE,
  xlab = "Reduced Direction",
  ylab = "Time",
  zlab = "Survival"
)
```

Arguments

x	A matrix or data.frame for features (continuous only). The algorithm will not scale the columns to unit variance
y	A vector of observed time
censor	A vector of censoring indicator
B	The dimension reduction subspace, can only be 1 dimensional
bw	A Kernel bandwidth (3D plot only) for approximating the survival function, default is the Silverman's formula
FUN	A scaling function applied to the time points y. Default is "log".

type	2D or 3D plot
legend.add	Should legend be added (2D plot only)
xlab	x axis label
ylab	y axis label
zlab	z axis label

Value

Invisible NULL. Called for its side-effect (producing a plot).

References

Sun, Q., Zhu, R., Wang, T. and Zeng, D. (2017). Counting process based dimension reduction method for censored outcomes. *arXiv preprint* doi:[10.48550/arXiv.1704.05046](https://doi.org/10.48550/arXiv.1704.05046).

See Also

[orthoDr_surv](#), [distance](#)

Examples

```
# generate some survival data
N = 100; P = 4; dataX = matrix(rnorm(N*P), N, P)
Y = exp(-1 + dataX[,1]) + rnorm(N)
Censor = rbinom(N, 1, 0.8)

orthoDr.fit = orthoDr_surv(dataX, Y, Censor, ndr = 1, method = "dm")
view_dr_surv(dataX, Y, Censor, orthoDr.fit$B)
```

Index

- * **skcm.clinical**
 - skcm.clinical, [19](#)
- * **skcm.melgene**
 - skcm.melgene, [19](#)
- CP_SIR, [2](#), [14](#), [15](#)
- dist_cross, [3](#)
- distance, [4](#), [21](#)
- hMave, [5](#)
- kernel_weight, [6](#)
- ortho_optim, [5](#), [7](#), [10](#), [13](#), [15](#)
- orthoDr_pdose, [9](#), [15](#), [17](#), [18](#)
- orthoDr_reg, [11](#)
- orthoDr_surv, [2](#), [3](#), [14](#), [21](#)
- predict, [12](#)
- predict.orthoDr, [13](#), [16](#)
- pSAVE, [9](#), [17](#)
- silverman, [2](#), [3](#), [18](#)
- skcm.clinical, [19](#)
- skcm.melgene, [19](#)
- view_dr_surv, [15](#), [20](#)