Package ‘orthoDr’

March 13, 2024

Type Package

Title Semi-Parametric Dimension Reduction Models Using Orthogonality Constrained Optimization

Version 0.6.8

Description Utilize an orthogonality constrained optimization algorithm of Wen & Yin (2013) <DOI: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>, Ma & Zhu (2013) <DOI:10.1214/12-AOS1072>, Sun, Zhu, Wang & Zeng (2019) <DOI:10.1093/biomet/asy064> and Zhou, Zhu & Zeng (2021) <DOI:10.1093/biomet/asaa087>. The package also implements some existing dimension reduction methods such as hMave by Xia, Zhang, & Xu (2010) <DOI:10.1198/jasa.2009.tm09372> and partial SAVE by Feng, Wen & Zhu (2013) <DOI:10.1080/01621459.2012.746065>. It also serves as a general purpose optimization solver for problems with orthogonality constraints, i.e., in Stiefel manifold. Parallel computing for approximating the gradient is enabled through 'OpenMP'.

License GPL (>= 2)

Encoding UTF-8

LazyData TRUE

Imports Rcpp (>= 1.0.9), survival, dr, pracma, plot3D, rgl, MASS

LinkingTo Rcpp, RcppArmadillo

URL `https://github.com/teazrq/orthoDr`

BugReports `https://github.com/teazrq/orthoDr/issues`

RoxygenNote 7.3.1

NeedsCompilation yes

Author Ruilin Zhao [aut, cph],
Ruoqing Zhu [aut, cre, cph] (<`https://orcid.org/0000-0002-0753-5716`>),
Jiyang Zhang [aut, cph],
Wenzhuo Zhou [aut, cph],
Peng Xu [aut, cph],
James Joseph Balamuta [ctb] (<`https://orcid.org/0000-0003-2826-8458`>)

Maintainer Ruoqing Zhu <`teazrq@gmail.com`>

Repository CRAN

Date/Publication 2024-03-13 05:10:02 UTC

R topics documented:

`CP_SIR` *Counting process based sliced inverse regression model*

Description

The CP-SIR model for right-censored survival outcome. This model is correct only under very strong assumptions, however, since it only requires an SVD, the solution is used as the initial value in the orthoDr optimization.

Usage

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

Arguments

`x`	A matrix for features (continuous only).
`y`	A vector of observed time.
`censor`	A vector of censoring indicator.
`bw`	Kernel bandwidth for nonparametric estimations (one-dimensional), the default is using Silverman's formula.

Value

A `list` consisting of

`values`	The eigenvalues of the estimation matrix
`vectors`	The estimated directions, ordered by eigenvalues

References

Sun, Q., Zhu, R., Wang, T. and Zeng, D. (2019) "Counting Process Based Dimension Reduction Method for Censored Outcomes." Biometrika, 106(1), 181-196. DOI: doi:10.1093/biomet/asy064

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))
failEDR <- 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 %*% failEDR))
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(failEDR, cpsir.fit$vectors[, 1:ndr, drop = FALSE], "dist")
```

`distance` *Compute Distance Correlation*

Description

Calculate the distance correlation between two linear spaces.

Usage

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

Arguments

`s1`	First space
`s2`	Second space
`type`	Type of distance measures: `"dist"` (default), `"trace"`, `"canonical"` or `"sine"`
`x`	The covariate values, for canonical correlation only.

Value

The distance between s1 and s2.

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)
```

---

dist_cross *Cross distance matrix*

Description

Calculate the Gaussian kernel distance between rows of X1 and rows of X2. As a result, this is an extension to the stats::dist() function.

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 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)
```

`hMave` *Hazard Mave for Censored Survival Data*

Description

This is an almost direct R translation of Xia, Zhang & Xu's (2010) hMave `MATLAB` code. We implemented further options for setting a different initial value. The computational algorithm does not utilize the orthogonality constrained optimization.

Usage

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

Arguments

`x`	A matrix for features.
`y`	A vector of observed time.
`censor`	A vector of censoring indicator.
`m0`	number of dimensions to use
`B0`	initial value of B. This is a feature we implemented.

Value

A `list` consisting of

`B`	The estimated B matrix
`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: 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` *Kernel Weight*

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 *i* th 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)
```

`orthoDr_pdose` *Direct Learning & Pseudo-direct Learning Model*

Description

Performs the "Direct Learning & Pseudo-direct Learning" Method for personalized medicine.

Usage

```
orthoDr_pdose(
  x,
  a,
  r,
  ndr = ndr,
  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 = FALSE,
  ncore = 0
)
```

Arguments

`x`	A `matrix` or `data.frame` for features (continuous only).
`a`	A `vector` of observed dose
`r`	A `vector` of observed reward
`ndr`	A dimension structure
`B.initial`	Initial `B` values. Will use the partial SAVE pSAVE as the initial if leaving as `NULL`. If specified, must be a matrix with `ncol(x)` rows and `ndr` columns. Will be processed by Gram-Schmidt if not orthogonal.
`bw`	A Kernel bandwidth, assuming each variables have unit variance
`lambda`	The penalty level for kernel ridge regression. If a range of values is specified, the GCV will be used to select the best tuning
`K`	A number of grids in the range of dose
`method`	Either `"direct"` or `"pseudo_direct"`
`keep.data`	Should the original data be kept for prediction
`control`	A list of tuning variables for optimization. `epsilon` is the size for numerically approximating the gradient. For others, see Wen and Yin (2013).
`maxitr`	Maximum number of iterations

`verbose`	Should information be displayed
`ncore`	the number of cores for parallel computing

Value

A `orthoDr` object consisting of `list` with named elements:

`B`	The optimal `B` value
`fn`	The final functional value
`itr`	The number of iterations
`converge`	convergence code

References

Zhou, W., Zhu, R., & Zeng, D. (2021). A parsimonious personalized dose-finding model via dimension reduction. Biometrika, 108(3), 643-659. DOI: doi:10.1093/biomet/asaa087

Examples

```
# 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 = sqrt(n), keep.data = TRUE,
  maxitr = 150, verbose = FALSE, ncore = 2
)

dose <- predict(orthofit, test$X)

# ` # compare with the optimal dose
dosedistance <- mean((test$D_opt - dose$pred)^2)
print(dosedistance)

# 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
)

dose <- predict(orthofit, test$X)

# compare with the optimal dose

dosedistance <- mean((test$D_opt - dose$pred)^2)
print(dosedistance)
```

`orthoDr_reg` *Semiparametric dimension reduction method from Ma & Zhu (2012).*

Description

Performs the semiparametric dimension reduction method associated with Ma & Zhu (2012).

Usage

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

Arguments

x	A matrix or data.frame for features (continous only). The algorithm will not scale the columns to unit variance
y	A vector of continuous outcome
method	Dimension reduction methods (semi-): "sir", "save", "phd", "local" or "seff". Currently only "sir" and "phd" are available.
ndr	The number of directions
B.initial	Initial B values. If specified, must be a matrix with ncol(x) rows and ndr columns. Will be processed by Gram-Schmidt if not orthogonal. If the initial value is not given, three initial values ("sir", "save" and "phd") using the traditional method will be tested. The one with smallest l2 norm of the estimating equation will be used.
bw	A Kernel bandwidth, assuming each variables have unit variance
keep.data	Should the original data be kept for prediction. Default is FALSE.
control	A list of tuning variables for optimization. epsilon is the size for numerically approximating the gradient. For others, see Wen and Yin (2013).
maxitr	Maximum number of iterations
verbose	Should information be displayed
ncore	Number of cores for parallel computing. The default is the maximum number of threads.

Value

A orthoDr object consisting of list with named elements:

B	The optimal B value
fn	The final functional value
itr	The number of iterations
converge	convergence code

References

Ma, Y., & Zhu, L. (2012). A semiparametric approach to dimension reduction. Journal of the American Statistical Association, 107(497), 168-179. DOI: doi:10.1080/01621459.2011.646925

Ma, Y., & Zhu, L. (2013). Efficient estimation in sufficient dimension reduction. Annals of statistics, 41(1), 250. DOI: doi:10.1214/12AOS1072

Examples

```
# generate some regression data
set.seed(1)
N <- 100
P <- 4
dataX <- matrix(rnorm(N * P), N, P)
Y <- -1 + dataX[, 1] + rnorm(N)
```

```
# fit the semi-sir model
orthoDr_reg(dataX, Y, ndr = 1, method = "sir")

# fit the semi-phd model
Y <- -1 + dataX[, 1]^2 + rnorm(N)
orthoDr_reg(dataX, Y, ndr = 1, method = "phd")
```

`orthoDr_surv` *Counting Process based semiparametric dimension reduction (IR-CP) model*

Description

Models the data according to the counting process based semiparametric dimension reduction (IR-CP) model for right censored survival outcome.

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 = FALSE,
  ncore = 0
)
```

Arguments

`x`	A `matrix` or `data.frame` for features. The algorithm will not scale the columns to unit variance
`y`	A `vector` of observed time
`censor`	A `vector` of censoring indicator
`method`	Estimation equation to use. Either: `"forward"` (1-d model), `"dn"` (counting process), or `"dm"` (martingale).
`ndr`	The number of directions
`B.initial`	Initial `B` values. Will use the counting process based SIR model CP_SIR as the initial if leaving as `NULL`. If specified, must be a matrix with `ncol(x)` rows and `ndr` columns. Will be processed by Gram-Schmidt if not orthogonal.

`bw`	A Kernel bandwidth, assuming each variables have unit variance.
`keep.data`	Should the original data be kept for prediction. Default is `FALSE`
`control`	A list of tuning variables for optimization. `epsilon` is the size for numerically approximating the gradient. For others, see Wen and Yin (2013).
`maxitr`	Maximum number of iterations
`verbose`	Should information be displayed
`ncore`	Number of cores for parallel computing. The default is the maximum number of threads.

Value

A `orthoDr` object consisting of `list` with named elements:

`B`	The optimal `B` value
`fn`	The final functional value
`itr`	The number of iterations
`converge`	convergence code

References

Sun, Q., Zhu, R., Wang, T., & Zeng, D. (2019). Counting process-based dimension reduction methods for censored outcomes. Biometrika, 106(1), 181-196. DOI: doi:10.1093/biomet/asy064

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))
failEDR <- 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 %*% failEDR))
C <- rexp(N, exp(dataX %*% censorEDR - 1))
ndr <- 1
Y <- pmin(T, C)
Censor <- (T < C)

# fit the model
forward.fit <- orthoDr_surv(dataX, Y, Censor, method = "forward")
distance(failEDR, forward.fit$B, "dist")

dn.fit <- orthoDr_surv(dataX, Y, Censor, method = "dn", ndr = ndr)
distance(failEDR, dn.fit$B, "dist")

dm.fit <- orthoDr_surv(dataX, Y, Censor, method = "dm", ndr = ndr)
distance(failEDR, dm.fit$B, "dist")
```

`ortho_optim`	*Orthogonality constrained optimization*

Description

A general purpose optimization solver with orthogonality constraint. The orthogonality constrained optimization method is a nearly direct translation from Wen and Yin (2010)'s `MATLAB` code.

Usage

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

Arguments

`B`	Initial `B` values. Must be a matrix, and the columns are subject to the orthogonality constrains. Will be processed by Gram-Schmidt if not orthogonal
`fn`	A function that calculate the objective function value. The first argument should be `B`. Returns a single value.
`grad`	A function that calculate the gradient. The first argument should be `B`. Returns a matrix with the same dimension as `B`. If not specified, then numerical approximation is used.
`...`	Arguments passed to `fn` and `grad`
`maximize`	By default, the solver will try to minimize the objective function unless `maximize = TRUE`
`control`	A list of tuning variables for optimization. `epsilon` is the size for numerically approximating the gradient. For others, see Wen and Yin (2013).
`maxitr`	Maximum number of iterations
`verbose`	Should information be displayed

Value

A `orthoDr` object that consists of a `list` with named entries of:

`B`	The optimal `B` value
`fn`	The final functional value
`itr`	The number of iterations
`converge`	convergence code

References

Wen, Z., & Yin, W. (2013). A feasible method for optimization with orthogonality constraints. Mathematical Programming, 142(1), 397-434. DOI: doi:10.1007/s1010701205841

Examples

```
# an eigen value problem
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 solution from the eigen function
sol <- eigen(A)$vectors[, 1:k]
fx(sol, A)
```

`predict.orthoDr` *Predictions under* `orthoDr` *models*

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
`...`	Additional parameters, not used.

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)
```

print.orthoDr — *Print a* `orthoDr` *object*

Description

Provides a custom print wrapper for displaying `orthoDr` fitted models.

Usage

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

Arguments

`x`	A fitted `orthoDr` object
`...`	Additional parameters, not used.

Value

Sliently returns the `orthoDr` object supplied into the function to allow for use with pipes.

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
orthoDr_surv(dataX, Y, Censor, ndr = 1, method = "dm")
```

pSAVE *Partial Sliced Averaged Variance Estimation*

Description

The partial-SAVE model. This model is correct only under very strong assumptions, the solution is used as the initial value in the `orthoDr` optimization.

Usage

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

Arguments

`x`	A `matrix` for features (continuous only).
`a`	A `vector` of observed dose levels (continuous only).
`r`	A `vector` of reward (outcome).
`ndr`	The dimension structure
`nslices0`	Number of slides used for save

Value

A `list` consisting of:

`vectors`	The basis of central subspace, ordered by eigenvalues

References

Feng, Z., Wen, X. M., Yu, Z., & Zhu, L. (2013). On partial sufficient dimension reduction with applications to partially linear multi-index models. Journal of the American Statistical Association, 108(501), 237-246. DOI: doi:10.1080/01621459.2012.746065

silverman *Silverman's rule of thumb*

Description

A simple Silverman's rule of thumb bandwidth calculation.

Usage

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

Arguments

`d`	Number of dimension
`n`	Number of observation

Value

A simple bandwidth choice

Examples

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

`skcm.clinical` *Skin Cutaneous Melanoma Data set*

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`
- `Age`

Note: `Age` has 8 missing values.

References

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

skcm.melgene *Genes associated with Melanoma given by the MelGene Database*

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. Emmanouil I. Athanasiadis, Kyriaki Antonopoulou, Foteini Chatzinasiou, Christina M. Lill, Marilena M. Bourdakou, Argiris Sakellariou, Katerina Kypreou, Irene Stefanaki, Evangelos Evangelou, John P.A. Ioannidis, Lars Bertram, Alexander J. Stratigos, George M. Spyrou, A Web-based database of genetic association studies in cutaneous melanoma enhanced with network-driven data exploration tools, Database, Volume 2014, 2014, bau101, https://doi.org/10.1093/database/bau101 `https://www.cancer.gov/ccg/research/genome-sequencing/tcga`

view_dr_surv *2D or 2D view of survival data on reduced dimension*

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

An `rgl` object that is rendered.

References

Sun, Q., Zhu, R., Wang, T., & Zeng, D. (2019). Counting process-based dimension reduction methods for censored outcomes. Biometrika, 106(1), 181-196. DOI: doi:10.1093/biomet/asy064

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