# Package 'CVarE'

March 11, 2021

**Type** Package

**Title** Conditional Variance Estimator for Sufficient Dimension Reduction

**Version** 1.1

**Date** 2021-03-09

**Maintainer** Daniel Kapla <daniel@kapla.at>

**Author** Daniel Kapla [aut, cph, cre],
Lukas Fertl [aut, cph],
Efstathia Bura [ctb]

**Description** Implementation of the CVE (Conditional Variance Estimation) method proposed by Fertl, L. and Bura, E. (2021) <arXiv:2102.08782> and the ECVE (Ensemble Conditional Variance Estimation) method introduced in Fertl, L. and Bura, E. (2021) <arXiv:2102.13435>.
CVE and ECVE are sufficient dimension reduction methods in regressions with continuous response and predictors. CVE applies to general additive error regression models while ECVE generalizes to non-additive error regression models. They operate under the assumption that the predictors can be replaced by a lower dimensional projection without loss of information.
It is a semiparametric forward regression model based exhaustive sufficient dimension reduction estimation method that is shown to be consistent under mild assumptions.

**License** GPL-3

**Contact** <daniel@kapla.at>

**URL** https://git.art-ist.cc/daniel/CVE

**Encoding** UTF-8

**NeedsCompilation** yes

**Imports** stats, mda

**RoxygenNote** 7.0.2

**Repository** CRAN

**Date/Publication** 2021-03-11 15:00:06 UTC

# R topics documented:



---

`CVarE-package` *Conditional Variance Estimator (CVE) Package.*

---

## Description

Conditional Variance Estimation (CVE) is a novel sufficient dimension reduction (SDR) method for regressions satisfying $E(Y|X) = E(Y|B'X)$, where $B'X$ is a lower dimensional projection of the predictors and $Y$ is a univariate response. CVE, similarly to its main competitor, the mean average variance estimation (MAVE), is not based on inverse regression, and does not require the restrictive linearity and constant variance conditions of moment based SDR methods. CVE is data-driven and applies to additive error regressions with continuous predictors and link function. Let $X$ be a real $p$-dimensional covariate vector. We assume that the dependence of $Y$ and $X$ is modelled by

## Details

$$Y = g(B'X) + \epsilon$$

where $X$ is independent of $\epsilon$ with positive definite variance-covariance matrix $Var(X) = \Sigma_X$. $\epsilon$ is a mean zero random variable with finite $Var(\epsilon) = E(\epsilon^2)$, $g$ is an unknown, continuous non-constant function, and $B = (b_1, ..., b_k)$ is a real $p \times k$ matrix of rank $k \leq p$. Without loss of generality $B$ is assumed to be orthonormal.

Further, the extended Ensemble Conditional Variance Estimation (ECVE) is implemented which is a SDR method in regressions with continuous response and predictors. ECVE applies to general non-additive error regression models.

$$Y = g(B'X, \epsilon)$$

It operates under the assumption that the predictors can be replaced by a lower dimensional projection without loss of information.It is a semiparametric forward regression model based exhaustive sufficient dimension reduction estimation method that is shown to be consistent under mild assumptions.

## Author(s)

Daniel Kapla, Lukas Fertl, Bura Efstathia

## References

[1] Fertl, L. and Bura, E. (2021) "Conditional Variance Estimation for Sufficient Dimension Reduction" <arXiv:2102.08782>

[2] Fertl, L. and Bura, E. (2021) "Ensemble Conditional Variance Estimation for Sufficient Dimension Reduction" <arXiv:2102.13435>

## See Also

Useful links:

- https://git.art-ist.cc/daniel/CVE

---

`coef.cve` *Extracts estimated SDR basis.*

---

## Description

Returns the SDR basis matrix for dimension `k`, i.e. returns the cve-estimate of $B$ with dimension $p \times k$.

## Usage

```
## S3 method for class 'cve'
coef(object, k, ...)
```

## Arguments

| | |
|---|---|
| `object` | an object of class `"cve"`, usually, a result of a call to `cve` or `cve.call`. |
| `k` | the SDR dimension. |
| `...` | ignored (no additional arguments). |

## Value

The matrix $B$ of dimensions $p \times k$.

## Examples

```
# set dimensions for simulation model
p <- 8 # sample dimension
k <- 2 # real dimension of SDR subspace
n <- 100 # samplesize
# create B for simulation
b1 <- rep(1 / sqrt(p), p)
b2 <- (-1)^seq(1, p) / sqrt(p)
B <- cbind(b1, b2)

set.seed(21)
# creat predictor data x ~ N(0, I_p)
x <- matrix(rnorm(n * p), n, p)
# simulate response variable
#    y = f(B'x) + err
# with f(x1, x2) = x1^2 + 2 * x2 and err ~ N(0, 0.1^2)
y <- (x %*% b1)^2 + 2 * (x %*% b2) + 0.1 * rnorm(100)
# calculate cve for k = 2, 3
cve.obj <- cve(y ~ x, min.dim = 2, max.dim = 3)
# get cve-estimate for B with dimensions (p, k = 2)
B2 <- coef(cve.obj, k = 2)

# Projection matrix on span(B)
# equivalent to `B %*% t(B)` since B is semi-orthonormal
PB <- B %*% solve(t(B) %*% B) %*% t(B)
# Projection matrix on span(B2)
# equivalent to `B2 %*% t(B2)` since B2 is semi-orthonormal
PB2 <- B2 %*% solve(t(B2) %*% B2) %*% t(B2)
# compare estimation accuracy by Frobenius norm of difference of projections
norm(PB - PB2, type = 'F')
```

---

cve *Conditional Variance Estimator (CVE).*

---

## Description

This is the main function in the CVE package. It creates objects of class "cve" to estimate the mean subspace. Helper functions that require a "cve" object can then be applied to the output from this function.

Conditional Variance Estimation (CVE) is a sufficient dimension reduction (SDR) method for regressions studying $E(Y|X)$, the conditional expectation of a response $Y$ given a set of predictors $X$. This function provides methods for estimating the dimension and the subspace spanned by the columns of a $p \times k$ matrix $B$ of minimal rank $k$ such that

$$E(Y|X) = E(Y|B'X)$$

or, equivalently,

$$Y = g(B'X) + \epsilon$$

where $X$ is independent of $\epsilon$ with positive definite variance-covariance matrix $Var(X) = \Sigma_X$. $\epsilon$ is a mean zero random variable with finite $Var(\epsilon) = E(\epsilon^2)$, $g$ is an unknown, continuous non-constant function, and $B = (b_1, ..., b_k)$ is a real $p \times k$ matrix of rank $k \leq p$.

Both the dimension $k$ and the subspace $span(B)$ are unknown. The CVE method makes very few assumptions.

A kernel matrix $\hat{B}$ is estimated such that the column space of $\hat{B}$ should be close to the mean subspace $span(B)$. The primary output from this method is a set of orthonormal vectors, $\hat{B}$, whose span estimates $span(B)$.

The method central implements the Ensemble Conditional Variance Estimation (ECVE) as described in [2]. It augments the CVE method by applying an ensemble of functions (parameter `func_list`) to the response to estimate the central subspace. This corresponds to the generalization

$$F(Y|X) = F(Y|B'X)$$

or, equivalently,

$$Y = g(B'X, \epsilon)$$

where $F$ is the conditional cumulative distribution function.

## Usage

```
cve(formula, data, method = "mean", max.dim = 10L, ...)
```

## Arguments

| | |
|---|---|
| `formula` | an object of class `"formula"` which is a symbolic description of the model to be fitted like $Y \sim X$ where $Y$ is a $n$-dimensional vector of the response variable and $X$ is a $n \times p$ matrix of the predictors. |
| `data` | an optional data frame, containing the data for the formula if supplied like `data <-data.frame(Y,X)` with dimension $n \times (p+1)$. By default the variables are taken from the environment from which `cve` is called. |
| `method` | This character string specifies the method of fitting. The options are<br>• `"mean"` method to estimate the mean subspace, see [1].<br>• `"central"` ensemble method to estimate the central subspace, see [2].<br>• `"weighted.mean"` variation of `"mean"` method with adaptive weighting of slices, see [1].<br>• `"weighted.central"` variation of `"central"` method with adaptive weighting of slices, see [2]. |
| `max.dim` | upper bounds for `k`, (ignored if `k` is supplied). |
| `...` | optional parameters passed on to `cve.call`. |

## Value

an S3 object of class `cve` with components:

**X** design matrix of predictor vector used for calculating cve-estimate,

**Y** $n$-dimensional vector of responses used for calculating cve-estimate,

**method** Name of used method,

**call** the matched call,

**res** list of components `V,L,B,loss,h` for each `k = min.dim,...,max.dim`. If `k` was supplied in the call `min.dim = max.dim = k`.

- `B` is the cve-estimate with dimension $p \times k$.
- `V` is the orthogonal complement of $B$.
- `L` is the loss for each sample seperatels such that it's mean is `loss`.
- `loss` is the value of the target function that is minimized, evaluated at $V$.
- `h` bandwidth parameter used to calculate `B,V,loss,L`.

## References

[1] Fertl, L. and Bura, E. (2021) "Conditional Variance Estimation for Sufficient Dimension Reduction" <arXiv:2102.08782>

[2] Fertl, L. and Bura, E. (2021) "Ensemble Conditional Variance Estimation for Sufficient Dimension Reduction" <arXiv:2102.13435>

## See Also

For a detailed description of `formula` see `formula`.

## Examples

```
# set dimensions for simulation model
p <- 5
k <- 2
# create B for simulation
b1 <- rep(1 / sqrt(p), p)
b2 <- (-1)^seq(1, p) / sqrt(p)
B <- cbind(b1, b2)
# sample size
n <- 100
set.seed(21)

# creat predictor data x ~ N(0, I_p)
x <- matrix(rnorm(n * p), n, p)
# simulate response variable
#     y = f(B'x) + err
# with f(x1, x2) = x1^2 + 2 * x2 and err ~ N(0, 0.25^2)
y <- (x %*% b1)^2 + 2 * (x %*% b2) + 0.25 * rnorm(n)

# calculate cve with method 'mean' for k unknown in 1, ..., 3
cve.obj.s <- cve(y ~ x, max.dim = 2) # default method 'mean'
```

```
# calculate cve with method 'weighed' for k = 2
cve.obj.w <- cve(y ~ x, k = 2, method = 'weighted.mean')
B2 <- coef(cve.obj.s, k = 2)

# get projected X data (same as cve.obj.s$X %*% B2)
proj.X <- directions(cve.obj.s, k = 2)
#  plot y against projected data
plot(proj.X[, 1], y)
plot(proj.X[, 2], y)

# creat 10 new x points and y according to model
x.new <- matrix(rnorm(10 * p), 10, p)
y.new <- (x.new %*% b1)^2 + 2 * (x.new %*% b2) + 0.25 * rnorm(10)
# predict y.new
yhat <- predict(cve.obj.s, x.new, 2)
plot(y.new, yhat)

# projection matrix on span(B)
# same as B %*% t(B) since B is semi-orthogonal
PB <- B %*% solve(t(B) %*% B) %*% t(B)
# cve estimates for B with mean and weighted method
B.s <- coef(cve.obj.s, k = 2)
B.w <- coef(cve.obj.w, k = 2)
# same as B.s %*% t(B.s) since B.s is semi-orthogonal (same vor B.w)
PB.s <- B.s %*% solve(t(B.s) %*% B.s) %*% t(B.s)
PB.w <- B.w %*% solve(t(B.w) %*% B.w) %*% t(B.w)
# compare estimation accuracy of mean and weighted cve estimate by
# Frobenius norm of difference of projections.
norm(PB - PB.s, type = 'F')
norm(PB - PB.w, type = 'F')
```

---

`cve.call` *Conditional Variance Estimator (CVE).*

---

## Description

This is the main function in the `CVE` package. It creates objects of class `"cve"` to estimate the mean subspace. Helper functions that require a `"cve"` object can then be applied to the output from this function.

Conditional Variance Estimation (CVE) is a sufficient dimension reduction (SDR) method for regressions studying $E(Y|X)$, the conditional expectation of a response $Y$ given a set of predictors $X$. This function provides methods for estimating the dimension and the subspace spanned by the columns of a $p \times k$ matrix $B$ of minimal rank $k$ such that

$$E(Y|X) = E(Y|B'X)$$

or, equivalently,

$$Y = g(B'X) + \epsilon$$

where $X$ is independent of $\epsilon$ with positive definite variance-covariance matrix $Var(X) = \Sigma_X$. $\epsilon$ is a mean zero random variable with finite $Var(\epsilon) = E(\epsilon^2)$, $g$ is an unknown, continuous non-constant function, and $B = (b_1, ..., b_k)$ is a real $p \times k$ matrix of rank $k \leq p$.

Both the dimension $k$ and the subspace $span(B)$ are unknown. The CVE method makes very few assumptions.

A kernel matrix $\hat{B}$ is estimated such that the column space of $\hat{B}$ should be close to the mean subspace $span(B)$. The primary output from this method is a set of orthonormal vectors, $\hat{B}$, whose span estimates $span(B)$.

The method central implements the Ensemble Conditional Variance Estimation (ECVE) as described in [2]. It augments the CVE method by applying an ensemble of functions (parameter `func_list`) to the response to estimate the central subspace. This corresponds to the generalization

$$F(Y|X) = F(Y|B'X)$$

or, equivalently,

$$Y = g(B'X, \epsilon)$$

where $F$ is the conditional cumulative distribution function.

## Usage

```
cve.call(
  X,
  Y,
  method = c("mean", "weighted.mean", "central", "weighted.central"),
  func_list = NULL,
  nObs = sqrt(nrow(X)),
  h = NULL,
  min.dim = 1L,
  max.dim = 10L,
  k = NULL,
  momentum = 0,
  tau = 1,
  tol = 0.001,
  slack = 0,
  gamma = 0.5,
  V.init = NULL,
  max.iter = 50L,
  attempts = 10L,
  nr.proj = 1L,
  logger = NULL
)
```

## Arguments

| | |
|---|---|
| X | Design predictor matrix. |
| Y | $n$-dimensional vector of responses. |
| method | This character string specifies the method of fitting. The options are<br>• "mean" method to estimate the mean subspace, see [1].<br>• "central" ensemble method to estimate the central subspace, see [2].<br>• "weighted.mean" variation of "mean" method with adaptive weighting of slices, see [1].<br>• "weighted.central" variation of "central" method with adaptive weighting of slices, see [2]. |
| func_list | a list of functions applied to Y used by ECVE (see [2]) for central subspace estimation. The default ensemble are indicator functions of the $[0, 10], (10, 20], ..., (90, 100]$ percent response quantiles. (only relevant if method is "central" or "weighted.central", ignored otherwise) |
| nObs | parameter for choosing bandwidth h using estimate.bandwidth (ignored if h is supplied). |
| h | bandwidth or function to estimate bandwidth, defaults to internaly estimated bandwidth. |
| min.dim | lower bounds for k, (ignored if k is supplied). |
| max.dim | upper bounds for k, (ignored if k is supplied). |
| k | Dimension of lower dimensional projection, if k is given only the specified dimension B matrix is estimated. |
| momentum | number of $[0, 1)$ giving the ration of momentum for eucledian gradient update with a momentum term. momentum = 0 corresponds to normal gradient descend. |
| tau | Initial step-size. |
| tol | Tolerance for break condition. |
| slack | Positive scaling to allow small increases of the loss while optimizing, i.e. slack = 0.1 allows the target function to increase up to $10\%$ in one optimization step. |
| gamma | step-size reduction multiple. If gradient step with step size tau is not accepted gamma * tau is set to the next step size. |
| V.init | Semi-orthogonal matrix of dimensions '(ncol(X), ncol(X) - k) used as starting value in the optimization. (If supplied, attempts is set to 0 and k to match dimension). |
| max.iter | maximum number of optimization steps. |
| attempts | If V.init not supplied, the optimization is carried out attempts times with starting values drawn from the invariant measure on the Stiefel manifold (see rStiefel). |
| nr.proj | The number of projection used for projective resampling for multivariate response $Y$ (under active development, ignored for univariate response). |
| logger | a logger function (only for advanced users, slows down the computation). |

## Value

an S3 object of class `cve` with components:

**X** design matrix of predictor vector used for calculating cve-estimate,

**Y** $n$-dimensional vector of responses used for calculating cve-estimate,

**method** Name of used method,

**call** the matched call,

**res** list of components `V,L,B,loss,h` for each `k = min.dim,...,max.dim`. If `k` was supplied in the call `min.dim = max.dim = k`.

- `B` is the cve-estimate with dimension $p \times k$.
- `V` is the orthogonal complement of $B$.
- `L` is the loss for each sample seperatels such that it's mean is `loss`.
- `loss` is the value of the target function that is minimized, evaluated at $V$.
- `h` bandwidth parameter used to calculate `B,V,loss,L`.

## References

[1] Fertl, L. and Bura, E. (2021) "Conditional Variance Estimation for Sufficient Dimension Reduction" <arXiv:2102.08782>

[2] Fertl, L. and Bura, E. (2021) "Ensemble Conditional Variance Estimation for Sufficient Dimension Reduction" <arXiv:2102.13435>

## Examples

```
# create B for simulation (k = 1)
B <- rep(1, 5) / sqrt(5)

set.seed(21)
# creat predictor data X ~ N(0, I_p)
X <- matrix(rnorm(500), 100, 5)
# simulate response variable
#     Y = f(B'X) + err
# with f(x1) = x1 and err ~ N(0, 0.25^2)
Y <- X %*% B + 0.25 * rnorm(100)

# calculate cve with method 'simple' for k = 1
set.seed(21)
cve.obj.simple1 <- cve(Y ~ X, k = 1)

# same as
set.seed(21)
cve.obj.simple2 <- cve.call(X, Y, k = 1)

# extract estimated B's.
coef(cve.obj.simple1, k = 1)
coef(cve.obj.simple2, k = 1)
```

`dataset` *Generates test datasets.*

## Description

Provides sample datasets M1-M7 used in the paper Conditional variance estimation for sufficient dimension reduction, Lukas Fertl, Efstathia Bura. The general model is given by:

$$Y = g(B'X) + \epsilon$$

## Usage

```
dataset(name = "M1", n = NULL, p = 20, sd = 0.5, ...)
```

## Arguments

| | |
|---|---|
| `name` | One of `"M1"`, `"M2"`, `"M3"`, `"M4"`, `"M5"`, `"M6"` or `"M7"`. Alternative just the dataset number 1-7. |
| `n` | number of samples. |
| `p` | Dimension of random variable $X$. |
| `sd` | standard diviation for error term $\epsilon$. |
| `...` | Additional parameters only for "M2" (namely `pmix` and `lambda`), see: below. |

## Value

List with elements

- Xdata, a $n \times p$ matrix.
- Yresponse.
- Bthe dim-reduction matrix
- nameName of the dataset (name parameter)

## M1

The predictors are distributed as $X \sim N_p(0, \Sigma)$ with $\Sigma_{i,j} = 0.5^{|i-j|}$ for $i, j = 1, ..., p$ for a subspace dimension of $k = 1$ with a default of $n = 100$ data points. $p = 20$, $b_1 = (1, 1, 1, 1, 1, 1, 0, ..., 0)'/\sqrt{6} \in \mathcal{R}^p$, and $Y$ is given as

$$Y = cos(b_1'X) + \epsilon$$

where $\epsilon$ is distributed as generalized normal distribution with location 0, shape-parameter 0.5, and the scale-parameter is chosen such that $Var(\epsilon) = 0.5$.

**M2**

The predictors are distributed as $X \sim Z1_p\lambda + N_p(0, I_p)$. with $Z \sim 2Binom(p_{mix}) - 1 \in \{-1, 1\}$ where $1_p$ is the $p$-dimensional vector of one's, for a subspace dimension of $k = 1$ with a default of $n = 100$ data points. $p = 20$, $b_1 = (1, 1, 1, 1, 1, 1, 0, ..., 0)'/\sqrt{6} \in \mathcal{R}^p$, and $Y$ is

$$Y = cos(b_1'X) + 0.5\epsilon$$

where $\epsilon$ is standard normal. Defaults for `pmix` is 0.3 and `lambda` defaults to 1.

**M3**

The predictors are distributed as $X \sim N_p(0, I_p)$ for a subspace dimension of $k = 1$ with a default of $n = 100$ data points. $p = 20$, $b_1 = (1, 1, 1, 1, 1, 1, 0, ..., 0)'/\sqrt{6} \in \mathcal{R}^p$, and $Y$ is

$$Y = 2log(|b_1'X| + 2) + 0.5\epsilon$$

where $\epsilon$ is standard normal.

**M4**

The predictors are distributed as $X \sim N_p(0, \Sigma)$ with $\Sigma_{i,j} = 0.5^{|i-j|}$ for $i, j = 1, ..., p$ for a subspace dimension of $k = 2$ with a default of $n = 100$ data points. $p = 20$, $b_1 = (1, 1, 1, 1, 1, 1, 0, ..., 0)'/\sqrt{6} \in \mathcal{R}^p$, $b_2 = (1, -1, 1, -1, 1, -1, 0, ..., 0)'/\sqrt{6} \in \mathcal{R}^p$ and $Y$ is given as

$$Y = \frac{b_1'X}{0.5 + (1.5 + b_2'X)^2} + 0.5\epsilon$$

where $\epsilon$ is standard normal.

**M5**

The predictors are distributed as $X \sim U([0, 1]^p)$ where $U([0, 1]^p)$ is the uniform distribution with independent components on the $p$-dimensional hypercube for a subspace dimension of $k = 2$ with a default of $n = 200$ data points. $p = 20$, $b_1 = (1, 1, 1, 1, 1, 1, 0, ..., 0)'/\sqrt{6} \in \mathcal{R}^p$, $b_2 = (1, -1, 1, -1, 1, -1, 0, ..., 0)'/\sqrt{6} \in \mathcal{R}^p$ and $Y$ is given as

$$Y = cos(\pi b_1'X)(b_2'X + 1)^2 + 0.5\epsilon$$

where $\epsilon$ is standard normal.

**M6**

The predictors are distributed as $X \sim N_p(0, I_p)$ for a subspace dimension of $k = 3$ with a default of $n = 200$ data point. $p = 20, b_1 = e_1, b_2 = e_2$, and $b_3 = e_p$, where $e_j$ is the $j$-th unit vector in the $p$-dimensional space. $Y$ is given as

$$Y = (b_1'X)^2 + (b_2'X)^2 + (b_3'X)^2 + 0.5\epsilon$$

where $\epsilon$ is standard normal.

### M7

The predictors are distributed as $X \sim t_3(I_p)$ where $t_3(I_p)$ is the standard multivariate t-distribution with 3 degrees of freedom, for a subspace dimension of $k = 4$ with a default of $n = 200$ data points. $p = 20, b_1 = e_1, b_2 = e_2, b_3 = e_3$, and $b_4 = e_p$, where $e_j$ is the $j$-th unit vector in the $p$-dimensional space. $Y$ is given as

$$Y = (b_1'X)(b_2'X)^2 + (b_3'X)(b_4'X) + 0.5\epsilon$$

where $\epsilon$ is distributed as generalized normal distribution with location 0, shape-parameter 1, and the scale-parameter is chosen such that $Var(\epsilon) = 0.25$.

## References

Fertl, L. and Bura, E. (2021) "Conditional Variance Estimation for Sufficient Dimension Reduction" <arXiv:2102.08782>

---

`directions.cve` *Computes projected training data* `X` *for given dimension 'k'.*

---

## Description

Returns $B'X$. That is, it computes the projection of the $nxp$ design matrix $X$ on the column space of $B$ of dimension $k$.

## Usage

```
## S3 method for class 'cve'
directions(object, k, ...)
```

## Arguments

| | |
|---|---|
| `object` | an object of class `"cve"`, usually, a result of a call to `cve` or `cve.call`. |
| `k` | SDR dimension to use for projection. |
| `...` | ignored (no additional arguments). |

## Value

the $n \times k$ dimensional matrix $XB$ where $B$ is the cve-estimate for dimension $k$.

## See Also

`cve`

## Examples

```
# create B for simulation (k = 1)
B <- rep(1, 5) / sqrt(5)
set.seed(21)
# creat predictor data x ~ N(0, I_p)
x <- matrix(rnorm(500), 100, 5)
# simulate response variable
#    y = f(B'x) + err
# with f(x1) = x1 and err ~ N(0, 0.25^2)
y <- x %*% B + 0.25 * rnorm(100)
# calculate cve with method 'mean' for k = 1
set.seed(21)
cve.obj.mean <- cve(y ~ x, k = 1, method = 'mean')
# get projected data for k = 1
x.proj <- directions(cve.obj.mean, k = 1)
# plot y against projected data
plot(x.proj, y)
```

---

`estimate.bandwidth` *Bandwidth estimation for CVE.*

---

## Description

If no bandwidth or function for calculating it is supplied, the CVE method defaults to using the following formula (version 1)

$$h = \frac{2tr(\Sigma)}{p}(1.2n^{\frac{-1}{4+k}})^2$$

Alternative version 2 is used for dimension prediction which is given by

$$h = \frac{2tr(\Sigma)}{p}\chi_k^{-1}(\frac{nObs-1}{n-1})$$

with $n$ the sample size, $p$ the dimension of $X$ and $\Sigma$ is $(n-1)/n$ times the sample covariance matrix of $X$.

## Usage

```
estimate.bandwidth(X, k, nObs, version = 1L)
```

## Arguments

| | |
|---|---|
| `X` | the $n \times p$ matrix of predictor values. |
| `k` | the SDR dimension. |
| `nObs` | number of points in a slice, only for version 2. |
| `version` | either 1 or 2. |

## Value

Estimated bandwidth h.

## Examples

```
# set dimensions for simulation model
p <- 5; k <- 1
# create B for simulation
B <- rep(1, p) / sqrt(p)
# samplsize
n <- 100
set.seed(21)
#creat predictor data x ~ N(0, I_p)
x <- matrix(rnorm(n * p), n, p)
# simulate response variable
#     y = f(B'x) + err
# with f(x1) = x1 and err ~ N(0, 0.25^2)
y <- x %*% B + 0.25 * rnorm(100)
# calculate cve with method 'simple' for k = 1
set.seed(21)
cve.obj.simple <- cve(y ~ x, k = k)
print(estimate.bandwidth(x, k = k))
```

---

# predict.cve *Predict method for CVE Fits.*

---

## Description

Predict response using projected data. The forward model $g(B'X)$ is estimated with `mars` in the **mda** package.

## Usage

```
## S3 method for class 'cve'
predict(object, newdata, k, ...)
```

## Arguments

| | |
|---|---|
| `object` | an object of class `"cve"`, usually, a result of a call to `cve` or `cve.call`. |
| `newdata` | Matrix of new predictor values, $C$. |
| `k` | dimension of SDR space to be used for data projection. |
| `...` | further arguments passed to `mars`. |

## Value

prediced respone(s) for `newdata`.

## See Also

cve, cve.call and **mars**.

## Examples

```
# create B for simulation
B <- rep(1, 5) / sqrt(5)

set.seed(21)
# creat predictor data x ~ N(0, I_p)
x <- matrix(rnorm(500), 100)

# simulate response variable
#    y = f(B'x) + err
# with f(x1) = x1 and err ~ N(0, 0.25^2)
y <- x %*% B + 0.25 * rnorm(100)

x.train <- x[1:80, ]
x.test  <- x[81:100, ]
y.train <- y[1:80, ]
y.test  <- y[81:100, ]

# calculate cve with method 'simple' for k = 1
cve.obj.simple <- cve(y.train ~ x.train, k = 1)

# predict y.test from x.test
yhat <- predict(cve.obj.simple, x.test, 1)

# plot prediction against y.test
plot(yhat, y.test)
```

---

predict_dim *Estimate Dimension of the Sufficient Reduction.*

---

## Description

This function estimates the dimension, i.e. the rank of $B$. The default method 'CV' performs leave-one-out (LOO) cross-validation using mars as follows for k = min.dim, . . . ,max.dim a cross-validation via mars is performed on the dataset $(Y_i, B_k' X_i)_{i=1,\ldots,n}$ where $B_k$ is the $p \times k$ dimensional CVE estimate. The estimated SDR dimension is the $k$ where the cross-validation mean squared error is minimal. The method 'elbow' estimates the dimension via $k = argmin_k L_n(V_{p-k})$ where $V_{p-k}$ is the space that is orthogonal to the column space of the CVE estimate of $B_k$. Method 'wilcoxon' finds the minimum using the Wilcoxon test.

## Usage

```
predict_dim(object, ..., method = "CV")
```

### Arguments

| | |
|---|---|
| object | an object of class "cve", usually, a result of a call to cve or cve.call. |
| ... | ignored. |
| method | This parameter specifies which method is used in dimension estimation. It provides three options: 'CV' (default), 'elbow' and 'wilcoxon'. |

### Value

A list with

criterion for method and k = min.dim, ..., max.dim.

**k** estimated dimension is the minimizer of the criterion.

### Examples

```
# create B for simulation
B <- rep(1, 5) / sqrt(5)

set.seed(21)
# creat predictor data x ~ N(0, I_p)
x <- matrix(rnorm(500), 100)

# simulate response variable
#    y = f(B'x) + err
# with f(x1) = x1 and err ~ N(0, 0.25^2)
y <- x %*% B + 0.25 * rnorm(100)

# Calculate cve for unknown k between min.dim and max.dim.
cve.obj.simple <- cve(y ~ x)

predict_dim(cve.obj.simple)
```

---

## rStiefel *Random sample from Stiefel manifold.*

---

### Description

Draws a random sample from the invariant measure on the Stiefel manifold $S(p, q)$.

### Usage

```
rStiefel(p, q)
```

### Arguments

| | |
|---|---|
| p | row dimension |
| q | col dimension |

## Value

A $p \times q$ semi-orthogonal matrix.

## Examples

```
V <- rStiefel(6, 4)
```

---

summary.cve *Prints summary statistics of the $L$* cve *component.*

---

## Description

Prints a summary statistics of the L component of a cve object #' for k = min.dim,...,max.dim.

## Usage

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

## Arguments

| | |
|---|---|
| object | an object of class "cve", usually, a result of a call to cve or cve.call. |
| ... | ignored. |

## Value

No return value, prints human readable summary.

## Examples

```
# create B for simulation
B <- rep(1, 5) / sqrt(5)

set.seed(21)
# create predictor data x ~ N(0, I_p)
x <- matrix(rnorm(500), 100)

# simulate response variable
#    y = f(B'x) + err
# with f(x1) = x1 and err ~ N(0, 0.25^2)
y <- x %*% B + 0.25 * rnorm(100)

# calculate cve for unknown reduction dimension.
cve.obj.simple <- cve(y ~ x)

summary(cve.obj.simple)
```

# Index