Package 'cpr'

October 12, 2022

Title Control Polygon Reduction

Version 0.2.3

Description Implementation of the Control Polygon Reduction and Control Net Reduction methods for finding parsimonious B-spline regression models.

Depends R (>= 3.0.2)

License GPL (>= 2)

URL https://github.com/dewittpe/cpr/

LazyData true

Imports dplyr, ggplot2 (>= 2.2.0), lazyeval, lme4, magrittr, plot3D, Rcpp (>= 0.11.0), rgl, tibble, tidyr

LinkingTo Rcpp (>= 0.11.0), RcppArmadillo

Suggests covr, geepack, knitr, Matrix, testthat, rmarkdown

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation yes

Author Peter DeWitt [aut, cre],
Samantha MaWhinney [ths],
Nichole Carlson [ths]

Maintainer Peter DeWitt <dewittpe@gmail.com>

Repository CRAN

Date/Publication 2017-03-07 13:41:34

R topics documented:

`cpr-package` *cpr: Control Polygon Reduction*

Description

The cpr package implements the control polygon reduction and control net reduction methods for finding parsimonious B-spline regression models.

This package was part of Peter DeWitt's Ph.D. dissertation which was overseen by his advisors Samantha MaWhinney and Nichole Carlson.

bsplineD *B-spline Derivatives*

Description

Generate the first and second derivatives of a B-spline Basis.

Usage

```
bsplineD(x, iknots = NULL, df = NULL, bknots = range(x), order = 4L,
  derivative = 1L)
```

Arguments

x	a numeric vector
iknots	internal knots
df	degrees of freedom: sum of the order and internal knots. Ignored if iknots is specified.
bknots	boundary knot locations, defaults to range(x).
order	order of the piecewise polynomials, defaults to 4L.
derivative,	(integer) first or second derivative

References

C. de Boor, "A practical guide to splines. Revised Edition," Springer, 2001.

H. Prautzsch, W. Boehm, M. Paluszny, "Bezier and B-spline Techniques," Springer, 2002.

See Also

bsplines

Examples

```
set.seed(42)

xvec <- seq(0.1, 9.9, length = 1000)
iknots <- sort(runif(rpois(1, 3), 1, 9))
bknots <- c(0, 10)

# basis matrix and the first and second derivatives thereof, for cubic (order =
# 4) b-splines
bmat  <- bsplines(xvec, iknots, bknots = bknots)
bmat1 <- bsplineD(xvec, iknots, bknots = bknots, derivative = 1)
bmat2 <- bsplineD(xvec, iknots, bknots = bknots, derivative = 2)

# control polygon ordinates
```

```
theta <- runif(length(iknots) + 4L, -5, 5)

# plot data
plot_data <-
  dplyr::data_frame(x = xvec,
                    Spline = as.numeric(bmat %*% theta),
                    "First Derivative" = as.numeric(bmat1 %*% theta),
                    "Second Derivative" = as.numeric(bmat2 %*% theta))
plot_data <- tidyr::gather(plot_data, key = key, value = value, -x)

ggplot2::ggplot(plot_data) +
ggplot2::aes(x = x, y = value, color = key) +
ggplot2::geom_line() +
ggplot2::geom_hline(yintercept = 0) +
ggplot2::geom_vline(xintercept = iknots, linetype = 3)
```

`bsplines` *B-Splines*

Description

An implementation of Carl de Boor's recursive algorithm for building B-splines.

Usage

```
bsplines(x, iknots = NULL, df = NULL, bknots = range(x), order = 4L)
```

Arguments

`x`	a numeric vector
`iknots`	internal knots
`df`	degrees of freedom: sum of the order and internal knots. Ignored if `iknots` is specified.
`bknots`	boundary knot locations, defaults to `range(x)`.
`order`	order of the piecewise polynomials, defaults to 4L.

Details

The difference between this function and `splines::bs` come in the attributes associated with the output and default options. The `cpr::bsplines` call is intended to simplify the work needed with respect to the control polygon reduction. Further, the implementation of `cpr::bsplines` is in C++ and tends to be faster than `splines::bs`.

See the `vignette("cpr-pkg", package = "cpr")` for a detailed comparison between the `bsplines` and `bs` calls.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

References

C. de Boor, "A practical guide to splines. Revised Edition," Springer, 2001.

H. Prautzsch, W. Boehm, M. Paluszny, "Bezier and B-spline Techniques," Springer, 2002.

See Also

`plot.cpr_bs` for plotting the basis, `bsplineD` for building the basis matrices for the first and second derivative of a B-spline.

See `update_bsplines` for info on a tool for updating a `cpr_bs` object. This is a similar method to the `update` function from the `stats` package.

Examples

```
# build a vector of values to transform
xvec <- seq(-3, 5, length = 100)

# cubic b-spline
bmat <- bsplines(xvec, iknots = c(-2, 0, 0.2))
bmat

# plot the splines
plot(bmat)

# If you want a second x-axis to show the x-values try the following:
second_x_axis <- round(stats::quantile(xvec, probs = seq(0, 1, by = .2)), 2)

plot(bmat) +
ggplot2::annotate(geom = "text", x = second_x_axis, y = -0.02, label = second_x_axis) +
ggplot2::annotate(geom = "linerange", x = second_x_axis, ymin = -0.05, ymax = -0.04) +
ggplot2::coord_cartesian(ylim = c(0, 1))

# quadratic splines
bmat <- bsplines(xvec, iknots = c(-2, 0, 0.2), order = 3L)
bmat
plot(bmat) + ggplot2::ggtitle("Quadratic B-splines")
```

`btensor` *btensor*

Description

Tensor products of B-splines.

Usage

```
btensor(x, df = NULL, iknots = NULL, bknots, order)
```

Arguments

x	a list of variables to build B-spline transforms of. The tensor product of these B-splines will be returned.
df	degrees of freedom. a list of the degrees of freedom for each marginal.
iknots	a list of internal knots for each x. If omitted, the default is to place no internal knots for all x. If specified, the list needs to contain the internal knots for all x. If `df` and `iknots` are both given, the `df` will take precedence.
bknots	a list of boundary knots for each x. As with the iknots, if omitted the default will be to use the range of each x. If specified, the use must specify the bknots for each x.
order	a list of the order for each x; defaults to 4L for all x.

Details

The return form this function is the tensor product of the B-splines transformations for the given variables. Say we have variables X, Y, and Z to build the tensor product of. The columns of the returned matrix correspond to the column products of the three B-splines:

x1y1z1 x2y1z1 x3y1z1 x4y1z1 x1y2z1 x2y2z1 ... x4y4z4

for three fourth order B-splines with no internal knots. The columns of X cycle the quickest, followed by Y, and then Z. This would be the same result as `model.matrix( ~ bsplines(X) : bsplines(Y) : bsplines(Z) + 0)` .

See `vignette("cpr-pkg", package = "cpr")` for more details.

Value

A matrix with a class cpr_bt

Author(s)

Peter DeWitt <dewittpe@gmail.com>

Examples

```
tp <- with(mtcars,
           btensor(x = list(disp, hp, mpg),
                   iknots = list(numeric(0), c(100, 150), numeric(0)))
           )
tp
utils::str(tp)

# The equivalent matrix is could be generated as follows
tp2 <- model.matrix( ~ splines::bs(disp, intercept = TRUE) :
                       splines::bs(hp, knots = c(100, 150), intercept = TRUE) :
                       splines::bs(mpg, intercept = TRUE) + 0,
```

```
                    data = mtcars)

all.equal(tp2, unclass(tp), check.attributes = FALSE)
```

---

build_tensor *Build Tensor*

Description

Tensor products of Matrices.

Usage

```
build_tensor(x, ...)
```

Arguments

x	a matrix, or list of numeric matrices, build the tensor product
...	additional numeric matrices to build the tensor product

Value

A matrix

Author(s)

Peter DeWitt <dewittpe@gmail.com>

Examples

```
# see vignette("cpr-pkg", package = "cpr") for details on tensor products.

A <- matrix(1:4, nrow = 10, ncol = 20)
B <- matrix(1:6, nrow = 10, ncol = 6)

# Two ways of building the same tensor product
tensor1 <- build_tensor(A, B)
tensor2 <- build_tensor(list(A, B))
all.equal(tensor1, tensor2)

# a three matrix tensor product
tensor3 <- build_tensor(A, B, B)
str(tensor3)
```

cn *Control Nets*

Description

Generate the control net for a uni-variable B-spline

Usage

```
cn(x, ...)

## S3 method for class 'cpr_bt'
cn(x, theta, ...)

## S3 method for class 'formula'
cn(formula, data, method = stats::lm, ...,
  keep_fit = FALSE, check_rank = TRUE)

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

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

Arguments

x	a cpr_bs object
...	arguments passed to the regression method
theta	a vector of (regression) coefficients, the ordinates of the control net.
formula	a formula that is appropriate for regression method being used.
data	a required data.frame
method	the regression method such as lm, glm, lmer, geeglm, ...
keep_fit	(logical, defaults to FALSE). If TRUE the regression model fit is retained and returned in the the fit element. If FALSE the regression model is not saved and the fit element will be NA.
check_rank	(logical, defaults to TRUE) if TRUE check that the design matrix is full rank.
object	a cpr_cn object

Details

cn generates the control net for the given B-spline function. There are several methods for building a control net.

Value

a cpr_cn object. This is a list with the following elements. Some of the elements are omitted when the using the cn.cpr_bt method.

cn the control net, data.frame with each row defining a vertex of the control net

bspline_list A list of the marginal B-splines

call the call

keep_fit logical, indicates if the regression models was retained

fit if isTRUE(keep_fit) then the regression model is here, else NA.

coefficients regression coefficients, only the fixed effects if a mixed effects model was used.

vcov The variance-covariance matrix for the coefficients

loglik The log-likelihood for the regression model

rmse The root mean squared error for the regression models

Author(s)

Peter DeWitt <dewittpe@gmail.com>

---

cnr *Control Net Reduction*

Description

Run the Control Net Reduction Algorithm.

Usage

```
cnr(x, keep = -1, p = 2, margin, n_polycoef = 20L,
  progress = interactive(), ...)

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

Arguments

x	a cnr_cp or cnr_tensor object
keep	keep (store) the regression fit for the first keep cpr_cn objects in the list returned by cnr.
p	defaults to 2L, the L^p norm used in determining the influence weight of each internal knot. Passed to influence_weights.
margin	the margins to apply the CNR algorithm to. Passed to influence_weights.
n_polycoef	the number of polynomial coefficients to use when assessing the influence of each internal knot.
progress	show a progress bar.
...	not currently used
object	a cpr_cnr object

Details

cnr runs the control net reduction algorithm.

keep will keep the regression fit as part of the cnr_cp object for models with up to and including keep fits. For example, if keep = 10 then the resulting cnr_cnr object will have the regression fit stored in the first keep + 1 (zero internal knots, one internal knot, ..., keep internal knots) cnr_cp objects in the list. The limit on the number of stored regression fits is to keep memory usage down.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

See Also

influence_weights, cpr for the uni-variable version, Control Polygon Reduction.

---

cp *Control Polygons*

Description

Generate the control polygon for a uni-variable B-spline

Usage

```
cp(x, ...)

## S3 method for class 'cpr_bs'
cp(x, theta, ...)

## S3 method for class 'formula'
cp(formula, data, method = stats::lm, ...,
  keep_fit = FALSE, check_rank = TRUE)

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

## S3 method for class 'cpr_cp'
summary(object, wiggle = FALSE, integrate.args = list(),
  ...)

## S3 method for class 'cpr_cp'
plot(x, ..., show_cp = TRUE, show_spline = FALSE,
  show_xi = TRUE, color = FALSE, n = 100)
```

Arguments

x	a cpr_bs object
...	arguments passed to the regression method
theta	a vector of (regression) coefficients, the ordinates of the control polygon.
formula	a formula that is appropriate for regression method being used.
data	a required data.frame
method	the regression method such as lm, glm, lmer, geeglm, ...
keep_fit	(logical, default value is FALSE). If TRUE the regression model fit is retained and returned in as the fit element. If FALSE the fit element with be NA.
check_rank	(logical, defaults to TRUE) if TRUE check that the design matrix is full rank.
object	a cpr_cp object
wiggle	logical, if TRUE then the integral of the squared second derivative of the spline function will be calculated via stats::integrate.
integrate.args	a list of arguments passed to cpr::wiggle and ultimately stats::integrate.
show_cp	logical (default TRUE), show the control polygon(s)?
show_spline	logical (default FALSE) to plot the spline function?
show_xi	logical (default TRUE) use geom_rug to show the location of the knots in the respective control polygons.
color	Boolean (default FALSE) if more than one cpr_cp object is to be plotted, set this value to TRUE to have the graphic in color (linetypes will be used regardless of the color setting).
n	the number of data points to use for plotting the spline

Details

cp generates the control polygon for the given B-spline function.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

Examples

```
# Support
xvec <- seq(0, 6, length = 500)

# Define the basis matrix
bmat1 <- cpr::bsplines(x = xvec, iknots = c(1, 1.5, 2.3, 4, 4.5))
bmat2 <- cpr::bsplines(x = xvec)

# Define the control vertices ordinates
theta1 <- c(1, 0, 3.5, 4.2, 3.7, -0.5, -0.7, 2, 1.5)
theta2 <- c(1, 3.4, -2, 1.7)
```

```
# build the two control polygons
cp1 <- cp(bmat1, theta1)
cp2 <- cp(bmat2, theta2)

# black and white plot
plot(cp1)
plot(cp1, show_spline = TRUE)

# multiple control polygons
plot(cp1, cp2, show_spline = TRUE)
plot(cp1, cp2, color = TRUE)
plot(cp1, cp2, show_spline = TRUE, color = TRUE)

# via formula
dat <- dplyr::data_frame(x = xvec, y = sin((x - 2)/pi) + 1.4 * cos(x/pi))
cp3 <- cp(y ~ cpr::bsplines(x) + 0, data = dat)

# plot the control polygon, spline and target data.
plot(cp3, show_spline = TRUE) +
  ggplot2::geom_line(mapping = ggplot2::aes_string(x = "x", y = "y"),
                     data = dat, linetype = 2, color = "red")
```

---

cpr *Control Polygon Reduction*

Description

Run the Control Polygon Reduction Algorithm.

Usage

```
cpr(x, keep = -1, p = 2, progress = interactive(), ...)

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

Arguments

x	a cpr_cp object
keep	keep (store) the regression fit for models with keep or fewer internal knots, e.g., keep = 3 will result in the regression fit for models with 0, 1, 2, and 3 internal knots being saved in their respective cpr_cp objects. The default is keep = -1 so that no regression models are retained.
p	defaults to 2L, the L^p norm used in determining the influence weight of each internal knot.
progress	show a progress bar.
...	not currently used
object	a cpr_cpr object

Details

cpr runs the control polygon reduction algorithm.

keep will keep the regression fit as part of the cpr_cp object for models with up to and including keep fits. For example, if keep = 10 then the resulting cpr_cpr object will have the regression fit stored in the first keep + 1 (zero internal knots, one internal knot, ..., keep internal knots) cpr_cp objects in the list. The limit on the number of stored regression fits is to keep memory usage down.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

---

cp_value *Control Polygon Diagnostics*

Description

Collection of functions for inspection and analysis of the control polygons

Usage

```
cp_value(obj, x)

cp_diff(cp1, cp2)
```

Arguments

obj	a cpr_cp object or data.frame where the first column is the abscissa and the second column is the ordinate for the control polygon vertices.
x	absicissa at which to determine the ordinate on control polygon cp
cp1	a cpr_cp object
cp2	a cpr_cp object

Value

cp_value returns the ordinate on the control polygon line segment for the abscissae x given. x could be a control vertex or on a line segment defined by two control vertices of the control polygon provided.

cp_diff returns the absolute vertical distance between the control vertices of cp1 to the control polygon cp2.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

extract_cpr_bsplines *Extract the bspline or btensor call from a formula*

Description

Non-exported function. Might be depericated.

Usage

```
extract_cpr_bsplines(form)
```

Arguments

form	a formula

Author(s)

Peter DeWitt <dewittpe@gmail.com>

generate_cp_formula_data *Generate Control Polygon Formula and Data*

Description

Construct a `data.frame` and `formula` to be passed to the regression modeling tool to generate a control polygon.

Usage

```
generate_cp_formula_data(f, .data)
```

Arguments

f	a formula
.data	the data set containing the variables in the formula

Details

This function is expected to be called from within the cpr::cp function and is not expected to be called by the end user directly.

generate_cp_data exists because of the need to build what could be considered a varying means model. y ~ bsplines(x1) + x2 will generate a rank deficient model matrix—the rows of the bspline basis matrix sum to one with is perfectly collinear with the implicit intercept term. Specifying a formula y ~ bsplines(x1) + x2 - 1 would work if x2 is a continuous variable. If, however, x2 is a factor, or coerced to a factor, then the model matrix will again be rank deficient as a column for all levels of the factor will be generated. We need to replace the intercept column of the model matrix with the bspline. This also needs to be done for a variety of possible model calls, lm, lmer, etc.

By returning an explicit formula and data.frame for use in the fit, we hope to reduce memory use and increase the speed of the cpr method.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

---

get_spline *Get the Control Polygon and the Spline Function*

Description

Generate data.frames for interpolating and plotting a spline function, given a cpr_cp or cpr_cn object.

Usage

```
get_spline(x, margin = 1, at, n = 100)
```

Arguments

x	a cpr_cp or cpr_cn object.
margin	an integer identifying the marginal of the control net to slice along. Only used when working x is a cpr_cn object.
at	point value for marginals not defined in the margin. Only used when x is a cpr_cn object. Expected input is a list of length length(attr(x, "bspline_list")). Entries for elements marginal are ignored. If omitted, the midpoint between the boundary knots for each marginal is used.
n	the length of sequence to use for interpolating the spline function.

Details

A control polygon, cpr_cp object, has a spline function f(x). get_spline returns a list of two data.frame. The cp element is a data.frame with the (x, y) coordinates control points and the spline element is a data.frame with n rows for interpolating f(x).

For a control net, cpr_cn object, the return is the same as for a cpr_cp object, but conceptually different. Where a cpr_cp objects have a uni-variable spline function, cpr_cn have multivariable spline surfaces. get_spline returns a "slice" of the higher dimensional object. For example, consider a three-dimensional control net defined on the unit cube with marginals x1, x2, and x3. The implied spline surface is the function f(x1, x2, x3). get_spline(x, margin = 2, at = list(0.2, NA, 0.5)) would return the control polygon and spline surface for f(0.2, x, 0.5).

See get_surface for taking a two-dimensional slice of a three-plus dimensional control net, or, for generating a useful data set for plotting the surface of a two-dimensional control net.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

See Also

get_surface

Examples

```
data(spdg, package = "cpr")

## Extract the control polygon and spline for plotting.  We'll use base R
## graphics for this example.
a_cp <- cp(pdg ~ bsplines(day, df = 10), data = spdg)

cp_and_spline <- get_spline(a_cp)
plot(cp_and_spline$cp, type = "b")
points(cp_and_spline$spline, type = "l")
grid()

# compare to the cpr:::plot.cpr_cp method
plot(a_cp, show_spline = TRUE)
```

---

get_surface *Get Two-Dimensional Control Net and Surface from n-dimensional Control Nets*

Description

Get Two-Dimensional Control Net and Surface from n-dimensional Control Nets

Usage

```
get_surface(x, margin = 1:2, at, n = 100)
```

Arguments

x	a cpr_cn object
margin	an integer identifying the marginal of the control net to slice along. Only used when working x is a cpr_cn object.
at	point value for marginals not defined in the margin. Only used when x is a cpr_cn object. Expected input is a list of length length(attr(x, "bspline_list")). Entries for elements marginal are ignored. If omitted, the midpoint between the boundary knots for each marginal is used.
n	the length of sequence to use for interpolating the spline function.

See Also

get_spline

Examples

```
## Extract the control net and surface from a cpr_cn object.
a_cn <- cn(pdg ~ btensor(list(day, age), df = list(15, 3), order = list(3, 2)),
           data = spdg)

cn_and_surface <- get_surface(a_cn, n = 50)
str(cn_and_surface, max.level = 2)

par(mfrow = c(1, 2))
with(cn_and_surface$cn,
     plot3D::persp3D(unique(Var1),
                     unique(Var2),
                     matrix(z,
                            nrow = dplyr::n_distinct(Var1),
                            ncol = dplyr::n_distinct(Var2)),
                     main = "Control Net")
     )
with(cn_and_surface$surface,
     plot3D::persp3D(unique(Var1),
                     unique(Var2),
                     matrix(z,
                            nrow = dplyr::n_distinct(Var1),
                            ncol = dplyr::n_distinct(Var2)),
                     main = "Surface")
     )
```

`influence_of` *Influence Of Internal Knots*

Description

Given a control polygon and set of indices, return the influence weight for the specified internal knots and generate graphics showing the original, coarsened, and approximated control polygons.

Usage

```
influence_of(x, indices, ...)
```

Arguments

`x`	a `cpr_cp` object
`indices`	an integer vector specifying the elements of `attr(x,"knots")` to assess. Defaults to all internal knots.
`...`	Additional arguments passed to `influence_weights`.

Value

A `cpr_influence_of` object. This is a list with the following elements:

weight A `tibble` (data.frame) showing the influence weight and relative rank of each internal knot

orig_cp The original control polygon

indices The indices of the internal knots assessed.

coarsened_cp A list of control polygons. These are the control polygons built on the coarsened knot sequence

reinserted_cp A list of control polygons. These are the control polygons resulting from the reinsertion of the assessed internal knot.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

See Also

`plot.cpr_influence_of`, `cp`

influence_weights *Influence Weights*

Description

Determine the influence weight of each internal knot on each marginal of a tensor product.

Usage

```
influence_weights(x, p = 2, margin = seq_along(x$bspline_list),
  n_polycoef = 20L)
```

Arguments

x	a cpr_cp or cpr_cn object
p	the order of the norm, default p = 2.
margin	The margins to apply CNR to. Ignored for cpr_cp objects.
n_polycoef	for influence weights in tensor products, this parameter set the number of polynomial coefficients to use in each of the marginal calculations. Ignored for cpr_cp objects.

Value

A data_frame with two elements, the internal knots (iknots) and the weights.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

is.cpr_bs *Tests for Objects*

Description

Test if an R object is of a class specific to the cpr package

Usage

```
is.cpr_bs(x)

is.cpr_bt(x)

is.cpr_cp(x)

is.cpr_cpr(x)
```

```
is.cpr_cn(x)

is.cpr_cnr(x)
```

Arguments

x	an R object

Author(s)

Peter DeWitt <dewittpe@gmail.com>

`loglikelihood` *Determine the (quasi) Log Likelihood for a regression object.*

Description

Return, via stats::logLik or a custom S3 method, the (quasi) log likelihood of a regression object.

Usage

```
loglikelihood(x, ...)
```

Arguments

x	a regression fit object
...	passed through to `stats::logLik`

Details

This function is used by `cpr::cpr` and `cpr::cnr` to determine the (quasi) log likelihood returned in the `cpr_cpr` and `cpr_cnr` objects.

Generally this function defaults to stats::logLik. Therefore, if an S3 method for determining the (quasi) log likelihood exists in the workspace everything should work. If an S3 method does not exist you should define one.

See `methods(loglikelihood)` for a list of the provided methods. The default method uses `stats::logLik`.

Value

the numeric value of the (quasi) log likelihood.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

See Also

`cpr` `cnr` `logLik`

`matrix_rank` *Rank of a Matrix*

Description

Determine the rank (number of linearly independent columns) of a matrix.

Usage

```
matrix_rank(x)
```

Arguments

`x`	a numeric matrix

Details

Implimentation via the Armadillo C++ linear algrebra library. The function returns the rank of the matix `x`. The computation is based on the singular value decomposition of the matrix; a std::runtime_error excetion will be thrown if the decomposition fails. Any singular values less than the tolerance are treated as zeros. The tolerance is max(m, n) * max_sv * datum::eps, where m is the number of rows of x, n is the number of columns of x, max_sv is the maximal singular value of x, and datum::eps is the difference between 1 and the least value greater than 1 that is representable.

Value

the rank of the matrix as a numeric value.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

References

Conrad Sanderson and Ryan Curtin. Armadillo: a template-based C++ library for linear algebra. Journal of Open Source Software, Vol. 1, pp. 26, 2016.

Examples

```
# Check the rank of a matrix
mat <- matrix(rnorm(25000 * 120), nrow = 25000)
Matrix::rankMatrix(mat)[1]
matrix_rank(mat)

# A full rank B-spline basis
bmat <- bsplines(seq(0, 1, length = 100), df = 15)
matrix_rank(bmat)
```

```
# A rank deficient B-spline basis
bmat <- bsplines(seq(0, 1, length = 100), iknots = c(0.001, 0.002))
ncol(bmat)
matrix_rank(bmat)
```

`plot.cpr_bs` *Plot B-splines*

Description

Plot B-splines

Usage

```
## S3 method for class 'cpr_bs'
plot(x, ..., show_xi = TRUE, show_x = FALSE,
  color = TRUE, digits = 2, n = 100)
```

Arguments

`x`	a `cpr_bs` object
`show_xi`	logical, show the knot locations, using the Greek letter xi, on the x-axis
`show_x`	logical, show the x values of the knots on the x-axis
`color`	logical, if `TRUE` (default) the splines are plotted in color. If `FALSE` all splines are black lines.
`digits`	number of digits to the right of the decimal place to report for the value of each knot.
`n`	number of values to use to plot the splines, defaults to 100
`...`	not currently used

See Also

`bsplines`

Examples

```
bmat <- bsplines(seq(-3, 2, length = 1000), iknots = c(-2, 0, 0.2))
plot(bmat, show_xi = TRUE,  show_x = TRUE)
plot(bmat, show_xi = FALSE, show_x = TRUE)
plot(bmat, show_xi = TRUE,  show_x = FALSE)  ## Default
plot(bmat, show_xi = FALSE, show_x = FALSE)
```

plot.cpr_cn *Plotting Control Nets*

Description

Three-dimensional plots of control nets and/or surfaces

Usage

```
## S3 method for class 'cpr_cn'
plot(x, ..., xlab = "", ylab = "", zlab = "",
  show_net = TRUE, show_surface = FALSE, get_surface_args, net_args,
  surface_args, rgl = TRUE)
```

Arguments

x	a cpr_cn object
...	common arguments which would be used for both the plot of the control net and the surface, e.g., xlim, ylim, zlim.
xlab, ylab, zlab	labels for the axes.
show_net	logical, show the control net
show_surface	logical, show the tensor product surface
get_surface_args	a list of arguments passed to the get_surface call. This call generates the needed data sets used in the plotting.
net_args	arguments to be used explicitly for the control net. Ignored if show_net = FALSE.
surface_args	arguments to be used explicitly for the surface. Ignored if show_surface = FALSE.
rgl	If TRUE, the default, generate use rgl::persp3d to generate the graphics. If FALSE, use plot3D::persp3D to generate the graphics.

Details

This plotting method generates three-dimensional plots of the control net, surface, or both, for a cpr_cn objects. The three-dimensional plots are generated by either persp3D form the plot3D package or persp3d from the rgl package.

Building complex and customized graphics might be easier for you if you use get_surface to generate the needed data for plotting. See vignette("cpr-pkg", package = "cpr") for examples of building different plots.

For rgl graphics, the surface_args and net_args are lists of rgl.material and other arguments passed to persp3d. Defaults are col = "black", front = "lines", back = "lines" for the net_args and col = "grey20", front = "fill", back = "lines" for the surface_args.

For plot3D graphics there are no defaults values for the net_args and surface_args.

Value

The result of the `get_surface` call is returned invisibly.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

See Also

`plot.cpr_cp` for plotting control polygons and splines, `persp3d` and `rgl.material` for generating and controlling rgl graphics. `persp3D` for building plot3D graphics. `get_surface` for generating the data sets needed for the plotting methods.

Examples

```
## see vignette("cpr-pkg", package = "cpr")
```

---

plot.cpr_cnr *Control Net Reduction Plots*

Description

A collection of function for the inspection and evaluation of the control polygon reduction.

Usage

```
## S3 method for class 'cpr_cnr'
plot(x, type = "rmse", from = 1, to, ...)
```

Arguments

`x`	a `cpr_cnr` object
`type`	type of diagnostic plot. `"loglik"` for the loglikihood by degrees of freedom, `"rmse"` for root mean squared residuals by model index
`from`	the first index of `x` to plot
`to`	the last index of `x` to plot
`...`	ignored

Author(s)

Peter DeWitt <dewittpe@gmail.com>

plot.cpr_cpr *Control Polygon Reduction Plots*

Description

A collection of function for the inspection and evaluation of the control polygon reduction.

Usage

```
## S3 method for class 'cpr_cpr'
plot(x, type = "cps", from = 1, to, ...)
```

Arguments

x	a cpr_cpr object
type	type of diagnostic plot. "cps" for control polygons, "loglik" for the loglikihood by degrees of freedom, "rmse" for root mean squared residuals by degrees of freedom
from	the first index of x to plot
to	the last index of x to plot
...	args passed to cpr::plot.cpr_cp

Author(s)

Peter DeWitt <dewittpe@gmail.com>

See Also

plot.cpr_cp

plot.cpr_influence_of *Plotting for cpr_influence_of objects*

Description

Plot method for cpr_influence_of objects

Usage

```
## S3 method for class 'cpr_influence_of'
plot(x, ...)
```

Arguments

x	a cpr_influence_of object
...	Arguments passed to plot.cpr_cp

Value

a ggplot2 graphic

See Also

influence_of

---

predict.cpr_cp *Model Prediction*

Description

Model prediction for cpr_cp and cpr_cn objects.

Usage

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

Arguments

object	a cpr_cp or cpr_cn object
newdata	a data.frame
...	passed to predict from the stats package.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

---

print.cpr_bs *Print bsplines*

Description

Print bsplines

Usage

```
## S3 method for class 'cpr_bs'
print(x, n = 6L, ...)
```

Arguments

x	a cpr_bs object.
n,	number of rows of the B-spline basis matrix to display, defaults to 6L.
...	not currently used.

refine_ordinate *Knot Insertion, Removal, and Reinsertion*

Description

Functions for the insertion, removal, and reinsertion of internal knots for B-splines.

Usage

```
refine_ordinate(x, xi, theta, order = 4L)

coarsen_ordinate(x, xi, theta, order = 4L)

hat_ordinate(x, xi, theta, order = 4L)

insertion_matrix(x, xi, order = 4L)
```

Arguments

x	the value of the knot to be inserted into the knot vector
xi	the (whole) knot vector, including the repeated boundary knots. Regardless of refinement or coarsening, this vector should be the 'reduced' vector such that x will be added to it. See details and examples.
theta	the ordinates of the control polygon vertices
order	the order of the B-spline, defaults to 4 for cubic splines

Value

numeric vectors

Author(s)

Peter DeWitt <dewittpe@gmail.com>

Examples

```
## Not run:
# See the vignette
vignette("cpr-pkg", package = "cpr")

## End(Not run)
```

spdg *Simulated Prognanediol-glucuronid (PDG) Data*

Description

A Simulated data set based on the Study of Women's Health Across the Nation (SWAN) Daily Hormone Study (DHS).

Usage

```
spdg
```

Format

a `data.frame`, carries the `tbl_df` and `tbl` classes from `dplyr`. Variables in the data set:

id Subject ID

age Age, in years of the subject

ttm Time-to-menopause, in years

ethnicity Ethnicity, a factor with five levels: Caucasian, Black, Chinese, Hispanic, and Japanese

bmi Body Mass Index

day_from_dlt A integer value for the number of days from Day of Luteal Transition (DLT). The DLT is `day_from_dlt == 0`. Negative values indicate the follicular phase, positive values for the luteal phase.

day_of_cycle the day of cycle

day A scaled day-of-cycle between [-1, 1] with 0 for the DLT. See Details

pdg A simulated PDG value

Details

Prognanediol-glucuronid (PDG) is the urine metabolite of progesterone. This data set was simulated to have similar characteristics to a subset of the SWAN DHS data. The SWAN DHS data was the motivating data set for the method development that lead to the `cpr` package. The DHS data cannot be made public, so this simulated data set has been provided for use in examples and instructions for use of the `cpr` package.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

Source

This is simulated data. To see the script that generated the data set please visit https://github.com/dewittpe/cpr and look at the scripts in the data-raw directory.

References

Santoro, Nanette, et al. "Body size and ethnicity are associated with menstrual cycle alterations in women in the early menopausal transition: The Study of Women's Health across the Nation (SWAN) Daily Hormone Study." The Journal of Clinical Endocrinology & Metabolism 89.6 (2004): 2622-2631.

`theta` *Extract Regression Coefficients for B-Splines and Tensor Products of B-splines*

Description

An S3 method for extracting the regression coefficients of the `bsplines` and `btensor` terms. By Default this uses `stats::coef` to extract all the regression coefficients. A specific method for `lmerMod` objects has been provided. If you are using a regression method which `stats::coef` will not return the regression coefficients, you'll need to define an S3 method for `stats::coef` to do so.

Usage

```
theta(fit)
```

Arguments

`fit`	a regression model fit

Details

This function is implicitly called in the `cpr::cp` and `cpr::cn` calls.

Value

the regression coefficients associated with terms with names containing either "bsplines" or "btensor".

Author(s)

Peter DeWitt <dewittpe@gmail.com>

See Also

`coef` `cp` `cn`

trimmed_quantile *Trimmed Quantiles*

Description

For data X = x1, x2, ..., xn, with order statistics x(1), x(2), ..., x(r) return the quantiles for a trimmed data set, e.g., X \ x(1), x(r) (trim = 1), or X \ x(1), x(2), x(r-1), x(r) (trim = 2).

Usage

```
trimmed_quantile(x, trim = 1L, use_unique = TRUE, ...)
```

Arguments

x	a numeric vector
trim	defaults to 1, omitting the min and the max
use_unique	logical, if true (defaults), base the quantiles on unique values, if false, base the quantiles on all data, after trimming.
...	other arguments to pass to stats::quantile

Author(s)

Peter DeWitt <dewittpe@gmail.com>

Examples

```
trimmed_quantile(1:100, prob = 1:23 / 24, name = FALSE)

# Warning
# trimmed_quantile(1:100, trim = .3, prob = 1:23 / 24, name = FALSE)

# no warning
trimmed_quantile(1:100, trim = 3, prob = 1:23 / 24, name = FALSE)
```

update_bsplines *Update bspline or btensor calls*

Description

Update cpr_bs and cpr_bt objects alone or within cpr_cp and cpr_cn objects.

Usage

```
update_bsplines(object, ..., evaluate = TRUE)

update_btensor(object, ..., evaluate = TRUE)
```

Arguments

object	an object to update.
...	things to update, expected to be iknots, df, bknots, or order.
evaluate	If true evaluate the new call else return the call.

Author(s)

Peter DeWitt <dewittpe@gmail.com>

See Also

update, bsplines, btensor

Examples

```
############################ Updating a cpr_bs object ############################
# construct a B-spline basis
bmat <- bsplines(seq(1, 10, length = 15), df = 5, order = 3)

# look at the structure of the basis
str(bmat)

# change the order
str(update_bsplines(bmat, order = 4))

# change the order and the degrees of freedom
str(update_bsplines(bmat, df = 12, order = 4))

############################ Updating a cpr_bt object ############################
# construct a tensor product
tpmat <- btensor(list(x1 = seq(0, 1, length = 10), x2 = seq(0, 1, length = 10)),
                 df = list(4, 5))

tpmat

# update the degrees of freedom
update_btensor(tpmat, df = list(6, 7))

####### Updating bsplines or btensor on the right and side of a formula ########

f1 <- y ~ bsplines(x, df = 14) + var1 + var2
f2 <- y ~ btensor(x = list(x1, x2), df = list(50, 31), order = list(3, 5))  + var1 + var2

update_bsplines(f1, df = 13, order = 5)
update_btensor(f2, df = list(13, 24), order = list(3, 8))

############################ Updating a cpr_cp object ############################
data(spdg, package = "cpr")
init_cp <- cp(pdg ~ bsplines(day, df = 30) + age + ttm, data = spdg)
updt_cp <- update_bsplines(init_cp, df = 5)
```

```
########################## Updating a cpr_cn object ##########################
init_cn <- cn(pdg ~ btensor(list(day, age), df = list(30, 4)) + ttm, data = spdg)
updt_cn <- update_btensor(init_cn, df = list(30, 2), order = list(3, 2))
```

`wiggle` *Wiggliness of a Spline function*

Description

Calculate the integral of the squared second derivative of the spline function.

Usage

```
wiggle(object, lower, upper, stop.on.error = FALSE, ...)
```

Arguments

`object`	a `cpr_cp` object
`lower`	the lower limit of the integral
`upper`	the upper limit of the integral
`stop.on.error`	default to `FALSE`, see `integrate`.
`...`	arguments passed to `stats::integrate`

Details

$$\int \left(\frac{\mathrm{d}^2}{\mathrm{d}x^2} f\left(x\right) \right)^2 \mathrm{d}x.$$

Author(s)

Peter DeWitt <dewittpe@gmail.com>

See Also

`cp` `integrate`

Index