Package 'DierckxSpline'

July 2, 2014

Type Package

Title R companion to ``Curve and Surface Fitting with Splines''

Version 1.1-9

Date 2009-01-17

Author Sundar Dorai-Raj and Spencer Graves

Maintainer Spencer Graves <spencer.graves@prodsyse.com>

Description This package provides a wrapper to the FITPACK routines written by
Paul Dierckx. The original Fortran is available from http://www.netlib.org/dierckx
2012.02.15: This package is broken.
Currently, it seems NOT to have any users and is not worth fixing.
These FITPACK routines are over 20 years old and may be obsolete.
If you want to fix it, I will happily make you the maintainer. Spencer Graves

License GPL (>= 2)

Depends R (>= 2.4.0), stats, lattice, PolynomF, fda

Suggests splines

Repository CRAN

Repository/R-Forge/Project dierckxspline

Repository/R-Forge/Revision 52

Repository/R-Forge/DateTimeStamp 2013-07-06 20:49:13

Date/Publication 2013-08-04 07:30:36

NeedsCompilation yes

R topics documented:

`as.dierckx` *Convert a spline object to class 'dierckx'*

Description

Translate a spline object of another class into DierckxSpline (class `dierckx`) format.

Usage

```
as.dierckx(x)
## S3 method for class 'fd'
as.dierckx(x)
```

Arguments

`x`	an object to be converted to class `dierckx`.

Details

The behavior depends on the `class` and nature of `x`. In particular, the `DierckxSpline` package only supports splines of order between 2 and 6, degree between 1 and 5, for linear through quintic splines. Other packages, e.g., `fda` support splines over a wider range and non-spline basis functions such as finite Fourier series. When these are encountered, `as.dierckx` throws an error.

- as.dierckx.fd The following describes how the components of a `dierckx` object are constructed from a spline object of a different class:
  - xRestored from the knots.
  - y Restored from spline predictions at the restored values of knot locations 'x'.
  - wlost. Restored as rep(1, length(x)).
  - from, tofd[["basis"]][["rangeval"]]
  - k coded indirectly as fd[["basis"]][["nbasis"]] - length(fd[["basis"]][["params"]]) - 1.
  - slost, restored as 0.
  - nestlost, restored as length(x) + k + 1
  - n coded indirectly as 2*fd[["basis"]][["nbasis"]] - length(fd[["basis"]][["params"]]).
  - knots The end knots are stored (unreplicated) in fd[["basis"]][["rangeval"]], while the interior knots are stored in fd[["basis"]][["params"]].
  - fplost. Restored as 0.
  - wrk, lwrk, iwrk lost. Restore by refitting to the knots.
  - ierlost. Restored as 0.
  - messagelost. Restored as character(0).
  - g stored indirectly as length(fd[["basis"]][["params"]]).
  - methodlost. Restored as "ss".
  - periodic 'dierckx2fd' only translates 'dierckx' objects with coincident boundary knots. Therefore, 'periodic' is restored as FALSE.
  - routinelost. Restored as 'curfit.default'.
  - xlabfd[["fdnames"]][["args"]]
  - ylabfd[["fdnames"]][["funs"]]

Value

`as.fd.dierckx` converts an object of class 'dierckx' into one of class `fd`.

Author(s)

Spencer Graves

References

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

Ramsay, James O., and Silverman, Bernard W. (2006), *Functional Data Analysis, 2nd ed.*, Springer, New York.

Ramsay, James O., and Silverman, Bernard W. (2002), *Applied Functional Data Analysis*, Springer, New York.

See Also

`as.fd` `fd` `curfit`

Examples

```
x <- 0:24
y <- c(1.0,1.0,1.4,1.1,1.0,1.0,4.0,9.0,13.0,
       13.4,12.8,13.1,13.0,14.0,13.0,13.5,
       10.0,2.0,3.0,2.5,2.5,2.5,3.0,4.0,3.5)
curfit.xy <- curfit(x, y, s=0)

if(require(fda)) {
#  as.fd was not exported;  now it is,
#  but I need fda:::as.fd temporarily
  curfit.fd <- fda:::as.fd(curfit.xy)
  plot(curfit.fd) # as an 'fd' object
  points(x, y) # Curve goes through the points.

  x. <- seq(0, 24, length=241)
  pred.y <- predict(curfit.xy, x.)
  lines(x., pred.y, lty="dashed", lwd=3, col="blue")
# dierckx and fd objects match.


  all.equal(knots(curfit.xy, FALSE), knots(curfit.fd, FALSE))


  all.equal(coef(curfit.xy), as.vector(coef(curfit.fd)))


  curfit2 <- as.dierckx(curfit.fd)
  pred.y2 <- predict(curfit2, x.)
  sum(abs(pred.y-pred.y2))/(sum(abs(pred.y)+abs(pred.y2))/2)
# 1.3e-7


  all.equal(knots(curfit.xy, FALSE), knots(curfit2, FALSE))

# TRUE
  all.equal(coef(curfit.xy), coef(curfit2))
# "Mean relative  difference: 4.5e-7

  preserved <- c("from", "to", "n", "g", "periodic", "xlim","ylim")
  all.equal(curfit.xy[preserved], curfit2[preserved])
# TRUE
# Other components are NOT preserved in translation
# and so can NOT be restored.
}
```

---

cam *cam data*

Description

cam data used in a Dierckx example

Usage

```
data(cam)
```

Format

a data.frame with 61 observations on one variable: 'X3.109'.

Details

Dierckx (1993)

Source

Dierckx, Paul (1993), *Curve and Surface Fitting with Splines*, Springer.

Examples

```
data(cam)
plot(1:61, cam$X3.109)
```

---

concon *Curve fitting with convexity constraints*

Description

General curve fitting with splines with convexity constraints. Wrapper for the Fortran function CONCON.

Usage

```
concon(x, ...)
## Default S3 method:
concon(x, y = NULL, w = NULL, v = 0, s = 0, ...)
```

Arguments

x	A `data.frame`, `matrix`, or `numeric` vector. See details.
y	Optional numeric vector. See details.
w	Optional vector of weights
v	Convexity constraints. See details.
s	Smoothing parameter
...	Additional arguments used only in `update.concon`. Otherwise, ignored.

Details

As with smooth.spline, the x vector should contain at least four distinct values. *Distinct* here means "distinct after rounding to 6 significant digits", i.e., x will be transformed to unique(sort(signif(x, 6))), and y, w, and v are pooled accordingly.

For the default method, arguments x and y are supplied to xy.coords to determine abscissa and ordinate values. The actual function used is DierckxSpline:::xyw.coords which is not exported in the NAMESPACE.

concon determines a smooth cubic spline approximation s(x). See chapters 1 and 3 in the reference for definition of symbols.

The vector v should be the same length as x. v[i] must be set to 1 if s(x) must be locally concave at x[i], to -1 if s(x) must be locally convex at x[i] and to 0 if no convexity constraint is imposed at x[i]. If all v are 0 (no consraints) an error is thrown suggesting the use of curfit instead.

Value

An object of class dierckx with the following components:

iopt	method used
m	length of x
x	abscissa values
y	ordinate values
w	input weights
s	input smoothing parameter
nest	Estimated number of knots
n	Actual number of knots
knots	Knot locations. Do NOT modify before call to update.dierckx
g	Number of interior knots
coef	b-Spline coefficients. Use coef.dierckx to extract.
fp	sum of squares residuals. Use deviance.dierckx to extract.
wrk	Work space. Do NOT modify before call to update.dierckx
lwrk	Length of wrk. Do NOT modify before call to update.dierckx
iwrk	Integer work space. Do NOT modify before call to update.dierckx
kwrk	Length of iwrk. Do NOT modify before call to update.dierckx
bind	Indicates the knots where s''(x)=0. Do NOT modify before call to update.dierckx
sx	Indicates the knots where s''(x)=0. Do NOT modify before call to update.dierckx
ier	Error code. Should always be zero.
method	input method value
k	Always 3 (cubic spline)
periodic	Always FALSE
routine	Always 'concon.default'
xlab	The x-label determined from deparse(substitute(x)).
ylab	The y-label determined from deparse(substitute(y)).

Author(s)

Sundar Dorai-Raj

References

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

curfit, spline, smooth.spline

Examples

```
data(moisture)
## Not run:
f1 <- with(moisture, concon(x, y, w, v, s = 0.2))
f2 <- update(f1, s = 0.04)
f3 <- update(f1, s = 0.0002)
g1 <- with(moisture, curfit(x, y, w, knots = knots(f1, interior=FALSE)))
#must include end knots, via interior = FALSE
g2 <- update(g1, knots = knots(f2, FALSE))
g3 <- update(g1, knots = knots(f3, FALSE))

newx <- with(moisture, seq(min(x), max(x), length = 100))
method <- c("Convexity Constrained", "Unconstrained Least Squares")
group <- c("0 interior knots", "1 interior knot", "3 interior knots")
out <- data.frame(x = rep(newx, times = 6),
         y = c(predict(f1, newx), predict(f2, newx),
               predict(f3, newx), predict(g1, newx),
               predict(g2, newx), predict(g3, newx)),
         group = rep(group, each = 100, times = 2),
         method = rep(method, each = 3 * 100))

library(lattice)
xyplot(y ~ x | method, out, groups = group, panel = function(...) {
         panel.superpose(...)
         panel.xyplot(moisture$x, moisture$y,
                      col = "#800000", pch = 16, cex = 1.2)
       },
       xlim = c(-1, 11), xlab = "", ylab = "", layout = c(1, 2),
       as.table = TRUE, scales = list(cex = 1.2),
       par.strip.text = list(cex = 1.5), type = "l", lwd = 3,
       key = list(space = "top", cex = 1.2, columns = 3,
         text = list(levels(out$group)),
         lines = list(lwd = 3, col = trellis.par.get("superpose.line")$col[1:3])))

## End(Not run)
```

controlPolygon *Control polygon of a spline*

Description

The control polygon matches running means of k knots with the spline coefficients, where k = the degree of the knots. See Dierckx(p. 20).

Usage

```
controlPolygon(object)
```

Arguments

object	an object of class 'dierckx' or 'fd'

Details

Dierckx(p. 20-22, including Figure 1.2) explains how the control polygon can help isolate the zeros of a spline.

Value

An array of dimension nbasis x 2, where the first column contains running means of k knots, and the second is the coefficients of the spline.

Author(s)

Spencer Graves

References

Dierckx, P. (1993) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

curfit, fd

Examples

```
# spline object of Figure 1.2
knots1.2 <- c(0,0,0,0, 2, 4, 7, 8, 10,10,10,10)
wts <- c(1, 2, 5, -5, 5, -1, 3, 3)

library(fda)
fig1.2basis <- create.bspline.basis(c(0, 10),
                                    breaks=c(0, 2, 4, 7, 8, 10))
# fda:::fd is used temporarily;  fd should work later
# but doesn't while debugging DierckxSpline and fda together
```

```
fig1.2a.fda <- fda:::fd(c(1, 2, 5, -5, 5, -1, 3, 3), fig1.2basis)
plot(fig1.2a.fda, ylim=c(-5, 5))

# NOTE:
# An attempt to create this Figure using 'curfit'
# failed, because 'curfit' dropped the knots at 2 and 8.

cP <- controlPolygon(fig1.2a.fda)
lines(cP[, 1], cP[, 2], lty="dotted")

fig1.2a.dierckx <- fd2dierckx(fig1.2a.fda)
cPdierckx <- controlPolygon(fig1.2a.dierckx)
all.equal(cP, cPdierckx)
#  "Mean relative  difference: 5.745888e-07"
```

---

curfit *Curve fitting with splines*

Description

General curve fitting with splines. Wrapper for Fortran function CURFIT.

Usage

```
curfit(x, ...)
percur(x, ...)
## Default S3 method:
curfit(x, y = NULL, w = NULL, s=NULL,
       knots = NULL, n = NULL, from = min(x), to = max(x),
       k = 3, periodic = FALSE, ...)
```

Arguments

x	A data.frame, matrix, or numeric vector. See details.
y	Optional numeric vector. See details.
w	Optional vector of weights
s	a nonnegative number of NULL.
knots	Optional vector of knots including end knots. See details.
n	The number of knots. If both knots and n are provided, n must equal length(knots).
from	Lower bound from which to fit spline.
to	Upper bound to which to fit spline.
k	Degree of the spline. Valid options for k are 1 to 5, inclusively.
periodic	logical; if TRUE then s(a) == s(b)
...	Additional arguments used only in update.curfit. Otherwise, ignored.

Details

curfit determines a spline approximation s(x) of degree k (order = k+1). See Dierckx (1993, ch. 1 and 3) for definition of symbols.

curfit uses two alternative methods: least squares an smoothing spline.

- least squares The least squares method is selected if the following three conditions are all satisfied:

  (1) s is NULL.

  (2) Either knots or n is provided.

  (3) The number of knots minus the order of the spline (k+1) does not exceed the number of distinct values of x (to 6 significant digits).

  The least squares method seems to use all supplied knots (though this must be checked).

- smoothing spline The smoothing spline method is selected in all other cases, namely when s is provided or when the number of knots (inferred or supplied) minus the order of the spline exceeds the number of distinct values of x (to 6 significant digits).

  NOTE: The DierckxSpline smoothing spline method seems to use a subset of the supplied knots. This is different from other smoothing spline software, which typically uses all supplied knots. The details of exactly how these knots are selected can be assertained by studying the Fortran source. This may also be documented in Dierckx' book, but the authors of this R package have not yet processed these details.

As with smooth.spline, the x vector should contain at least four distinct values. *Distinct* here means "distinct after rounding to 6 significant digits", i.e., x will be transformed to unique(signif(x, 6)), and y and w are pooled accordingly.

NOTE: curfit.f calls fpchec.f, which checks to ensure that the number of knots n is at least twice the order of the spline, i.e., 2*(k+1), where k is the degree of the spline. It also checks that the number of distinct data points exceeds the degree k of the spline.

For the default method, arguments x, y and w are supplied to xyw.coords to determine abscissa and ordinate values. If any non-distinct x values are found, they are combined, replacing the corresponding values of y and w by the mean and sum, respectively.

When supplying knots, the end knots must be included. Thus, if periodic = FALSE, the first k+1 values should be min(x) and the last n+k-1 value should be max(x). See below for examples.

If neither knots nor n are provided, the algorithm places one knot at each distinct x value. If n is provided but not knots, n-2*(k+1) interior knots are evenly spaced between from and to. An error message is given if both knots and n are provided and n does not equal length(knots).

percur is equivalent to calling curfit(..., periodic=TRUE).

Value

An object of class dierckx with the following components:

iopt method used coded as follows:

-1 least squares using all knots

0 smoothing spline with a subset of knots

1 smoothing spline update from a previous call

m	length of 'x'
x	abscissa values
y	ordinate values
w	input weights
from	input from value
to	input to value
k	degree of the spline
s	input smoothing parameter
nest	Estimated number of knots
n	Actual number of knots
knots	Knot locations. Use knots.dierckx to extract.
coef	B-spline coefficients. Use coef.dierckx to extract.
fp	sum of squares residuals. Use deviance.dierckx to extract.
wrk	Work space. Do NOT modify before call to update.dierckx.
lwrk	Length of wrk. Do NOT modify before call to update.dierckx.
iwrk	Integer work space. Do NOT modify before call to update.dierckx.
ier	Error code. Should always be zero.
message	brief character string description of the fit
g	Number of interior knots
method	method coded 'ls' for least squares (if 's' is not provided), or 'ss' for smoothing spline (if 's' is provided).
periodic	input 'periodic' parameter
routine	Always 'curfit.default'
xlab	The x-label determined from deparse(substitute(x)).
ylab	The y-label determined from deparse(substitute(y)).

Author(s)

Sundar Dorai-Raj and Spencer Graves

References

Dierckx, P. (1993) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

concon, spline, smooth.spline

Examples

```
##
## titanium example
##
data(titanium)
# The following kills R:
# with(titanium, curfit(x, y))


##
## made up example
##
x <- 0:24
y <- c(1.0,1.0,1.4,1.1,1.0,1.0,4.0,9.0,13.0,
       13.4,12.8,13.1,13.0,14.0,13.0,13.5,
       10.0,2.0,3.0,2.5,2.5,2.5,3.0,4.0,3.5)

#fitLS0 <- curfit(x, y)
#fitSS0 <- curfit(x, y, s=0)

ks <- c(3, 5, 2)
kk <- length(ks)
z <- vector("list", kk)
names(z) <- ks
for(i in 1:kk) {
  k <- ks[i]
  z1 <- curfit(x, y, s = 1000, k = k)
  z2 <- update(z1, s = 60)
  z3 <- update(z2, s = 10)
  z4 <- update(z3, s = 30)
  z5 <- curfit(x, y, s = 30, k = k)
  z6 <- update(z5, s = 0)
  knots <- c(rep(0, k + 1), seq(3, 21, 3), rep(24, k + 1))
  z7 <- curfit(x, y, s = 30, knots = knots, k = k)
  z[[i]] <- list(z1, z2, z3, z4, z5, z6, z7)
}

p <- unlist(z, recursive = FALSE)
n <- sapply(lapply(p, knots), length)
s <- sapply(p, "[[", "s")
i <- sapply(p, "[[", "iopt")
m <- ifelse(i == -1, "ls", ifelse(i == 0, "ss", "ss1"))
k <- sprintf("k = %d", sapply(p, "[[", "k"))
g <- sprintf("%s(s=%d)", m, s, i)
sp <- data.frame(x = rep(x, times = length(p)),
     y = rep(y, times = length(p)), z = unlist(lapply(p, fitted)),
     k = factor(rep(k, each = length(x))), g = rep(g, each = length(x)))

library(lattice)
xyplot(z ~ x | k, data = sp, groups = g,
   panel = function(x, y, subscripts, groups, obs, ...) {
     panel.superpose(x, y, subscripts, groups, lwd = 3, type = "l", ...)
```

```
     x <- unique(x)
     y <- unique(obs)
     panel.xyplot(x, obs, pch = 16, cex = 1.2, col = "darkblue")
   },
   auto.key = list(space = "right", points = FALSE, lines = TRUE),
   obs = sp$y)

## periodic spline
set.seed(42)
n <- 100
r <- 1:n
x <- 0.01 * (r - 1)
e <- rnorm(n, 0, 0.1)
w <- rep(1/sd(e), n + 1)
y <- cos(2 * pi * x) + 0.25 * sin(8 * pi * x) + e
x <- c(x, 1)
y <- c(y, y[1])
kn <- seq(0.01, 0.99, length = 12)
f1 <- percur(x, y, w = w, s = 90, k = 5)

library(lattice)
top <- xyplot(y ~ x,
              panel = function(x, y, ...) {
                panel.abline(v = knots(f1), lty = 2, lwd = 3, col = "gray")
                panel.xyplot(x, y, pch = 16, col = "#800000", cex = 1.2)
                panel.xyplot(x, fitted(f1), type = "l", lwd = 3, col = "#000080")
              },
           par.settings = list(layout.widths = list(left.padding = 0, right.padding = 0)),
              scales = list(cex = 1.2),
              xlab = "", ylab = "")
newx <- seq(-2, 2, 0.01)
newy <- predict(f1, newx)
bot <- xyplot(newy ~ newx, type = "l",
              panel = function(...) {
                panel.abline(v = -2:2, lty = 2, col = "salmon", lwd = 3)
                panel.xyplot(...)
              },
              col = "#000080", lwd = 3,
           par.settings = list(layout.widths = list(left.padding = 0, right.padding = 0)),
              scales = list(cex = 1.2),
              xlab = "", ylab = "")
print(top, c(0, 0.2, 1, 1))
print(bot, c(0.008, 0, 0.992, 0.25), newpage = FALSE)

## example borrowed from ?smooth.spline
plot(cars$speed, cars$dist,
     main = "data(cars)  &  smoothing splines",
     xlab = "SPEED", ylab = "DISTANCE",
     cex.lab = 1.2, cex.axis = 1.2,
     cex.main = 2, cex = 1.5, col = "blue")
## This example has duplicate points, so avoid cv=TRUE
cars.spl.0 <- smooth.spline(cars$speed, cars$dist)
cars.spl.1 <- smooth.spline(cars$speed, cars$dist, df = 10)
```

```
cars.spl.2 <- curfit(cars$speed, cars$dist, s = 5e3)
newx <- seq(min(cars$speed), max(cars$speed), len = 200)
lines(predict(cars.spl.0, newx), col = "blue", lwd = 3, lty = 2)
lines(predict(cars.spl.1, newx), lty="dashed", col = "red", lwd = 3)
lines(newx, predict(cars.spl.2, newx), lty="dotted", lwd = 3)
legend(5, 120, c(paste("smooth.spline( * , df = ", round(cars.spl.0$df, 1), ")", sep = ""),
                 "smooth.spline( * , df = 10)", "curfit( * , s = 5e3)"),
       col = c("blue", "red", "black"),
       lty = c("solid", "dashed", "dotted"), lwd = 3,
       bg = 'bisque', cex = 1.5)
```

---

curfit.free.knot *free-knot splines*

Description

Least squares splines with variable knots.

Usage

```
#curfit.free.knot(x, y, w=NULL, k = 3, g = 10, eps = 0.5e-3,
#                 prior = NULL, fixed = NULL, trace=1, ...)

#NOTE:  The following is required by CRAN rules that
# function names like "as.numeric" must follow the documentation
# standards for S3 generics, even when they are not.
# Please ignore the following line:
## S3 method for class 'free.knot'
curfit(x, y, w=NULL, k = 3, g = 10, eps = 0.5e-3,
                 prior = NULL, fixed = NULL, trace=1, ...)
```

Arguments

x	A data.frame, matrix, or numeric vector.
y	Optional numeric vector.
w	Observation weights.
k	degree of the spline; k=3 for cubic spline.
g	integer vector specifying the number(s) of knots to try. This is ignored if 'prior' is provided. If(length(g) == 1) g = 1:g. If(length(g)>1) g = seq(min(g), max(g)).
eps	weight on the reciprocal differences between successive knots: $penalty = eps{*}(diff(range(x))/(length(knots){+}1)^2){*}sigma0{*}sum(1/diff(unique(knots)))$
prior	initial values for the free knots

fixed	locations of fixed knots
trace	a number indicating the trace level. If 'g' is parsed as a vector, then with trace > 0, after a model for each g[i] is fit, the algorithm reports the number of knots, the variance of the residuals, and the z-score for the lag 1 autocorrelation of residuals. For g[i] > 1, 'optim' is called with control=list(trace=trace-1).
...	Additional arguments used by curfit.

Details

1. If(!is.null(prior)) fit only a model with that number if free knots.

2. Otherwise, decode 'g' and fit one model for each level of 'g'.

3. Return the first model with a negative lag 1 autocorrelation, if any. Otherwise, return the last model fit.

Value

An object of class dierckx with the following components:

iopt	method used
m	length of 'x'
x	abscissa values
y	ordinate values
w	input weights
from	input from value
to	input to value
k	degree of the spline
s	input smoothing parameter
nest	Estimated number of knots
n	Actual number of knots
knots	Knot locations.
coef	b-Spline coefficients. Use coef.dierckx to extract.
fp	sum of squares residuals. Use deviance.dierckx to extract.
wrk	Work space. Do NOT modify before call to update.dierckx
lwrk	Length of wrk. Do NOT modify before call to update.dierckx
iwrk	Integer work space. Do NOT modify before call to update.dierckx
ier	Error code. Should always be zero.
message	convergence message or character(0)
g	Number of interior knots
method	input method value, e.g, 'ls' for least squares
periodic	logical: TRUE for a periodic spline

routine	Always 'curfit.default'
xlab	character x axis label for a plot
ylab	character y axis label for a plot
fits	This component is only present if(is.null(prior)). In that case it is a list with other objects of class 'dierckx' with attributes 'iopt', 'm', 'x', 'y', 'w', 'from', 'to', 'k', 's', 'nest', 'n', 'knots', 'coef', 'fp', 'wrk', 'lwrk', 'iwrk', 'ier', 'message', 'g', 'method', 'periodic', 'routine', 'xlab', 'ylab', as just described, one for each value of 'g' tested.
summary	a data.frame with the following columns: • gvector of the values of 'g' tested • sigma variance of the residuals = (sum of squares of residuals) / (degrees of freedom) • T z-score for the lag 1 autocorrelation of residuals = sqrt(length(x)-1) * (lag 1 autocorrelation of the residuals).

Author(s)

Sundar Dorai-Raj

References

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

curfit, concon, spline, smooth.spline

Examples

```
# Skip this test on CRAN
# because it may exceed their 5 second limit
if(!CRAN()){

data(titanium)
titan10 <- with(titanium, curfit.free.knot(x, y))
#titan10.1 <- with(titanium, curfit.free.knot(x, y, k=1))
#titan10.1 <- with(titanium, curfit.free.knot(x, y, k=1, g=8, trace=1.1))
#titan10.1 <- with(titanium, curfit.free.knot(x, y, k=1,g=7:8,trace=1.1))
}
```

deprecated *depricated*

Description

Use `as.fd.dierckx` (in the `fda` package) instead of `dierckx2fd` and `as.dierckx.fd` in place of `fd2dierckx`.

Usage

```
dierckx2fd(object)
fd2dierckx(object)
```

Arguments

`object`	an object of class dierckx (for 'dierckx2fd') or fd (for 'fd2dierckx') to be translated into the other class.

Details

See the documentation for `as.dierckx`.

Author(s)

Spencer Graves

See Also

`as.dierckx` `as.fd`

deriv.dierckx *Spline Differentiation*

Description

Evaluates in a number of points x(i),i=1,2,...,m the derivative of order nu of a spline s(x) of degree k,given in its b-spline representation.

Usage

```
## S3 method for class 'dierckx'
deriv(expr, at = NULL, order = 1, ...)
```

Arguments

expr	An object of class dierckx.
at	Optional numeric vector where the derivatives should be calculated. If missing, the initial abscissa values are used.
order	Order of the derivative of the derivative to calculate. Default is 1 (first derivative). Valid values are 0<=order<=k.
...	ignored

Value

A numeric vector the same length as at containing the derivatives.

Author(s)

Sundar Dorai-Raj

References

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

curfit, integral.dierckx, spline, smooth.spline

Examples

```
x <- seq(0, 1, 0.1)
y <- (1 - x)^3
z <- curfit(x, y, method = "ls", knots = seq(0, 1, 0.2), k = 3)

plot(x, y, type = "p")
lines(x, fitted(z), col = "blue")

D1 <- deriv(z, order = 1)
D2 <- deriv(~(1 - x)^3, "x", func = TRUE)(z$x)
D3 <- numericDeriv(quote((1 - x)^3), "x")
D4 <- -3 * (1 - z$x)^2
cbind(D1 = D1,
     D2 = attr(D2, "gradient")[, 1],
     D3 = diag(attr(D3, "gradient")),
     D4 = D4)
```

`insert.dierckx` *Spline Knot Insertion*

Description

Inserts a knot into a spline object.

Usage

```
insert(object, ...)
## S3 method for class 'dierckx'
insert(object, at, ...)
```

Arguments

`object`	An object of class `dierckx`.
`at`	A vector of knots to insert. If missing, it uses the midpoints of sort(unique(knots(object, interior=FALSE))).
`...`	ignored

Details

See Dierckx (1993, pp. 16-22, sec. 1.3.4-1.3.6).

Value

An updated spline representation.

Author(s)

Sundar Dorai-Raj

References

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

`curfit`

Examples

```
xi <- 1:10
yi <- sin(xi)
spli <- curfit(xi, yi, s=10)
plot(xi, yi)
lines(spli)
```

```
spli2 <- insert(spli)
lines(spli2, col="red")
kni <- knots(spli, FALSE) # 1 1 1 1 10 10 10 10
all.equal(c(kni[1:4], 5.5, kni[5:8]), knots(spli2, FALSE))
# add 1 knot at 5.5
ci <- coef(spli)
ci2 <- coef(spli2)
all.equal(c(ci[1], 0.5*(ci[-1]+ci[-4]), ci[4]), ci2)
# "Mean relative  difference: 2.700131e-08"
```

`integral.dierckx` *Spline Integration*

Description

Calculates the integral of a spline function s(x) of degree k, which is given in its normalized b-spline representation

Usage

```
## S3 method for class 'dierckx'
integral(expr, from = NULL, to = NULL, ...)
```

Arguments

`expr`	An object of class `dierckx`.
`from`	Lower integration bound. If `NULL`, the minimum knot value is used.
`to`	Upper integration bound. If `NULL`, the maximum knot value is used.
`...`	ignored

Details

`s(x)` is considered to be identically zero outside the interval `(t(k+1),t(n-k))`, where `t` are the knot values. For this reason, `from` and `to` are forced to be in or on the boundaries of the knots.

Value

The value of the integral.

Author(s)

Sundar Dorai-Raj with help from William Venables on how to eliminate a conflict between the generic `integral` functions in the `PolynomF` and `DierckxSpline` packages.

References

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

integral, curfit, deriv.dierckx, spline, smooth.spline

Examples

```
x <- seq(0, 1, 0.1)
y <- (1 - x)^3
z <- curfit(x, y, knots = seq(0, 1, 0.2))

plot(x, y, type = "p")
lines(x, fitted(z), col = "blue")

(answer <- integrate(function(x) (1 - x)^3, 0, 1))
#0.25 with absolute error < 2.8e-15
integral(z)-answer$value
# 0

(ans2 <- integrate(function(x) (1 - x)^3, 0.5, 0.6))
#0.009225 with absolute error < 1.0e-16
integral(z, 0.5, 0.6)-ans2$value
# 6e-9
```

---

knee *knee flexion-extension during walking*

Description

rotation of the knee throughout a walking cycle.

Usage

```
data(knee)
```

Format

a data.frame with the following columns:

- percentpercent of the cycle
- degreeKnee flexion in degrees.

Details

Example used in Dierckx (1993)

Source

Soudan, K., and Dierckx, P. (1979) Calculation of derivatives and Fourier coefficients of human motion data, while using spline functions, *ACM Transactions on Mathematical Software*, 4: 290-294.

Dierckx, Paul (1993), *Curve and Surface Fitting with Splines*, Springer.

knots.dierckx *Extract the knots from a dierckx object*

Description

Extract either all or only the interior knots from an object of class dierckx.

Usage

```
## S3 method for class 'dierckx'
knots(Fn, interior=TRUE, ...)
```

Arguments

Fn	an object of class dierckx
interior	logical: if TRUE, the first Fn[["k"]]+1 of Fn[["knots"]] are dropped, and the next Fn[["g"]] are returned. Otherwise, the first Fn[["n"]] of Fn[["knots"]] are returned.
...	ignored

Details

The knots component of at least some dierckx objects is of length Fn[["nest"]], but only the first Fn[["n"]] of them are actually used. Moreover, the first and last Fn[["k"]]+1 of them are end knots while the remaining Fn[["g"]] are interior knots.

Value

Numeric vector. If 'interior' is TRUE, this vector has length = Fn[["g"]]. Otherwise, it has length Fn[["n"]].

Author(s)

Sundar Dorai-Raj

References

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

curfit, concon, spline, smooth.spline

Examples

```
x <- 0:24
y <- c(1.0,1.0,1.4,1.1,1.0,1.0,4.0,9.0,13.0,
       13.4,12.8,13.1,13.0,14.0,13.0,13.5,
       10.0,2.0,3.0,2.5,2.5,2.5,3.0,4.0,3.5)

z1 <- curfit(x, y, method = "ss", s = 0, k = 3)

knots(z1)
knots(z1, interior=FALSE) # to see all knots
```

`moisture` *Volumetric moisture content*

Description

16 measurements of volumetric moisture content

Usage

```
data(moisture)
```

Format

a data.frame with the following columns:

- xproperty
- ymeasurement
- vconstant 1
- w Weight: constant 1 except for the first 2 and the last observations, being 10, 3, 10, respectively.

Details

Dierckx (1993, Table 7.2, pp. 129-131)

Source

Dierckx, Paul (1993), *Curve and Surface Fitting with Splines*, Springer.

Examples

```
data(moisture)
moisture # Dierckx, Table 7.2, p. 130
## Not run:
f1 <- with(moisture,
  concon(x, y, w, v, s = 0.2))
f2 <- update(f1, s = 0.04)
f3 <- update(f1, s = 0.0002)

g1 <- with(moisture,
  curfit(x, y, w, method = "ls",
         knots = knots(f1)))
g2 <- update(g1, knots = f2)
g3 <- update(g1, knots = f3)

## End(Not run)
```

`panel.dierckx` *Panel function for xyplot.dierckx*

Description

This is the default panel function for 'xyplot.dierckx'.

Usage

```
panel.dierckx(x, y, newx, newy, knots = NULL, knots.y = NULL, lty = 2,
              knot.cex = 1.5, knot.col = "red", knot.fill = "lightgray",
              ...)
```

Arguments

`x, y`	points to be plotted
`newx, newy`	line to be drawn
`knots`	a vector of x-axis position to mark as interior 'knots' of the spline.
`knots.y`	a vector of y-axis positions corresponding to 'knots'.
`lty`	line type for 'panel.xyplot'.
`knot.cex`	character expansion for plotted knots.
`knot.col`	color to use when plotting the knots
`knot.fill`	'fill' argument passed to 'lpoints' when plotting the knots; see `lpoints`.
`...`	other graphics parameters passed to `lpoints` and `panel.xyplot`.

Details

Creates a scatterplot of 'x' and 'y' with a line plot of 'newx' and 'newy', optionally marking the knots at 'knots' and 'knots.y'.

Author(s)

Sundar Dorai-Raj and Spencer Graves

References

Dierckx, P. (1993) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

concon, spline, smooth.spline

---

predict.dierckx *Predict method for object of class 'dierckx'.*

Description

Evaluate the spline for a desired set of x values.

Usage

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

Arguments

object	An object of class 'dierckx'.
newdata	an optional numeric vector at which 'dierckx' spline predictions are desired.
...	Ignored.

Details

if newdata is provided, evaluate the spline at newdata and return the predictions. Else, return fitted(object).

Value

A numeric vector giving spline predictions for 'newdata' if provided or for each distinct level of 'x' in the spline fit.

Author(s)

Sundar Dorai-Raj and Spencer Graves

References

Dierckx, P. (1993) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

curfit

Examples

```
x <- 0:24
y <- c(1.0,1.0,1.4,1.1,1.0,1.0,4.0,9.0,13.0,
       13.4,12.8,13.1,13.0,14.0,13.0,13.5,
       10.0,2.0,3.0,2.5,2.5,2.5,3.0,4.0,3.5)
fitSS0 <- curfit(x, y, s=0)
predict(fitSS0, seq(2, 3, length=5))
```

---

splineZeros *Find the zeros of a spline*

Description

Find the zeros of a spline starting with the control polygon.

Usage

```
splineZeros(object, maxiter)
```

Arguments

object	An object of class dierckx.
maxiter	maximum number of iterations allowed so the signs of the control polygon match those of the spline.

Details

As described in Dierckx(1993, pp. 16-22):

1. cP <- controlPolygon(object)

2. Evaluate the spline at the zeros of the control polygon.

3. While there are sign changes in the control polygon that are not matched by sign changes in the spline, insert knots, recompute the control polygon and the spline at the zeros of the control polygon. Iterate.

4. Use 'uniroot' to obtain the zero in each interval containing a sign change in the spline.

Value

a matrix with one column for each of the outputs of 'uniroot':

root	estimated zero
f.root	value of the spline at 'root'
iter	number of iterations required by 'uniroot'
estim.prec	estimated precision for [root]

Author(s)

Spencer Graves

References

Dierckx, P. (1993) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

curfit,insert.dierckx,predict.dierckx,uniroot

Examples

```
xi <- 1:10
yi <- sin(xi)
spli <- curfit(xi, yi, s=10)

cPi <- controlPolygon(spli)
x. <- seq(1, 10, length=201)
plot(x., predict(spli, x.), type="l")
points(xi, yi)

abline(h=0)
lines(cPi[, 1], cPi[, 2], lty=2)
Zeros <- splineZeros(spli)
abline(v=Zeros[, 1])
```

---

titanium *titanium heat data*

Description

49 values expressing a thermal property of titanium.

Usage

```
data(titanium)
```

Format

a data.frame with the following columns:

- x Temperature (in unknown units. The primary source, de Boor and Rice (1968), does not say. However, standard practice with these kinds of experimens would be to record data like this in degrees Kelvin, and the numbers seem consistent with that practice.)
- yphysical property
- yhatpredictions from some method
- residual100*(y-yhat)

Details

Famous example used by de Boor and Rice (1968), Jupp (1978), Dierckx (1993) and perhaps others. (de Boor and Rice do not mention the source nor provide other information on the units or how the data were collected.)

Source

de Boor, C., and Rice, J. R. (1986), Least-squares cubic spline approximation. II: variable knots. *Report CSD TR 21*, Purdue U., Lafayette, IN. (available from http:// citeseer.ist.psu.edu/cache/papers/cs/24666/ ftp:zSzzSzftp.cs.wisc.eduzSzApproxzSztr21.pdf/ deboor68least.pdf; accessed 2008.07.22)

Dierckx, Paul (1993), *Curve and Surface Fitting with Splines*, Springer.

Jupp, D. L. B. (1975), *Approximation to data by splines with free knots*, SIAM Journal on Numerical Analysis, 15: 328-343.

Examples

```
data(titanium)
# Skip this test on CRAN,
# because it may exeed their 5 second limit
if(!CRAN()){

r <- curfit.free.knot(titanium$x2,
   titanium$y, g = 10, eps = 5e-4)
xyplot(r, show.knots = TRUE)
}
```

`update.dierckx` *Update method for object of class 'dierckx'.*

Description

Update a fit using any of several different functions that produce an object of class 'dierckx'.

Usage

```
  ## S3 method for class 'dierckx'
update(object, knots, k, s, ...)
```

Arguments

`object`	an object of class 'dierckx'
`knots`	either a numeric vector or an object from which knots(knots, interior=FALSE) will produce the desired numeric vector.
`k`	a positive integer giving the degree of the spline = one more than the order of the polynomial segments. Valid options for `k` are 1 to 5, inclusively.
`s`	a nonnegative number or NULL.
`...`	Currently ignored. Additional arguments used only in `update.curfit`. Otherwise, ignored.

Value

An object as produced by the function named in 'object[["routine"]]'.

Author(s)

Sundar Dorai-Raj and Spencer Graves

References

Dierckx, P. (1993) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

concon spline smooth.spline

Examples

```
x <- 0:24
y <- c(1.0,1.0,1.4,1.1,1.0,1.0,4.0,9.0,13.0,
       13.4,12.8,13.1,13.0,14.0,13.0,13.5,
       10.0,2.0,3.0,2.5,2.5,2.5,3.0,4.0,3.5)

#fitLS0 <- curfit(x, y)
#fitSS0 <- curfit(x, y, s=0)

ks <- c(3, 5, 2)
kk <- length(ks)
z <- vector("list", kk)
names(z) <- ks
for(i in 1:kk) {
  k <- ks[i]
  z1 <- curfit(x, y, s = 1000, k = k)
  z2 <- update(z1, s = 60)
  z3 <- update(z2, s = 10)
  z4 <- update(z3, s = 30)
  z5 <- curfit(x, y, s = 30, k = k)
  z6 <- update(z5, s = 0)
  knots <- c(rep(0, k + 1), seq(3, 21, 3), rep(24, k + 1))
  z7 <- curfit(x, y, s = 30, knots = knots, k = k)
  z[[i]] <- list(z1, z2, z3, z4, z5, z6, z7)
}

p <- unlist(z, recursive = FALSE)
n <- sapply(lapply(p, knots), length)
s <- sapply(p, "[[", "s")
i <- sapply(p, "[[", "iopt")
m <- ifelse(i == -1, "ls", ifelse(i == 0, "ss", "ss1"))
k <- sprintf("k = %d", sapply(p, "[[", "k"))
g <- sprintf("%s(s=%d)", m, s, i)
sp <- data.frame(x = rep(x, times = length(p)),
     y = rep(y, times = length(p)), z = unlist(lapply(p, fitted)),
     k = factor(rep(k, each = length(x))), g = rep(g, each = length(x)))
```

```
library(lattice)
xyplot(z ~ x | k, data = sp, groups = g,
   panel = function(x, y, subscripts, groups, obs, ...) {
     panel.superpose(x, y, subscripts, groups, lwd = 3, type = "l", ...)
     x <- unique(x)
     y <- unique(obs)
     panel.xyplot(x, obs, pch = 16, cex = 1.2, col = "darkblue")
   },
   auto.key = list(space = "right", points = FALSE, lines = TRUE),
   obs = sp[["y"]])

## periodic spline
set.seed(42)
n <- 100
r <- 1:n
x <- 0.01 * (r - 1)
e <- rnorm(n, 0, 0.1)
w <- rep(1/sd(e), n + 1)
y <- cos(2 * pi * x) + 0.25 * sin(8 * pi * x) + e
x <- c(x, 1)
y <- c(y, y[1])
kn <- seq(0.01, 0.99, length = 12)
f1 <- percur(x, y, w = w, s = 90, k = 5)

library(lattice)
top <- xyplot(y ~ x,
              panel = function(x, y, ...) {
                panel.abline(v = knots(f1), lty = 2, lwd = 3, col = "gray")
                panel.xyplot(x, y, pch = 16, col = "#800000", cex = 1.2)
                panel.xyplot(x, fitted(f1), type = "l", lwd = 3, col = "#000080")
              },
           par.settings = list(layout.widths = list(left.padding = 0, right.padding = 0)),
              scales = list(cex = 1.2),
              xlab = "", ylab = "")
newx <- seq(-2, 2, 0.01)
newy <- predict(f1, newx)
bot <- xyplot(newy ~ newx, type = "l",
              panel = function(...) {
                panel.abline(v = -2:2, lty = 2, col = "salmon", lwd = 3)
                panel.xyplot(...)
              },
              col = "#000080", lwd = 3,
           par.settings = list(layout.widths = list(left.padding = 0, right.padding = 0)),
              scales = list(cex = 1.2),
              xlab = "", ylab = "")
print(top, c(0, 0.2, 1, 1))
print(bot, c(0.008, 0, 0.992, 0.25), newpage = FALSE)

## example borrowed from ?smooth.spline
plot(cars[["speed"]], cars[["dist"]],
     main = "data(cars)  &  smoothing splines",
     xlab = "SPEED", ylab = "DISTANCE",
```

```
     cex.lab = 1.2, cex.axis = 1.2,
     cex.main = 2, cex = 1.5, col = "blue")
## This example has duplicate points, so avoid cv=TRUE
cars.spl.0 <- smooth.spline(cars[["speed"]], cars[["dist"]])
cars.spl.1 <- smooth.spline(cars[["speed"]], cars[["dist"]], df = 10)
cars.spl.2 <- curfit(cars[["speed"]], cars[["dist"]], s = 5e3)
newx <- seq(min(cars[["speed"]]), max(cars[["speed"]]), len = 200)
lines(predict(cars.spl.0, newx), col = "blue", lwd = 3, lty = 2)
lines(predict(cars.spl.1, newx), lty="dashed", col = "red", lwd = 3)
lines(newx, predict(cars.spl.2, newx), lty="dotted", lwd = 3)
legend(5, 120, c(paste("smooth.spline( * , df = ",
             round(cars.spl.0[["df"]], 1), ")", sep = ""),
                 "smooth.spline( * , df = 10)", "curfit( * , s = 5e3)"),
       col = c("blue", "red", "black"),
       lty = c("solid", "dashed", "dotted"), lwd = 3,
       bg = 'bisque', cex = 1.5)
```

`xyw.coords` *Extract fitting structures with weights*

Description

Extract arguments, eliminate duplicates in x, averaging corresponding y's, summing corresponding weights, and maintaining the dominant sign of convexity constraints.

Usage

```
xyw.coords(x, y = NULL, w = NULL, v = NULL, ...)
```

Arguments

`x`	A `data.frame`, `matrix`, or `numeric` vector. See details.
`y`	Optional numeric vector. See details.
`w`	Optional vector of weights
`v`	Convexity constraints. See details.
`...`	Optional arguments, currently ignored.

Details

'x' and 'y' are first passed to `xy.coords` for initial parsing. Then duplicate 'x' values are identified, and corresponding values of y, w and v are processed to average y's, sum w's, and use sign(sum(v's)). If(missing(w)) w = 1. If(missing(v)) v = 0.

Value

A list with the following components:

x	unique of input x values, sorted
y	input y, arranged to match x, with values corresponding to duplicate x values replaced by their mean.
w	input w, arranged to match x, with values corresponding to duplicate x values replaced by their sum.
v	input v, arranged to match x, with values corresponding to duplicate x values replaced by the sign of their sum.
xin	input x values
yin	input y values

Author(s)

Sundar Dorai-Raj

References

Dierckx, P. (1991) *Curve and Surface Fitting with Splines*, Oxford Science Publications.

See Also

xy.coords curfit, concon

Examples

```
x <- c(3, 1, 3+1e-7, 6, 3+5e-7, 5)
y <- c(3, 1, 2, 6, 4, 5)
tst <- xyw.coords(x, y)
ans <- list(x = c(1, 3+2e-7, 5:6), y = c(1, 3, 5:6), w = c(1, 3, 1, 1),
            v = rep(0, 4), xin=x, yin=y)
all.equal(tst, ans)
# TRUE
#all.equal(tst$x, c(1, 3, 5:6))
#[1] "Mean relative  difference: 6.666666e-08"
```

Index