Package ‘PolynomF’

March 28, 2019

Type    Package
Title   Polynomials in R
Description Implements univariate polynomial operations in R, including polynomial arithmetic, finding zeros, plotting, and some operations on lists of polynomials.
Version 1.1-0
Date    2019-03-26
NeedsCompilation yes
Imports stats, Rcpp
Depends R (>= 3.0.0), graphics
Author Bill Venables, with contribution by Kurt Hornik and Georgi Boshnakov
Maintainer Bill Venables <Bill.Venables@gmail.com>
License GPL-2
Encoding UTF-8
LinkingTo Rcpp
RoxygenNote 6.1.1
Suggests knitr, rmarkdown
VignetteBuilder knitr
Repository CRAN
Date/Publication 2019-03-28 08:30:03 UTC

R topics documented:

as.character.polynom ................................. 2
as.function.polynom ..................................... 3
c.polynom .......................... 3
change.origin ................................. 4
coef.polynom ................................. 5
deriv.polynom ................................. 5
GCD ........................................... 6
as.character.polynom

Description

Produce a text representation of a polynomial object

Usage

## S3 method for class 'polynom'
as.character(x, variable = "x", decreasing = FALSE, ...

Arguments

x The polynomial object in question
variable Character string: what variable name should be used?
decreasing Logical: in decreasing powers or increasing powers?
... Additional arguments (ignored as yet)

Value

A character string representation of the polynomial

Examples

p <- poly.from.zeros(-2:3)
as.character(p, "z", FALSE)
as.character(p, "z", TRUE)
parse(text = as.character(p, "z", TRUE))[[1]]
as.function.polynom  Coercion to function

Description
PolynomF objects ARE functions, but this coercion method creates from a polynomial object a pure function with the coefficients fully exposed in the code and which evaluates the polynomial more efficiently.

Usage

---

## S3 method for class 'polynom'
as.function(x, variable = "x", ...)

## S3 method for class 'polylist'
as.function(x, ...)

Arguments

- `x`: A polynomial object
- `variable`: Character string: what variable name should be used?
- `...`: Additional arguments

Value
An explicit R function evaluating the polynomial

Examples

```r
p <- poly.from.zeros(-2:3)
p
as.function(p)
```

---

c.polynom  Concatenation of polynomial objects into lists

Description
Concatenation of polynomial objects into lists

Usage

---

## S3 method for class 'polynom'
c(..., recursive = FALSE)

## S3 method for class 'polylist'
c(..., recursive = FALSE)
Arguments

... Polynomial or polylist objects
recursive Logical, should the concatenation flatten all component lists?

Value

A polylist object with all arguments included

change.origin

Chenge origin of a polynomial

Description

Given a polynomial P(x) and a new origin o, find the polynomial Q(x) = P(x + o). I.e. Q(0) = P(o)

Usage

change.origin(p, o, ...)

## Default S3 method:
change.origin(p, o, ...)

## S3 method for class 'polynom'
change.origin(p, o, ...)

## S3 method for class 'polylist'
change.origin(p, o, ...)

Arguments

p A polynom or polylist object
o A single numeric quantity specifying the new x-origin
... currently not used

Value

A polynom or polylist object with x measured from the new origin
**coef.polynom**

**Polynomial coefficients**

**Description**

Extract polynomial coefficients

**Usage**

```r
## S3 method for class 'polynom'
coef(object, ...)
```

```r
## S3 method for class 'polylist'
coef(object, ...)
```

**Arguments**

- `object`: A polynomial object or list thereof
- `...`: Ignored

**Value**

A numeric vector of coefficients

**Examples**

```r
p <- polynom(1:3)*polynom(5:1)
coef(p)
```

**deriv.polynom**

**Polynomial Calculus**

**Description**

Find the derivative or indefinite integral of a polynomial object, or list thereof.

**Usage**

```r
## S3 method for class 'polynom'
deriv(expr, ...)
```

```r
integral(expr, ...)
```

```r
## Default S3 method:
integral(expr, ...)
```
# GCD

## Method

```r
## S3 method for class 'polynom'
integral(expr, limits = NULL, ...)
```

```r
## S3 method for class 'polylist'
deriv(expr, ...)
```

```r
## S3 method for class 'polylist'
integral(expr, ...)
```

### Arguments

- `expr`: A polynomial object, or list thereof
- `...`: Unused as yet
- `limits`: Real limits of a definite integral

### Value

A coefficient vector, or list thereof

### Examples

```r
p <- poly.from.roots(-2:3)
p
deriv(p)
integral(p)
```

---

## GCD

### Greatest common divisor

**Description**

Find a monic polynomial of maximal degree that divides each of a set of polynomials exactly

**Usage**

```r
GCD(...) 
```

```r
greatest_common_divisor(...) 
```

```r
## S3 method for class 'polynom'
GCD(...) 
```

```r
## S3 method for class 'polylist'
GCD(...) 
```

### Arguments

- `...`: A list of polynomials or polylist objects
Value

A polynomial giving the greatest common divisor, as defined above

Examples

```r
p <- poly.calc(0:5)
r <- poly.calc(1:6)
greatest_common_divisor(p, r)
solve(greatest_common_divisor(p, r))
lowest_common_multiple(p, r)
solve(lowest_common_multiple(p, r))
```

Description

These provide methods for the generic function `summary` and `math` for polynomial and polylist objects. For `summary` only `sum` and `prod` members are implemented.

Usage

```r
## S3 method for class 'polynom'
Summary(..., na.rm = FALSE)

## S3 method for class 'polylist'
Summary(..., na.rm = FALSE)

## S3 method for class 'polynom'
Math(x, ...)

## S3 method for class 'polylist'
Math(x, ...)
```

Arguments

- `...` Additional arguments
- `na.rm` Logical: should missing values be removed?
- `x` a "polynom" or "polylist" objects.

Value

The result of the group generic operation
Examples

```R
lis <- as.polylist(lapply(-2:3, function(x) polynom() - x))
prod(lis)
sum(lis)
solve(prod(lis))
solve(sum(lis))
```

---

**LCM**

*Lowest Common Multiple*

Description

For a list of polynomials, find the lowest degree monic polynomial into which each divides exactly.

Usage

```R
LCM(...)  
lowest_common_multiple(...)  
```

```R
## S3 method for class 'polynom'
LCM(...)  
```

```R
## S3 method for class 'polylist'
LCM(...)  
```

Arguments

...  

A list of polynomials or polylist objects

Value

A polynomial giving the lowest common multiple

Examples

```R
p <- poly.calc(0:5)
r <- poly.calc(1:6)
greatest_common_divisor(p, r)
solve(greatest_common_divisor(p, r))
lowest_common_multiple(p, r)
solve(lowest_common_multiple(p, r))
```
Ops.polynom

Polynomial arithmetic

Description
Group generic function to implement arithmetic operations on polynomial objects

Usage

## S3 method for class 'polynom'
Ops(e1, e2)

## S3 method for class 'polylist'
Ops(e1, e2)

Arguments

d1, e2  
A numeric vector of a polynomial object. At least one of e1 or e2 must be an object of class "polynom" or "polylist".

Value
A polynomial or polylist object representing the result of the operation.

Examples

x <- polynom()
(p <- (x-1)^5 - 1)
(p1 <- (p + 1)/(x - 1)^2 - 1)
for(i in 0:10) cat(coef((x+1)^i), "\n")

plot.polylist

Plot method for polynomials

Description
Plot methods for polynom or polylist objects

Usage

## S3 method for class 'polylist'
plot(x, xlim = 0:1, ylim = range(Px), type = "l",
    xlab = "x", ylab = "P(x)", ..., len = 1000)

## S3 method for class 'polynom'
plot(x, xlim = 0:1, ylim = range(Px), type = "l",
    xlab = "x", ylab = "P(x)", ...,
xlab = "x", ylab = "p(x)", ..., len = 1000, limits = pu[1:2])

## S3 method for class 'polynom'
lines(x, ..., len = 1000, limits = pu[1:2])

## S3 method for class 'polynom'
points(x, ..., len = 100, limits = pu[1:2])

## S3 method for class 'polylist'
lines(x, ..., len = 1000, limits = pu[1:2],
      col = seq_along(x), lty = seq_along(x))

## S3 method for class 'polylist'
points(x, ..., len = 100)

Arguments

- **x**: A polynom or polylist object to be plotted
- **xlim, ylim**: as for graphics::plot
- **type**: as for graphics::plot
- **xlab, ylab**: as for graphics::plot
- **...**: additional arguments passed on t methods
- **len**: positive integer defining the point or curve resolution
- **limits**: x-limits for the polynomial, default: the entire plot. For polylist objects this may be a two column matrix.
- **col, lty**: Colour(s) and line type(s) as for graphics::plot

Value

Nothing of interest, invisibly

Examples

```r
p <- poly.from.zeros((-3):4)
plot(p)
lines(deriv(p), col = "red")
```

Description

Calculate the Lagrange interpolation polynomial, or list of polynomials, given a set of (x, y) points to fit
Usage

```r
cpyoly.calc(x, y, tol = sqrt(.Machine$double.eps),
      lab = dimnames(y)[[2]])

ppy.from.zeros(...)

ppy.from.roots(...)

ppy.from.values(x, y, tol = sqrt(.Machine$double.eps),
      lab = dimnames(y)[[2]])
```

Arguments

- `x`: A numeric vector of x-points at which the y-values are specified.
- `y`: Either a numeric vector of the same length as `x` or a numeric matrix with rows matching the length of `x`. If `y` is missing (not specified) then a polynomial with zero at `x` is returned.
- `tol`: A numeric tolerance for duplicated `x` values.
- `lab`: A character string vector of names for the list result when `y` is a matrix.
- `...`: A list of specified zeros (for subsidiary functions)

Value

An interpolation polynomial, or list of interpolating polynomials.

Examples

```r
(p <- ploy.calc(0:5)) # same as ploy.from.zeros(0:5)
(p <- ploy.calc(0:5, exp(0:5)))
plot(p)
curve(exp, add = TRUE, col = "red")
```

---

**poly.orth**

Orthogonal polynomials

Description

Generate a list of polynomials up to a specified degree, orthogonal with respect to the natural inner product.

Usage

```r
ppy.orth(x, degree = length(unique(x)) - 1, norm = TRUE)
```
Arguments

- **x**  
  A numeric vector

- **degree**  
  The desired maximum degree

- **norm**  
  Logical: should polynomials be normalised to length one?

Value

A list of orthogonal polynomials

Examples

```r
x <- c(0:3, 5)
P <- poly.orth(x)
plot(P)
Pf <- as.function(P)
zapsmall(crossprod(Pf(x)))
```

---

**polynom**

*Polynomial construction*

Description

Functions to construct polynomial objects and check class membership

Usage

```r
polynom(a = c(0, 1), ..., eps = 0)
apolynom(a)
is.polynom(a)
polylist(...)
is.polylist(x)
as.polylist(x)
```

Arguments

- **a**  
  A `polynom` object, or a numeric vector of coefficients (in "power series" order)  
  or a vector object which can be coerced to one.

- **...**  
  Additional arguments, currently ignored.

- **eps**  
  A small non-negative tolerance to check for zero components.

- **x**  
  An object of class "polylist", at least potentially.
**predict.polynom**

**Value**

A polynomial object.

**Examples**

```r
(x <- polynom())
(p <- polynom(c(1, 5, 2)/10))
plot(p, xlim = 0:1, ylim = 0:1)
P <- p
for(j in 1:7) {
  lines(P, col = j)
  P <- p(P)
}
(r <- solve(p-x))
segments(r, 0, r, p(r), lty = "dashed")
```

---

**predict.polynom**  
*Evaluate a polynomial*

**Description**

Evaluate a polynomial, or polylist object components.

**Usage**

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

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

**Arguments**

- **object**: A polynomial or polylist object
- **newdata**: A target object at which to evaluate.
- **...**: Not used

**Value**

If `newdata` is a numeric vector, a numeric vector of results. If `newdata` is a polynomial, then the composition is returned as a polynomial, or polylist object.
print.polylist  

*Print method for polynomial objects*

**Description**

Print method for polynomial objects

**Usage**

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

**Arguments**

- `x`: A polynomial object or list thereof
- `...`: Additional arguments passed on to methods

**Value**

The original object, invisibly.

---

print.polynom  

*Print method for polynomial objects*

**Description**

Standard method for printing polynomial objects

**Usage**

```r
## S3 method for class 'polynom'
print(x, variable = "x",
      digits = getOption("digits"), decreasing = FALSE, ...)
```

**Arguments**

- `x`: A polynomial object
- `variable`: Character string: what variable name should be given?
- `digits`: Integer: how many decimal digits to use?
- `decreasing`: Logical: in descending powers, or ascending?
- `...`: Additional arguments

**Value**

The original object `x`, invisibly
rep.polylist

Component replication

Description
Repeat components of a polylist object

Usage
```r
## S3 method for class 'polylist'
rep(x, times, ...)
```

```r
## S3 method for class 'polynom'
rep(x, times, ...)
```

Arguments
- `x` A single polynom or polylist object
- `times, ...` As for the base package function `rep`.

Value
The resulting polylist object.

solve.polynom
Find Polynomial Zeros

Description
Solve polynomial equations, \( a(x) = b(x) \), or alternatively find the zeros of the polynomial \( a(x) - b(x) \)

Usage
```r
## S3 method for class 'polynom'
solve(a, b, ...)
```

```r
## S3 method for class 'polylist'
solve(a, b, ...)
```

Arguments
- `a, b` Polynomials for the LHS and RHS respectively
- `...` Currently unused
Value

A vector of roots, usually complex

Examples

```r
p <- poly.calc(0:5)
solve(p)
solve(p, 1)
```

---

**summary.polynom**  
*Polynomial summary*

**Description**

Provide a succinct summary of the critical points of a polynomial, or list thereof

**Usage**

```r
## S3 method for class 'polynom'
support(object, ...)

## S3 method for class 'polylist'
support(object, ...)

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

**Arguments**

- `object, x`  
  A polynomial or polylist object
- `...`  
  Currently unused

**Value**

A list giving the zeros, stationary points and points of inflexion of the polynomial(s)

**Examples**

```r
p <- poly.calc(0:5)
support(p)
```
Tangent lines

Description

Find the tangent line to a polynomial at one or more x-points

Usage

tangent(p, x0)

Arguments

p A polynomial object
x0 A numeric vector of values at which the tangent line(s) are required

Value

A linear polynomial giving the tangent line, or a list of such polynomials

Examples

```r
p <- poly.from.zeros(c(0, 0.5, 4))
plot(p, xlab = expression(italic(x)), ylab = expression(italic(P(x))),
    main = paste("italic(P(x) =", as.character(p, decreasing = TRUE), ")")))
x0 <- solve(deriv(p))  ## stationary points
lines(tangent(p, x0), col = "dark green", lty = "solid",
    limits = cbind(x0-1/4, x0+1/4))
points(x0, p(x0), col = "dark green")

x0 <- solve(deriv(deriv(p)))  ## points of inflexion
lines(tangent(p, x0), col = "red", lty = "solid", lwd = 2,
    limits = cbind(x0-1/4, x0+1/4))
points(x0, p(x0), col = "red")
legend("bottomleft", c("Stationary points", "Points of inflexion"),
    pch = 19, col = c("dark green", "red"), lty = "solid",
    cex = 0.7, bg = "beige", box.lwd = 0.25)
```
### unique.polylist

**Unique components**

**Description**

Remove duplicated polynomials in a polylist object

**Usage**

```r
## S3 method for class 'polylist'
unique(x, incomparables = FALSE, ...)
```

**Arguments**

- `x` A polylist object
- `incomparables` Logical: as for the base function `unique`
- `...` As for the base function `unique`

**Value**

A polylist object with no duplicated components

### [.polylist

**Extract components of a list of polynomials**

**Description**

Extract components of a list of polynomials

**Usage**

```r
## S3 method for class 'polylist'
x[i]
```

**Arguments**

- `x` A polylist object
- `i` An index vector of any congruent form

**Value**

A polylist object of the components
Index

[.polylist, 18
as.character.polylist, 2
as.function.polylist
(as.function.polylist), 3
as.function.polylist (as.function.polylist), 3
as.polylist (polylist), 12
as.polylist (polylist), 12
c.polylist (c.polylist), 3
c.polylist (c.polylist), 3
change.origin, 4
coeff.polylist (coeff.polylist), 5
coeff.polylist, 5
deriv.polylist (deriv.polylist), 5
deriv.polylist (deriv.polylist), 5
GCD, 6
greatest_common_divisor (GCD), 6
GroupGenerics, 7
integral (deriv.polylist), 5
is.polylist (polylist), 12
is.polylist (polylist), 12
LCM, 8
lines.polylist (plot.polylist), 9
lines.polylist (plot.polylist), 9
lowest_common_multiple (LCM), 8
Math.polylist (GroupGenerics), 7
Math.polylist (GroupGenerics), 7
Ops.polylist (Ops.polylist), 9
Ops.polylist, 9
plot.polylist, 9
plot.polylist (plot.polylist), 9
points.polylist (plot.polylist), 9
points.polylist (plot.polylist), 9
poly.calc, 10
poly.from.roots (poly.calc), 10
poly.from.values (poly.calc), 10
poly.from.zeros (poly.calc), 10
poly.orth, 11
polylist (polylist), 12
polylist (polylist), 12
predict.polylist (predict.polylist), 13
predict.polylist (predict.polylist), 13
print.polylist, 14
print.polylist, 14
print.summary.polylist
(print.summary.polylist), 16
rep.polylist, 15
rep.polylist (rep.polylist), 15
solve.polylist (solve.polylist), 15
solve.polylist, 15
summary.polylist (GroupGenerics), 7
summary.polylist (summary.polylist), 16
summary.polylist (summary.polylist), 16
tangent, 17
unique.polylist, 18