The polynom Package

July 13, 2007

Version 1.3-3

Date 2007-07-12

Title A collection of functions to implement a class for univariate polynomial manipulations

Description A collection of functions to implement a class for univariate polynomial manipulations.

Author S original by Bill Venables <Bill.Venables@adelaide.edu.au>. Packaged for R by Kurt Hornik <Kurt.Hornik@R-project.org> and Martin Maechler <maechler@stat.math.ethz.ch>.

Maintainer Kurt Hornik <Kurt.Hornik@R-project.org>

Imports stats, graphics

License GPL-2

R topics documented:

`Math.polynomial` *Math Group Methods for Polynomials*

Description

Group method for functions in the Math group.

Usage

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

Arguments

`x`	an object of class `"polynomial"`.
`...`	further arguments to be passed to or from methods, such as `digits`.

Details

Most math group functions are disallowed with polynomial arguments. The only exceptions are `ceiling`, `floor`, `round`, `trunc`, and `signif` which may be used to transform the coefficients accordingly.

Value

A polynomial with transformed coefficients.

See Also

`Ops.polynomial`, `Summary.polynomial`.

Examples

```
op <- options(digits=18)
p <- poly.from.values(1:4, (2:5)^2)
## 1 + 2.00000000000001*x + x^2
p <- round(p)
## 1 + 2*x + x^2
options(op)
```

Ops.polynomial *Arithmetic Ops Group Methods for Polynomials*

Description

Allows arithmetic operators to be used for polynomial calculations, such as addition, multiplication, division, etc.

Usage

```
## S3 method for class 'polynomial':
Ops(e1, e2)
```

Arguments

e1	an object of class "polynomial".
e2	an object of class "polynomial".

Value

A polynomial got by performing the operation on the two arguments.

See Also

Math.polynomial, Summary.polynomial.

Examples

```
p <- polynomial(c(1, 2, 1))
## 1 + 2*x + x^2
r <- poly.calc(-1 : 1)
## -1*x + x^3
(r - 2 * p)^2
## 4 + 20*x + 33*x^2 + 16*x^3 - 6*x^4 - 4*x^5 + x^6
```

as.function.polynomial *Coerce a Polynomial to a Function*

Description

Takes a polynomial argument and constructs an R function to evaluate it at arbitrary points.

Usage

```
## S3 method for class 'polynomial':
as.function(x, ...)
```

Arguments

x	An object of class "polynomial".
...	further arguments to be passed to or from methods.

Details

This is a method for the generic function as.function.

The polynomial is evaluated within the function using the Horner scheme.

Note that you can use the model-oriented predict method for polynomials for purpose of evaluation (without explicit coercion to a function), see the example below.

Value

A function to evaluate the polynomial p.

See Also

as.function, predict.polynomial

Examples

```
pr <- (poly.calc(-1:1) - 2 * polynomial(c(1, 2, 1)))^2
pr
## 4 + 20*x + 33*x^2 + 16*x^3 - 6*x^4 - 4*x^5 + x^6
prf <- as.function(pr)
prf
## function (x)
## 4 + x * (20 + x * (33 + x * (16 + x * (-6 + x * (-4 + x * (1))))))
## <environment: 0x402440f0>
prf(-3:3)
##  1024 64  0 4 64 144 64
predict(pr, -3:3)
##  1024 64  0 4 64 144 64
```

---

change.origin *Change Origin for a Polynomial*

Description

Calculate the coefficients of a polynomial relative to a new origin on the x axis.

Usage

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

Arguments

p	an object of class "polynomial".
o	a numeric scalar representing the new origin on the original scale.

Details

Let $P(x) = \sum_i p_i x^i$ be a given polynomial and consider writing $P(x) = \sum_j q_j (x - o)^j$. This function calculates the coefficients q_j and returns the result as a polynomial.

Value

A polynomial with coefficients relative to the re-located x axis.

Examples

```
pr <- poly.calc(1:5)
pr
## -120 + 274*x - 225*x^2 + 85*x^3 - 15*x^4 + x^5
change.origin(pr, 3)
## 4*x - 5*x^3 + x^5
```

---

deriv.polynomial *Differentiate a Polynomial*

Description

Calculates the derivative of a univariate polynomial.

Usage

```
## S3 method for class 'polynomial':
deriv(expr, ...)
```

Arguments

expr	an object of class "polynomial".
...	further arguments to be passed to or from methods.

Details

This is a method for the generic function deriv.

Value

Derivative of the polynomial.

See Also

integral.polynomial, deriv.

Examples

```
pr <- poly.calc(1:5)
pr
## -120 + 274*x - 225*x^2 + 85*x^3 - 15*x^4 + x^5
deriv(pr)
##  274 - 450*x + 255*x^2 - 60*x^3 + 5*x^4
```

---

gcd *GCD and LCM for Polynomials*

Description

Compute the greatest common divisor (GCD) and least common multiple (LCM) of a collection of polynomials and polylists.

Usage

```
## S3 method for class 'polylist':
GCD(...)
## S3 method for class 'polynomial':
GCD(...)
## S3 method for class 'polylist':
LCM(...)
## S3 method for class 'polynomial':
LCM(...)
```

Arguments

...	a list of objects of class polynomial or polylist.

Examples

```
pl <- polylist(poly.from.roots(-1),
               poly.from.roots(c(-1, -1)),
               poly.from.roots(1))
GCD(pl)
GCD(pl[-3])
LCM(pl)
LCM(pl, pl, pl[[2]])
```

`integral.polynomial` *Integrate a Polynomial*

Description

Find the integral of a univariate polynomial.

Usage

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

Arguments

`expr`	an object of class `"polynomial"`.
`limits`	numeric vector of length 2 giving the integration limits.
`...`	further arguments to be passed to or from methods.

Value

If `limits` is not given, the integral of `p` from 0 to 'x". Otherwise, the integral with the given integration limits.

See Also

`deriv.polynomial`

Examples

```
p <- poly.calc(1:5)
p
## -120 + 274*x - 225*x^2 + 85*x^3 - 15*x^4 + x^5
deriv(p)
## 274 - 450*x + 255*x^2 - 60*x^3 + 5*x^4
integral(deriv(p)) - 120
## -120 + 274*x - 225*x^2 + 85*x^3 - 15*x^4 + x^5
```

`lines.polynomial` *Lines Method for Polynomials*

Description

Add a polynomial to an existing plot usually as a line plot.

Usage

```
## S3 method for class 'polynomial':
lines(x, len = 100, xlim = NULL, ylim = NULL, ...)
```

Arguments

`x`	an object of class `"polynomial"`.
`len`	size of vector at which evaluations are to be made.
`xlim, ylim`	the range of x and y values with sensible defaults.
`...`	additional arguments as for the `lines` generic.

Details

This is a method for the generic function `lines`.

Lines representing the given polynomial are added to an existing plot. Values outside the current plot region are not shown.

See Also

`lines`, `points`, `points.polynomial`, `plot`, `plot.polynomial`.

Examples

```
plot  (poly.calc(-1:5))
lines (poly.calc( 2:4), lty = 2)
points(poly.calc(-2:6), pch = 4)
```

monic *Monic Polynomials*

Description

Convert a polynomial to monic form by dividing by the leading coefficient.

Usage

```
monic(p)
```

Arguments

p	A polynomial. A warning is issued if the polynomial is identically zero.

Details

Similar in effect to `p/as.numeric(p[length(p)])` but with some safeguards against leading zero coefficients.

Value

A polynomial proportional to `p` with leading coefficient 1.

See Also

`change.origin`

plot.polynomial *Plot Method for Polynomials*

Description

Plots polynomials, optionally allowing the "interesting" region to be automatically determined.

Usage

```
## S3 method for class 'polynomial':
plot(x, xlim = 0:1, ylim = range(Px), type = "l",
                len = 100, ...)
```

Arguments

x	an object of class "polynomial".
xlim	the range to be encompased by the x axis.
ylim	the range to be encompased by the y axis.
type	as for plot().
len	number of x points drawn.
...	additional arguments as for plot.

Details

This is a method for the generic function plot.

A plot of the polynomial is produced on the currently active device. Unless otherwise specified, the domain is chosen to enclose the real parts of all zeros, stationary points and zero itself.

See Also

plot, lines, points, lines.polynomial, points.polynomial.

Examples

```
plot(p <- poly.calc(-1:5))
```

---

points.polynomial *Points Method for Polynomials*

Description

Add a polynomial to an existing plot usually as a point plot.

Usage

```
## S3 method for class 'polynomial':
points(x, length = 100, ...)
```

Arguments

x	an object of class "polynomial".
length	size of x vector at which evaluations are to be made.
...	additional arguments as for the points generic.

Details

This is a method for the generic function points.

Points representing the given polynomial are added to an existing plot. Values outside the current plot region are not shown.

See Also

plot, lines, points, plot.polynomial, lines.polynomial.

Examples

```
plot(poly.calc(-1:5))
lines(poly.calc(2:4), lty=2)
points(poly.calc(-2:6), pch=4)
```

---

poly.calc *Calculate Polynomials from Zeros or Values*

Description

Calculate either the monic polynomial with specified zeros, or the Lagrange interpolation polynomial through the (x,y) points.

Usage

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

Arguments

x	numeric vector specifying either the zeros of the desired polynomial if this is the only non-missing argument, or the x-values for Lagrange interpolation.
y	numeric vector or matrix specifying the y-values for the Lagrange interpolation polynomial. If y is a matrix, nrow(y) must equal length(x), and each column of y is used separately with x.
tol	An absolute value tolerance, below which coefficients are treated as zero.
lab	If y is a matrix, lab is used as the names vector for the list result.

Details

If y is a matrix, the result is a list of polynomials using each column separately.

If x only is given, repeated zeros are allowed. If x and y are given, repeated values in the x vector must have identical y values associated with them (up to tol), otherwise the first y-value only is used and a warning is issued.

Value

Either a polynomial object, or a list of polynomials, as appropriate. In the latter case the object is of class "polylist".

See Also

polynomial

Examples

```
poly.calc(rep(1,3))
## -1 + 3*x - 3*x^2 + x^3
poly.calc(0:4, (0:4)^2 + 1)
## 1 + x^2
poly.calc(0:4, cbind(0:4, (0:4)^2 + 1), lab = letters[1:2])
## List of polynomials:
## $a:
## x
##
## $b:
## 1 + x^2
```

---

poly.orth *Construct Orthogonal Polynomials*

Description

Construct the orthogonal polynomials on a given vector, up to a specified degree.

Usage

```
poly.orth(x, degree = length(unique(x)) - 1, norm = TRUE)
```

Arguments

x	a numeric vector of abscissae. When evaluated at x the polynomials will generate an orthonormal set.
degree	maximum degree requried. The default is one fewer than the number of distinct values in x, which is maximum possible.
norm	a logical indicating whether the polynomials should be normalized.

Value

A list of class "polylist" of objects of class "polynomial" of degree 1, 2, ..., degree.

Examples

```
x <- rep(1:4, 1:4)              # x with repetitions for weighting
x
## [1] 1 2 2 3 3 3 4 4 4 4
polx <- poly.orth(x, 3)         # calculate orthogonal polynomials
polx
## List of polynomials:
## [[1]]
## 0.3162278
##
## [[2]]
```

```
## -0.9486833 + 0.3162278*x
##
## [[3]]
## 2.139203 - 1.863177*x + 0.3450328*x^2
##
## [[4]]
## -5.831564 + 8.80369*x - 3.803194*x^2 + 0.4930066*x^3
v <- sapply(polx, predict, x)   # orthonormal basis
round(crossprod(v), 10)         # check orthonormality
```

`polylist` *Lists of Polynomials*

Description

Create and manipulate lists of polynomials.

Usage

```
polylist(...)
as.polylist(x)
is.polylist(x)
```

Arguments

`...`	a list of R objects.
`x`	an R object.

Details

`polylist` takes a list of arguments, tries to convert each into a polynomial (see `polynomial`), and sets the class of the list to `"polylist"`.

`as.polylist` tries to coerce its arguments to a polylist, and will do so for arguments which are polynomials or lists thereof.

`is.polylist` tests whether its argument is a polylist.

This class has several useful methods, such as taking derivatives (`deriv`) and antiderivatives (`integral`), printing and plotting, subscripting, computing sums and products of the elements, and methods for `c`, `rep`, and `unique`.

Examples

```
## Calculate orthogonal polynomials
pl <- poly.orth(rep(1:4, 1:4), 3)
pl
plot(pl)
deriv(pl)
integral(pl)
sum(pl)
```

```
prod(pl)
unique(rep(pl, 3)[c(8, 12)])
```

`polynomial` *Polynomials*

Description

Construct, coerce to, test for, and print polynomial objects.

Usage

```
polynomial(coef = c(0, 1))
as.polynomial(p)
is.polynomial(p)

## S3 method for class 'polynomial':
as.character(x, decreasing = FALSE, ...)
## S3 method for class 'polynomial':
print(x, digits = getOption("digits"), decreasing = FALSE, ...)
```

Arguments

`coef`	numeric vector, giving the polynomial coefficients in *in*creasing order.
`p`	an arbitrary R object.
`x`	a `polynomial` object.
`decreasing`	a logical specifying the order of the terms; in increasing (default) or decreasing powers.
`digits`	the number of significant digits to use for printing.
`...`	potentially further arguments passed to and from other methods.

Details

`polynomial` constructs a polynomial from its coefficients, i.e., `p[1:k]` specifies the polynomial

$$p_1 + p_2 x + p_3 x^2 + \ldots + p_k x^{k-1}.$$

Internally, polynomials are simply numeric coefficient vectors of class `"polynomial"`. Several useful methods are available for this class, such as coercion to character (`as.character()`) and function (`as.function.polynomial`), extraction of the coefficients (`coef()`), printing (using `as.character`), plotting (`plot.polynomial`), and computing sums and products of arbitrarily many polynomials.

`as.polynomial` tries to coerce its arguments to a polynomial.

`is.polynomial` tests whether its argument is a polynomial (in the sense that it has class `"polynomial"`.

Examples

```
polynomial(1:4)
p <- as.polynomial(c(1,0,3,0))
p
print(p, decreasing = TRUE)
stopifnot(coef(p) == c(1,0,3))

polynomial(c(2,rep(0,10),1))
```

---

predict.polynomial *Evaluate a Polynomial*

Description

Evaluate a polynomial at a given numeric or polynomial argument.

Usage

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

Arguments

object	A polynomial object to be evaluated.
newdata	Argument at which evaluation is requested. May be numeric or itself a polynomial
...	Not used by this method.

Details

This is a method for the generic function predict.

The polynomial is evaluated according to the Horner scheme for speed and numerical accuracy.

Value

Evaluated object of the same class as newdata.

See Also

as.function.polynomial

`solve.polynomial` *Zeros of a Polynomial*

Description

Find the zeros, if any, of a given polynomial.

Usage

```
## S3 method for class 'polynomial':
solve(a, b, ...)
```

Arguments

`a`	A polynomial object for which the zeros are required.
`b`	a numeric value specifying an additional intercept. If given, the zeros of `a - b` are found.
`...`	Not used by this method.

Details

This is a method for the generic function `solve`.

The zeros are found as the eigenvalues of the companion matrix, sorted according to their real parts.

Value

A numeric vector, generally complex, of zeros.

See Also

`polyroot`, `poly.calc`, `summary.polynomial`

Examples

```
p <- polynomial(6:1)
p
## 6 + 5*x + 4*x^2 + 3*x^3 + 2*x^4 + x^5
pz <- solve(p)
pz
## [1] -1.49180+0.0000i -0.80579-1.2229i -0.80579+1.2229i
## [4]  0.55169-1.2533i  0.55169+1.2533i
## To retrieve the original polynomial from the zeros:
poly.calc(pz)
## Warning: imaginary parts discarded in coercion
## 6 + 5*x + 4*x^2 + 3*x^3 + 2*x^4 + x^5
```

`summary.polynomial` *Summary of a Polynomial*

Description

Summarize a polynomial by describing its "key" points.

Usage

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

Arguments

`object`	an object of class `"polynomial"`.
`...`	Not used by this method.

Details

This is a method for the generic function `summary`.

Value

A list of class `"summary.polynomial"` (which has its own `print` method) containing information on zeros, stationary and inflexion points.

Examples

```
p <- polynomial(6:1)
p
## 6 + 5*x + 4*x^2 + 3*x^3 + 2*x^4 + x^5
pz <- summary(p)
pz
## [1] -1.49180+0.0000i -0.80579-1.2229i -0.80579+1.2229i
## [4]  0.55169-1.2533i  0.55169+1.2533i
## To retrieve the original polynomial from the zeros:
poly.calc(pz)
## Warning: imaginary parts discarded in coercion
## 6 + 5*x + 4*x^2 + 3*x^3 + 2*x^4 + x^5
```

`zSummary.polynomial` *Summary Group Methods for Polynomials*

Description

Allows summary group generics to be used on polynomial arguments.

Usage

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

Arguments

`...`	R objects, the first supplied of class `"polynomial"`.
`na.rm`	logical: should missing values be removed?

Details

For the `sum` and `prod` functions, the sum and product of the given polynomials, respectively. For the other members of the Summary group, an error is returned.

Note that one could *order* polynomials by divisibility, and define `min` and `max` as the corresponding lattice meet and join, i.e., the greatest common divisor and the least common multiple, respectively. This is currently not provided: instead, functions `GCD` and `LCM` should be called directly.

See Also

`Math.polynomial`, `Ops.polynomial`

Index