

# Package ‘PolynomF’

January 2, 2012

**Type** Package

**Title** Polynomials in R

**Version** 0.94

**Date** 2010-07-15

**Author** Bill Venables

**Maintainer** Bill Venables <Bill.Venables@gmail.com>

**Description** Implements univariate polynomial operations in R

**License** GPL-2

**Repository** CRAN

**Date/Publication** 2010-07-24 19:41:35

## R topics documented:

PolynomF-package . . . . .	2
as.character.polynom . . . . .	3
as.function.polynom . . . . .	4
c.polylist . . . . .	5
change.origin . . . . .	6
GCD . . . . .	7
integral . . . . .	8
Math.polynom . . . . .	9
plot.polynom . . . . .	11
poly.calc . . . . .	12
polynom . . . . .	14
solve.polynom . . . . .	16
summary.polynom . . . . .	17

<b>Index</b>	<b>19</b>
--------------	-----------

## Description

A package to implement a class of objects that behave like univariate polynomials. Arithmetic operations (addition, subtraction, multiplication, division, remainder, raising to a non-negative integer power) are supported in a natural way. The objects also act as R functions. This package is a successor to the 'polynom' package, but has a simpler and more convenient representation for the objects. Like 'polynom' it uses S3 classes and methods.

## Details

Package:	PolynomF
Type:	Package
Version:	1.0
Date:	2008-05-05
License:	GPL-2
LazyLoad:	yes

The constructor function `polynom` is used to create polynomial objects from their coefficient vector, in power series order. Once polynomials are constructed they may be used as objects in arithmetic operations, integration and differentiation, and as R functions that evaluate the polynomial either at a numeric or complex vector, or at another polynomial, i.e. substituting one polynomial into another. Facilities are also provided for graphical presentation and calculation of complex zeros.

The constructor function `polylist` may be used to create a *list* of polynomial objects. Operations on `polylist` objects include simultaneous graphical display of all components and coercion to function. The function may then be used to evaluate all polynomials on the list simultaneously at the same argument.

## Author(s)

Bill Venables, with some code inherited from the original package by Bill Venables and Kurt Hornik.

Maintainer: <Bill.Venables@gmail.com>

## References

None.

## Examples

```
x <- polynom()
p <- (x-1)^2 + 1
p
```

```

plot(p)

pv <- p(-3:4); pv

p1 <- p(p-1); p1;
plot(polylist(p, p1))

## Hermite polynomials to degree 10
H <- polylist(1, x)
for(n in 2:10)
  H[[n+1]] <- x*H[[n]] - (n-1)*H[[n-1]]
H
### normalisation to unit length
for(n in 1:11)
  H[[n]] <- H[[n]]*exp(-lgamma(n)/2)

plot(H, xlim = c(-3,3))

## orthogonality relationship check:
f <- function(i,j) stats::integrate(function(z)
  dnorm(z)*H[[i+1]](z)*H[[j+1]](z), -Inf, Inf)

f(2,3)
f(4,4)

```

---

as.character.polynom *Coerce polynomial object to character*

---

## Description

Coerces a polynom object to a printable character representation.

## Usage

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

## Arguments

x	A polynom object
variable	Character string with the desired variable name.
decreasing	Should the powers be decreasing, or increasing as in power series form?
...	Not presently used

## Details

The character string may be parsed into an expression for the polynomial itself.

**Value**

A character string.

**Author(s)**

Bill Venables, with contributions from Kurt Hornik

**References**

None

**Examples**

```
p <- poly.from.zeros(-2:4)
as.character(p)
# "48*x - 28*x^2 - 56*x^3 + 35*x^4 + 7*x^5 - 7*x^6 + x^7"
```

---

as.function.polynom    *Coerce polynom or polylist objects to function.*

---

**Description**

Since polynom objects are represented as functions, as.function.polynom simply removes the class attribute. The function as.function.polylist produces allows all polynomials on the list to be evaluated simultaneously at the same argument value.

**Usage**

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

**Arguments**

x	A polynom or polylist object.
variable	A character string giving the name to be used for the formal argument of the resulting function, (for as.polynom only).
...	Not currently used.

**Details**

Since polynom objects are already stored as functions, as.function.polynom is rarely needed and may be replaced by unclass. An explicit call to as.function.polynom, however, will generate a slightly faster version of the function as an unrolled loop, and does allow the user to specify a different name for the formal argument to be used.

Since arithmetic on polynomials is provided, both of these functions may be evaluated at a polynom object as well as a numeric argument.

**Value**

An R function implementing the evaluation.

**Author(s)**

Bill Venables

**References**

None

**Examples**

```
x <- polynom()
H <- polylist(polynom(1), x)
for(j in 2:5)
  H[[j+1]] <- x*H[[j]] - (j-1)*H[[j-1]]
H5 <- as.function(H)
H5(-4:4)
H5(x+1)
```

---

c.polylist

*Utility methods for manipulating polylist objects*


---

**Description**

These three functions implement methods for well-known generic functions.

**Usage**

```
## S3 method for class 'polylist'
c(..., recursive = FALSE)
## S3 method for class 'polylist'
rep(x, times, ...)
## S3 method for class 'polylist'
unique(x, incomparables = FALSE, ...)
```

**Arguments**

...	As for the generic function.
x	As for the generic function.
recursive	As for the generic function.
times	As for the generic function.
incomparables	As for the generic function.

**Details**

Perform familiar operations, retaining `polylist` class for the result.

**Value**

An object of class `polylist`

**Author(s)**

Kurt Hornik, slightly modified by Bill Venables

**References**

None

**Examples**

```
p <- poly.from.zeros(-3:4)
p5 <- rep(polylist(p), 5)
p5
p6 <- c(p, p5)
p6
unique(p6)
```

---

change.origin

*Shift oritin*

---

**Description**

These small convenience functions shift polynomials by relocating the origin to a new position. The primary function is generic and active methods are provided for `polynom` and `polylist` objects.

**Usage**

```
## 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**

<code>p</code>	A polynim or <code>polylist</code> object.
<code>o</code>	The value of the original variable to become the new origin, that is the zero value in the shifted variable.
<code>...</code>	Not used

**Details**

The function `change.origin.default` is a trap for unimplemeted methods.

**Value**

A polynomial or `polylist` object with the origin shifted to 0.

**Author(s)**

Bill Venables

**References**

None

**Examples**

```
x <- polynom()
p <- 1 - 2*x + x^2
change.origin(p, 1)
## x^2
```

---

GCD

*GCD and LCM of two or more polynomials*


---

**Description**

Functions to find the Greatest Common Divisor (GCD) or Least Common Multiple (LCM) of two or more polynomials, specified either as individual arguments or as a `polylist` object.

**Usage**

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

**Arguments**

... Either individual `polynom` arguments or a single `polylist` object with all polynomials.

**Details**

Uses the classical GCD and LCM algorithms with polynomial arithmetic.

**Value**

A single polynomial object giving the GCD or LCM respectively, normalised to have the leading coefficient unity (i.e. a monic polynomial).

**Author(s)**

Kurt Hornik, slightly modified by Bill Venables.

**References**

None.

**Examples**

```
p1 <- poly.from.zeros(-3:2)
p2 <- poly.from.zeros(0:4)

pgcd <- GCD(p1, p2)

p1 <- polylist(p1, p2)
plcm <- LCM(p1)

polylist(pgcd, plcm)
```

---

integral

*Integral and differential calculus on polynomials or lists of polynomials.*

---

**Description**

Performs calculus operations on polynomial objects: differentiation and indefinite or definite integration.

**Usage**

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

**Arguments**

<code>expr</code>	A <code>polynom</code> or <code>polylist</code> object.
<code>limits</code>	Either <code>NULL</code> , implying an indefinite integral, or a numeric (or complex) vector of length two, implying a definite integral between two fixed real or complex limits.
<code>...</code>	Not used, except in the case of <code>integral.polylist</code> where it may pass on common limits of integration to all integrals in the list.

**Details**

`integral` is a local generic function. The generic function `deriv` is already defined in the `stats` package.

**Value**

A `polynom` or numeric object giving the result of the calculus operation, or list of such results. If the result is `polynom` the list is a `polylist`.

**References**

None

**Examples**

```
p <- poly.from.zeros(-2:5)
ip <- integral(p)
ipv <- integral(p, limits = c(-2, 5))

plot(polylist(p, deriv(p)))

x <- polynom()
H <- polylist(1, x)
for(n in 2:10)
  H[[n+1]] <- x * H[[n]] - (n-1)*H[[n-1]]

solve(deriv(H))
```

**Description**

The function `Ops.polynom` allows arithmetic operations on `polynom` objects: addition, subtraction, multiplication, division (with remainder), remainder, raising to a non-negative integer power. It also allows exact equality and exact inequality tests.

Math.polynom allows round, signif, floor, ceiling and trunc operations on a polynom coefficient vector, returning a polynom result.

Summary.polynom and Summary.polylist allow sum and prod operations on polynom arguments, specified either as individual polynom objects or in a single polylist object.

### Usage

```
## S3 method for class 'polynom'
Math(x, ...)
## S3 method for class 'polylist'
Math(x, ...)
## S3 method for class 'polynom'
Ops(e1, e2)
## S3 method for class 'polynom'
Summary(..., na.rm = FALSE)
## S3 method for class 'polylist'
Summary(..., na.rm = FALSE)
```

### Arguments

e1, e2	Objects of class polynom or numeric arguments which may be coerced to class polynom. At least one must be a polynom object.
x	An object of class polymon or polylist.
...	polynom objects or scalar numeric, as appropriate. In the case of Summary.polylist, may be a single polylist object.
na.rm	logical: should missing values be removed first?

### Details

None of these functions is called directly. Ops.polynom is the workhorse of the entire package. Ops.polylist allows vectorised arithmetic computations on polylists.

### Value

A polynom (or polylist) object giving the result of the operation.

### Author(s)

Bill Venables, with some code provided by Kurt Hornik

### References

None

**Examples**

```
x <- polynom(0:1)
p <- (x + 1)^2 - 3

round(p/3)
sum(p, p^2, p^3, p^4)
```

---

plot.polynom

*Graphical display of polynomial objects*


---

**Description**

Standard display methods for polynomial or lists of polynomials.

**Usage**

```
## S3 method for class 'polynom'
plot(x, xlim = 0:1, ylim = range(Px), type = "l",
     xlab = "x", ylab = "p(x)", ..., len = 1000)
## 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 'polylist'
lines(x, ..., len = 1000)
## S3 method for class 'polynom'
lines(x, ..., len = 1000)
## S3 method for class 'polylist'
points(x, ..., len = 100)
## S3 method for class 'polynom'
points(x, ..., at = seq(pu[1], pu[2], len = len), len = 100)
```

**Arguments**

**x** A polynom or polylist object.

**xlim, ylim, xlab, ylab, type** As for plot.

**...** Additional arguments sent to plot, points or lines

**len** The number of linear line segments to use to present the polynomial curve.

**at** the \$x-\$values where the points are to appear.

**Details**

`plot.polynom` will by default choose  $x$  limits to cover the (real parts of) the zeros, stationary points and points of inflexion of the polynomial being plotted. `plot.polylist` chooses by default an  $x$  region to accommodate all polynomials on the list in this way. The current palette of colours is used for different components. `lines.polynom` may be used to add individual polynomials to the plot. The argument `len` may be used to increase or decrease the number of straight line segments used to represent the curves.

**Value**

Nothing of use.

**Author(s)**

Bill Venables, with contributions by Kurt Hornik.

**References**

Nont

**See Also**

`curve`

**Examples**

```
x <- polynom()
L <- polylist(1, 1-x)
for(j in 2:10) L[[j+1]] <- (2*j - 1 - x)*L[[j]] - (j-1)^2*L[[j-1]]
plot(L[1:5], xlim = c(0,5), xaxs = "r", ylab = expression(L[[j]](z)),
      xlab = "z", main = "Laguerre polynomials to degree 4")

lines(L[[6]], col = "grey", lwd = 2)
```

---

poly.calc

*Functions to generate polynomials in several standard ways*

---

**Description**

`poly.calc` (alias `poly.from.values`) computes the Lagrange interpolating polynomial. `poly.from.zeros` (alias `poly.from.roots`) computes the monic polynomial with specified zeros. `poly.orth` calculates polynomials orthogonal over a discrete set of  $x$ -values, as done numerically by the standard R function `poly`.

**Usage**

```
poly.calc(x, y, tol = sqrt(.Machine$double.eps), lab = dimnames(y)[[2]])
poly.from.values(x, y, tol = sqrt(.Machine$double.eps), lab = dimnames(y)[[2]])
poly.from.zeros(...)
poly.from.roots(...)
poly.orth(x, degree = length(unique(x)) - 1, norm = TRUE)
```

**Arguments**

x	A numeric vector of values for the polynomial variable.
y	A numeric vector or matrix specifying values for the polynomial.
tol	A numeric tolerance
lab	A character vector providing names for the <code>polylist</code> of polynomials.
degree	The maximum degree for the orthogonal polynomials required.
norm	Logical value. Should the polynomials be normalised to be of length 1?)
...	Not presently used.

**Details**

Given a vector of distinct values `x` and a vector `y` of the same length, `poly.calc` computes the Lagranging intrepolating polynomial they define. If `y` is a matrix, its row size must match the length of `x` and interpolating polynomials are computed for all columns. In this case the value is a `polylist` object.

`poly.from.values` is a complete alias for `poly.calc`.

The function `poly.from.zeros` computes the monic polynomial with zeros as given by the arguments. The zeros may be specified either as separate artuments or as a single numeric vector.

`poly.from.roots` is a complete alias for `poly.from.zeros`.

`poly.orth` calculates polynomials orthogonal with respect to the uniform measure over a discrete set of `x`-values given by the artument `x`. These are the polynomials for which the standard function `poly` can be used to compute numerical values.

**Value**

A polynom object, or, in the case of `poly.calc` and `poly.orth`, possibly a `polylist` object

**Author(s)**

Bill Venables

**References**

None

**See Also**

`poly`

**Examples**

```

x <- polynom()
H <- polylist(1, x)
for(j in 2:10)
  H[[j+1]] <- x*H[[j]] - (j-1)*H[[j-1]]
Hf <- as.function(H)
x0 <- -5:5
y0 <- Hf(x0)
J <- poly.from.values(x0, y0)
all.equal(H[[11]], J[[11]])

p1 <- poly.from.zeros(-3:2)
p2 <- poly.from.zeros(0:4)
p3 <- GCD(p1, p2)
p4 <- LCM(p1, p2)

solve(polylist(p1, p2, p3, p4))

po <- poly.orth(-4:4, degree = 4)
plot(po)

round(crossprod(as.function(po)(-4:4)), 10)

```

---

polynom

*Constructor, coercion, predicate and print functions for polynom and polylist objects*

---

**Description**

The function `polynom` is the constructor function for objects of the eponymous S3 class. `as.polynom` is the standard coercion function to the same class and `is.polynom` tests for inheritance from the class. `polylist` objects are lists of `polynom` objects, again with the eponymous class.

**Usage**

```

polynom(a = c(0, 1), ..., eps = 0)
as.polynom(a)
is.polynom(a)
polylist(...)
as.polylist(x)
is.polylist(x)
## S3 method for class 'polynom'
print(x, variable = "x",
      digits = getOption("digits"), decreasing = FALSE, ...)
## S3 method for class 'polylist'
print(x, ...)

```

**Arguments**

a	Numeric coefficient vector for the constructor, specifying the coefficients of the powers from 0 to the maximum, without gaps, in that order, that is, in power series order. For the coercion and predicate functions this may be a coefficient vector, or an existing polynom object.
...	As yet, unused for polynom but may be used in future releases. With polylist, objects which are, or may be coerced to, polynom objects. With print methods, additional arguments either for the generic or for the print.polynom method.
x	Object which may be coerced to, or tested for, class polylist.
eps	Absolute tolerance below which components are considered zero.
variable	Character string giving name to be used for the independent variable.
digits	Integer giving the number of significant digits to use in the printed representation of the coefficients.
decreasing	Logical value. Should the powers appear in decreasing order?

**Details**

polynom objects are represented as R functions with the coefficient vector held in an enclosing environment, or closure. They may be used as R functions, or as mathematical objects which behave under arithmetic and calculus operations like polynomials.

Note that polynomials with complex coefficients are not (yet) supported. Nevertheless the root-finding methods described elsewhere will usually give complex zeros.

polylist objects are lists of polynom objects. Methods are available for dealing with all polynomials on the list simultaneously.

**Value**

For polynom and as.polynom, an R function with class polynom. For is.polynom, a logical value

**Author(s)**

Bill Venables

**References**

None

**Examples**

```
x <- polynom() # polynomial 'x'
p <- (x-1)^2 + 10*x^3 + 5*x^4
p
# 1 - 2*x + x^2 + 10*x^3 + 5*x^4

plot(polylist(p, deriv(p), integral(p)),
      xlim = c(-2, 1), ylim = c(-10, 10))
abline(h=0, lty = "dashed", col = "grey")
```

```
print(p, variable = "z", decreasing = TRUE)
```

---

solve.polynom

*Find roots of polynomial equations*

---

### Description

Method functions for the solve generic function to find the roots of a single polynomial equation or the roots of a list of polynomial equations

### Usage

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

### Arguments

a	A polynom or polylist object
b	A polynom or scalar numeric object. The polynomial equation(s) to be solved is $a - b = 0$ .
...	Not used.

### Details

The roots of the equation will in general be complex. In the case of polylist the same equation,  $a - b = 0$ , is solved for all polynomials in the list. In this case the result is a list of root vectors.

### Value

A vector of roots, or a list of such vectors.

### Author(s)

Bill Venables

### References

None

**Examples**

```
x <- polynom()
H <- polylist(1, x)
for(n in 2:10)
  H[[n+1]] <- x * H[[n]] - (n-1)*H[[n-1]]
solve(H)
solve(deriv(H))
```

summary.polynom

*Summary, coefficient and prediction methods for polynomial objects.***Description**

These function implement methods for `summary`, `coef`, and `predict` generic functions for `polynom` and `polylist` objects.

**Usage**

```
## S3 method for class 'polynom'
summary(object, ...)
## S3 method for class 'polynom'
summary(object, ...)
## S3 method for class 'summary.polynom'
print(x, ...)
## S3 method for class 'polynom'
coef(object, ...)
## S3 method for class 'polylist'
coef(object, ...)
## S3 method for class 'polynom'
predict(object, newdata, ...)
## S3 method for class 'polylist'
predict(object, newdata, ...)
```

**Arguments**

<code>object</code>	An object of class <code>polynom</code> or <code>polylist</code> .
<code>x</code>	A <code>summary.polynom</code> object to be printed.
<code>newdata</code>	A numeric or <code>polynom</code> object.
<code>...</code>	Not currently used.

**Details**

The `summary` method for `polynom` objects provides information on the zeros, stationary points and points of inflexion for the object. For `polylist` objects this information is provided for each polynomial in the list. The result is a list.

**Value**

For `summary.polynom` and `summary.polylist` a list of numeric vectors.

For `predict.polynom` and `predict.polylist` an object of the same class as the input argument: either numeric or `polynom`.

For `coef.polynom` and `coef.polylist` a numeric vector or matrix, or a list of numeric vectors.

**Author(s)**

Bill Venables

**References**

None.

**Examples**

```
x <- polynom()
L <- polylist(1, 1-x)
for(j in 2:10)
  L[[j+1]] <- (2*j - 1 - x)*L[[j]] - (j-1)^2*L[[j-1]]

summary(L[[5]])
predict(L[[5]], x-1)
L[[5]](x-1)

coef(L)
```

# Index

## \*Topic **symbolmath**

- as.character.polynom, 3
  - as.function.polynom, 4
  - c.polylist, 5
  - change.origin, 6
  - GCD, 7
  - integral, 8
  - Math.polynom, 9
  - plot.polynom, 11
  - poly.calc, 12
  - polynom, 14
  - PolynomF-package, 2
  - solve.polynom, 16
  - summary.polynom, 17
- as.character.polynom, 3
- as.function.polylist  
(as.function.polynom), 4
- as.function.polynom, 4
- as.polylist (polynom), 14
- as.polynom (polynom), 14
- c.polylist, 5
- change.origin, 6
- coef.polylist (summary.polynom), 17
- coef.polynom (summary.polynom), 17
- deriv.polylist (integral), 8
- deriv.polynom (integral), 8
- GCD, 7
- integral, 8
- is.polylist (polynom), 14
- is.polynom (polynom), 14
- LCM (GCD), 7
- lines.polylist (plot.polynom), 11
- lines.polynom (plot.polynom), 11
- Math.polylist (Math.polynom), 9
- Math.polynom, 9
- Ops.polylist (Math.polynom), 9
- Ops.polynom (Math.polynom), 9
- plot.polylist (plot.polynom), 11
- plot.polynom, 11
- points.polylist (plot.polynom), 11
- points.polynom (plot.polynom), 11
- poly.calc, 12
- poly.from.roots (poly.calc), 12
- poly.from.values (poly.calc), 12
- poly.from.zeros (poly.calc), 12
- poly.orth (poly.calc), 12
- polylist (polynom), 14
- polynom, 14
- PolynomF (PolynomF-package), 2
- PolynomF-package, 2
- predict.polylist (summary.polynom), 17
- predict.polynom (summary.polynom), 17
- print.polylist (polynom), 14
- print.polynom (polynom), 14
- print.summary.polynom  
(summary.polynom), 17
- rep.polylist (c.polylist), 5
- solve.polylist (solve.polynom), 16
- solve.polynom, 16
- Summary.polylist (Math.polynom), 9
- summary.polylist (summary.polynom), 17
- Summary.polynom (Math.polynom), 9
- summary.polynom, 17
- unique.polylist (c.polylist), 5