

Ryacas – an R interface to the `yacas` computer algebra system – Sym version

Rob Goodman, Gabor Grothendieck, Søren Højsgaard,
Ayal Pinkus

October 20, 2006

Contents

1	Introduction	1
2	A sample session	2
3	Simple Yacas calculations	3
3.1	Setting and clearing a variable	3
3.2	Symbolic and numerical evaluations, precision	4
3.3	Rational numbers	5
3.4	Symbolic calculation	5
3.5	Complex numbers and the imaginary unit	5
3.6	Recall the most recent line – the <code>%</code> operator	6
3.7	Printing with <code>PrettyForm</code> , <code>PrettyPrint</code> , <code>TeXForm</code> and <code>TeXForm</code>	6
4	Commands	7
4.1	Factorial	7
4.2	Taylor expansions	7
4.3	Solving equations	8
4.3.1	Solving equations symbolically	8
4.3.2	Solving equations numerically	8
4.4	expanding polynomials	8
4.5	Simplifying an expression	8
4.6	Analytical derivatives	8
4.7	Integration	9
4.8	Limits	9
4.9	Variable substitution	9
4.10	Solving ordinary differential equations	9
5	Matrices	10
5.1	Inverse	10
5.2	Determinant	10

1 Introduction

`Ryacas` makes the `yacas` computer algebra system available from within R. (`yacas` is short for “Yet Another Computer Algebra System”).

This document is based on the `yacas` Sym interface and is similar to another document based on the interface provided by R `yacas` function.

`yacas` is developed by Ayal Pinkhuis (who is also the maintainer) and others, and is available at yacas.sourceforge.org for various platforms. There is a comprehensive documentation (300+ pages) of `yacas` (also available at yacas.sourceforge.org) and the documentation contains many examples. The examples given here are largely taken from the `yacas` documentation (especially from the introductory chapter) but are organised differently.

A `Sym` object is a `yacas` character string that has the "Sym" class. Using `+`, `-` and other similar R operators. One can combine `Sym` objects with other `Sym` objects as well as to other R objects. `Sym(x)` coerces object `x` to a `Sym` object by first coercing it to character and then changing its class to "Sym". One can also apply `sin`, `cos`, `tan`, `deriv`, `Integrate` and other provided functions to `Sym` objects as shown in the examples below.

2 A sample session

Algebraic calculations:

```
> (Sym(10) + Sym(2)) * Sym(5) + Sym(7)^Sym(7)
expression(823603)
> Sym("10 * 2") * 5 + Sym(7)^7
expression(823643)
> (Sym(10) + 2) * 5 + Sym(7)^7
expression(823603)
> Sym("(10+2)*5 + 7^7")
expression(823603)
> Sym("1/14 + 5/21 * (30 - 1+1/2)")
expression(149/21)
```

Numerical evaluations:

```
> Sym("-12/2")
expression(-6)
> Eval(Sym("-12/2"))
[1] -6
```

Working with symbolic expressions:

```
> x <- Sym("x")
> Factor(x^2 - 1)
expression((x + 1) * (x - 1))
> exp1 <- x^2 + 2 * x^2
> exp0 <- Sym("exp0")
> exp2 <- 2 * Sym(exp0)
> exp3 <- 6 * Pi * x
> exp4 <- exp1 * (1 - sin(exp3))/exp2
> exp4
expression(3 * x^2 * (1 - sin(6 * x * pi))/(2 * exp0))
```

Working with numerical expressions:

Combining symbolic and numerical expressions:

```
> N(sin(1)^2 + cos(x)^2)
expression(cos(x)^2 + 0.708073418273571)
```

Differentiation:

```
> deriv(sin(x), x)
expression(cos(x))
```

Integration: [!!! This is odd: I thought yacas was case sensitive...]]

```
> Set(A, "ssss")
expression(ssss)
> a <- Sym("a")
> b <- Sym("b")
> Integrate(sin(x), x, a, b)

In function "Check" : CommandLine(1) : "Found bound variable A which should have been unbound, in MatchLin
> Clear("A")
expression(TRUE)
> Integrate(sin(x), x, a, b)
expression(cos(a) - cos(b))
```

Expanding polynomials:

```
> Expand((1 + x)^3)
expression(x^3 + 3 * x^2 + 3 * x + 1)
```

Taylor expansion:

```
> texp <- Taylor(exp(x), x, 0, 3)
```

Printing the result in nice forms:

```
> PrettyForm(texp)
      2      3
      x      x
x + -- + -- + 1
      2      6
> TeXForm(texp)
expression("$x + \frac{x ^{2}}{2} + \frac{x ^{3}}{6} + 1$")
```

3 Simple Yacas calculations

3.1 Setting and clearing a variable

The function Set() command can both be used to assign values to global variables.

```

> Set(n, "(10 + 2) * 5")
expression(60)
> n <- Sym("n")
> Set(n, n + n)
expression(120)
> a <- Sym("a")
> Set(z, cos(a))
expression(cos(a))
> z <- Sym("z")
> z + z
expression(2 * cos(a))

```

To clear a variable binding execute `Clear()`:

```

> Set(n, 1)
expression(1)
> n <- Sym("n")
> n
expression(1)
> Clear(n)
expression(TRUE)
> n
expression(n)

```

3.2 Symbolic and numerical evaluations, precision

Evaluations are generally exact:

```

> exp(0)
[1] 1
> exp(1)
[1] 2.718282
> sin(Pi/4)
expression(root(1/2, 2))
> Sym("355/113")
expression(355/113)

```

To obtain a numerical evaluation (approximation), the `N()` function can be used:

```

> N(exp(1))
expression(2.71828182845905)
> N(sin(Pi/4))
expression(0.707134)
> N(355/113)
expression(3.14159292035398)

```

The `N()` function has an optional second argument, the required precision:

```
> N("355/133", 20)
expression(2.66917293233083)
```

The command `Precision(n)` can be used to specify that all floating point numbers should have a fixed precision of `n` digits:

```
> Precision(5)
expression(TRUE)
> N("355/113")
expression(3.14159)
```

3.3 Rational numbers

Rational numbers will stay rational as long as the numerator and denominator are integers:

```
> Sym("55 / 10")
expression(11/2)
```

3.4 Symbolic calculation

Some exact manipulations :

```
> Sym("1/14+5/21*(1*30-(1+1/2)*5^2)")
expression(-12/7)
> x <- Sym("x")
> 0 + x
expression(x)
> y <- Sym("y")
> x + y
expression(x + y)
> sin(asin(x)) + tan(atan(y))
expression(x + y)
```

3.5 Complex numbers and the imaginary unit

The imaginary unit i is denoted `I` and complex numbers can be entered as either expressions involving `I` or explicitly `Complex(a,b)` for $a+ib$.

```
> I^2
expression(-1)
> 7 + 3 * I
expression(complex_cartesian(7, 3))
> Conjugate("%")
expression(complex_cartesian(7, -3))
> exp(3 * I)
expression(complex_cartesian(cos(3), sin(3)))
```

3.6 Recall the most recent line – the % operator

The operator % automatically recalls the result from the previous line.

```
> (1 + x)^3
expression((x + 1)^3)
> z <- Sym("%")
> z
expression((x + 1)^3)
```

3.7 Printing with PrettyForm, PrettyPrint, TexForm and TeX-Form

There are different ways of displaying the output. The (standard) yacas form is:

```
> a <- Sym("a")
> b <- Sym("b")
> c <- Sym("c")
> d <- Sym("d")
> A <- List(List(a, b), List(c, d))
> k <- Sym("k")
> B <- (1 + x)^2 + k^3
> A
expression(list(list(a, b), list(c, d)))
> B
expression((x + 1)^2 + k^3)
```

The Pretty form is:

```
> PrettyForm(A)
/
| ( a ) ( b ) |
|             |
| ( c ) ( d ) |
\             /

> PrettyForm(B)
      2    3
( x + 1 ) + k
```

An alternative is the PrettyPrinter [!!! Why does this give the same result as before??? Earlier I got XML output as well... Is something not reset???

```

> PrettyPrinter("PrettyForm")
True
> A
/          \
| ( a ) ( b ) |
|          |
| ( c ) ( d ) |
\          /
> PrettyPrinter()
True;
> A
{{a,b},{c,d}};
> PrettyPrinter("OMForm")
expression(TRUE)
> A
expression(list(list(a, b), list(c, d)))

```

The output can be displayed in TeX form as well:

```

> TeXForm(B)
expression("$\left( x + 1\right) ^{2} + k ^{3}$")

```

4 Commands

4.1 Factorial

```

> Factorial(40)
expression(Factorial(40))

```

4.2 Taylor expansions

Expand $\exp(x)$ in three terms around 0 and a:

```

> Taylor(exp(x), x, 0, 3)
expression(x + x^2/2 + x^3/6 + 1)
> a <- Sym("a")
> Taylor(exp(x), x, a, 3)
expression(exp(a) + exp(a) * (x - a) + (x - a)^2 * exp(a)/2 +
(x - a)^3 * exp(a)/6)

```

The `InverseTaylor()` function builds the Taylor series expansion of the inverse of an expression. For example, the Taylor expansion in two terms of the inverse of $\exp(x)$ around $x=0$ (which is the Taylor expansion of $\log(y)$ around $y=1$):

```

> InverseTaylor(exp(x), x, 0, 2)
expression(x + x^2/2 + 1)
> Taylor(log(y), y, 1, 2)
expression(y - 1 - (y - 1)^2/2)

```

4.3 Solving equations

4.3.1 Solving equations symbolically

Solve equations symbolically with:

```
> Solve(x/(1 + x) == a, x)
expression(list(x == a/(1 - a)))
> Solve(x^2 + x == 0, x)
expression(list(x == 0, x == -1))
```

(Note the use of the == operator, which does not evaluate to anything, to denote an "equation" object.) Solve() is rather limited.

4.3.2 Solving equations numerically

To solve an equation (in one variable) like $\sin(x) - \exp(x) = 0$ numerically taking 0.5 as initial guess and an accuracy of 0.0001 do:

```
> Newton(sin(x) - exp(x), x, 0.5, 1e-04)
expression(-3.18306)
```

4.4 expanding polynomials

```
> Expand((x + 1)^3)
expression(x^3 + 3 * x^2 + 3 * x + 1)
```

4.5 Simplifying an expression

The function Simplify() attempts to reduce an expression to a simpler form.

```
> (x + y)^3 - (x - y)^3
expression((x + y)^3 - (x - y)^3)
> Simplify("%")
expression(6 * (x^2 * y) + 2 * y^3)
```

4.6 Analytical derivatives

Analytical derivatives of functions can be evaluated:

```
> deriv(sin(x), x)
expression(cos(x))
> deriv(deriv(sin(x)))
expression(-sin(x))
```

The deriv function also accepts an argument specifying how often the derivative has to be taken, e.g:

```
> deriv(sin(x), x, 2)
expression(-sin(x))
```


4.7 Integration

!!! Problem arises because A was defined above (a is not defined, though) (Seems ok in Sym version.)

```
> Integrate(sin(x), x, a, b)
expression(cos(a) - cos(b))
> Integrate(log(x), x, a, b)
expression(b * log(b) - b - (a * log(a) - a))
> Integrate(1/(x^2 - 1), x)
expression(log(2 * (x - 1))/2 - log(2 * (x + 1))/2)
> Integrate(sin(a * x)^2 * cos(b * x), x)
expression((2 * sin(b * x)/b - (sin(-2 * x * a - b * x)/(-2 *
a - b) + sin(-2 * x * a + b * x)/(-2 * a + b)))/4)
```

4.8 Limits

```
> Limit(sin(x)/x, x, 0)
expression(1)
> n <- Sym("n")
> Limit((1 + (1/n))^n, n, Infinity)
expression(exp(1))
> h <- Sym("h")
> Limit((sin(x + h) - sin(x))/h, h, 0)
expression(cos(x))
```

4.9 Variable substitution

```
> Subst(x + x, x, cos(a))
expression(2 * cos(a))
```

4.10 Solving ordinary differential equations

OdeSolve notation not supported by Sym currently.

```
> yacas("OdeSolve(y' ==4*y)")
expression(C435 * exp(2 * x) + C439 * exp(-2 * x))
> yacas("OdeSolve(y' ==8*y)")
expression(C469 * exp(8 * x))
```

5 Matrices

```
> u1 <- Sym("u1")
> u2 <- Sym("u2")
> E4 <- List(List(u1, u1, 0), List(u1, 0, u2), List(0,
+      u2, 0))
> PrettyForm(E4)
```

$$\begin{array}{c} / \\ | \ (\ u1 \) \ (\ u1 \) \ (\ 0 \) \ | \\ | \ | \\ | \ (\ u1 \) \ (\ 0 \) \ (\ u2 \) \ | \\ | \ | \\ | \ (\ 0 \) \ (\ u2 \) \ (\ 0 \) \ | \\ \backslash \end{array}$$

5.1 Inverse

```
> E4i <- Inverse(E4)
> Simplify(E4i)
expression(list(list(1/u1, 0, -1/u2), list(0, 0, 1/u2), list(-1/u2,
1/u2, u1/u2^2)))
> PrettyForm(Simplify(E4i))
```

$$\begin{array}{c} / \\ | \ / \ 1 \ \backslash \ (\ 0 \) \ / \ -1 \ \backslash \ | \\ | \ | \ -- \ | \ | \ | \ -- \ | \ | \\ | \ \backslash \ u1 \ / \ \backslash \ u2 \ / \ | \\ | \ | \\ | \ (\ 0 \) \ (\ 0 \) \ / \ 1 \ \backslash \ | \\ | \ | \ | \ | \ | \ -- \ | \ | \\ | \ | \ | \ \backslash \ u2 \ / \ | \\ | \ | \\ | \ / \ -1 \ \backslash \ / \ 1 \ \backslash \ / \ u1 \ \backslash \ | \\ | \ | \ -- \ | \ | \ -- \ | \ | \ --- \ | \ | \\ | \ \backslash \ u2 \ / \ \backslash \ u2 \ / \ | \ 2 \ | \ | \\ | \ | \ | \ \backslash \ u2 \ / \ | \\ \backslash \end{array}$$

5.2 Determinant

```
> determinant(E4)
expression(-(u1 * u2^2))
> determinant(E4i)
expression(-(u1 * u2 * (u1 * u2^3))/(u1 * u2^2)^3)
> Simplify(E4i)
expression(list(list(1/u1, 0, -1/u2), list(0, 0, 1/u2), list(-1/u2,
1/u2, u1/u2^2)))
> Simplify(determinant(E4i))
expression(-1/(u1 * u2^2))
```