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

Parlamis Franklin, Rob Goedman, Gabor Grothendieck,
Søren Højsgaard, Ayal Pinkus

September 15, 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 % operator . . . 5
3.7 Printing with PrettyForm, PrettyPrint, TexForm and TeXForm . . . 6

4 Commands **7**
4.1 Factorial . . . 7
4.2 Taylor expansions . . . 7
4.3 Solving equations . . . 7
4.3.1 Solving equations symbolically . . . 7
4.3.2 Solving equations numerically . . . 7
4.4 Expanding polynomials . . . 8
4.5 Simplifying an expression . . . 8
4.6 Analytical derivatives . . . 8
4.7 Integration . . . 8
4.8 Limits . . . 9
4.9 Variable substitution . . . 9
4.10 Solving ordinary differential equations . . . 9

5 Matrices **9**
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)

823603;

> Sym("10 * 2") * 5 + Sym(7)^7

823643;

> (Sym(10) + 2) * 5 + Sym(7)^7

823603;

> Sym("(10+2)*5 + 7^7")

823603;

> Sym("1/14 + 5/21 * (30 - 1+1/2)")

149/21;
```

Numerical evaluations:

```
> Sym("-12/2")

-6;

> Eval(Sym("-12/2"))

NULL
```

Working with symbolic expressions:

```
> x <- Sym("x")
> Factor(x^2 - 1)

(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

(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)

Cos(x)^2+0.70807;
```

Differentiation:

```
> Deriv(Sin(x), x)

Cos(x);
```

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

```
> Set(A, "ssss")

ssss;

> A <- Sym("A")
> 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")

True;

> Integrate(Sin(x), x, a, b)

In function "Check" : CommandLine(1) : "Found bound variable B which should have been unbound, in MatchLin
```

Expanding polynomials:

```
> Expand((1 + x)^3)

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)

"$x + \frac{x ^{2}}{2}  + \frac{x ^{3}}{6}  + 1$";
```

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")

60;

> n <- Sym("n")
> Set(n, n + n)

120;

> a <- Sym("a")
> Set(z, Cos(a))

Cos(a);

> z <- Sym("z")
> z + z

2*Cos(a);
```

To clear a variable binding execute Clear():

```
> Set(n, 1)

1;

> n <- Sym("n")
> n

1;

> Clear(n)

True;

> n

n;
```

3.2 Symbolic and numerical evaluations, precision

Evaluations are generally exact:

```
> Exp(0)

1;

> Exp(1)

Exp(1);

> Sin(Pi/4)

Sqrt(1/2);

> Sym("355/113")

355/113;
```

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

```
> N(Exp(1))

2.71828;

> N(Sin(Pi/4))

0.707134;

> N(355/113)

3.14159292035398;
```

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

```
> N(355/133, 20)

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)

True;

> N("355/113")

3.14159;
```

3.3 Rational numbers

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

```
> Sym("55 / 10")

11/2;
```

3.4 Symbolic calculation

Some exact manipulations :

```
> Sym("1/14+5/21*(1*30-(1+1/2)*5^2)")

(-12)/7;

> x <- Sym("x")
> 0 + x

x;

> y <- Sym("y")
> x + y

x+y;

> Sin(ArcSin(x)) + Tan(ArcTan(y))

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

-1;

> 7 + 3 * I

Complex(7,3);

> Conjugate("%")

Complex(7,-3);

> Exp(3 * I)

Complex(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

(x+1)^3;

> z <- Sym("%")
> z

(x+1)^3;
```

3.7 Printing with PrettyForm, PrettyPrint, TexForm and TeXForm

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

{{a,b},{c,d}};

> B

(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 Ln(y) around y=1):

```
> InverseTaylor(Exp(x), x, 0, 2)

expression(x - 1 - (x - 1)^2/2)

> Taylor(Ln(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 D. function also accepts an argument specifying how often the derivative has to be taken, e.g:

```
> D.(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)

In function "Check" : CommandLine(1) : "Found bound variable B which should have been unbound, in MatchLin

> Integrate(Ln(x), x, a, b)

In function "Check" : CommandLine(1) : "Found bound variable B which should have been unbound, in MatchLin

> Integrate(1/(x^2 - 1), x)

In function "Check" : CommandLine(1) : "Found bound variable B which should have been unbound, in MatchLin

> Integrate(Sin(a * x)^2 * Cos(b * x), x)

In function "Check" : CommandLine(1) : "Found bound variable B which should have been unbound, in MatchLin
```

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(C800 * exp(2 * x) + C804 * exp(-2 * x))

> yacas("OdeSolve(y' ==8*y)")

expression(C834 * 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)
/                      \
| ( u1 ) ( u1 ) ( 0 )   |
|                       |
| ( u1 ) ( 0 )  ( u2 )  |
|                       |
| ( 0 )  ( u2 ) ( 0 )   |
\                      /
```

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))
/                         \
| / 1  \ ( 0 )  / -1 \     |
| | -- |        | -- |    |
| \ u1 /        \ u2 /    |
|                         |
| ( 0 )  ( 0 )  / 1  \    |
|               | -- |    |
|               \ u2 /    |
|                         |
| / -1 \ / 1  \ / u1  \   |
| | -- | | -- | | --- |   |
| \ u2 / \ u2 / |   2 |   |
|               \ u2  /   |
\                        /
```

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))
```