

# Ryacas – an R interface to the yacas computer algebra system

Rob Goedman,      Gabor Grothendieck,      Søren Højsgaard,  
Ayal Pinkus

September 9, 2007

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>R expressions, yacas expressions and Sym objects</b>	<b>2</b>
2.1	R expressions . . . . .	2
2.2	yacas expressions . . . . .	2
2.3	Sym objects . . . . .	2
<b>3</b>	<b>A sample session</b>	<b>3</b>
<b>4</b>	<b>Simple Yacas calculations</b>	<b>6</b>
4.1	Setting and clearing a variable . . . . .	6
4.2	Symbolic and numerical evaluations, precision . . . . .	7
4.3	Rational numbers . . . . .	8
4.4	Symbolic calculation . . . . .	8
4.5	Complex numbers and the imaginary unit . . . . .	9
4.6	Recall the most recent line – the % operator . . . . .	10
4.7	Printing with PrettyForm, PrettyPrint, TexForm and TeXForm . . . . .	10
4.7.1	Standard form . . . . .	10
4.7.2	Pretty form . . . . .	11
4.7.3	TeX form . . . . .	11
<b>5</b>	<b>Commands</b>	<b>12</b>
5.1	Factorial . . . . .	12
5.2	Taylor expansions . . . . .	12
5.3	Solving equations . . . . .	13
5.3.1	Solving equations symbolically . . . . .	13
5.3.2	Solving equations numerically . . . . .	13
5.4	Expanding polynomials . . . . .	13
5.5	Simplifying an expression . . . . .	13
5.6	Analytical derivatives . . . . .	14
5.7	Integration . . . . .	14
5.8	Limits . . . . .	15
5.9	Variable substitution . . . . .	15
5.10	Solving ordinary differential equations . . . . .	15
<b>6</b>	<b>Matrices</b>	<b>15</b>
6.1	Inverse . . . . .	16
6.2	Determinant . . . . .	17

## 1 Introduction

**Ryacas** makes the **yacas** computer algebra system available from within **R**. The name **yacas** is short for “Yet Another Computer Algebra System”. The **yacas** program is developed by Ayal Pinkhuis (who is also the maintainer) and others, and is available at <http://yacas.sourceforge.net> for various platforms. There is a comprehensive documentation (300+ pages) of **yacas** (also available at <http://yacas.sourceforge.net>) and the documentation contains many examples.

## 2 R expressions, yacas expressions and Sym objects

The **Ryacas** package works by sending “commands” to **yacas** which makes the calculations and returns the result to **R**. There are various different formats of the return value as well

### 2.1 R expressions

A call to **yacas** may be in the form of an **R** expression which involves valid **R** calls, symbols or constants (though not all valid **R** expressions are valid). For example:

```
> exp1 <- yacas(expression(Factor(x^2 - 1)))
[1] "Starting Yacas!"
expression((x + 1) * (x - 1))
```

The result **exp1** is not an expression in the **R** sense but an object of class “**yacas**”. To evaluate the resulting expression numerically, we can do

```
> Eval(exp1, list(x = 4))
[1] 15
```

### 2.2 yacas expressions

Some commands are not proper **R** expressions. For example, typing

```
yacas(expression(D(x)Sin(x)))
```

produces an error. For such cases we can make a specification using the **yacas** syntax:

```
> yacas("D(x)Sin(x)")
expression(cos(x))
```

### 2.3 Sym objects

Probably the most elegant way of working with **yacas** is by using **Sym** objects. A **Sym** object is a **yacas** character string that has the “**Sym**” class. One can combine **Sym** objects with other **Sym** objects as well as to other **R** objects using **+**, **-** and other similar **R** operators.

The function `Sym(x)` coerces an object `x` to a `Sym` object by first coercing it to character and then changing its class to "Sym":

```
> x <- Sym("x")
expression(x)
```

Operations on `Sym` objects lead to new `Sym` objects:

```
> x + 4
expression(x + 4)
```

One can apply `sin`, `cos`, `tan`, `deriv`, `Integrate` and other provided functions to `Sym` objects. For example:

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

In this way the communication with `yacas` is "tacit".

It is important to note the difference between the R name `x` and the symbol `"x"` as illustrated below:

```
> x <- Sym("xs")
expression(xs)
> x
expression(xs)
> x + 4
expression(xs + 4)
> Eval(x + 4, list(xs = 5))
[1] 9
```

The convention in the following is 1) that `Sym` objects match with their names that they end with an 's', e.g.

```
> xs <- Sym("xs")
```

### 3 A sample session

Algebraic calculations:

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

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

Numerical evaluations:

```
> yacas(expression(N(-12/2)))
expression(-6)
```

```
> Sym("-12/2")
expression(-6)
```

Symbolic expressions:

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

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

Combining symbolic and numerical expressions:

```
> yacas(expression(N(Sin(1)^2 + Cos(x)^2)))
expression(cos(x)^2 + 0.7080734182)
```

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

Differentiation:

```
> yacas("D(x)Sin(x)")
expression(cos(x))
```

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

Integration:

```
> yacas("Integrate(x,a,b)Sin(x)")
expression(cos(a) - cos(b))
```

```
> as <- Sym("as")
> bs <- Sym("bs")
> Integrate(sin(xs), xs, as, bs)
expression(cos(as) - cos(bs))
```

Expanding polynomials:

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

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

Taylor expansion:

```
> yacas("texp := Taylor(x,0,3) Exp(x)")
expression(x + x^2/2 + x^3/6 + 1)
```

```
> texp <- Taylor(exp(xs), xs, 0, 3)
expression(xs + xs^2/2 + xs^3/6 + 1)
```

Printing the result in nice forms:

```
> yacas("PrettyForm(texp)")
      2      3
      x      x
x + -- + -- + 1
      2      6

> yacas("TeXForm(texp)", retclass = "unquote")
$x + \frac{x ^{2}}{2} + \frac{x ^{3}}{6} + 1$
```

```
> PrettyForm(texp)
      2      3
      xs     xs
xs + --- + --- + 1
      2      6

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

## 4 Simple Yacas calculations

### 4.1 Setting and clearing a variable

The function `Set()` and the operator `:=` can both be used to assign values to global variables.

```
> yacas("n := (10 + 2) * 5")
expression(60)
> yacas("n := n+n")
expression(120)
```

The same can be achieved with `Sym` objects: Consider:

```
> Set(ns, (10 + 2) * 5)
expression(60)
```

Now `ns` exists as a variable in `yacas` (and we can make computations on this variable as above). However we have no handle on this variable in R. Such a handle is obtained with

```
> ns <- Sym("ns")
```

Now the R variable `ns` refers to the `yacas` variable `ns` and we can make calculations directly from R, e.g.:

```
> Set(ns, 123)
expression(123)
> ns
expression(123)
> ns^2
expression(15129)
```

Likewise:

```
> as <- Sym("as")
> zs <- Sym("zs")
> Set(zs, cos(as))
expression(cos(as))
> zs + zs
expression(2 * cos(as))
```

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

```
> yacas(expression(n))
expression(120)
> yacas("Clear(n)")
expression(TRUE)
> yacas(expression(n))
expression(n)
```

```

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

```

## 4.2 Symbolic and numerical evaluations, precision

Evaluations are generally exact:

```

> yacas("Exp(0)")
expression(1)
> yacas("Exp(1)")
expression(exp(1))
> yacas("Sin(Pi/4)")
expression(root(1/2, 2))
> yacas("355/113")
expression(355/113)

```

```

> exp(Sym(0))
expression(1)
> exp(Sym(1))
expression(exp(1))
> 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:

```

> yacas("N(Exp(1))")
expression(2.7182818284)
> yacas("N(Sin(Pi/4))")
expression(0.70710678118)
> yacas("N(355/113)")
expression(3.1415929203)

```

```

> N(exp(1))
expression(2.71828182845905)
> N(sin(Pi/4))
expression(0.70710678118)
> N(355/113)
expression(3.14159292035398)

```

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

```

> yacas("N(355/113,20)")
expression(2.66917293233083)

```

```

> N("355/113", 20)
expression(3.14159292035398)

```

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

```

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

```

```

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

```

### 4.3 Rational numbers

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

```

> yacas(expression(55/10))
expression(11/2)

```

```

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

```

### 4.4 Symbolic calculation

Some exact manipulations :

```

> yacas("1/14+5/21*(30-(1+1/2)*5^2)")
expression(-12/7)
> yacas("0+x")
expression(x)
> yacas("x+1*y")
expression(x + y)
> yacas("Sin(ArcSin(alpha))+Tan(ArcTan(beta))")
expression(alpha + beta)

```

```

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

```

## 4.5 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  $\text{Complex}(a,b)$  for  $a+ib$ .

```

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

```

```

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

```

## 4.6 Recall the most recent line – the % operator

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

```

> yacas("(1+x)^3")
expression((x + 1)^3)
> yacas("%")
expression((x + 1)^3)
> yacas("z:= %")
expression((x + 1)^3)

```

```

> (1 + x)^3
expression((xs + 1)^3)
> zs <- Sym("%")
> zs
expression((xs + 1)^3)

```

## 4.7 Printing with PrettyForm, PrettyPrint, TexForm and TeX-Form

There are different ways of displaying the output.

### 4.7.1 Standard form

The (standard) yacas form is:

```

> yacas("A:={{a,b},{c,d}}")
expression(list(list(a, b), list(c, d)))
> yacas("B:= (1+x)^2+k^3")
expression((x + 1)^2 + k^3)
> yacas("A")
expression(list(list(a, b), list(c, d)))
> yacas("B")
expression((x + 1)^2 + k^3)

```

```

> as <- Sym("as")
> bs <- Sym("bs")
> cs <- Sym("cs")
> ds <- Sym("ds")
> A <- List(List(as, bs), List(cs, ds))
> ks <- Sym("ks")
> B <- (1 + xs)^2 + ks^3
> A
expression(list(list(as, bs), list(cs, ds)))
> B
expression((xs + 1)^2 + ks^3)

```

#### 4.7.2 Pretty form

The Pretty form is:

```

> yacas("PrettyForm(A)")
/
| ( a ) ( b ) |
|             |
| ( c ) ( d ) |
\             /
> yacas("PrettyForm(B)")
      2    3
( x + 1 ) + k

```

```

> PrettyForm(A)
/
| ( as ) ( bs ) |
|             |
| ( cs ) ( ds ) |
\             /
> PrettyForm(B)
      2    3
( xs + 1 ) + ks

```

#### 4.7.3 TeX form

The output can be displayed in TeX form:

```

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

```

```

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

```

## 5 Commands

### 5.1 Factorial

```
> yacas("40!")  
expression(8.15915283247898e+47)
```

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

### 5.2 Taylor expansions

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

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

```
> xs <- Sym("xs")  
> Taylor(exp(xs), xs, 0, 3)  
expression(xs + xs^2/2 + xs^3/6 + 1)  
  
> as <- Sym("as")  
> Taylor(exp(x), x, as, 3)  
expression(exp(as) + exp(as) * (xs - as) + (xs - as)^2 * exp(as)/2 +  
           (xs - as)^3 * exp(as)/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  $y = 1$ ):

```
> yacas("InverseTaylor(x,0,2)Exp(x)")  
expression(x - 1 - (x - 1)^2/2)  
  
> yacas("Taylor(y,1,2)Ln(y)")  
expression(y - 1 - (y - 1)^2/2)
```

```
> InverseTaylor(exp(xs), xs, 0, 2)  
expression(xs + xs^2/2 + 1)  
  
> Taylor(log(ys), ys, 1, 2)  
expression(ys - 1 - (ys - 1)^2/2)
```

## 5.3 Solving equations

### 5.3.1 Solving equations symbolically

Solve equations symbolically with the `Solve()` function:

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

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

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

### 5.3.2 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:

```
> yacas("Newton(Sin(x)-Exp(x),x, 0.5, 0.0001)")
expression(-3.18306)
```

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

## 5.4 Expanding polynomials

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

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

## 5.5 Simplifying an expression

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

```

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

```

```

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

```

## 5.6 Analytical derivatives

Analytical derivatives of functions can be evaluated with the `D()` and `deriv()` functions:

```

> yacas("D(x) Sin(x)")
expression(cos(x))

```

```

> deriv(sin(xs), xs)
expression(cos(xs))

```

These functions also accept an argument specifying how often the derivative has to be taken, e.g.:

```

> yacas("D(x,4)Sin(x)")
expression(sin(x))

```

```

> deriv(sin(xs), xs, 4)
expression(sin(xs))

```

## 5.7 Integration

```

> yacas("Integrate(x)Sin(a*x)^2*cos(b*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)

```

```

> a <- Sym("a")
> b <- Sym("b")
> Integrate(sin(a * x)^2 * cos(b * x), x)
expression((2 * sin(b * xs)/b - (sin(-2 * xs * a - b * xs)/(-2 *
a - b) + sin(-2 * xs * a + b * xs)/(-2 * a + b)))/4)

```

## 5.8 Limits

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

```
> ns <- Sym("ns")
> Limit((1 + (1/ns))^ns, ns, Infinity)
expression(exp(1))
> hs <- Sym("hs")
> Limit((sin(xs + hs) - sin(xs))/hs, hs, 0)
expression(cos(xs))
```

## 5.9 Variable substitution

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

```
> Subst(xs + xs, xs, cos(as))
expression(2 * cos(as))
```

## 5.10 Solving ordinary differential equations

```
> yacas("OdeSolve(y'==4*y)")
expression(C345 * exp(2 * x) + C349 * exp(-2 * x))
> yacas("OdeSolve(y'==8*y)")
expression(C379 * exp(8 * x))
```

## 6 Matrices

```
> yacas("E4:={ {u1,u1,0},{u1,0,u2},{0,u2,0} }")
expression(list(list(u1, u1, 0), list(u1, 0, u2), list(0, u2,
0)))
> yacas("PrettyForm(E4)")
/
| ( u1 ) ( u1 ) ( 0 ) |
|
| ( u1 ) ( 0 ) ( u2 ) |
|
| ( 0 ) ( u2 ) ( 0 ) |
\
```

```

> 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 ) |
\
/

```

## 6.1 Inverse

```

> yacas("E4i:=Inverse(E4)")
expression(list(list(u2^2/(u1 * u2^2), 0, -(u1 * u2)/(u1 * u2^2)),
  list(0, 0, u1 * u2/(u1 * u2^2)), list(-(u1 * u2)/(u1 * u2^2),
    u1 * u2/(u1 * u2^2), u1^2/(u1 * u2^2))))
> yacas("Simplify(E4i)")
expression(list(list(1/u1, 0, -1/u2), list(0, 0, 1/u2), list(-1/u2,
  1/u2, u1/u2^2)))
> yacas("PrettyForm(Simplify(E4i))")

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

```

```

> 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 \ |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| / -1 \ / 1 \ / u1 \ |
| | -- | | -- | | --- | |
| \ u2 / \ u2 / | 2 | |
| | | | | | | |
| | | | | | | |
\
/

```

## 6.2 Determinant

```

> yacas("Determinant(E4)")

expression(-(u1 * u2^2))

> yacas("Determinant(E4i)")

expression(-(u1 * u2 * (u1 * u2^3))/(u1 * u2^2)^3)

> yacas("Simplify(E4i)")

expression(list(list(1/u1, 0, -1/u2), list(0, 0, 1/u2), list(-1/u2,
1/u2, u1/u2^2)))

> yacas("Simplify(Determinant(E4i))")

expression(-1/(u1 * u2^2))

```

```

> 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))

```

## 7 Miscellaneous

Note that the value returned by `yacas` can be of different types:

```
> yacas(expression(Factor(x^2 - 1)), retclass = "unquote")
*" ("+" (x ,1 ),"- (x ,1 ))
> yacas(expression(Factor(x^2 - 1)), retclass = "character")
"*" ("+" (x ,1 ),"- (x ,1 ))
```