Ryacas – an R interface to the yacas computer algebra system

Rob Goedman, Gabor Grothendieck, Søren Højsgaard, Ayal Pinkus

February 26, 2017

Contents

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)))

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

expression(823643)

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

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

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

<OMOBJ>
  <OMS cd="logic1" name="true"/>
</OMOBJ>
> 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

<OMOBJ>
  <OMS cd="logic1" name="true"/>
</OMOBJ>
> 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)

> #yacas("Set(z, Cos(a))")
> #yacas("z+z")
```

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(0.7071067811)
> yacas("N(355/113)")
expression(3.1415929203)
```

```
> N(exp(1))
expression(2.71828182845905)
> N(sin(Pi/4))
expression(0.7071067811)
> N(355/113)
expression(3.14159292035398)
```

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

```
> yacas("N(355/133,20)")
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(Precision(5))
> yacas("N(355/113)")
expression(3.1415929203)
```

```
> Precision(5)
expression(Precision(5))
> N("355/113")
expression(3.1415929203)
```

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 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 TeXForm

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 )  |
\             /

<OMOBJ>
  <OMS cd="logic1" name="true"/>
</OMOBJ>

> yacas("PrettyForm(B)")

         2    3
( x + 1 )  + k

<OMOBJ>
  <OMS cd="logic1" name="true"/>
</OMOBJ>
```

```
> PrettyForm(A)

/               \
| ( as ) ( bs )  |
|               |
| ( cs ) ( ds )  |
\               /

<OMOBJ>
  <OMS cd="logic1" name="true"/>
</OMOBJ>

> PrettyForm(B)

          2     3
( xs + 1 )  + ks

<OMOBJ>
  <OMS cd="logic1" name="true"/>
</OMOBJ>
```