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

Rob Goedman, Gabor Grothendieck, Søren Højsgaard, Ayal Pinkus

Ryacas version 0.2-12.1 as of 2015-06-01

Contents

1 Introduction

Ryacas is an `R` package that makes the `yacas` computer algebra system available from within `R`. The name `yacas` is short for "Yet Another Computer Algebra System". **Ryacas** is available on the The

Comprehensive R Archive Network (CRAN) at `http://cran.r-project.org/`. In addtition, **Ryacas** is also described on `https://code.google.com/p/ryacas/`.

The `yacas` program is developed by Ayal Pinkhuis (who is also the maintainer of `yacas`) and others, and is available at http://yacas.sourceforge.net for various platforms. This site also containsa comprehensive documentation 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.

Ressources:

`http://yacas.sourceforge.net/ref.book.pdf`

2 A small session – polynomials

The **Ryacas** package works by sending "commands" to `yacas` which makes the calculations and returns the result to `R`. This can be done in various ways.

2.1 Calling with `yacas` commands as text strings

A simple way of calling `yacas` from `R` is by submitting `yacas` commands as a text string to the `yacas()` function.

```
> out <- yacas( "Expand( (1 + u)^4 )" ); out
 4        3        2
u  + 4 * u  + 6 * u  + 4 * u + 1
> class( out )
[1] "yacas"
```

Notice that the output from `yacas()` looks like an `R` expression, but it is not; see below.

The result can be displayed nicer in different forms. For example

```
> PrettyForm( out )
PrettyForm(   )
> TeXForm( out )
TeXForm(   )
```

To obtain the tex form as a string to put into a file we can do:

```
> yacas(TeXForm(out), retclass = "unquote")
TeXForm(   )
```

2.2 The output from `yacas`

The output from `yacas()` looks like an `R` expression, bit it is not; it is a `"yacas"` object.

```
> class( out )
[1] "yacas"
> names( out )
[1] ""           "PrettyForm"
> lapply( out, head )
[[1]]
NULL

$PrettyForm
[1] " 4        3        2           " "u  + 4 * u  + 6 * u  + 4 * u + 1"
```

The `text` slot in the output is an `R` expression that can be evaluated numerically as

```
> eval( out$text, list(u=1) )
NULL
```

in **Ryacas**, there is an `Eval()` function that operation that operates on `yacas` objects so one can also do

```
> Eval( out, list(u=1) )
NULL
```

2.3 Calling `yacas` with `R` expressions

In **Ryacas**, `R` expressions can be sent to `yacas` so one can do

```
> e <- expression( Expand( (1 + u)^4 ) )
> yacas( e )
 4        3        2
u  + 4 * u  + 6 * u  + 4 * u + 1
```

This works fine as long as a valid `R` expression can be created – but consider expanding a polynomial in two variables

```
> e <- expression( Expand((1+x-y)^2, x) )         ## This works
> ## e <- expression( Expand((1+x-y)^2, {x,y}) )  ## This fails
```

The latter case fails because of the `yacas` syntax with curly braces which `R`'s `expression()` function can not interpret. So in the latter case we have to pass the argument as a text string:

```
> yacas( "Expand((1+x-y)^2, {x,y})" )
 2                       2
x  + ( -2 * y + 2 ) * x + y  - 2 * y + 1
```

Another example of an operation that can not be handled using `R` expressions is differentiation. Doing

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

produces an error. For such cases we can do

```
> yacas("D(x)Sin(x)")
Cos( x )
```

2.4 Using Sym objects

An elegant way of working with yacas is by using Sym objects. A Sym object is a character string that has class Sym. 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. For clarity we use two different symbols here:

```
> x_ <- Sym("x")
> x_
x
> dput( x_ )
structure("x", class = c("Sym", "character"))
```

One can combine Sym objects with other Sym objects as well as to other R objects using +, - and other similar R operators. For example

```
> x_ + 4
x + 4
> Eval(x_+4, list(x=1))
NULL
```

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

```
> x <- Sym("x")
> Integrate(sin(x), x)
In function "Check" : CommandLine(1) : "Found bound variable A which should have been unbound, in Mat
> deriv(cos(x), x)
-( Sin( x ) )
```

2.5 Recall the most recent line – the % operator

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

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

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

2.6 Setting and clearing a variable

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

```
> yacas("n := 10")
10
> yacas("n := n + n")
20
> yacas("Set(z, Cos(a))")
True
> yacas("z+z")
2 * Cos( a )
```

Notice that these are variables in `yacas`, not in `R`. Variables in `yacas` can be removed `Clear()`

```
> yacas("n")
20
> yacas("Clear(n, z)")
True
> yacas("n")
n
```

Variables in `yacas` can also be set with `Set()` from `R`:

```
> Set(m, 10)
10
```

Now `m` 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 with `Sym` objects:

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

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

```
> Set(m, 123)
123
> m^2
15129
```

3 Yacas calculations

3.1 Symbolic and numerical calculations, precision

```
> yacas(expression( (1/2) / (4/5) ))
5
-
8
> Sym("1/2") / Sym("4/5") ## WRONG RESULT
1
--
40
> Sym("1/14 + 5/21 * (30 - 1 + 1/2)")
149
---
21
> Sym("55/10")
11
--
2
```

Notice that rational numbers will stay rational as long as numerators and denominators are integers.

Evaluations are generally exact:

```
> yacas("Exp(0)")
1
> yacas("Exp(1)")
Exp( 1 )
> yacas("Sin(Pi/4)")
    / 1 \
Sqrt| - |
    \ 2 /
> yacas("355/113")
355
---
113
```

```
> exp(Sym(0))
1
> exp(Sym(1))
Exp( 1 )
> sin(Pi/4)
    / 1 \
Sqrt| - |
    \ 2 /
> Sym("355/113")
355
---
113
```

To obtain a numerical evaluation (approximation), the `N()` function can be used: The `N()` function has an optional second argument, the required precision:

```
> x <- Sym("55/10")
> N( x )
5.5
> Eval( N( x ) )
NULL
> yacas("355/113")
355
---
113
> N( yacas("355/113"), 8 )
CommandLine(1) : Expecting ) closing bracket for sub-expression, but got 8 instead
> N( "355/113", 8 )
3.14159292
>
```

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)")
True
> yacas("N(355/113)")
3.14159
> ## Alternative
> ## Precision(5)
> ## N("355/113")
```

Combining symbolic and numerical expressions:

```
> x <- Sym("x")
> N(sin(1)^2 + cos(x)^2)
        2
Cos( x )  + 0.708073418273571
```

3.2 Differentiation

```
> yacas("D(x) Sin(x)")
Cos( x )
> yacas("D(x, 2) Sin(x)")
-( Sin( x ) )
```

```
> x <- Sym("x")
> deriv(sin(x), x)
Cos( x )
```

3.3 Integration

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

```
> a <- Sym("a"); b <- Sym("b")
> Integrate(sin(x), x, a, b)
Cos( a ) - Cos( b )
```

3.4 Expanding polynomials

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

```
> xs <- Sym("xs")
> Expand((1+xs)^3)
  3         2
xs  + 3 * xs  + 3 * xs + 1
> Factor(xs^2-1)
( xs + 1 ) * ( xs - 1 )
```

3.5 Taylor expansion

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

```
> xs <- Sym("xs")
xs
> texp <- Taylor(exp(xs), xs, 0, 3)
       2     3
     xs    xs
xs + --- + --- + 1
      2     6
```

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

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

         3
( x - a )  * Exp( a )
---------------------
          6
```

```
> xs <- Sym("xs")
> Taylor(exp(xs),xs,0,3)
       2     3
     xs    xs
xs + --- + --- + 1
      2     6
> as <- Sym("as")
> Taylor(exp(x),x,as,3)
Exp( as ) + Exp( as ) * ( x - as ) +

          2                          3
( x - as )  * Exp( as )   ( x - as )  * Exp( as )
----------------------- + -----------------------
           2                          6
```

3.6 Inverse Taylor

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)")
                 2
        ( x - 1 )
x - 1 - ----------
            2
> yacas("Taylor(y,1,2)Ln(y)")
                 2
        ( y - 1 )
y - 1 - ----------
            2
```

```
> ys <- Sym("ys"); xs <- Sym("xs")
> InverseTaylor(exp(xs),xs,0,2)
       2
     xs
xs + --- + 1
      2
> Taylor(log(ys),ys,1,2)
                   2
         ( ys - 1 )
ys - 1 - -----------
              2
```

3.7 Factorial

```
> yacas("40!")
815915283247897734345611269596115894272000000000
> yacas("40!", retclass = "character")
815915283247897734345611269596115894272000000000
> Factorial(40)
815915283247897734345611269596115894272000000000
```

3.8 Limits

```
> yacas( "Limit(x,0) Sin(x)/x" )
1
> yacas( "Limit(n,Infinity) (1+(1/n))^n" )
Exp( 1 )
> yacas( "Limit(h,0) (Sin(x+h)-Sin(x))/h" )
Cos( x )
```

```
> x <- Sym("x"); n <- Sym("n"); h <- Sym("h")
> Limit( sin(x) / x, x, 0)
1
> Limit( (1 + (1 / n))^n, n, Infinity )
Exp( 1 )
> Limit( (sin(x + h) - sin(x) ) / h, h, 0)
Cos( x )
```

3.9 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")
         3           3
( x + y )  - ( x - y )
> yacas("Simplify(%)")
     2           3
6 * x  * y + 2 * y
```

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

3.10 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")
Complex( 0 , 1 )
> yacas("I^2")
-1
> yacas("7+3*I")
Complex( 7 , 3 )
> yacas("Conjugate(%)")
Complex( 7 , -3 )
> yacas("Exp(3*I)")
Complex( Cos( 3 ) , Sin( 3 ) )
```

```
> I
Complex( 0 , 1 )
> I^2
-1
> 7+3*I
Complex( 7 , 3 )
> Conjugate("%")
Complex( 7 , -3 )
> exp(3*I)
Complex( Cos( 3 ) , Sin( 3 ) )
```

```
> z <- 7+3*I
> 1/z
       / 7    -3 \
Complex| -- , -- |
       \ 58   58 /
> z*(1/z)
1
```

We get the famous identity $\exp(i\pi) = -1$ with

```
> exp(I*Pi)
-1
```

3.11 Solving equations

3.11.1 Solving equations symbolically

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

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

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

```
> Solve(List(xs^2+ys^2==6, xs-ys==3), List(xs,ys))
/                                          \
| /          /       2 \ \ ( ys == ys )  |
| \ xs == Sqrt\ 6 - ys  / /              |
\                                          /
```

```
> mu <- Sym("mu") # mean
> v <- Sym("v") # variance
> Solve(List(mu==(xs/(xs+ys)), v==((xs*ys)/(((xs+ys)^2) * (xs+ys+1)))),
+     List(xs,ys))
/                                                    \
| /         2                   \ /       xs      \  |
| |       mu  * ( 1 - mu )       | | ys == -- - xs |  |
| | xs == ---------------- - mu | \       mu      /  |
| \              v              /                    |
\                                                    /
```

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

3.11.2 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)")
-3.18306
```

```
> Newton(sin(xs)-exp(xs),xs, 0.5, 0.0001)
-3.18306
```

3.12 Solving ordinary differential equations

```
> yacas("OdeSolve(y''==4*y)")
C13766 * Exp( 2 * x ) + C13770 * Exp( -2 * x )
> yacas("OdeSolve(y'==8*y)")
C13800 * Exp( 8 * x )
```

4 Matrices

```
> PrettyPrinter()
True;
```

```
> A <- yacas( "A:={{4,-2,4,2},{-2,10,-2,-7},{4,-2,8,4},{2,-7,4,7}}" )
> PrettyForm(A) ## FIXME: SHOULD THIS WORK??
PrettyForm();
> yacas("PrettyForm(A)")
/                              \
| ( 4 )  ( -2 ) ( 4 )  ( 2 )   |
|                              |
| ( -2 ) ( 10 ) ( -2 ) ( -7 )  |
|                              |
| ( 4 )  ( -2 ) ( 8 )  ( 4 )   |
|                              |
| ( 2 )  ( -7 ) ( 4 )  ( 7 )   |
\                              /
> R <- yacas("R := Cholesky(A)")
> yacas("PrettyForm(R)")
/                            \
| ( 2 ) ( -1 ) ( 2 ) ( 1 )   |
|                            |
| ( 0 ) ( 3 )  ( 0 ) ( -2 )  |
|                            |
| ( 0 ) ( 0 )  ( 2 ) ( 1 )   |
|                            |
| ( 0 ) ( 0 )  ( 0 ) ( 1 )   |
\                            /
> yacas("RtR:=Transpose(R) * R")
{{4,-2,4,2},{-2,10,-2,-7},{4,-2,8,4},{2,-7,4,7}};
> yacas("PrettyForm(RtR)")
/                              \
| ( 4 )  ( -2 ) ( 4 )  ( 2 )   |
|                              |
| ( -2 ) ( 10 ) ( -2 ) ( -7 )  |
|                              |
| ( 4 )  ( -2 ) ( 8 )  ( 4 )   |
|                              |
| ( 2 )  ( -7 ) ( 4 )  ( 7 )   |
\                              /
```

```
> yacas("M:={ {u1,u1,0},{u1,0,u2},{0,u2,0} }")
{{u1,u1,0},{u1,0,u2},{0,u2,0}};
> yacas("PrettyForm(M)")
/                       \
| ( u1 ) ( u1 ) ( 0 )   |
|                       |
| ( u1 ) ( 0 )  ( u2 )  |
|                       |
| ( 0 )  ( u2 ) ( 0 )   |
\                       /
```

```
> u1 <- Sym("u1"); u2 <- Sym("u2")
> M <- List(List(u1, u1, 0), List(u1, 0, u2), List(0, u2, 0))
> PrettyForm( M )
/                       \
| ( u1 ) ( u1 ) ( 0 )   |
|                       |
| ( u1 ) ( 0 )  ( u2 )  |
|                       |
| ( 0 )  ( u2 ) ( 0 )   |
\                       /
```

```
> yacas(TeXForm(M), retclass = "unquote")
"\begin{pmatrix} u_1 & u_1 & 0 \\ u_1 & 0 & u_2 \\ 0 & u_2 & 0 \end{pmatrix}";
```

4.1 Inverse

```
> yacas("Mi:=Inverse( M )")
{{u2^2/(u1*u2^2),0,(-u1*u2)/(u1*u2^2)},{0,0,(u1*u2)/(u1*u2^2)},{(-u1*u2)/(u1*u2^2),(u1*u2)/(u1*u2^2),
> yacas("Simplify( Mi )")
{{1/u1,0,(-1)/u2},{0,0,1/u2},{(-1)/u2,1/u2,u1/u2^2}};
> yacas("PrettyForm(Simplify( Mi ))")
/                           \
| / 1  \ ( 0 )  / -1 \      |
| | -- |        | -- |     |
| \ u1 /        \ u2 /      |
|                           |
| ( 0 )  ( 0 )  / 1  \      |
|               | -- |      |
|               \ u2 /      |
|                           |
| / -1 \ / 1  \ / u1  \     |
| | -- | | -- | | --- |    |
| \ u2 / \ u2 / |   2 |     |
|               \ u2  /     |
\                           /
```

```
> Mi <- Inverse( M )
> Simplify( Mi )
{{1/u1,0,(-1)/u2},{0,0,1/u2},{(-1)/u2,1/u2,u1/u2^2}};
> PrettyForm(Simplify( Mi ))
/                          \
| / 1  \ ( 0 )  / -1 \     |
| | -- |        | -- |     |
| \ u1 /        \ u2 /     |
|                          |
| ( 0 )  ( 0 )  / 1  \     |
|               | -- |     |
|               \ u2 /     |
|                          |
| / -1 \ / 1  \ / u1  \    |
| | -- | | -- | | --- |   |
| \ u2 / \ u2 / |   2 |    |
|               \ u2  /    |
\                          /
```

4.2 Determinant

```
> yacas("Determinant( M )")
-u1*u2^2;
> yacas("Determinant( Mi )") ## FIXME: Whats up here?
(-u1*u2*u1*u2^3)/(u1*u2^2)^3;
> yacas("Simplify( Mi )")
{{1/u1,0,(-1)/u2},{0,0,1/u2},{(-1)/u2,1/u2,u1/u2^2}};
> yacas("Simplify(Determinant( Mi ))") ## FIXME: Whats up here?
(-1)/(u1*u2^2);
```

```
> determinant( M )
-u1*u2^2;
> determinant( Mi )
(-u1*u2*u1*u2^3)/(u1*u2^2)^3;
> Simplify( Mi )
{{1/u1,0,(-1)/u2},{0,0,1/u2},{(-1)/u2,1/u2,u1/u2^2}};
> Simplify(determinant( Mi ))
(-1)/(u1*u2^2);
```

5 Printing with PrettyForm, PrettyPrint, TexForm and TeXForm

Printing the result in nice forms:

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

```
> 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)
"$xs + \frac{xs ^{2}}{2}  + \frac{xs ^{3}}{6}  + 1$";
```

There are different ways of displaying the output.

5.0.1 Standard form

The (standard) yacas form is:

```
> yacas("A:={{a,b},{c,d}}")
{{a,b},{c,d}};
> yacas("B:= (1+x)^2+k^3")
(x+1)^2+k^3;
> yacas("A")
{{a,b},{c,d}};
> yacas("B")
(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
{{as,bs},{cs,ds}};
> B
(xs+1)^2+ks^3;
```

5.0.2 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
```

5.0.3 TeX form

The output can be displayed in TeX form:

```
> yacas("TeXForm(B)", retclass = "character")
"$\left( x + 1\right)  ^{2} + k ^{3}$";
```

This function sets up the function printer to print out the results on the command line. This can be reset to the internal printer with PrettyPrinter().

Currently implemented prettyprinters are: PrettyForm, TeXForm, Print and DefaultPrint.

```
> PrettyPrinter()
True;
```