

Package ‘Ryacas’

July 21, 2025

Version 1.1.5

Title R Interface to the 'Yacas' Computer Algebra System

Maintainer Mikkel Meyer Andersen <mikl@math.aau.dk>

Encoding UTF-8

Description Interface to the 'yacas' computer algebra system (<<http://www.yacas.org/>>).

Depends R (>= 3.3.0)

Imports Rcpp (>= 0.12.0), stats, methods, magrittr

LinkingTo Rcpp

Suggests devtools, exams, knitr, Matrix, pkgload, rmarkdown, igraph,
testthat (>= 2.1.0), unix, Rmpfr

License GPL

SystemRequirements C++14

URL <https://github.com/r-cas/ryacas>, <http://www.yacas.org>

BugReports <https://github.com/r-cas/ryacas/issues>

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation yes

Author Mikkel Meyer Andersen [aut, cre, cph],
Rob Goedman [aut, cph],
Gabor Grothendieck [aut, cph],
Søren Højsgaard [aut, cph],
Grzegorz Mazur [aut, cph],
Ayal Pinkus [aut, cph],
Nemanja Trifunovic [cph] (UTF-8 part of yacas
(src/yacas/include/yacas/utf8*))

Repository CRAN

Date/Publication 2023-01-16 18:30:02 UTC

Contents

Ryacas-package	3
as.character.yac_symbol	3
as_r	3
as_y	4
cbind.yac_symbol	4
deriv.yac_symbol	5
det	5
diag	6
diag<-	6
Hessian	6
integrate	7
Jacobian	7
lim	8
lower.tri	8
Math.yac_symbol	9
Ops.yac_symbol	9
pow	10
prod.yac_symbol	10
rbind.yac_symbol	11
simplify	11
solve.yac_symbol	12
sum.yac_symbol	13
t	13
tex	14
tr	14
upper.tri	15
vec	15
vech	16
with_value	16
yac	17
yac_assign	17
yac_cli	18
yac_expr	19
yac_silent	19
yac_str	20
yac_symbol	20
ysym	21
ysym_ls	21
y_eval	22
y_fn	22
y_print	23
y_rmvars	24
[.yac_symbol	24
[<-.yac_symbol	25
[[.yac_symbol	25
%*%	26

Ryacac-package

*R interface to yacas computer algebra package***Description**

Ryacac allows one to use the yacas computer algebra package entirely from within R.

Details

Please read the "Getting started" vignette.

as.character.yac_symbol

*Convert yac symbol to character***Description**

Convert yac symbol to character

Usage

```
## S3 method for class 'yac_symbol'
as.character(x, ...)
```

Arguments

x	A yac_symbol
...	not used

as_r

*Convert yacas object to R***Description**

If x is a yacas command as string, convert to a character vector/matrix in R. If x is a yac_symbol (e.g. from `ysym()`), then convert it to a numeric object if there are no variables or a character type if there are variables.

Usage

```
as_r(x)
```

Arguments

x yacas list or list of lists to convert

Details

In yacas a vector is a list, and a matrix is a list of lists.

<code>as_y</code>	<i>Convert R vector/matrix to yacas vector (list) or matrix (list of lists)</i>
-------------------	---

Description

Convert R vector/matrix to yacas vector (list) or matrix (list of lists)

Usage

`as_y(x)`

Arguments

x R vector to convert

<code>cbind.yac_symbol</code>	<i>Combine R Objects by Columns</i>
-------------------------------	-------------------------------------

Description

Combine R Objects by Columns

Usage

```
## S3 method for class 'yac_symbol'
cbind(..., deparse.level = 1)
```

Arguments

... Objects to bind
deparse.level Not used

deriv.yac_symbol	<i>Find the derivative of yac symbol</i>
------------------	--

Description

Find the derivative of yac symbol

Usage

```
## S3 method for class 'yac_symbol'
deriv(expr, ...)
```

Arguments

expr	A yac_symbol
...	variables as character vector to take derivate with respect to

det	<i>Matrix Determinant</i>
-----	---------------------------

Description

From `base::det()`.

Usage

```
det(x, ...)
```

Arguments

x	If yac_symbol treat as such, else call <code>base::det()</code> .
...	further arguments passed to <code>base::det()</code>

Examples

```
(x <- matrix(1:4, ncol = 2))
det(x)
det(ysym(x))
```

diag	<i>Matrix diagonals</i>
------	-------------------------

Description

From `base::diag()`.

Usage

`diag(x, ...)`

Arguments

x	If <code>yac_symbol</code> treat as such, else call <code>base::diag()</code> .
...	further arguments passed to <code>base::diag()</code>

diag<-	<i>Matrix diagonals</i>
--------	-------------------------

Description

From `base::diag()`.

Usage

`diag(x) <- value`

Arguments

x	If <code>yac_symbol</code> treat as such, else call <code>base::diag<-()</code> .
value	New value for <code>diag(x)</code>

Hessian	<i>Find the Hessian matrix of <code>yac</code> symbol</i>
---------	---

Description

Find the Hessian matrix of `yac` symbol

Usage

`Hessian(expr, ...)`

Arguments

expr	A <code>yac_symbol</code>
...	variables as character vector to take Hessian with respect to

integrate	<i>Integration of Functions</i>
-----------	---------------------------------

Description

If `f` is a `yac_symbol`, `yacas's Integrate()` is used. Else, `stats::integrate()` is used.

Usage

`integrate(f, ...)`

Arguments

<code>f</code>	Function to integrate. See details.
<code>...</code>	See details.

Details

Additional arguments:

- `yac_symbol`: `var`, `lower`, `upper`
- Else (`stats::integrate()`): `lower`, `upper`

Jacobian	<i>Find the Jacobian matrix of <code>yac</code> symbol</i>
----------	--

Description

Find the Jacobian matrix of `yac` symbol

Usage

`Jacobian(expr, ...)`

Arguments

<code>expr</code>	A <code>yac_symbol</code>
<code>...</code>	variables as character vector to take Jacobian with respect to

<code>lim</code>	<i>Limits</i>
------------------	---------------

Description

If first argument is a `yac_symbol`, `yacas`'s `Limit()` is used.

Usage

`lim(...)`

Arguments

`...` See details.

Details

Arguments:

- `yac_symbol`: `f`, `var`, `val`, `from_left`, `from_right`

<code>lower.tri</code>	<i>Lower and upper triangular part of a matrix</i>
------------------------	--

Description

Lower and upper triangular part of a matrix

Usage

`lower.tri(x, diag = FALSE)`

Arguments

`x` If `yac_symbol` treat as such, else call `base::lower.tri()/base::upper.tri()`.
`diag` Whether diagonal is included.

Math.yac_symbol	<i>Math functions</i>
-----------------	-----------------------

Description

Math functions

Usage

```
## S3 method for class 'yac_symbol'  
Math(x, ...)
```

Arguments

- x yac_symbol.
- ... further arguments passed to methods

Ops.yac_symbol	<i>Math operators</i>
----------------	-----------------------

Description

Math operators

Usage

```
## S3 method for class 'yac_symbol'  
Ops(e1, e2)
```

Arguments

- e1 A yac_symbol.
- e2 A yac_symbol.

pow	<i>Matrix Power</i>
-----	---------------------

Description

Matrix Power

Usage

```
pow(x, n, ...)  
  
## Default S3 method:  
pow(x, n, ...)
```

Arguments

- x If `yac_symbol` treat as such, else call `pow.default()`.
- n nth power of the square matrix.
- ... further arguments passed to `pow.default()`

Examples

```
(x <- matrix(c(1, 2, 2, 3), ncol = 2))  
pow(x, 2)  
pow(ysym(x), 2)
```

prod.yac_symbol	<i>Product of Vector Elements</i>
-----------------	-----------------------------------

Description

Product of Vector Elements

Usage

```
## S3 method for class 'yac_symbol'  
prod(expr, ..., na.rm = FALSE)
```

Arguments

- expr Expression to be multiplied
- ... Not used
- na.rm Not used

rbind.yac_symbol	<i>Combine R Objects by Rows</i>
------------------	----------------------------------

Description

Combine R Objects by Rows

Usage

```
## S3 method for class 'yac_symbol'  
rbind(..., deparse.level = 1)
```

Arguments

...	Objects to bind
deparse.level	Not used

simplify	<i>Simplify expression</i>
----------	----------------------------

Description

Simplify expression

Usage

```
simplify(x, timeout = 2)
```

Arguments

x	A yac_symbol
timeout	timeout in seconds before simplification is aborted; only works when package unix is available

solve.yac_symbol	<i>Solve a system of equations</i>
------------------	------------------------------------

Description

This generic function solves the equation $a x = b$ for x .

Usage

```
## S3 method for class 'yac_symbol'
solve(a, b, ...)
```

Arguments

<code>a</code>	A <code>yac_symbol</code>
<code>b</code>	A <code>yac_symbol</code> or a value, see details and examples.
<code>...</code>	See details and examples.

Details

When `a` is a matrix and `b` not provided, this finds the inverse of `a`. When `a` is a matrix and a vector `b` is provided, the linear system of equations is solved.

Note that solving non-linear equations:

- `solve(a, b)`: find roots of `a` for variable `b`, i.e. `yacas Solve(a == 0, b)`
- `solve(a, b, v)`: find solutions to `a == b` for variable `v`, i.e. `yacas Solve(a == b, v)`

This also works for a system of equations (when `a` is a vector)

Examples

```
A <- outer(0:3, 1:4, "-") + diag(2:5)
a <- 1:4
B <- ysym(A)
b <- ysym(a)
solve(A)
solve(B)
solve(A, a)
solve(B, b)

poly <- ysym("x^2 - x - 6")
solve(poly, "x") # Solve(poly == 0, x)
solve(poly, 3, "x") # Solve(poly == 3, x)
```

sum.yac_symbol	<i>Summation</i>
----------------	------------------

Description

If only expr given: sum elements.

Usage

```
## S3 method for class 'yac_symbol'  
sum(expr, var, lower, upper, ..., na.rm = FALSE)
```

Arguments

expr	Expression to be summed
var	Variable to sum
lower	Lower limit
upper	Upper limit
...	Not used
na.rm	Not used

Details

Else: sums expr by letting var taking values from lower to upper (potentially Inf)

t	<i>t</i>
---	----------

Description

t

Usage

```
## S3 method for class 'yac_symbol'  
t(x)
```

Arguments

x	If yac_symbol treat as such, else call <code>base::t()</code> .
---	---

tex	<i>Export object to TeX</i>
-----	-----------------------------

Description

Export object to TeX

Usage

```
tex(x)
```

Arguments

x	A yac_symbol
---	--------------

tr	<i>Matrix Trace</i>
----	---------------------

Description

The trace of a square matrix is the sum of the diagonal elements.

Usage

```
tr(x, ...)

## Default S3 method:
tr(x, ...)
```

Arguments

x	If yac_symbol treat as such, else call <code>tr.default()</code> .
...	further arguments passed to <code>tr.default()</code>

Examples

```
(x <- matrix(1:4, ncol = 2))
tr(x)
tr(ysym(x))
```

upper.tri	<i>Lower and upper triangular part of a matrix</i>
-----------	--

Description

Lower and upper triangular part of a matrix

Usage

```
upper.tri(x, diag = FALSE)
```

Arguments

x	If <code>yac_symbol</code> treat as such, else call <code>base::lower.tri()/base::upper.tri()</code> .
diag	Whether diagonal is included.

vec	<i>Vectorize</i>
-----	------------------

Description

Vectorize

Usage

```
vec(x, ...)

## Default S3 method:
vec(x, ...)
```

Arguments

x	If <code>yac_symbol</code> treat as such, else call <code>base::as.vector()</code> .
...	further arguments passed to <code>base::as.vector()</code>

Examples

```
(x <- matrix(1:9, ncol = 3))
vec(x)
vec(ysym(x))
```

vech	<i>Half-Vectorize</i>
------	-----------------------

Description

Half-Vectorize

Usage

```
vech(x, ...)  
  
## Default S3 method:  
vech(x, ...)
```

Arguments

x If yac_symbol treat as such, else call `vech.default()`.
... further arguments passed to `vech.default()`

Examples

```
A <- mtcars[, c(1, 3, 4, 5, 6, 7)]  
x <- cov(A)  
vech(x)  
vech(ysym(x))
```

with_value	<i>Give a variable a value</i>
------------	--------------------------------

Description

Give a variable a value

Usage

```
with_value(x, var, val)
```

Arguments

x yac_symbol
var Variable
val Value

`yac`*Run yacas command*

Description

Run yacas command

Usage

```
yac(x, rettype = c("str", "expr", "silent"))
```

Arguments

<code>x</code>	yacas command
<code>rettype</code>	str for string/character, expr for expression, silent for silent

Examples

```
yac("D(x) x^2 + 4*x")  
yac("D(x) x^2 + 4*x", rettype = "str")  
yac("D(x) x^2 + 4*x", rettype = "expr")  
yac("D(x) x^2 + 4*x", rettype = "silent")
```

`yac_assign`*Assign yacas variable*

Description

Assign yacas variable

Usage

```
yac_assign(value, x)
```

Arguments

<code>value</code>	Expression
<code>x</code>	Variable name

`yac_cli`*yacas command line interface*

Description

Interactive interface to the yacas

Usage

```
yac_cli(enable_history = TRUE)
```

Arguments

`enable_history` Use R history such that previous yacas commands can be used. Default is TRUE.

Details

The user types valid yacas input and presses return. Type 'quit' to return to R prompt.

Value

Output of yacas is returned.

Note

Note that command will use R `history()` and modify it by default. Yacas is given a limited amount of time to complete, otherwise `\[1\]` `CommandLine(1)` : User interrupted calculation is returned. E.g. `Taylor(x,0,5) 1/(1+x)` will work, but `Taylor(x,0,12) 1/(1+x)` is likely to take too long.

References

<https://yacas.sourceforge.io/>

Examples

```
## Not run:
yac_cli()
(x+y)^3-(x-y)^3
Simplify(%)
q

## End(Not run)
```

`yac_expr`*Run yacas command returning R expression*

Description

Run yacas command returning R expression

Usage

```
yac_expr(x)
```

Arguments

x yacas command

Examples

```
yac_expr("D(x) x^2 + 4*x")
yac_expr("Limit(x, 1) (x^2 - 1)/(x - 1)")
yac_expr("Sum(n, 1, Infinity, (1/2)^n)")
yac_expr("Fibonacci(10)")
yac_expr("Sum(n, 1, 10, Fibonacci(n))")
```

`yac_silent`*Run yacas command silently*

Description

Run yacas command silently

Usage

```
yac_silent(x)
```

Arguments

x yacas command

yac_str	<i>Run yacas command returning string/character</i>
---------	---

Description

Run yacas command returning string/character

Usage

yac_str(x)

Arguments

x	yacas command
---	---------------

Examples

```
yac_str("D(x) x^2 + 4*x")
yac_str("Limit(x, 1) (x^2 - 1)/(x - 1)")
yac_str("Sum(n, 1, Infinity, (1/2)^n)")
yac_str("Fibonacci(10)")
yac_str("Sum(n, 1, 10, Fibonacci(n))")
yac_str("TeXForm(x^2 - 1)")
```

yac_symbol	<i>Make a yacas symbol</i>
------------	----------------------------

Description

This is an alias for [ysym\(\)](#). See description there.

Usage

yac_symbol(x)

Arguments

x	A vector or a matrix
---	----------------------

Value

A yac_symbol

ysym	<i>Make a yacas symbol</i>
------	----------------------------

Description

Note that this results in multiple calls to yacas and the performance may be slower than manually using e.g. `yac_str()`.

Usage

```
ysym(x)
```

Arguments

x	A vector or a matrix
---	----------------------

Value

A yac_symbol

ysym_ls	<i>List defined yac_symbols</i>
---------	---------------------------------

Description

List defined yac_symbols

Usage

```
ysym_ls(print_details = FALSE)
```

Arguments

print_details	print content of symbols
---------------	--------------------------

y_eval

Evaluate a yacas expression

Description

Evaluate a yacas expression by replacing variables with values as for the given list.

Usage

```
y_eval(expr, ..., as.r = FALSE)
```

Arguments

expr	a valid yacas expression
...	a list of assignments (see example)
as.r	if TRUE, then the expression is evaluated as R (if any variable to be substituted in the expression is a vector, then a vector is returned). If it is FALSE (default), a yacc expression is returned, replacing scalar variables.

Examples

```
# Evaluate as yacas object
eq <- ysym("2*y+x^2+2*x-3")
y_eval(eq, x=3, y=2)

# Evaluate as R expression:
y_eval(eq, x=3, y=2, as.r=TRUE)
# This allows to use vectors:
y_eval(eq, x=1:10, y=2, as.r=TRUE)
# and to plot functions:
curve(y_eval(eq, x=x, y=2, as.r=TRUE), xlim=c(0,10))
```

y_fn

Prepare simple yacas call

Description

Prepare simple yacas call

Usage

```
y_fn(x, fn, ...)
```

Arguments

x	parameter to function fn
fn	function with parameter x
...	additional arguments to fn

Examples

```

y_fn("x^2 - 1", "TeXForm")
yac_str(y_fn("x^2 - 1", "TeXForm"))

y_fn("x^2 - 1", "Factor")
yac_str(y_fn("x^2 - 1", "Factor"))

cmd <- "x^2 - 1 == 0" %>% y_fn("Solve", "x")
cmd
sol <- yac_str(cmd)
sol
yac_str(y_rmvars(sol))

```

y_print

*Pretty print yacas strings***Description**

Pretty print yacas strings

Usage

```
y_print(x)
```

Arguments

x	yacas string, e.g. a matrix
---	-----------------------------

Examples

```

A <- diag(4)
Ayac <- as_y(A)
y_print(Ayac)

B <- A
B[2, 2] <- "-t"
Byac <- as_y(B)
Byac
y_print(Byac)

```

`y_rmvars`*Remove/strip variable names*

Description

This only builds a yacas command. You need to also call `yac_str()`, `yac_expr()` or similar. This is the reason that it does not call yacas: it depends on how you want it returned (string, expression).

Usage

```
y_rmvars(x)
```

Arguments

<code>x</code>	yacas command
----------------	---------------

Examples

```
cmd <- "{x == 2, y == 4}"
yac_str(cmd)
yac_str(y_rmvars(cmd))
```

`[.yac_symbol`*Extract or replace parts of an object*

Description

Extract or replace parts of an object

Usage

```
## S3 method for class 'yac_symbol'
x[i, j]
```

Arguments

<code>x</code>	A yac_symbol.
<code>i</code>	row indices specifying elements to extract or replace
<code>j</code>	column indices specifying elements to extract or replace

[<-.yac_symbol	<i>Extract or replace parts of an object</i>
----------------	--

Description

Extract or replace parts of an object

Usage

```
## S3 replacement method for class 'yac_symbol'
x[i, j] <- value
```

Arguments

x	A yac_symbol.
i	row indices specifying elements to extract or replace
j	column indices specifying elements to extract or replace
value	the value to replace x[i, j] by

[.yac_symbol	<i>Extract parts of an object</i>
--------------	-----------------------------------

Description

Extract parts of an object

Usage

```
## S3 method for class 'yac_symbol'
x[[i]]
```

Arguments

x	A yac_symbol.
i	indices specifying elements to extract

%%*	<i>Matrix multiplication</i>
-----	------------------------------

Description

Matrix multiplication

Usage

x %*% y

Arguments

- x A yac_symbol
- y A yac_symbol

Index

- * **helper**
 - as_r, 3
 - as_y, 4
 - y_eval, 22
 - y_fn, 22
 - y_print, 23
 - y_rmvars, 24
- * **programming**
 - Ryacas-package, 3
- * **symbolmath**
 - yac_cli, 18
- * **yac_communication**
 - yac, 17
 - yac_assign, 17
 - yac_cli, 18
 - yac_expr, 19
 - yac_silent, 19
 - yac_str, 20
- * **yac_symbol**
 - %%%, 26
 - as.character.yac_symbol, 3
 - cbind.yac_symbol, 4
 - deriv.yac_symbol, 5
 - det, 5
 - diag, 6
 - diag<-, 6
 - Hessian, 6
 - integrate, 7
 - Jacobian, 7
 - lim, 8
 - lower.tri, 8
 - pow, 10
 - prod.yac_symbol, 10
 - rbind.yac_symbol, 11
 - simplify, 11
 - sum.yac_symbol, 13
 - t, 13
 - tex, 14
 - tr, 14
 - upper.tri, 15
 - vec, 15
 - vech, 16
 - with_value, 16
 - y_fn, 22
 - yac, 17
 - yac_assign, 17
 - yac_expr, 19
 - yac_silent, 19
 - yac_str, 20
 - yac_symbol, 20
 - ysym, 21
 - [.yac_symbol, 24
 - [<-.yac_symbol, 25
 - [|.yac_symbol, 25
 - %%%, 26
 - as.character.yac_symbol, 3
 - as_r, 3
 - as_y, 4
 - base::as.vector(), 15
 - base::det(), 5
 - base::diag(), 6
 - base::lower.tri(), 8, 15
 - base::t(), 13
 - base::upper.tri(), 8, 15
 - cbind.yac_symbol, 4
 - deriv.yac_symbol, 5
 - det, 5
 - diag, 6
 - diag<-, 6
 - Hessian, 6
 - integrate, 7
 - Jacobian, 7

`lim`, [8](#)
`lower.tri`, [8](#)
`Math.yac_symbol`, [9](#)
`Ops.yac_symbol`, [9](#)

`pow`, [10](#)
`pow.default()`, [10](#)
`prod.yac_symbol`, [10](#)

`rbind.yac_symbol`, [11](#)
`Ryacas-package`, [3](#)

`simplify`, [11](#)
`solve.yac_symbol`, [12](#)
`stats::integrate()`, [7](#)
`sum.yac_symbol`, [13](#)

`t`, [13](#)
`tex`, [14](#)
`tr`, [14](#)
`tr.default()`, [14](#)

`upper.tri`, [15](#)

`vec`, [15](#)
`vech`, [16](#)
`vech.default()`, [16](#)

`with_value`, [16](#)

`y_eval`, [22](#)
`y_fn`, [22](#)
`y_print`, [23](#)
`y_rmvars`, [24](#)
`yac`, [17](#)
`yac_assign`, [17](#)
`yac_cli`, [18](#)
`yac_expr`, [19](#)
`yac_expr()`, [24](#)
`yac_silent`, [19](#)
`yac_str`, [20](#)
`yac_str()`, [21](#), [24](#)
`yac_symbol`, [7](#), [8](#), [20](#)
`ysym`, [21](#)
`ysym()`, [3](#), [20](#)
`ysym_ls`, [21](#)