Package 'caracas'

May 21, 2020

Version 1.0.1

Title Computer Algebra

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

Encoding UTF-8

Description Computer algebra via the 'SymPy' library (<https://www.sympy.org/>).
This makes it possible to solve equations symbolically,
find symbolic integrals, symbolic sums and other important quantities.

Depends R (>= 3.0)

Imports reticulate (>= 1.14)

Suggests testthat (>= 2.1.0), knitr, rmarkdown

License GPL

SystemRequirements Python (>= 3.6.0)

URL https://github.com/r-cas/caracas

BugReports https://github.com/r-cas/caracas/issues

RoxygenNote 7.1.0

VignetteBuilder knitr

NeedsCompilation no

Author Mikkel Meyer Andersen [aut, cre, cph],
Søren Højsgaard [aut, cph]

Repository CRAN

Date/Publication 2020-05-21 12:50:13 UTC

R topics documented:

as.character.caracas_symbol *Convert symbol to character*

Description

Convert symbol to character

Usage

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

Arguments

x	A caracas_symbol
...	not used

as_character_matrix *Get matrix as character matrix*

Description

Get matrix as character matrix

Usage

```
as_character_matrix(x)
```

Arguments

x	caracas symbol

as_r *Convert caracas object to R*

Description

Potentially calls `doit()`.

Usage

```
as_r(x, first_doit = TRUE)
```

Arguments

`x`	caracas_symbol
`first_doit`	Try `doit()` first

as_symbol *Convert object to symbol*

Description

Variables are detected as a character followed by a number of either: character, number or underscore.

Usage

```
as_symbol(x, declare_variables = TRUE)
```

Arguments

`x`	R object to convert to a symbol
`declare_variables`	declare detected variables automatically

Details

Default is to declare used variables. Alternatively, the user must declare them first, e.g. by `symbol()`.

Note that matrices can be defined by specifying a Python matrix, see below in examples.

Examples

```
if (have_sympy()) {
  x <- symbol("x")
  A <- matrix(c("x", 0, 0, "2*x"), 2, 2)
  A
  B <- as_symbol(A)
  B
  2*B
  dim(B)
  sqrt(B)
  D <- as_symbol("[[1, 4, 5], [-5, 8, 9]]")
  D
}
```

---

der *Symbolic differentiation of an expression*

Description

Symbolic differentiation of an expression

Usage

```
der(expr, vars)
```

Arguments

expr	A caracas_symbol
vars	variables to take derivate with respect to

Examples

```
if (have_sympy()) {
  x <- symbol("x")
  y <- symbol("y")
  f <- 3*x^2 + x*y^2
  der(f, x)
}
```

der2 *Symbolic differentiation of second order of an expression*

Description

Symbolic differentiation of second order of an expression

Usage

```
der2(expr, vars)
```

Arguments

expr	A caracas_symbol
vars	variables to take derivate with respect to

Examples

```
if (have_sympy()) {
  x <- symbol("x")
  y <- symbol("y")
  f <- 3*x^2 + x*y^2
  der2(f, x)
}
```

determinant.caracas_symbol *Calculate the Determinant of a Matrix*

Description

Note that there is no argument for `logarithm` as with the generic method.

Usage

```
## S3 method for class 'caracas_symbol'
determinant(x, ...)
```

Arguments

x	A caracas_symbol
...	Not used

diag *Matrix diagonal*

Description

Matrix diagonal

Usage

```
diag(x, ...)
```

Arguments

x	Object x
...	Passed on

diag.caracas_symbol *Matrix diagonal*

Description

Matrix diagonal

Usage

```
## S3 method for class 'caracas_symbol'
diag(x, ...)
```

Arguments

x	Object x
...	Not used

diag<- *Replace matrix diagonal*

Description

Replace matrix diagonal

Usage

```
diag(x) <- value
```

Arguments

x	Object x
value	Replacement value

diag<-.caracas_symbol *Replace diagonal*

Description

Replace diagonal

Usage

```
## S3 replacement method for class 'caracas_symbol'
diag(x) <- value
```

Arguments

x	A caracas_symbol.
value	Replacement value

Examples

```
if (have_sympy()) {
  A <- matrix(c("a", 0, 0, 0, "a", "a", "a", 0, 0), 3, 3)
  B <- as_symbol(A)
  B
  diag(B)
  diag(B) <- "b"
  B
  diag(B)
}
```

dim.caracas_symbol *Dimensions of a caracas symbol*

Description

Dimensions of a caracas symbol

Usage

```
## S3 method for class 'caracas_symbol'
dim(x)
```

Arguments

x	caracas symbol

doit *Perform calculations setup previously*

Description

Perform calculations setup previously

Usage

```
doit(x)
```

Arguments

x	A caracas_symbol

Examples

```
if (have_sympy()) {
   x <- symbol('x')
   res <- limf(sin(x)/x, "x", 0, doit = FALSE)
   res
   doit(res)
}
```

eigen_val *Eigenvalues*

Description

Eigenvalues

Usage

```
eigen_val(x)
```

Arguments

x	Matrix to find eigenvalues for

Examples

```
if (have_sympy()) {
  A <- matrix(c("a", 0, 0, 0, "a", "a", "a", 0, 0), 3, 3)
  B <- as_symbol(A)
  eigen_val(B)
  eigen_vec(B)
  eigen(eval(as_r(B), list(a = 2)))
}
```

eigen_vec *Eigenvectors and eigenvalues*

Description

Eigenvectors and eigenvalues

Usage

```
eigen_vec(x)
```

Arguments

x	Matrix to find eigenvectors and eigenvalues for

Examples

```
if (have_sympy()) {
  A <- matrix(c("a", 0, 0, 0, "a", "a", "a", 0, 0), 3, 3)
  B <- as_symbol(A)
  eigen_val(B)
  eigen_vec(B)
  eigen(eval(as_r(B), list(a = 2)))
}
```

`eval_to_symbol` *Create a symbol from a string*

Description

Create a symbol from a string

Usage

```
eval_to_symbol(x)
```

Arguments

`x`	String to evaluate

Value

A `caracas_symbol`

Examples

```
if (have_sympy()) {
   x <- symbol('x')
   (1+1)*x^2
   limf(sin(x)/x, "x", 0)
}
```

expand *Expand expression*

Description

Expand expression

Usage

```
expand(x)
```

Arguments

x	A caracas_symbol

expand_log *Expand a logarithmic expression*

Description

Note that force as described at https://docs.sympy.org/latest/tutorial/simplification.html#expand-log is used meaning that some assumptions are taken.

Usage

```
expand_log(x)
```

Arguments

x	A caracas_symbol

Examples

```
if (have_sympy()) {
  x <- symbol('x')
  y <- symbol('y')
  z <- log(x*y)
  z
  expand_log(z)
}
```

expand_trig *Expand a trigonometric expression*

Description

Expand a trigonometric expression

Usage

```
expand_trig(x)
```

Arguments

x	A caracas_symbol

get_sympy *Access 'SymPy' directly*

Description

Get the 'SymPy' object. Note that it gives you extra responsibilities when you choose to access the 'SymPy' object directly.

Usage

```
get_sympy()
```

Value

The 'SymPy' object with direct access to the library.

Examples

```
if (have_sympy()) {
  sympy <- get_sympy()
  sympy$solve("x**2-1", "x")
}
```

have_sympy *Check if 'SymPy' is available*

Description

Check if 'SymPy' is available

Usage

```
have_sympy()
```

Value

TRUE if 'SymPy' is available, else FALSE

Examples

```
have_sympy()
```

install_sympy *Install 'SymPy'*

Description

Install the 'SymPy' Python package into a virtual environment or Conda environment.

Usage

```
install_sympy(method = "auto", conda = "auto")
```

Arguments

method	Installation method. By default, "auto" automatically finds a method that will work in the local environment. Change the default to force a specific installation method. Note that the "virtualenv" method is not available on Windows.
conda	Path to conda executable (or "auto" to find conda using the PATH and other conventional install locations).

Value

None

intf *Integrate a function*

Description

If no limits are provided, the indefinite integral is calculated. Otherwise, if both limits are provided, the definite integral is calculated.

Usage

```
intf(f, var, lower, upper, doit = TRUE)
```

Arguments

f	Function to integrate
var	Variable to integrate with respect to (either string or caracas_symbol)
lower	Lower limit
upper	Upper limit
doit	Evaluate the integral immediately (or later with doit())

Examples

```
if (have_sympy()) {
  x <- symbol("x")

  intf(1/x, x, 1, 10)
  intf(1/x, x, 1, 10, doit = FALSE)
  intf(1/x, x)
  intf(1/x, x, doit = FALSE)
}
```

inv *Find inverse of matrix*

Description

Find inverse of matrix

Usage

```
inv(A)
```

Arguments

A	matrix

limf *Limit of a function*

Description

Limit of a function

Usage

```
limf(f, var, val, dir = NULL, doit = TRUE)
```

Arguments

f	Function to take limit of
var	Variable to take limit for (either string or caracas_symbol)
val	Value for var to approach
dir	Direction from where var should approach val: '+' or '-'
doit	Evaluate the limit immediately (or later with doit())

Examples

```
if (have_sympy()) {
  x <- symbol("x")
  limf(sin(x)/x, "x", 0)
  limf(1/x, "x", 0, dir = '+')
  limf(1/x, "x", 0, dir = '-')
}
```

Math.caracas_symbol *Math functions*

Description

If x is a matrix, the function is applied component-wise.

Usage

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

Arguments

x	caracas_symbol.
...	further arguments passed to methods

Ops.caracas_symbol *Math operators*

Description

Math operators

Usage

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

Arguments

e1	A caracas_symbol.
e2	A caracas_symbol.

print.caracas_solve_sys_sol *Print solution*

Description

Print solution

Usage

```
## S3 method for class 'caracas_solve_sys_sol'
print(
  x,
  simplify = getOption("caracas.print.sol.simplify", default = TRUE),
  ...
)
```

Arguments

x	A caracas_symbol
simplify	Print solution in a simple format
...	Passed to print.caracas_symbol()

print.caracas_symbol *Print symbol*

Description

Print symbol

Usage

```
## S3 method for class 'caracas_symbol'
print(
  x,
  caracas_prefix = TRUE,
  prettyascii = getOption("caracas.print.prettyascii", default = FALSE),
  ascii = getOption("caracas.print.ascii", default = FALSE),
  rowvec = getOption("caracas.print.rowvec", default = TRUE),
  ...
)
```

Arguments

x	A caracas_symbol
caracas_prefix	Print 'caracas' prefix
prettyascii	TRUE to print in pretty ASCII format rather than in utf8
ascii	TRUE to print in ASCII format rather than in utf8
rowvec	FALSE to print column vectors as is
...	not used

prodf *Product of a function*

Description

Product of a function

Usage

```
prodf(f, var, lower, upper, doit = TRUE)
```

Arguments

f	Function to take product of
var	Variable to take product for (either string or caracas_symbol)
lower	Lower limit
upper	Upper limit
doit	Evaluate the product immediately (or later with doit())

Examples

```
if (have_sympy()) {
  x <- symbol("x")
  p <- prodf(1/x, "x", 1, 10)
  p
  as_r(p)
  prod(1/(1:10))
  n <- symbol("n")
  prodf(x, x, 1, n)
}
```

`simplify` *Simplify expression*

Description

Simplify expression

Usage

```
simplify(x)
```

Arguments

x	A `caracas_symbol`

`solve_lin` *Solve a linear system of equations*

Description

Find `x` in `Ax = b`. If `b` not supplied, the inverse of `A` is returned.

Usage

```
solve_lin(A, b)
```

Arguments

A	matrix
b	vector

solve_sys *Solves a system of non-linear equations*

Description

If called as `solve_sys(lhs,vars)` the roots are found. If called as `solve_sys(lhs,rhs,vars)` the solutions to `lhs = rhs` for `vars` are found.

Usage

```
solve_sys(lhs, rhs, vars)
```

Arguments

`lhs`	Equation (or equations as row vector/1xn matrix)
`rhs`	Equation (or equations as row vector/1xn matrix)
`vars`	vector of variable names or symbols

Value

A list with solutions (with class `caracas_solve_sys_sol` for compact printing), each element containing a named list of the variables' values.

subs *Substitute symbol for value*

Description

Substitute symbol for value

Usage

```
subs(s, x, v)
```

Arguments

`s`	Expression
`x`	Name of symbol (character)
`v`	Value for `x`

Examples

```
if (have_sympy()) {
   x <- symbol('x')
   e <- 2*x^2
   e
   subs(e, "x", "2")
   y <- as_symbol("2")
   subs(e, "x", y)
}
```

---

subs_lst *Substitute symbol for of value given by a list*

Description

Useful for substituting solutions into expressions.

Usage

```
subs_lst(s, x)
```

Arguments

s	Expression
x	Named list of values

Examples

```
if (have_sympy()) {
     p <- as_symbol(paste0("p", 1:3))
     y <- as_symbol(paste0("y", 1:3))
     a <- as_symbol("a")
     l <- sum(y*log(p))
     L <- -l + a*(sum(p) - 1)
     g <- der(L, c(a, p))
     sols <- solve_sys(g, c(a, p))
     sol <- sols[[1L]]
     sol
     H <- der2(L, c(p, a))
     H
     H_sol <- subs_lst(H, sol)
     H_sol
}
```

sum.caracas_symbol *Summation*

Description

Summation

Usage

```
## S3 method for class 'caracas_symbol'
sum(..., na.rm = FALSE)
```

Arguments

...	Elements to sum
na.rm	Not used

sumf *Sum of a function*

Description

Sum of a function

Usage

```
sumf(f, var, lower, upper, doit = TRUE)
```

Arguments

f	Function to take sum of
var	Variable to take sum for (either string or caracas_symbol)
lower	Lower limit
upper	Upper limit
doit	Evaluate the sum immediately (or later with doit())

Examples

```
if (have_sympy()) {
  x <- symbol("x")
  s <- sumf(1/x, "x", 1, 10)
  as_r(s)
  sum(1/(1:10))
  n <- symbol("n")
  simplify(sumf(x, x, 1, n))
}
```

symbol *Create a symbol*

Description

Create a symbol

Usage

```
symbol(x)
```

Arguments

x	Name to turn into symbol

Value

A caracas_symbol

sympy_version *Get 'SymPy' version*

Description

Get 'SymPy' version

Usage

```
sympy_version()
```

Value

The version of the 'SymPy' available

Examples

```
if (have_sympy()) {
  sympy_version()
}
```

t.caracas_symbol *Transpose of matrix*

Description

Transpose of matrix

Usage

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

Arguments

x	If caracas_symbol treat as such, else call base::t().

tex *Export object to TeX*

Description

Export object to TeX

Usage

```
tex(x)
```

Arguments

x	A caracas_symbol

[.caracas_symbol *Extract or replace parts of an object*

Description

Extract or replace parts of an object

Usage

```
## S3 method for class 'caracas_symbol'
x[i, j, ..., drop = TRUE]
```

Arguments

x	A caracas_symbol.
i	row indices specifying elements to extract or replace
j	column indices specifying elements to extract or replace
...	Not used
drop	Simplify dimensions of resulting object

Examples

```
if (have_sympy()) {
  A <- matrix(c("a", 0, 0, 0, "a", "a", "a", 0, 0), 3, 3)
  B <- as_symbol(A)
  B[1:2, ]
  B[, 2]
  B[2, , drop = FALSE]
}
```

[<-.caracas_symbol *Extract or replace parts of an object*

Description

Extract or replace parts of an object

Usage

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

Arguments

x	A caracas_symbol.
i	row indices specifying elements to extract or replace
j	column indices specifying elements to extract or replace
...	Not used
value	Replacement value

Examples

```
if (have_sympy()) {
  A <- matrix(c("a", 0, 0, 0, "a", "a", "a", 0, 0), 3, 3)
  B <- as_symbol(A)
  B[, 2] <- "x"
  B
}
```

%*%.caracas_symbol *Matrix multiplication*

Description

Matrix multiplication

Usage

```
## S3 method for class 'caracas_symbol'
x %*% y
```

Arguments

x	Object x
y	Object y

%*% *Matrix multiplication*

Description

Matrix multiplication

Usage

```
x %*% y
```

Arguments

x	Object x
y	Object y

Index