Package: rim (via r-universe)

November 26, 2026

Type Package

Title Interface to 'Maxima', Enabling Symbolic Computation

Version 0.9.0

Date 2026-10-28

Description An interface to the powerful and fairly complete computer algebra system 'Maxima'. It can be used to start and control 'Maxima' from within R by entering 'Maxima' commands. Results from 'Maxima' can be parsed and evaluated in R. It facilitates outputting results from 'Maxima' in 'LaTeX' and 'MathML'. 2D and 3D plots can be displayed directly. This package also registers a 'knitr'-engine enabling 'Maxima' code chunks to be written in 'RMarkdown' documents.

URL https://rcst.github.io/rim/

BugReports https://github.com/rcst/rim/issues

SystemRequirements Maxima (https://sourceforge.net/projects/maxima/, tested with versions 5.42.0 - 5.46.0), needs to be on PATH

License GPL (>= 3)

Imports methods, Rcpp, R6, knitr, GlobalOptions

LinkingTo Rcpp

Depends R (>= 3.5.0)

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0), rmarkdown

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation yes

Author Eric Stemmler [aut, cre], Kseniia Shumelchyk [aut], Hans W. Borchers [aut]

Maintainer Eric Stemmler <stemmler.eric@gmail.com>

Repository https://cranhaven.r-universe.dev

Date/Publication 2026-11-26 02:02:00 UTC

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/rim

RemoteSha d9389a91260fc78bd6cb0f68c0d7a690ca5a4618

RemoteSubdir rim

Contents

rim-package 2
maxima.engine 5
maxima.options 6
maxima.repl 7
rim_global 8

Index **9**

`rim-package` *rim*

Description

Provides an interface to Maxima, a computer algebra system.

Usage

```
maxima.start(restart = FALSE)

maxima.stop(engine = FALSE)

maxima.get(command)

maxima.load(lib, quote = FALSE, stuck = TRUE, suppress = TRUE)

maxima.apropos(keystring)

maxima.version()

maxima.isInstalled()

iprint(x)

## S3 method for class 'maxima'
print(x, ...)

maxima.eval(x, code = FALSE, envir = globalenv())
```

Arguments

restart	if FALSE (default), then Maxima is started provided it is not running already. If TRUE starts or restarts Maxima.
engine	if FALSE (default), quits the (running) maxima instance and closes the connection, otherwise quits and closes the (running) maxima instance used for the knitr engine.
command	character string containing the Maxima command.
lib	character vector naming the Maxima module (typically a *.mac or *.lisp file) to be loaded.
quote	logical; whether lib should be quoted if it isn't already (default FALSE).
stuck	logical; whether the load command should avoid advancing Maxima's input/output labels (default TRUE).
suppress	logical; whether the load command should be run as a suppressed (default TRUE), suppressing any output.
keystring	character vector containing a search term.
x	Either a character vector of length 1L or an S3 object of class "maxima"
...	other arguments (ignored).
code	A logical vector of length 1L, whether to attach the original expression (TRUE) or not (FALSE, default)
envir	A environment object. globalenv() (default), is passed to eval().

Details

Note: You need to install the Maxima software separately in order to make use of this package.

Maxima is set up automatically on attachment via library(rim) and automatically started when a command is send (if it isn't running already) using maxima.get(). If environment variable RIM_MAXIMA_PATH is not set, rim will search for the Maxima executable, or use the former otherwise. Using maxima.start() and maxima.stop(), one can stop and (re-)start the current Maxima session if needed, e.g. to clear Maxima command and output history.

To send a single command to Maxima and receive the corresponding output use maxima.get(). This function returns a S3 object of class "maxima". The output is printed by printing the object and will be printed in a format currently set by maxima.options(format). The output format can be changed by setting it, e.g. maxima.options(format = "ascii"). Output labels are printed according to option maxima.options(label).

Value

invisibly returns NULL.

Character vector of length 1 of the input command. Depending on whether option "label" is set to TRUE, the corresponding input reference label is printed preceding the input command.

The evaluated R-object

Functions

- `maxima.start()`: (re-)starts Maxima.
- `maxima.stop()`: Quits Maxima.
- `maxima.get()`: Executes a single Maxima command provided by `command`. If no command ending character (`;` or `$` is provided, `;` is appended.
- `maxima.load()`: A convenience wrapper to load a Maxima library named by `lib`
- `maxima.apropos()`: A wrapper to the Maxima helper function `apropos` to lookup existing Maxima functions that match `keystring`.
- `maxima.version()`: Returns the version number of Maxima that is used
- `maxima.isInstalled()`: Returns TRUE when an installation of Maxima has been detected, otherwise FALSE
- `iprint()`: Prints the input command of an maxima S3-object returned by `maxima.get()`
- `print(maxima)`: Prints the maxima output part of an S3 object returned by `maxima.get()`
- `maxima.eval()`: Evaluates the parsed and quoted R-expression which is part of an S3 object returned by `maxima.get()`

Author(s)

Maintainer: Eric Stemmler <stemmler.eric@gmail.com>

Authors:

- Kseniia Shumelchyk <shumelchyk@gmail.com>
- Hans W. Borchers <hwborchers@googlemail.com>

See Also

Useful links:

- https://rcst.github.io/rim/
- Report bugs at https://github.com/rcst/rim/issues

`maxima.engine`, `maxima.options`

Examples

```
if(maxima.isInstalled()) maxima.start(restart = TRUE)
if(maxima.isInstalled()) {
  maxima.start(restart = TRUE)
  maxima.stop()
}
if(maxima.isInstalled()) maxima.get("2+2;")
if(maxima.isInstalled()) maxima.load("ratpow", quote = FALSE, stuck = FALSE, suppress = FALSE)
if(maxima.isInstalled()) maxima.apropos("integrate")
maxima.version()
maxima.isInstalled()
if(maxima.isInstalled()) {
  a <- maxima.get("2+2;")
```

```
  iprint(a)
}
if(maxima.isInstalled()) {
  a <- maxima.get("2+2;")
  print(a)
}
if(maxima.isInstalled()) {
  a <- maxima.get("2+2;")
  maxima.eval(a)
  # same
  maxima.eval("2+2;")
  # evaluate with data.frame
  df <- data.frame(x = seq(0, 1, by = 0.1))
  maxima.eval("integrate(1 / (1 + x^4), x);", code = TRUE, envir = df)
  maxima.stop()
}
```

---

maxima.engine knitr *maxima engine*

Description

Functions to process Maxima code chunks by knitr.

Usage

```
maxima.engine(options)

maxima.inline(command)
```

Arguments

options	named list of knitr options. Supported options are echo, eval, include and output.var. To change the output format of the Maxima engine set the option maxima.options(engine.format) to either "linear" (default), "ascii", "latex" or "mathml".
command	character string containing the Maxima command to be executed.

Details

Upon attachment, i.e. library(rim) function maxima.engine is registered as a knitr engine. Thus, maxima.engine() is called by knit() to evaluate Maxima code chunks. When called upon the first code chunk of a document it starts Maxima in a separate process in server mode. This means that a single Maxima session is used for all Maxima code chunks of an RMarkdown document. Inputs and outputs can thus be used across chunks (e.g. by using Maxima reference labels). maxima.options(engine.format = ..., engine.label = ...) configures the output format and whether or not output reference labels should be printed.

The purpose of `maxima.inline` is to insert Maxima results as inline text, i.e. on the same line of the preceding text, if it is actually written on the same line of the `RMarkdown` file. It uses the same running Maxima process as `maxima.engine`. The output format for inline results can be configured separately from the settings of `maxima.engine`, i.e. `maxima.options(inline.format = ..., inline.label = ...)`.

Value

This functions prints the resulting output from maxima together with it's cod

character string containing the maxima result printed according options set by `maxima.options(inline.format = ..., inline.label = ...)`.

Functions

- `maxima.inline()`: This function can be used to insert maxima outputs as inline.

Examples

```
if (maxima.isInstalled()) {
  maxima.inline("2+2;")
  maxima.stop(engine = TRUE)
}
```

`maxima.options` *maxima.options*

Description

Function for globally setting and retrieving options.

Usage

```
maxima.options(
  ...,
  RESET = FALSE,
  READ.ONLY = NULL,
  LOCAL = FALSE,
  ADD = FALSE
)
```

Arguments

`...`	options to be accessed by using `name = value`. Options can be added by setting `ADD = TRUE`, but only those mentioned below will be used my rim.
`RESET`	logical of length 1, whether to reset all options to default values.
`READ.ONLY`	logical of length 1, can be used to output all options that are read-only.
`LOCAL`	logical of length 1, to output all options that are defined locally.
`ADD`	logical of length 1, whether to add the specified option in ... (TRUE), default is FALSE.

Details

`format`: character vector of length 1 setting the output format for `maxima.get()` and `maxima.repl()`. Can be one of `"linear"`, `"ascii"`, `"latex"` or `"mathml"`.

`engine.format`: same as option `format`, but for outputs in `RMarkdown` documents.

`inline.format`: character string setting the output format for `maxima.inline()`, for knitting outputs inline into `RMarkdown` documents. Can be one of `"linear"`, `"latex"` or `"mathml"`, but *not* `"ascii"`.

`label`: logical of length 1, whether reference labels should be printed for returned S3 objects from `maxima.get()` and `maxima.repl()` (TRUE, default), or not (FALSE). This also applies to printing of input commands using `iprint()`.

`engine.label`: same as `label`, but for outputs in `RMarkdown` documents.

`inline.label`: same as `label`, but for inline outputs in `RMarkdown` documents.

Examples

```
maxima.options(format = "latex")
maxima.options(label = FALSE)
maxima.options(label = TRUE, format = "ascii")
# reset to default
maxima.options(label = TRUE, format = "linear")
```

---

maxima.repl *Run a Maxima REPL*

Description

This function provides a Maxima REPL in the R session, which can be used to interactively run Maxima code. All code executed within the REPL is run within the interactive Maxima session that is started when rim is attached and any generated Maxima objects will persist in the Maxima session after the REPL is detached.

Usage

```
maxima.repl(history_filename = NA_character_)
```

Arguments

`history_filename`
: A (optional) filename to which all Maxima commands during the repl session are saved. If not provided (default), command history lookup will still be available (if supported by OS) in the same way as in the R session.

Details

When working with R and Maxima scripts interactively, one can activate the Python REPL with ‘maxima.repl()‘, run Maxima code, and later run ‘exit;‘ to return to the R console.

Note that, inside the REPL, multiple commands are allowed.

The output format displayed inside the REPL is controlled by ‘maxima.options(repl.format = ...)‘.

`rim_global` *rim_global*

Description

Returns knitr::knit_global() unless this returs the GlobalEnv() in which case it returns maxima.env, rim's internal environment. Main purpose is to prevent ‘knitr::knit_global()‘ from returning ‘globalenv()‘.

Usage

```
rim_global()
```

Value

An environment, either the result of knitr::knit_global or maxima.env

Index

iprint, *7*
iprint *(*rim-package*)*, 2

maxima.apropos *(*rim-package*)*, 2
maxima.engine, *4*, 5
maxima.eval *(*rim-package*)*, 2
maxima.get, *3, 4*, *7*
maxima.get *(*rim-package*)*, 2
maxima.get(), *7*
maxima.inline *(*maxima.engine*)*, 5
maxima.inline(), *7*
maxima.isInstalled *(*rim-package*)*, 2
maxima.load *(*rim-package*)*, 2
maxima.options, *3, 4*, 6
maxima.repl, 7
maxima.repl(), *7*
maxima.start, *3*
maxima.start *(*rim-package*)*, 2
maxima.stop, *3*
maxima.stop *(*rim-package*)*, 2
maxima.version *(*rim-package*)*, 2

print.maxima *(*rim-package*)*, 2

rim *(*rim-package*)*, 2
rim-package, 2
rim_global, 8