Package ‘RcppOctave’

December 17, 2012

Type Package

Title Seamless Interface to Octave -- and Matlab

Version 0.8.12

Date 2011-06-07

Author Renaud Gaujoux

Maintainer Renaud Gaujoux `<renaud@mancala.cbio.uct.ac.za>`

Description Direct interface to Octave. The primary goal is to facilitate the port of Matlab/Octave scripts to R. The package enables to call any Octave functions from R and as well as browsing their documentation, passing variables between R and Octave, using R core RNGs in Octave, which ensure stochastic computations are also reproducible.

License GPL (>=2)

URL `http://r-forge.r-project.org/projects/rcppoctave/`

LazyLoad yes

SystemRequirements Octave (>= 3.2.4) and its development files

OS_type unix

Depends methods,stats,Rcpp (>= 0.10.1),pkgmaker (>= 0.10)

Imports digest,stringr,utils

Suggests RUnit,knitr,bibtex

LinkingTo Rcpp

Collate ’utils.R’’package.R’’interface.R’’Octave-class.R’’OctaveFunction-class.R’’base-functions.R’’eval.R’’random.R’

R topics documented:

`.CallOctave` *Calling an Octave Function*

Description

`.CallOctave` calls an Octave function and returns its value.

Usage

```
.CallOctave(.NAME, ..., argout = -1,
  unlist = !is.character(argout))
```

Arguments

`.NAME`	an Octave function name. The function must be a valid function name in the current Octave session.
`...`	arguments passed to the Octave function
`argout`	the number of output values, or a vector of names to use as output variable names. The names are directly used and applied to the result list in their original order. The default value `argout=-1` returns: • all output values when their number can be determined. This would typically be the case for functions defined in .m files. Please do read section *Details* for considerations about the functions that use varargout. • only the first one otherwise.
`unlist`	a logical that specifies if an output list of length one should be simplified and returned as a single value or kept as a list. The default is to unlist unless output names were passed in `argout`.

Value

the value returned by the Octave function – converted into standard R objects.

Examples

```
# data matrix
x <- matrix(1:9, 3)

# call Octave function 'svd': equivalent to [S] = svd(x). See o_help(svd)
.CallOctave('svd', x)

# call Octave function 'svd' asking for 3 output values: [U, S, V] = svd(x)
.CallOctave('svd', x, argout=3)

# call Octave function 'svd' asking for 3 named output values: [U, S, V] = svd(x)
.CallOctave('svd', x, argout=c('U', 'S', 'V'))
```

`.DollarNames,Octave-method`

Auto-completion for `Octave` objects

Description

Auto-completion for `Octave` objects

Usage

```
  ## S4 method for signature 'Octave'
.DollarNames(x, pattern = "")
```

Arguments

x	An R object for which valid names after "$" are computed and returned.
pattern	A regular expression. Only matching names are returned.

`.O` *Direct Interface to Octave*

Description

RcppOctave provides a simple interface to Octave via the object `.O`, an instance of class `Octave`, that allows for direct access to Octave functions and variables using calls such as: `.O$svd(matrix(1:9,3))`.

Usage

```
  .O

  ## S4 method for signature 'Octave'
x$name

  ## S4 replacement method for signature 'Octave'
x$name <- value
```

Arguments

x	object from which to extract element(s) or in which to replace element(s).
name	A literal character string or a name (possibly backtick quoted). For extraction, this is normally (see under 'Environments') partially matched to the `names` of the object.
value	typically an array-like R object of a similar class as `x`.

Format

`.O` is an object of class `Octave`.

Methods

$ `signature(x = "Octave")`: The method `$` provides a direct way of calling Octave functions or retrieving variables from Octave base context, via e.g. `.O$svd(x)` or `.O$a`. It is equivalent to `o_get(name)`.

$<- `signature(x = "Octave")`: The method `$<-` allow to directly assign/set Octave variables via e.g. `.O$a <- 10`.

See Also

`o_get`

Examples

```
.O
# assign/get Octave variables
.O$a <- 10
.O$a

# call Octave functions
.O$help()
.O$svd(matrix(runif(9), 3))
```

`check.equal` *Compare Lists or Environments*

Description

This function compares two lists or environments. It is useful for comparing results obtained in R and Octave.

Usage

```
check.equal(x, y, msg)
```

Arguments

`x`	a `list` or an `environment`
`y`	a `list` or an `environment`
`msg`	a character string used (if not missing) in a message that is printed before the comparison. It is useful for separating multiple sequential comparisons.

Value

No value is returned, but prints out:

- the element/variable names of each input list or environment,
- the result of the comparison of the elements in `x` and the corresponding element in `y` – if present.

Examples

```
X <- matrix(1:64, 8)
ref <- svd(X)
res <- .O$svd(X, argout=3)

check.equal(ref, res, "R and Octave function 'svd'")
```

mfiles *M Files*

Description

mfiles converts source code or .m filenames into real paths to .m files that can be sourced with o_source.

Usage

```
mfiles(..., pattern = "mfile_", dir = tempdir())
```

Arguments

...	specification of a .m files as character arguments. The elements of the vector can be either file paths or plain Octave/Matlab code, which are then written to disk in – temporary – .m files. Note that the paths do not need to correspond to existing files.
dir	existing directory where to write the .m files generated from the plain code elements of *x*.
pattern	a non-empty character vector giving the initial part of the name.

Examples

```
cat('', file='test.m')
f <- mfiles('test.m')
f <- mfiles('test.m', '')
# remove all files
unlink(f)
```

OctaveFunction-class *Wrapping and Defining Octave Functions from R*

Description

Wrapping and Defining Octave Functions from R

The function OctaveFunction is a constructor/factory method for OctaveFunction objects, which wrap calls to Octave functions into plain R functions.

Usage

```
  OctaveFunction(name)

  ## S4 method for signature 'OctaveFunction'
show(object)
```

Arguments

`name`	the name or definition of an Octave function.
`object`	Any R object

Slots

name name of the wrapped Octave function

Examples

```
osvd <- OctaveFunction('svd')
osvd
osvd(matrix(1:9,3))

orand <- OctaveFunction('rand')
orand()
orand(2)
orand(2, 3)

# From source code
myfun <- OctaveFunction('function [Y] = somefun(x)
Y = x * x;
end
')
myfun
myfun(10)
```

`ostart` *Low-level Function Interfacing with Octave*

Description

`ostart` Initialize an Octave session.

`oend` clears and terminates the current Octave session.

`overbose` toggles the verbosity of RcppOctave calls: messages tracks any function call, or conversion of objects between R and Octave (e.g. arguments and results).

`oconfig` retrieves Octave configuration variables.

`omodules` add the Octave modules shipped with RcppOctave to Octave path.

Usage

```
ostart(verbose = FALSE)

oend()

overbose(value)
```

```
oconfig(varname, verbose = FALSE, warn = TRUE)

omodules(verbose = getOption("verbose"))
```

Arguments

`verbose`	logical value used as the inital verbosity status.
`value`	logical value to toggle verbosity
`varname`	Name (as a character string) of the Octave configuration variable to retrieve. It is used in following system call '`octave-config -p <varname>`'. This function is vectorised and returns a character vector of the same length as its argument.
`warn`	logical that indicates if a warning should be thrown when a variable is returned empty, which generally means that `x` is not a valid config variable name.

See Also

OctaveConfig

`o_addpath` *Manipulating Octave Search Path*

Description

Adds a directory at the beginning of Octave search path.

`o_inpath` tells if a directory or files are in Octave path.

Usage

```
o_addpath(DIR1, ..., OPTION = "-begin")

o_inpath(...)
```

Arguments

`DIR1`	path specification to add to Octave search path. See section *Octave Documentation*.
`...`	other path specifications
`OPTION`	option that specifies how the path should be added. Possible values are: `'-begin', 0, '-end', 1`. See section *Octave Documentation*.

Details

The .oct files present in directories from the search path are looked up when an object or function is requested but not loaded in the current session. The files are watched and automatically reloaded in case modification.

Value

returns invisibly the old value of search path.

Octave Documentation for *addpath*

```
 -- Built-in Function:  addpath (DIR1, ...)
 -- Built-in Function:  addpath (DIR1, ..., OPTION)
     Add DIR1, ... to the current function search path.  If OPTION is
     "-begin" or 0 (the default), prepend the directory name to the
     current path.  If OPTION is "-end" or 1, append the directory name
     to the current path.  Directories added to the path must exist.

     In addition to accepting individual directory arguments, lists of
     directory names separated by ‘pathsep’ are also accepted.  For
     example:

          addpath ("dir1:/dir2:~/dir3");

     See also: path, rmpath, genpath, pathdef, savepath, pathsep
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

See Also

Other Octave_files: o_source

Examples

```
# call an undefined function
try(.CallOctave('fun1'))

# add to the path a directory with a .oct file that contains a definition for 'fun1'
o_addpath(system.file('scripts', package='RcppOctave'))

# re-call the function
#.CallOctave('fun1')

# change the .oct file
o_addpath(tempdir())
o_inpath(tempdir())
o_inpath(tempfile())
```

---

o_assign *Assign/Get Octave Variables*

Description

`o_assign` assigns a variable in Octave. `o_assignin` is an alias for `o_assign`.

`o_get` fetches Octave variables/functions and possibly rename them on the fly with the provided argument names when present. Functions are returned as objects of class `OctaveFunction`, that can be called subsequently (see the examples).

Usage

```
o_assign(...)

o_assignin(...)

o_get(..., unlist = TRUE, rm.ans = TRUE, pattern)
```

Arguments

`...`	variables to assign in Octave global context for `o_assign` , or object names to retrieve from Octave for `o_get`.
`unlist`	a logical that specifies it single variables should be returned as a single value (default), or as a list.
`rm.ans`	a logical that indicates if the automatic Octave variable `ans` should be included in the result. Default is not to include it unless otherwise explicitly specified with this argument, or if it is part of the requested variables in `...`. When present, argument `rm.ans` is always honoured.
`pattern`	regular expression used to filter the requested variable names. Only names matching the pattern are returned.

Details

`o_assign` assigns the variables using the arguments' names if present. Variables can also be specified as a single named list or environment. Single variable assignments can also be specified as `o_assign('a', 10)`. See *Examples* for more details.

Value

`o_assign` returns invisibly the names of the assigned variables.

`o_get` returns a list of the retrieved variable/object. If `unlist=TRUE` and a single – not re-named – variable/object is requested then only its value is returned.

Octave Documentation for *assignin*

```
 -- Built-in Function:  assignin (CONTEXT, VARNAME, VALUE)
     Assign VALUE to VARNAME in context CONTEXT, which may be either
     '"base"' or '"caller"'.

     See also: evalin
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

Note

The function `o_get` is the equivalent of R `get` function and is not related to the Octave function `get` which returns graphics' properties.

Examples

```
## directly assign variables
o_assign(a=1, b=2, c=matrix(1:9, 3))
# retrieve their values
o_get()


## assign a variable for each element in a list
x <- list(a=10, b=20, c=matrix(101:109, 3))
o_assign(x)
o_get()


## assign the content of an environment
e <- list2env(setNames(x, paste('env', names(x), sep='_')))
o_assign(e)
o_get(pattern="^env_")


# get all currently defined variables
o_get()

# by default, the automatic variable `ans` is not returned but might be there
# from unstored previous computation
o_eval('svd(rand(3,3))')
o_get()
o_get(rm.ans=FALSE)

# load some variables
x <- list(b=1, c=3, d=matrix(1:9, 3))
o_assign(x)

# re-fetch all variables
o_get()


# only fetch specific variables
o_get('b')
o_get('b', 'c')
# one can rename variables on the fly
o_get(a='b', 'c')
o_get(c(othername='b'))
o_get(c(othername='b', 'c'))

# fetching using a regular expression
o_assign( setNames(1:3, paste("test", 1:3, sep='_')))
o_get()
o_get(pattern="^test")

# works with functions
f <- o_get('svd')
f
f(matrix(1:9,3))
f(matrix(1:9,3), argout=3)

# an error is thrown in the case of multiple matches (the alternatives are shown)
o_clear()
o_assign(aaa=1, ab=2)
try(o_get('a'))
```

`o_clear` *Deleting Octave Variables*

Description

Deletes variables from Octave global context.

The function `o_rm` is an alias to `o_clear`.

Usage

```
o_clear(..., all = FALSE, options)

o_rm(..., all = FALSE, options)
```

Arguments

`...`	names or pattern of the variables to delete, as character strings.
`all`	a logical indicating whether all user-defined objects should be deleted. See section *Octave Documentation* for details.
`options`	options passed to Octave function `clear`. See section *Octave Documentation*.

Value

None

Octave Documentation for *clear*

```
-- Command:  clear [options] pattern ...
    Delete the names matching the given patterns from the symbol
    table.  The pattern may contain the following special characters:

   '?'
        Match any single character.

   '*'
        Match zero or more characters.

   '[ LIST ]'
        Match the list of characters specified by LIST.  If the first
        character is '!' or '^', match all characters except those
        specified by LIST.  For example, the pattern '[a-zA-Z]' will
        match all lowercase and uppercase alphabetic characters.

    For example, the command

         clear foo b*r

    clears the name 'foo' and all names that begin with the letter 'b'
    and end with the letter 'r'.

    If 'clear' is called without any arguments, all user-defined
    variables (local and global) are cleared from the symbol table.  If
    'clear' is called with at least one argument, only the visible
    names matching the arguments are cleared.  For example, suppose
    you have defined a function 'foo', and then hidden it by
    performing the assignment 'foo = 2'.  Executing the command 'clear
    foo' once will clear the variable definition and restore the
    definition of 'foo' as a function.  Executing 'clear foo' a second
    time will clear the function definition.

    The following options are available in both long and short form
   '-all, -a'
        Clears all local and global user-defined variables and all
        functions from the symbol table.

   '-exclusive, -x'
        Clears the variables that don't match the following pattern.

   '-functions, -f'
        Clears the function names and the built-in symbols names.

   '-global, -g'
        Clears the global symbol names.
```

```
‘-variables, -v’
      Clears the local variable names.

‘-classes, -c’
      Clears the class structure table and clears all objects.

‘-regexp, -r’
      The arguments are treated as regular expressions as any
      variables that match will be cleared.
 With the exception of ‘exclusive’, all long options can be used
 without the dash as well.
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

Examples

```
# Assign a variable in Octave
o_assign('a', 10)
o_who()

# Clear
o_clear()
o_who()


# Assign other variables in Octave
.O$a <- 10
.O$b <- 100
.O$ba <- 1000
o_who()
o_get()


# Clear variable starting with 'b'
o_clear('b*')
o_who()
```

`o_eval` *Evaluate an Octave Expression*

Description

Evaluates an Octave expression in the current embedded Octave session. The variables assigned in the expression are available for subsequent `o_eval` calls.

Usage

```
o_eval(..., CATCH, unlist = TRUE)
```

Arguments

`...`	The Octave expression(s) to evaluate, as a character string.
`CATCH`	The Octave expression(s) to evaluate if the evaluation(s) of `...` fails. See section *Octave Documentation* for more details.
`unlist`	a logical that specifies it single variables should be returned as a single value (default), or as a list.

Value

the result of the evaluation

Octave Documentation for *evalin*

```
 -- Built-in Function:  evalin (CONTEXT, TRY)
 -- Built-in Function:  evalin (CONTEXT, TRY, CATCH)
     Like `eval', except that the expressions are evaluated in the
     context CONTEXT, which may be either `"caller"' or `"base"'.

     See also: eval, assignin
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

Examples

```
# assign some variable
o_eval("a=10")

# retrieve its value in a subsequent call
o_eval("a")

o_get('a')

# use its value
o_eval("b = a^2")


# multiple expression can be evaluated
o_eval(a="10^3", singular="svd(rand(4,4))", random="rand(10, 1)")
# or from a list
l <- list(a="10^3", singular="svd(rand(4,4))", random="rand(10, 1)")
o_eval(l)

# if the evaluation fails then an error is thrown
## Not run:  o_eval("a=svd()")

# except if argument CATCH is provided
o_eval("a=svd()", CATCH="a=2")
```

o_help — *Accessing Octave Help and Documentation Pages*

Description

o_help retrieves the Octave help page associated with a given symbol. By default the page is printed out, but may also be silently retrieved or formatted for direct inclusion in R documentation files (i.e. Rd files).

o_doc displays documentation for the function FUNCTION_NAME directly from an on-line version of the printed manual, using the GNU Info browser. Type ‘q‘ to quit the browser.

Usage

```
o_help(NAME, character.only = FALSE,
  show = interactive(), rd = FALSE)

o_doc(FUNCTION_NAME)
```

Arguments

`NAME`	Octave symbol (e.g. command, function, operator) passed to Octave function `help` to retrieve the related documentation.
`character.only`	a logical indicating whether `NAME` can be assumed to be a character string (`TRUE`) or should be substituted with `substitute` before using them (default).
`show`	logical that specifies if the help page should be shown using the as R documentation file (default), e.g. using a pager, or only returned as a single string. Note that when `show=TRUE`, the string is still returned but invisibly.
`rd`	a logical that specifies if the result should be returned in a suitable way for including in Rd files. If `TRUE`, it wraps the Octave documentation string in Rd code that is rendered as in the Octave console.
`FUNCTION_NAME`	the name of the function from which to show the documentation. See the relevant *Octave Documentation* section below.

Value

this function is usually called for its side effect of printing the help page on standard output (argument `show=TRUE`), but it invisibly returns the help page as a single character string.

Octave Documentation for *help*

```
-- Command:  help NAME
-- Command:  help '--list'
    Display the help text for NAME.   For example, the command 'help
    help' prints a short message describing the 'help' command.

    Given the single argument '--list', list all operators, keywords,
    built-in functions, and loadable functions available in the
    current session of Octave.

    If invoked without any arguments, 'help' display instructions on
    how to access help from the command line.

    The help command can give you information about operators, but not
    the comma and semicolons that are used as command separators.  To
    get help for those, you must type 'help comma' or 'help semicolon'.

    See also: doc, lookfor, which
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

Octave Documentation for *doc*

```
 -- Command:  doc FUNCTION_NAME
     Display documentation for the function FUNCTION_NAME directly from
     an on-line version of the printed manual, using the GNU Info
     browser.  If invoked without any arguments, the manual is shown
     from the beginning.

     For example, the command 'doc rand' starts the GNU Info browser at
     the 'rand' node in the on-line version of the manual.

     Once the GNU Info browser is running, help for using it is
     available using the command 'C-h'.

     See also: help
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

Examples

```
o_help(print)
o_help(rand)
# or equivalently
o_help('rand')

# to include in Rd files, use argument rd=TRUE in an \Sexpr:
## Not run:
 \Sexpr[results=rd,stage=render]{RcppOctave::o_help(rand,rd=TRUE)}

## End(Not run)
```

```
# to see the included Rd code
o_help(rand, rd=TRUE)
o_doc(text)
# or equivalently
o_doc('text')
```

`o_identity` *Octave Identity Function*

Description

This function calls the Octave function provided by the module shipped with RcppOctave. It Returns its arguments unchanged, and is mainly used to test and check the effect of object conversions between R and Octave.

Usage

```
o_identity(...)
```

Arguments

`...`	any R object supported by RcppOctave.

Value

its argument – list – after its conversion from R to Octave and from Octave to R.

Examples

```
o_identity(1L)
o_identity(1:10)
o_identity(matrix(1:10, 2,5))

o_identity(1)
o_identity(runif(10))
o_identity(matrix(runif(10), 2,5))
```

`o_load` *Loading Variables into Octave*

Description

Loads variables from a file, a list or an environment.

Usage

```
o_load(from, ..., options)
```

Arguments

`from`	a list or an environment from where the objects should be loaded
`...`	names of the variables to load
`options`	argument passed to the Octave function `load`. See section *Octave Documentation*.

Octave Documentation for *load*

```
 -- Command:  load file
 -- Command:  load options file
 -- Command:  load options file v1 v2 ...
 -- Command: S = load ("options", "file", "v1", "v2", ...)
 -- Command:  load file options
 -- Command:  load file options v1 v2 ...
 -- Command: S = load ("file", "options", "v1", "v2", ...)
     Load the named variables V1, V2, ..., from the file FILE.  If no
     variables are specified then all variables found in the file will
     be loaded.  As with 'save', the list of variables to extract can
     be full names or use a pattern syntax.  The format of the file is
     automatically detected but may be overridden by supplying the
     appropriate option.

     If load is invoked using the functional form

          load ("-option1", ..., "file", "v1", ...)

     then the OPTIONS, FILE, and variable name arguments (V1, ...) must
     be specified as character strings.

     If a variable that is not marked as global is loaded from a file
     when a global symbol with the same name already exists, it is
     loaded in the global symbol table.  Also, if a variable is marked
     as global in a file and a local symbol exists, the local symbol is
     moved to the global symbol table and given the value from the file.

     If invoked with a single output argument, Octave returns data
     instead of inserting variables in the symbol table.  If the data
     file contains only numbers (TAB- or space-delimited columns), a
     matrix of values is returned.  Otherwise, 'load' returns a
     structure with members corresponding to the names of the variables
     in the file.

     The 'load' command can read data stored in Octave's text and
     binary formats, and MATLAB's binary format.  If compiled with zlib
     support, it can also load gzip-compressed files.  It will
     automatically detect the type of file and do conversion from
     different floating point formats (currently only IEEE big and
     little endian, though other formats may be added in the future).
     Valid options for ‘load’ are listed in the following table.

    ‘-force’
          This option is accepted for backward compatibility but is
          ignored.  Octave now overwrites variables currently in memory
          with those of the same name found in the file.

    ‘-ascii’
          Force Octave to assume the file contains columns of numbers
          in text format without any header or other information.  Data
          in the file will be loaded as a single numeric matrix with
          the name of the variable derived from the name of the file.

    ‘-binary’
          Force Octave to assume the file is in Octave’s binary format.

    ‘-hdf5’
          Force Octave to assume the file is in HDF5 format.  (HDF5 is
          a free, portable binary format developed by the National
          Center for Supercomputing Applications at the University of
          Illinois.)  Note that Octave can read HDF5 files not created
          by itself, but may skip some datasets in formats that it
          cannot support.  This format is only available if Octave was
          built with a link to the HDF5 libraries.

    ‘-import’
          This option is accepted for backward compatibility but is
          ignored.  Octave can now support multi-dimensional HDF data
          and automatically modifies variable names if they are invalid
          Octave identifiers.

    ‘-mat’
    ‘-mat-binary’
    ‘-6’
    ‘-v6’
    ‘-7’
    ‘-v7’
          Force Octave to assume the file is in MATLAB’s version 6 or 7
          binary format.

    ‘-mat4-binary’
    ‘-4’
    ‘-v4’
    ‘-V4’
          Force Octave to assume the file is in the binary format
          written by MATLAB version 4.
    '-text'
          Force Octave to assume the file is in Octave's text format.

     See also: save, dlmwrite, csvwrite, fwrite
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

Examples

```
# Loading from a MATLAB/Octave file
#o_load

# Loading from an R list
o_clear()
l <- list(a=1, b=20, c=runif(10), d="this is a string", e=matrix(1:15, 3, 5))
o_load(l)

# Loading from an R environment
o_load( list2env(l) )

# Partial loading
o_clear()
o_load(l, a, b, c)
o_clear()
o_load(list2env(l), d, e)
```

`o_ls` *Listing Objects from the Current Octave Session*

Description

The function `o_ls` is an enhanced listing function, which also lists user-defined functions, as opposed to `o_who` or `o_whos`, which only list variables. Note that this function works properly on Octave >= 3.6.1, but is known not to list user-defined functions on Octave 3.4.1 (for some unknown reason the Octave function `completion_matches` does not return the names of user-defined functions).

Usage

```
o_ls(long = FALSE, rm.ans = TRUE)
```

Arguments

`rm.ans`	a logical that indicates if the automatic Octave variable `ans` should be included in the result. Default (`TRUE`) is not to include it.
`long`	logical that indicates the desired type of output: if `FALSE` (default) then only the names of the variables are returned (like dispatched `o_who`), otherwise a list with more detailed information about each variable is returned (like `o_whos`.

Value

a character vector or a list depending on the value of argument `long`.

See Also

Other listoct: o_who, o_whos

Examples

```
# only variables
o_assign(list(a=1, b=2, c=5))
o_ls()
# compare with the output of standard Octave functions
o_who() # should be the same output
o_whos()

# variables and user-defined functions
o_clear(all=TRUE) # first clear Octave session
o_source(system.file('scripts/ex_source.m', package='RcppOctave'))
o_ls()
o_ls(long=TRUE)
# compare with the output of standard Octave functions
o_who()
o_whos()
```

---

o_rexp — *Drawing from R Exponential Distribution in Octave*

Description

This function wraps a call to the standard Octave function `rande`, which is redefined by `RcppOctave` to call the R base function `rexp`. This enables to exactly reproduce stochastic computations in R and Octave, without changing the original Octave/Matlab code. See `o_runif` for more details.

Usage

```
o_rexp(n, p = n)
```

Arguments

n	number of output rows
p	number of output columns (default to n)

Octave Documentation for *rande*

```
USAGE: E = rande( n [, k])

Generates standard-exponential random variatesas R function 'rexp' -- using the current RNG from R.

Possible calls:
rande(n, k)   returns n*k matrix with uncorrelated E(0, 1) deviates drawn in columns
rande(n)      returns n*n matrix with uncorrelated E(0, 1) deviates drawn in columns


NOTE:
This function substitutes Octave original function in calls from RcppOctave
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

See Also

rexp

Other orandom: o_rgamma, o_rnorm, o_runif

Examples

```
# Draw random exponential values (in vector form)
set.seed(123)
o_rexp(1)
o_rexp(1, 10)

# Draw random normal values (in matrix form)
set.seed(123)
o_rexp(2)
o_rexp(2, 5)
```

o_rgamma *Drawing from R Gamma Distribution in Octave*

Description

This function wraps a call to the standard Octave function randg, which is redefined by RcppOctave to call the R base function rgamma. This enables to exactly reproduce stochastic computations in R and Octave, without changing the original Octave/Matlab code. See o_runif for more details.

Usage

```
o_rgamma(n, p = n, shape = 1, scale = 1)
```

Arguments

shape	Mean of the Gamma distribution
scale	Scale of the Gamma distribution
n	number of output rows
p	number of output columns (default to n)

Octave Documentation for *randg*

```
USAGE: E = randg( n [, k, shape, scale])

Generates Gamma random variates as R function 'rgamma' -- using the current RNG from R.

Possible calls:
randg(shape) returns a single draw from G(shape, 1)
randg(shape, n)  returns n*n matrix with uncorrelated G(shape, 1) deviates drawn in columns
randg(shape, n, p)  returns n*p matrix with uncorrelated G(shape, 1) deviates drawn in columns
randg(shape, n, p, scale)  returns n*p matrix with uncorrelated G(shape, scale) deviates drawn in c


NOTE:
This function substitutes Octave original function in calls from RcppOctave
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

See Also

rgamma

Other orandom: o_rexp, o_rnorm, o_runif

Examples

```
# Draw random gamma values (in vector form)
set.seed(123)
o_rgamma(1)
o_rgamma(1, 10)

# Draw random gamma values (in matrix form)
set.seed(123)
o_rgamma(2)
o_rgamma(2, 5)

# Draw random gamma values with shape and scale parameters
o_rgamma(1, 5, shape=2)
o_rgamma(1, 10, scale=0.5)
```

o_rnorm *Drawing from R Normal Distribution in Octave*

Description

This function wraps a call to the standard Octave function `randn`, which is redefined by `RcppOctave` to call the R base function `rnorm`. This enables to exactly reproduce stochastic computations in R and Octave, without changing the original Octave/Matlab code. See `o_runif` for more details.

Usage

```
o_rnorm(n, p = n)
```

Arguments

`n`	number of output rows
`p`	number of output columns (default to `n`)

Octave Documentation for *randn*

```
USAGE: N = randn( n [, k])

Generates standard-normal random variates as R function 'rnorm' -- using the current RNG from R.

Possible calls:
randn(n, k)   returns a n*k matrix with uncorrelated N(0, 1) deviates drawn in columns
randn(n)      returns a n*n matrix with uncorrelated N(0, 1) deviates draw in columns


NOTE:
This function substitutes Octave original function in calls from RcppOctave
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

See Also

rnorm

Other orandom: `o_rexp`, `o_rgamma`, `o_runif`

Examples

```
# Draw random normal values (in vector form)
set.seed(123)
o_rnorm(1)
o_rnorm(1, 10)

# Draw random normal values (in matrix form)
```

```
set.seed(123)
o_rnorm(2)
o_rnorm(2, 5)
```

---

o_runif *Drawing from R Uniform Distribution in Octave*

Description

This function wraps a call to the standard Octave function `rand`, which is redefined by `RcppOctave` to call the R base function `runif`. This enables to exactly reproduce stochastic computations in R and Octave, without changing the original Octave/Matlab code.

Usage

```
o_runif(n, p = n)
```

Arguments

`n`	number of output rows
`p`	number of output columns (default to `n`)

Value

a numeric vector or a matrix

Difference with plain `runif`

Since calling `o_runif` or `runif` is equivalent, this function may not be really useful for the end user, and is defined for testing purposes essentially. One possible advantage over plain `runif` however, is that it can generate random matrices, instead of only vectors (see examples).

Seeding

Because the RNG of R is called used, seeding computations is achieved by a standard call to `set.seed`.

Octave details

`RcppOctave` defines a set of functions like `rand` that shadow Octave built-in functions. These functions are defined in the Octave module `Rrng.oct` that is stored in the package *modules/* subdirectory (call `OctaveConfig()`) to see the exact path to this location).

Octave Documentation for *rand*

```
USAGE: U = rand( n [, k])

Generates uniform random variates as R function 'runif' -- using the current RNG from R.

Possible calls:
rand(n, k)   returns a n*k matrix with uncorrelated U(0, 1) deviates drawn in columns
rand(n)      returns a n*n matrix with uncorrelated U(0, 1) deviates drawn in columns


NOTE:
This function substitutes Octave original function in calls from RcppOctave
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

See Also

runif

Other orandom: `o_rexp`, `o_rgamma`, `o_rnorm`

Examples

```
# Draw random uniform values (in vector form)
set.seed(123)
o_runif(1)
o_runif(1, 10)
# The result is identical as calling runif
set.seed(123)
runif(1)
runif(10)

# Draw random uniform values (in matrix form)
set.seed(123)
o_runif(2)
o_runif(2, 5)
```

`o_source` *Sourcing Octave/Matlab Files*

Description

This function sources an Octave file within the current Octave session. The loaded functions are accessible by subsequent calls of `.CallOctave`.

Usage

```
o_source(file = "", text = NULL, sep = ";\n")
```

Arguments

`file`	the path to the Octave/Matlab source file – typically with extension ".m".
`text`	a character vector containing *Octave* statements, that are concatenated in a temporary file, which is then sourced. This argument typically enables the evaluation of multiple statements, as opposed to single statement evaluation performed by `o_eval`.
`sep`	single character string added as suffix to each element of `text`. The concatenation of all suffixed element should form a valid *Octave* block.

Value

None

Octave Documentation for *source*

```
 -- Built-in Function:  source (FILE)
     Parse and execute the contents of FILE.  This is equivalent to
     executing commands from a script file, but without requiring the
     file to be named 'FILE.m'.
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

See Also

Other Octave_files: `o_addpath`, `o_inpath`

Examples

```
# source file
mfile <- system.file("scripts/ex_source.m", package='RcppOctave')
o_source(mfile)

# pass multiple statements
o_source(text="a=1;b=3;c=randn(1,5);")
o_get('a','b','c')

# also works with a character vector of statements
o_source(text=c("a=10;b=30;", "c=randn(1,5)", "d=4"))
o_get('a','b','c', 'd')
```

o_version *Get Octave Version*

Description

Returns the version of Octave currently used by `RcppOctave`.

Usage

```
o_version()
```

Value

Octave version as a single character string

Examples

```
o_version()
```

o_who *Listing Octave Variables*

Description

Lists currently defined variables in Octave global context.

Usage

```
o_who(..., options, rm.ans = FALSE, unique = TRUE)
```

Arguments

`...`	filtering patterns or extra arguments passed to `o_who` and `o_whos`. Only names matching any of the patterns are returned.
`rm.ans`	a logical that indicates if the automatic Octave variable `ans` should be included in the result (`FALSE`) or removed (`TRUE`).
`options`	options passed to Octave function `who`. See section *Octave Documentation*.
`unique`	a logical that indicates whether unique names should be returned. This argument is relevant in the case multiple patterns are specified in `...`.

Value

None

Octave Documentation for *who*

```
 -- Command:  who
 -- Command:  who pattern ...
 -- Command:  who option pattern ...
 -- Command: C = who ("pattern", ...)
     List currently defined variables matching the given patterns.
     Valid pattern syntax is the same as described for the 'clear'
     command.  If no patterns are supplied, all variables are listed.
     By default, only variables visible in the local scope are
     displayed.

     The following are valid options but may not be combined.

    'global'
          List variables in the global scope rather than the current
          scope.

    '-regexp'
          The patterns are considered to be regular expressions when
          matching the variables to display.  The same pattern syntax
          accepted by the 'regexp' function is used.

    '-file'
          The next argument is treated as a filename.  All variables
          found within the specified file are listed.  No patterns are
          accepted when reading variables from a file.

     If called as a function, return a cell array of defined variable
     names matching the given patterns.

     See also: whos, isglobal, isvarname, exist, regexp
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

See Also

Other listoct: o_ls, o_whos

Examples

```
o_who()
l <- as.list(setNames(1:10, letters[1:10]))
o_assign(l)
o_who()


prefnames <- paste('pref', letters[1:10], sep='')
```

```
o_assign( setNames(l, prefnames) )
o_who()
o_who('pref*')
```

o_whos *Detailed Listing of Octave Variables*

Description

The function `o_whos` returns a detailed description of the variables defined in the current Octave session.

Usage

```
o_whos(..., options, rm.ans = FALSE)
```

Arguments

`...`	filtering patterns or extra arguments passed to `o_who` and `o_whos`. Only names matching any of the patterns are returned.
`options`	options passed to Octave function `who`. See section *Octave Documentation*.
`rm.ans`	a logical that indicates if the automatic Octave variable `ans` should be included in the result (`FALSE`) or removed (`TRUE`).

Octave Documentation for *whos*

```
 -- Command:  whos
 -- Command:  whos pattern ...
 -- Command:  whos option pattern ...
 -- Command: S = whos ("pattern", ...)
     Provide detailed information on currently defined variables
     matching the given patterns.  Options and pattern syntax are the
     same as for the `who' command.  Extended information about each
     variable is summarized in a table with the following default
     entries.

    Attr
          Attributes of the listed variable.  Possible attributes are:
         blank
               Variable in local scope

         `a'
               Automatic variable.  An automatic variable is one
               created by the interpreter, for example `argn'.

         `c'
```

```
            Variable of complex type.

       'f'
            Formal parameter (function argument).

       'g'
            Variable with global scope.

       'p'
            Persistent variable.

 Name
      The name of the variable.

 Size
      The logical size of the variable.  A scalar is 1x1, a vector
      is 1xN or Nx1, a 2-D matrix is MxN.

 Bytes
      The amount of memory currently used to store the variable.

 Class
      The class of the variable.  Examples include double, single,
      char, uint16, cell, and struct.

  The table can be customized to display more or less information
  through the function 'whos_line_format'.

  If 'whos' is called as a function, return a struct array of defined
  variable names matching the given patterns.  Fields in the
  structure describing each variable are: name, size, bytes, class,
  global, sparse, complex, nesting, persistent.

  See also: who, whos_line_format
```

[Generated from Octave-3.6.1 on 2012-12-17 16:29:11]

See Also

Other listoct: o_ls, o_who

Examples

```
.O$a <- 1
.O$b <- 10
o_whos()
```

```
o_eval("sqrt(2)")
o_whos()
```

`RcppOctave` *Interfacing R with Octave*

Description

Interfacing R with Octave.

Details

The primary goal is to facilitate the port of Matlab/Octave scripts to R. The package enables to call any Octave functions from R and as well as browsing their documentation, passing variables between R and Octave, using R core RNGs in Octave, which ensure stochastic computations are also reproducible.

Package:	RcppOctave
Type:	Package
Version:	1.0
Date:	2011-11-01
License:	GPL (>= 2)
LazyLoad:	yes

Author(s)

Renaud Gaujoux <`renaud@cbio.uct.ac.za`>

Maintainer: Renaud Gaujoux <`renaud@cbio.uct.ac.za`>

References

Eaton JW (2002). _GNU Octave Manual_. Network Theory Limited. ISBN 0-9541617-2-6, <URL: http://www.octave.org/>.

See Also

See `.CallOctave`, `o_source`, `o_help`

Examples

```
.CallOctave('svd', matrix(1:9, 3))
o_help('svd')
```

show,Octave-method *Show method for Octave objects*

Description

Show method for Octave objects

Usage

```
  ## S4 method for signature 'Octave'
show(object)
```

Arguments

object	Any R object

sourceExamples *Loading Example M-files*

Description

Source an example M-file in the sub-directory “scripts/” of RcppOctave installation.

Usage

```
  sourceExamples(file)
```

Arguments

file	filename of the example script to source. If missing, the function lists all the m-files from the “scripts/” sub-directory.

Examples

```
sourceExamples()
sourceExamples('ex_source.m')
```

Index