Package 'callr'

August 29, 2016

Title Call R from R

Version 1.0.0

Author Gábor Csárdi

Maintainer Gábor Csárdi <gcsardi@mango-solutions.com>

Description It is sometimes useful to perform a computation in a separate R process, without affecting the current R process at all. This packages does exactly that.

License MIT + file LICENSE

LazyData true

URL https://github.com/MangoTheCat/callr

BugReports https://github.com/MangoTheCat/callr/issues

RoxygenNote 5.0.1.9000

Suggests covr, testthat

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-18 15:24:15

R topics documented:

callr 2
r 2
rcmd 4
rcmd_safe 5
r_safe 6
r_vanilla 7

Index **9**

callr *Call R from R*

Description

It is sometimes useful to perform a computation in a separate R process, without affecting the current R process at all. This packages does exactly that.

r *Evaluate an expression in another R session*

Description

Evaluate an expression in another R session

Usage

```
r(func, args = list(), libpath = .libPaths(), repos = getOption("repos"),
  stdout = NULL, stderr = NULL, error = c("error", "stack", "debugger"),
  cmdargs = "--slave", show = FALSE, callback = NULL,
  system_profile = TRUE, user_profile = TRUE, env = character())
```

Arguments

func	Function object to call in the new R process. The function should be self-contained and only refer to other functions and use variables explicitly from other packages using the :: notation. The environment of the function is set to .GlobalEnv before passing it to the child process. Because of this, it is good practice to create an anonymous function and pass that to callr, instead of passing a function object from a (base or other) package. In particular
	`r(.libPaths)`
	does not work, because it is defined in a special environment, but
	`r(function() .libPaths())`
	works just fine.

args	Arguments to pass to the function. Must be a list. vector.
libpath	The library path.
repos	The 'repos' option. If NULL, then no repos option is set. This options is only used if user_profile or system_profile is set to FALSE, as it is set using the system or the user profile.
stdout	The name of the file the standard output of the child R process will be written to. If the child process runs with the --slave option (the default), then the commands are not echoed and will not be shown in the standard output. Also note that you need to call 'print()' explicitly to show the output of the command(s).

stderr	The name of the file the standard error of the child R process will be written to. In particular `message()` sends output to the standard error. If nothing was sent to the standard error, then this file will be empty.
error	What to do if the remote process throws an error. See details below.
cmdargs	Command line arguments to pass to the R process. Note that `c("-f", rscript)` is appended to this, `rscript` is the name of the script file to run. This contains a call to the supplied function and some error handling code.
show	Logical, whether to show the standard output on the screen while the child process is running. Note that this is independent of the `stdout` and `stderr` arguments. The standard error is not shown currently.
callback	A function to call for each line of the standard output from the child process. It works together with the `show` option; i.e. if `show = TRUE`, and a callback is provided, then the output is shown of the screen, and the callback is also called.
system_profile	Whether to use the system profile file.
user_profile	Whether to use the user's profile file.
env	Environment variables to set for the child process.

Value

Value of the evaluated expression.

Error handling

`callr` handles errors properly. If the child process throws an error, then `callr` throws an error with the same error message in the parent process.

The 'error' expert option may be used to specify a different behavior on error. The following values are possible:

- 'error' is the default behavior: throw an error in the parent, with the same error message. In fact the same error object is thrown again.
- 'stack' also throws an error in the parent, but the error is of a special kind, class `callr_condition`, and it contains both the original error object, and the call stack of the child, as written out by `dump.frames`.
- 'debugger' is similar to 'stack', but in addition to returning the complete call stack, it also start up a debugger in the child call stack, via `debugger`.

See Also

Other callr functions: `r_safe`, `r_vanilla`

Examples

```
# Workspace is empty
r(function() ls())

# library path is the same by default
```

```
r(function() .libPaths())
.libPaths()
```

---

rcmd *Run an R CMD command*

Description

Run an R CMD command form within R. This will usually start another R process, from a shell script.

Usage

```
rcmd(cmd, cmdargs = character(), libpath = .libPaths(),
  repos = getOption("repos"), stdout = NULL, stderr = NULL,
  show = FALSE, callback = NULL, system_profile = FALSE,
  user_profile = FALSE, env = character())
```

Arguments

cmd	Command to run. See `R --help` from the command line for the various commands. In the current version of R (3.2.4) these are: BATCH, COMPILE, SHLIB, INSTALL, REMOVE, build, check, LINK, Rprof, Rdconv, Rd2pdf, Rd2txt, Stangle, Sweave, Rdiff, config, javareconf, rtags.
cmdargs	Command line arguments.
libpath	The library path.
repos	The 'repos' option. If `NULL`, then no `repos` option is set. This options is only used if `user_profile` or `system_profile` is set to `FALSE`, as it is set using the system or the user profile.
stdout	Optionally a file name to send the standard output to.
stderr	Optionally a file name to send the standard error to.
show	Logical, whether to show the standard output on the screen while the child process is running. Note that this is independent of the `stdout` and `stderr` arguments. The standard error is not shown currently.
callback	A function to call for each line of the standard output from the child process. It works together with the `show` option; i.e. if `show = TRUE`, and a callback is provided, then the output is shown of the screen, and the callback is also called.
system_profile	Whether to use the system profile file.
user_profile	Whether to use the user's profile file.
env	Environment variables to set for the child process.

Value

A list with the standard output (`$stdout`), standard error (`stderr`) and exit status (`$status`) of the external `R CMD` command.

See Also

Other R CMD commands: rcmd_safe

Examples

```
rcmd("config", "CC")
```

`rcmd_safe` *Call R CMD <command> safely*

Description

Very similar to rcmd, but with different defaults, that tend to create a less error-prone execution environment for the child process.

Usage

```
rcmd_safe(cmd, cmdargs = character(), libpath = .libPaths(),
  repos = c(getOption("repos"), c(CRAN = "https://cran.rstudio.com")),
  system_profile = FALSE, user_profile = FALSE, env = c(CYGWIN =
  "nodosfilewarning", R_TESTS = "", R_BROWSER = "false", R_PDFVIEWER = "false"),
  ...)
```

Arguments

`cmd`	Command to run. See `R --help` from the command line for the various commands. In the current version of R (3.2.4) these are: BATCH, COMPILE, SHLIB, INSTALL, REMOVE, build, check, LINK, Rprof, Rdconv, Rd2pdf, Rd2txt, Stangle, Sweave, Rdiff, config, javareconf, rtags.
`cmdargs`	Command line arguments.
`libpath`	The library path.
`repos`	The 'repos' option. If `NULL`, then no `repos` option is set. This options is only used if `user_profile` or `system_profile` is set to `FALSE`, as it is set using the system or the user profile.
`system_profile`	Whether to use the system profile file.
`user_profile`	Whether to use the user's profile file.
`env`	Environment variables to set for the child process.
`...`	Additional arguments are passed to rcmd.

See Also

Other R CMD commands: rcmd

`r_safe` *Run an R child process in safe mode*

Description

The following options are set up:

- The library path is set to the current path.
- Makes sure that at least one reasonable CRAN mirror is set up.
- Some command line arguments are added to avoid saving `.RData` files, etc. See them above.
- The system and user profile files are ignored.
- Various environment variables are set: `CYGWIN` to avoid warnings about DOS-style paths, `R_TESTS` to avoid issues when `callr` is invoked from unit tests, `R_BROWSER` and `R_PDFVIEWER` to avoid starting a browser or a PDF viewer.

Usage

```
r_safe(func, args = list(), libpath = .libPaths(),
  repos = c(getOption("repos"), c(CRAN = "https://cran.rstudio.com")),
  cmdargs = c("--no-site-file", "--no-environ", "--slave", "--no-save",
  "--no-restore"), system_profile = FALSE, user_profile = FALSE,
  env = c(CYGWIN = "nodosfilewarning", R_TESTS = "", R_BROWSER = "false",
  R_PDFVIEWER = "false"), ...)
```

Arguments

`func`	Function object to call in the new R process. The function should be self-contained and only refer to other functions and use variables explicitly from other packages using the `::` notation. The environment of the function is set to `.GlobalEnv` before passing it to the child process. Because of this, it is good practice to create an anonymous function and pass that to `callr`, instead of passing a function object from a (base or other) package. In particular
	`r(.libPaths)`
	does not work, because it is defined in a special environment, but
	`r(function() .libPaths())`
	works just fine.

`args`	Arguments to pass to the function. Must be a list. vector.
`libpath`	The library path.
`repos`	The 'repos' option. If `NULL`, then no `repos` option is set. This options is only used if `user_profile` or `system_profile` is set to `FALSE`, as it is set using the system or the user profile.
`cmdargs`	Command line arguments to pass to the R process. Note that `c("-f", rscript)` is appended to this, `rscript` is the name of the script file to run. This contains a call to the supplied function and some error handling code.

system_profile	Whether to use the system profile file.
user_profile	Whether to use the user's profile file.
env	Environment variables to set for the child process.
...	Additional arguments are passed to r.

See Also

Other callr functions: r_vanilla, r

---

r_vanilla *Run an R child process, with no configuration*

Description

It tries to mimic a fresh R installation. In particular:

- No library path setting.
- No CRAN(-like) repository is set.
- The system and user profiles are not run.

Usage

```
r_vanilla(func, args = list(), libpath = character(), repos = c(CRAN =
  "@CRAN@"), cmdargs = "--slave", system_profile = FALSE,
  user_profile = FALSE, env = character(), ...)
```

Arguments

func	Function object to call in the new R process. The function should be self-contained and only refer to other functions and use variables explicitly from other packages using the :: notation. The environment of the function is set to .GlobalEnv before passing it to the child process. Because of this, it is good practice to create an anonymous function and pass that to callr, instead of passing a function object from a (base or other) package. In particular
	`r(.libPaths)`
	does not work, because it is defined in a special environment, but
	`r(function() .libPaths())`
	works just fine.

args	Arguments to pass to the function. Must be a list. vector.
libpath	The library path.
repos	The 'repos' option. If NULL, then no repos option is set. This options is only used if user_profile or system_profile is set to FALSE, as it is set using the system or the user profile.

cmdargs	Command line arguments to pass to the R process. Note that `c("-f", rscript)` is appended to this, `rscript` is the name of the script file to run. This contains a call to the supplied function and some error handling code.
system_profile	Whether to use the system profile file.
user_profile	Whether to use the user's profile file.
env	Environment variables to set for the child process.
...	Additional arguments are passed to `r`.

See Also

Other callr functions: `r_safe`, `r`

Examples

```
# Compare to r()
r(function() .libPaths())
r_vanilla(function() .libPaths())

r(function() getOption("repos"))
r_vanilla(function() getOption("repos"))
```

Index

callr, 2
callr-package (callr), 2

debugger, *3*
dump.frames, *3*

r, 2, *7, 8*
r_safe, *3*, 6, *8*
r_vanilla, *3*, *7*, 7
rcmd, 4, *5*
rcmd_safe, *5*, 5