Package ‘rhub’

April 8, 2019

Title Connect to ‘R-hub’

Version 1.1.1

Description Run 'R CMD check' on any of the 'R-hub' (<https://builder.r-hub.io/>) architectures, from the command line. The current architectures include 'Windows', 'macOS', 'Solaris' and various 'Linux' distributions.

License MIT + file LICENSE

LazyData true


BugReports https://github.com/r-hub/rhub/issues

RoxygenNote 6.1.1

Roxygen list(markdown = TRUE)

Imports assertthat,
callr,
cli (>= 1.1.0),
crayon,
desc,
digest,
httr,
jsonlite,
parsedate,
pillar,
prettyunits,
processx,
R6,
rappdirs,
rcmdcheck (>= 1.2.1),
rematch,
tibble,
tools,
uuid,
whoami,
withr

Suggests covr,
testthat,
knitr,
rmarkdown
**check**

Check an R package on R-hub

**Description**

Check an R package on R-hub

**Usage**

```r
check(path = ".", platform = NULL, email = NULL, valgrind = FALSE,
check_args = character(), env_vars = character(),
show_status = interactive())
```

**Arguments**

- **path**
  Path to a directory containing an R package, or path to source R package tarball built with `R CMD build` or `devtools::build()`.

- **platform**
  Platform to build/check the package on. See `platforms()` for the available platforms. If this is `NULL`, and the R session is interactive, then a menu is shown. If it is `NULL`, and the session is not interactive, then the default R-hub platforms is used. Can take a vector of platforms which saves time by building one R package tarball that is used for all the platforms specified.

- **email**
  Email address to send notification to about the check. It must be a validated email address, see `validate_email()`. If `NULL`, then the email address of the maintainer is used, as defined in the DESCRIPTION file of the package.

- **valgrind**
  Whether to run the check in valgrind. Only supported on Linux currently, and ignored on other platforms.

- **check_args**
  Extra arguments for the `R CMD check` command.
check_for_cran

**env_vars**
Environment variables to set on the builder machine before the check. A named character vector.

**show_status**
Whether to show the status of the build and check (live log) as it is happening.

**Value**
An rhub_check object.

**Examples**

```r
## Not run:
check(".
check("mypackage_1.0.0.tar.gz", platform = "fedora-clang-devel")
## End(Not run)
```

---

**Description**
This function calls `check()` with arguments and platforms, that are suggested for a CRAN submission.

**Usage**

```r
check_for_cran(path = ".", email = NULL, check_args = "--as-cran",
env_vars = c("_R_CHECK_FORCE_SUGGESTS_" = "true",
'_R_CHECK_CRAN_INCOMING_USE_ASPELL_' = "true"), platforms = NULL, ...)
```

**Arguments**

- **path**: Path to a directory containing an R package, or path to source R package tarball built with `R CMD build` or `devtools::build()`.
- **email**: Email address to send notification to about the check. It must be a validated email address, see `validate_email()`. If NULL, then the email address of the maintainer is used, as defined in the DESCRIPTION file of the package.
- **check_args**: Arguments for `R CMD check`. By default `--as-cran` is used.
- **env_vars**: Environment variables to set on the builder. By default `_R_CHECK_FORCE_SUGGESTS_=true` is set, to require all packages used. `_R_CHECK_CRAN_INCOMING_USE_ASPELL_=true` is also set, to use the spell checker.
- **platforms**: Character vector of platform ids to use (see `platforms()`), or NULL. If NULL, then a set of default platforms will be selected, see below.
- **...**: Additional arguments are passed to `check()`.

**Details**
In particular, if `platforms` is NULL (the default), then

- It checks the package on Windows, and Linux.
- It checks the package on R-release and R-devel.
- It uses the `--as-cran` argument to `R CMD check`.
- It requires all dependencies, including suggested ones.
Value

An `rhub_check` object.

Examples

```r
## Not run:
ch <- check_for_cran("package", show_status = FALSE)
ch$update()
ch$livelog(3)
## End(Not run)
```

---

`check_on_linux` Check an R package on an R-hub platform

Description

These functions provide a quick easy to use interface to check a package on a platform with some particular aspect. Which platform they use might change over time.

Usage

```r
check_on_linux(path = ".", ...)
check_on_windows(path = ".", ...)
check_on_macos(path = ".", ...)
check_on_debian(path = ".", ...)
check_on_ubuntu(path = ".", ...)
check_on_fedora(path = ".", ...)
check_on_solaris(path = ".", 
                 check_args = "--no-manual --no-build-vignettes", ...)
check_on_centos(path = ".", ...)
check_with_roldrel(path = ".", ...)
check_with_rrelease(path = ".", ...)
check_with_rpatched(path = ".", ...)
check_with_rdevel(path = ".", ...)
check_with_valgrind(path = ".", ...)
check_with_sanitizers(path = ".", ...)
```
get_check

Arguments

- **path**: Path to a directory containing an R package, or path to source R package tarball built with `R CMD build` or `devtools::build()`.
- **...**: Additional arguments are passed to `check()`.
- **check_args**: Extra arguments for the `R CMD check` command.

Value

An `rhub_check` object.

---

**get_check** Retrieve the result of R-hub checks

Description

Retrieve the result of R-hub checks

Usage

`get_check(ids)`

Arguments

- **ids**: One of the following:
  - A single R-hub check id.
  - A character vector of check ids.
  - An R-hub check group id. All ids can be abbreviated, see R-hub ids.

Value

An `rhub_check` object.

Examples

```r
cchk <- get_check("915ee61")
chk
chk$update()
chk$browse()
chk$cran_summary()
chk$urls()
```

See Also

`list_my_checks()` and `list_package_checks()` to list R-hub checks.
list_my_checks

Description

List all checks for an email address

Usage

list_my_checks(email = email_address(), package = NULL, howmany = 20)

Arguments

description
    Email address. By default it is guessed with whoami::email_address(). The
    address must be validated, see validate_email().

description
    NULL, or a character scalar. Can be used to restrict the search for a single pack-
    age.

howmany
    How many check groups (checks submitted simultaneously) to show. The cur-
    rent API limit is 20.
Value

A tibble::tibble with columns:

- package Name of the package.
- version Package version.
- result: More detailed result of the check. Can be NULL for errors. This is a list column with members: status, errors, warnings, notes.
- group: R-hub check group id.
- id: ‘R-hub check id.
- platform_name: Name of the check platform.
- build_time: Build time, a difftime object.
- submitted: Time of submission.
- started: Time of the check start.
- platform: Detailed platform data, a list column.
- builder: Name of the builder machine.
- status Status of the check. Possible values:
  - created: check job was created, but not running yet.
  - in-progress: check job is running.
  - parseerror: internal R-hub error parsing the check results.
  - preerror: check error, before the package check has started.
  - aborted: aborted by admin or user.
  - error: failed check. (Possibly warnings and notes as well.)
  - warning: R CMD check reported warnings. (Possibly notes as well.)
  - note: R CMD check reported notes.
  - ok: successful check.
- email: Email address of maintainer / submitter.

See Also

list_package_checks

Examples

```r
## Not run:
ch <- list_my_checks()
ch
chd$details()
## End(Not run)
```
**list_package_checks**  
List checks of a package

**Description**
List checks of a package

**Usage**
```
list_package_checks(package = ".", email = NULL, howmany = 20)
```

**Arguments**
- `package` Directory of an R package, or a package tarball.
- `email` Email address that was used for the check(s). If `NULL`, then the maintainer address is used.
- `howmany` How many checks to show. The current maximum of the API is 20.

**Value**
A `tibble::tibble` with columns:
- `package` Name of the package.
- `version` Package version.
- `result` More detailed result of the check. Can be `NULL` for errors. This is a list column with members: `status, errors, warnings, notes`.
- `group` R-hub check group id.
- `id` 'R-hub check id.
- `platform_name` Name of the check platform.
- `build_time` Build time, a `difftime` object.
- `submitted` Time of submission.
- `started` Time of the check start.
- `platform` Detailed platform data, a list column.
- `builder` Name of the builder machine.
- `status` Status of the check. Possible values:
  - `created`: check job was created, but not running yet.
  - `in-progress`: check job is running.
  - `parseerror`: internal R-hub error parsing the check results.
  - `preerror`: check error, before the package check has started.
  - `aborted`: aborted by admin or user.
  - `error`: failed check. (Possibly warnings and notes as well.)
  - `warning`: `R CMD check` reported warnings. (Possibly notes as well.)
  - `note`: `R CMD check` reported notes.
  - `ok`: successful check.
- `email` Email address of maintainer / submitter.
Examples

```r
## Not run:
ch <- list_package_checks()
ch
ch$details(1)
## End(Not run)
```

**list_validated_emails**  *List validated email addresses*

**Description**

List email addresses validated on R-hub on the current machine.

**Usage**

`list_validated_emails()`

**Value**

A `data.frame` with two columns: `email` and `token`. If in interactive mode, and there are no validated email addresses, then a message is printed and the data frame is returned invisibly.

**See Also**

Other email validation: `validate_email`

---

**local_check_linux**  *Run a package check locally, in a Docker container*

**Description**

Run a package check locally, in a Docker container. UNTESTED ON WINDOWS, bug reports welcome. :-)

**Usage**

`local_check_linux(path = ".", quiet = FALSE, image = NULL, valgrind = FALSE, check_args = character(), env_vars = character(), timeout = Inf, artifacts = tempfile())`
Arguments

path Path to a directory containing an R package, or path to source R package tarball built with `R CMD build` or `devtools::build()`. 
quiet Whether to print the check output 
image Docker image to use. If NULL, a default image is selected. 
valgrind Whether to run the check with Valgrind. 
check_args Extra arguments for the `R CMD check` command. 
env_vars Environment variables to set on the builder machine before the check. A named character vector. 
timeout Timeout for a check, a `difftime` object or a scalar that will be interpreted as seconds. 
artifacts Where to copy the build artifacts after the build.

Details

You’ll need to have bash and Docker installed.

Value

An `rcmdcheck::rcmdcheck` object, with extra fields:

- all_output: all output from the check, both standard output and error.
- container_name: name of the Docker container that performed the build. It is a random name.
- artifacts: directory of build artifacts.

Description

The images are pretty-printed in a short format. Use `as.data.frame()` to get all available platform metadata.

Usage

`local_check_linux_images()`
platforms

platforms List all R-hub platforms

Description
The platforms are pretty-printed in a short format. Use as.data.frame(platforms()) to get all available platform metadata.

Usage
platforms()

Examples
## Not run:
platforms()
as.data.frame(platforms())
## End(Not run)

rhub-ids R-hub check ids

Description
R-hub check ids

R-hub ids
Every R-hub check has a unique id, that is constructed from the name of the source package archive, and a random string. For example:

devtools_2.0.0.tar.gz-fe53bbba85de4a579f6dc3b852bf76a3

R-hub group ids
For every check submission, R-hub also creates a unique check group id. One check group may contain multiple checks. E.g. check_for_cran() typically creates three or four check groups. Group ids look the same as individual check ids.

Abbreviating ids
The rhub package keeps a list of all the checks that it has seen in the current session, and these checks can be also referenced by any unique prefix of the random string part of the id, e.g. in the get_check() function. E.g. if rhub already know the devtools check above, then

get_check("fe53bbb")

works.
This is only recommended in interactive mode, and we suggest that you always use the full ids when using rhub programmatically.
**rhub_check**

An *rhub_check* object holds status and results of rhub checks

**Description**

An *rhub_check* object holds status and results of rhub checks

**Usage**

```r
ch <- rhub_check$new(ids = NULL, status = NULL, group = NULL)
ch$update()
ch$print(...)
ch$browse(which = NULL)
ch$urls(which = NULL)
ch$livelog(which = 1)
ch$cran_summary()
```

**Arguments**

- **ch** An rhub check object. It can be created using `check()`, and other check functions including `check_for_cran`. See also `last_check()`.
- **ids** Character vector of check ids.
- **status** Check status for ids or group.
- **group** Check group id, string scalar. Either group or ids must be non-NULL.
- **...** Extra arguments are currently ignored.
- **which** Which check to show, if the object contains multiple checks. For `browse` the default is all checks. For `livelog` the default is the first check. A check can be selected via its number or id.

**Details**

An *rhub_check* object can be created by `check()`, `list_my_checks()`, or `list_package_checks()`. `last_check()` returns the last check(s) submitted from the current R session. Do not confuse `rhub_check`/`rhub_check_for_cran` (classes) with `check()` or `check_for_cran()` (functions).

`ch$update()` updates the status of the check. Printing the check status to the screen does not perform an update, unless the status of the check(s) is unknown.

`ch$print()` prints the status of the check(s) to the screen.

`ch$cran_summary()` prints text to be copy-pasted in cran-comments.md, it is especially useful on the output of `check_for_cran()`.

`ch$browse()` opens a tab or window in the default web browser, that points to the detailed logs of the check(s).

`ch$urls()` return a `tibble::tibble` with URL to the html log, text log and artifacts of the check(s).

For both `ch$browse()` and `ch$urls()`, note that the logs and artifacts are not kept forever, they are accessible for a few days after submission.

`ch$livelog()` shows the live log of the check. The live log can be interrupted using the usual interruption keyboard shortcut, usually CTRL+c or ESC.
validate_email

validate_email Validate an email address on R-hub

Description
To build and check R packages on R-hub, you need to validate your email address. This is because R-hub sends out emails about check results.

Usage
validate_email(email = NULL, token = NULL)

Arguments
email The email address to validate.
token Token obtained from rhub, to validate the email address.

Details
The rhub package stores validated email addresses in a user configuration file, at a platform-dependent location. On your current platform the file is at /Users/gaborcsardi/Library/Application Support/rhub/validated_emails.csv.

To validate a new email address, call this function from an interactive R session, without any arguments.

To add an email address that was validated before (probably on another machine), to the configuration file, call this function with the email and token arguments.

See Also
Other email validation: list_validated_emails
Index

check, 2
check(), 3, 5, 12
check_for_cran, 3, 12
check_for_cran(), 11, 12
check_on_centos (check_on_linux), 4
check_on_debian (check_on_linux), 4
check_on_fedora (check_on_linux), 4
check_on_linux, 4
check_on_macos (check_on_linux), 4
check_on_solaris (check_on_linux), 4
check_on_ubuntu (check_on_linux), 4
check_on_windows (check_on_linux), 4
check_with_rdevel (check_on_linux), 4
check_with_roldrel (check_on_linux), 4
check_with_rpatched (check_on_linux), 4
check_with_rrrelease (check_on_linux), 4
check_with_sanitizers (check_on_linux), 4
check_with_valgrind (check_on_linux), 4
difftime, 7, 8
get_check, 5
get_check(), 11
last_check, 6
last_check(), 12
list_my_checks, 6
list_my_checks(), 5, 12
list_package_checks, 8
list_package_checks(), 5, 12
list_validated_emails, 9, 13
local_check_linux, 9
local_check_linux_images, 10
platforms, 11
platforms(), 2, 3
R-hub ids, 5
rhub-ids, 11
rhub_check, 3–5, 12
tibble::tibble, 7, 8, 12
validate_email, 9, 13
validate_email(), 2, 3, 6
whoami::email_address(), 6