Package 'checked'

October 25, 2024

Title Systematically Run R CMD Checks

Version 0.2.4

Description Systematically Run R checks against multiple packages. Checks are run in parallel with strategies to minimize dependency installation. Provides out of the box interface for running reverse dependency check.

URL https://Genentech.github.io/checked/, https://github.com/Genentech/checked

BugReports https://github.com/Genentech/checked/issues

License MIT + file LICENSE

Encoding UTF-8

Imports callr, cli, igraph, jsonlite, options, R6, rcmdcheck, utils (>= 3.6.2), tools

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0), withr

Config/Needs/website r-lib/asciicast

Config/testthat/edition 3

NeedsCompilation no

Author Szymon Maksymiuk [cre, aut] (<https://orcid.org/0000-0002-3120-1601>), Doug Kelkhoff [aut] (<https://orcid.org/0009-0003-7845-4061>), F. Hoffmann-La Roche AG [cph, fnd]

Maintainer Szymon Maksymiuk <sz.maksymiuk@gmail.com>

Repository CRAN

Date/Publication 2024-10-25 15:20:02 UTC

Contents

`checked-task-df` *Check schedule data frame*

Description

Create data.frame which each row defines a package for which R CMD check should be run. Such data.frame is a prerequisite for generating `check_design()` which orchestrates all the processes including dependencies installation.

Arguments

`path`	path to the package source. Can be either a single source code directory or a directory containing multiple package source code directories.
`...`	parameters passed to the task specs allowing to customize subprocesses.

Details

`_tasks_df()` functions generate check task `data.frame` for all source packages specified by the `path`. Therefore it accepts it to be a vector of an arbitrary length.

Value

The check schedule `data.frame` with the following columns:

- `alias`: The alias of the check to run. It also serves the purpose of providing a unique identifier and node name in the task graph.

- `version`: Version of the package to be checked.
- `package`: Object that inherits from `check_task_spec()`. Defines how package to be checked can be acquired.
- `custom`: Object that inherits from `custom_install_task_spec()`. Defines custom package, for instance only available from local source, that should be installed before checking the package.

See Also

Other tasks: `check_task_spec()`, `custom_install_task_spec()`, `install_task_spec()`, `rev_dep_check_tasks_df()`, `revdep_check_task_spec()`, `source_check_tasks_df()`, `task_spec()`

`check_design` `R6` *Checks Coordinator*

Description

A stateful object that orchestrates all separate processes required to manage installation, library setup and run `R CMD checks` in sequence.

Public fields

`graph (igraph::igraph())`
: A dependency graph, storing information about which dependencies are required prior to execution of each check task. Created with `task_graph_create()`

`input (data.frame())`
: Checks task `data.frame` which is the source of all the checks.

`output (character(1))`
: Output directory where raw results and temporary library will be created and stored.

Methods

Public methods:

- `check_design$new()`
- `check_design$active_processes()`
- `check_design$failed_tasks()`
- `check_design$terminate()`
- `check_design$step()`
- `check_design$start_next_task()`
- `check_design$is_done()`
- `check_design$clone()`

Method `new()`**:** Initialize a new check design

Use checks data.frame to generate task graph in which all dependencies and installation order are embedded.

Usage:

```
check_design$new(
  df,
  n = 2L,
  output = tempfile(paste(packageName(), Sys.Date(), sep = "-")),
  lib.loc = .libPaths(),
  repos = getOption("repos"),
  restore = options::opt("restore"),
  ...
)
```

Arguments:

`df` check_design data.frame.

`n` `integer` value indicating maximum number of subprocesses that can be simultaneously spawned when executing tasks.

`output` `character` value specifying path where the output should be stored.

`lib.loc` `character` vector with libraries allowed to be used when checking packages, defaults to entire .libPaths().

`repos` `character` vector of repositories which will be used when generating task graph and later pulling dependencies.

`restore` `logical` value, whether output directory should be unlinked before running checks. If `FALSE`, an attempt will me made to restore previous progress from the same `output`.

`...` Additional arguments unused

Returns: check_design.

Method `active_processes()`**:** Get Active Processes list

Usage:

```
check_design$active_processes()
```

Method `failed_tasks()`**:** Get Failed Tasks list

Usage:

```
check_design$failed_tasks()
```

Method `terminate()`**:** Kill All Active Design Processes

Immediately terminates all the active processes.

Usage:

```
check_design$terminate()
```

Method `step()`**:** Fill Available Processes with Tasks

Usage:

```
check_design$step()
```

Returns: A logical value, indicating whether processes are actively running.

Method `start_next_task()`**:** Start Next Task

Usage:

```
check_design$start_next_task()
```

Returns: A integer value, coercible to logical to indicate whether a new process was spawned, or `-1` if all tasks have finished.

Method `is_done()`**:** Check if checks are done

Checks whether all the scheduled tasks were successfully executed.

Usage:

```
check_design$is_done()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
check_design$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other checks: `check_dev_rev_deps()`, `check_dir()`, `check_pkgs()`, `check_rev_deps()`, `new_check_design()`

Examples

```
## Not run:
library(checked)
df <- source_check_tasks_df(c(
  system.file("example_packages", "exampleBad", package = "checked"),
  system.file("example_packages", "exampleGood", package = "checked")
))

plan <- check_design$new(df, n = 10, repos = "https://cran.r-project.org/")
while (!plan$is_done()) {
  plan$start_next_task()
}

## End(Not run)
```

`check_dev_rev_deps` *Run reverse dependency checks against a development version only*

Description

`check_dev_rev_deps()` works similarly to `check_rev_deps()` but it runs R CMD check only once for each package, with the development version of the package installed. It is advantageous to check whether adding a new package into a repository breaks existing packages that possibly take said package as a `Suggests` dependency.

Usage

```
check_dev_rev_deps(
  path,
  n = 2L,
  output = tempfile(paste(utils::packageName(), Sys.Date(), sep = "-")),
  lib.loc = .libPaths(),
  repos = getOption("repos"),
  restore = options::opt("restore"),
  reporter = reporter_default(),
  ...
)
```

Arguments

path	file path to the package source directory
n	integer value indicating maximum number of subprocesses that can be simultaneously spawned when executing tasks.
output	character value specifying path where the output should be stored.
lib.loc	character vector with libraries allowed to be used when checking packages, defaults to entire .libPaths().
repos	character vector of repositories which will be used when generating task graph and later pulling dependencies.
restore	logical indicating whether output directory should be unlinked before running checks. If FALSE, an attempt will me made to restore previous progress from the same output (Defaults to NA, overwritable using option 'checked.restore' or environment variable 'R_CHECKED_RESTORE')
reporter	A reporter to provide progress updates. Will default to the most expressive command-line reporter given your terminal capabilities.
...	Additional arguments passed to checked-task-df and run()

Value

check_design() R6 class storing all the details regarding checks that run. Can be combined with results and summary() methods to generate results.

See Also

Other checks: check_design, check_dir(), check_pkgs(), check_rev_deps(), new_check_design()

check_dir *Check all package source directories in current directory*

Description

check_dir() Identifies all R packages in the given directory (non-recursively) and passes them to the check_pkgs()

Usage

```
check_dir(
  path,
  n = 2L,
  output = tempfile(paste(utils::packageName(), Sys.Date(), sep = "-")),
  lib.loc = .libPaths(),
  repos = getOption("repos"),
  restore = options::opt("restore"),
  reporter = reporter_default(),
  ...
)
```

Arguments

path	file path to the package source directory
n	integer value indicating maximum number of subprocesses that can be simultaneously spawned when executing tasks.
output	character value specifying path where the output should be stored.
lib.loc	character vector with libraries allowed to be used when checking packages, defaults to entire .libPaths().
repos	character vector of repositories which will be used when generating task graph and later pulling dependencies.
restore	logical indicating whether output directory should be unlinked before running checks. If FALSE, an attempt will me made to restore previous progress from the same output (Defaults to NA, overwritable using option 'checked.restore' or environment variable 'R_CHECKED_RESTORE')
reporter	A reporter to provide progress updates. Will default to the most expressive command-line reporter given your terminal capabilities.
...	Additional arguments passed to checked-task-df and run()

Value

check_design() R6 class storing all the details regarding checks that run. Can be combined with results and summary() methods to generate results.

See Also

Other checks: check_design, check_dev_rev_deps(), check_pkgs(), check_rev_deps(), new_check_design()

---

check_pkgs *Check one or more package source directories*

Description

check_pkgs() Installs all dependencies and runs R CMD checks in parallel for all source packages whose source code is found in the path directory

Usage

```
check_pkgs(
  path,
  n = 2L,
  output = tempfile(paste(utils::packageName(), Sys.Date(), sep = "-")),
  lib.loc = .libPaths(),
  repos = getOption("repos"),
  restore = options::opt("restore"),
  reporter = reporter_default(),
  ...
)
```

Arguments

path	file path to the package source directory
n	integer value indicating maximum number of subprocesses that can be simultaneously spawned when executing tasks.
output	character value specifying path where the output should be stored.
lib.loc	character vector with libraries allowed to be used when checking packages, defaults to entire .libPaths().
repos	character vector of repositories which will be used when generating task graph and later pulling dependencies.
restore	logical indicating whether output directory should be unlinked before running checks. If FALSE, an attempt will me made to restore previous progress from the same output (Defaults to NA, overwritable using option 'checked.restore' or environment variable 'R_CHECKED_RESTORE')
reporter	A reporter to provide progress updates. Will default to the most expressive command-line reporter given your terminal capabilities.
...	Additional arguments passed to checked-task-df and run()

Value

check_design() R6 class storing all the details regarding checks that run. Can be combined with results and summary() methods to generate results.

See Also

Other checks: check_design, check_dev_rev_deps(), check_dir(), check_rev_deps(), new_check_design()

---

check_rev_deps *Check reverse dependencies*

Description

Check a package's reverse dependencies in order to identify differences in reverse dependency check results when run alongside your package's development and release versions.

Usage

```
check_rev_deps(
  path,
  n = 2L,
  output = tempfile(paste(utils::packageName(), Sys.Date(), sep = "-")),
  lib.loc = .libPaths(),
  repos = getOption("repos"),
  reverse_repos = repos,
  restore = options::opt("restore"),
  reporter = reporter_default(),
  ...
)
```

Arguments

path	file path to the package source directory
n	integer value indicating maximum number of subprocesses that can be simultaneously spawned when executing tasks.
output	character value specifying path where the output should be stored.
lib.loc	character vector with libraries allowed to be used when checking packages, defaults to entire .libPaths().
repos	character vector of repositories which will be used when generating task graph and later pulling dependencies.
reverse_repos	character vector of repositories which will be used to pull sources for reverse dependencies. In some cases, for instance using binaries on Linux, we want to use different repositories when pulling sources to check and different when installing dependencies.
restore	logical indicating whether output directory should be unlinked before running checks. If FALSE, an attempt will me made to restore previous progress from the same output (Defaults to NA, overwritable using option 'checked.restore' or environment variable 'R_CHECKED_RESTORE')
reporter	A reporter to provide progress updates. Will default to the most expressive command-line reporter given your terminal capabilities.
...	Additional arguments passed to checked-task-df and run()

Details

Runs classical reverse dependency checks for the given source package. It first identifies reverse dependencies available in `repos`. Then, after installing all required dependencies, runs `R CMD check` twice for each package, one time with the release version of the given source package installed from `repos` and a second time with the development version installed from local source. Both `R CMD checks` are later compared to identify changes in reverse dependency behaviors.

Value

`check_design()` R6 class storing all the details regarding checks that run. Can be combined with `results` and `summary()` methods to generate results.

See Also

Other checks: `check_design`, `check_dev_rev_deps()`, `check_dir()`, `check_pkgs()`, `new_check_design()`

---

check_task_spec *Create a task to run* `R CMD check`

Description

Create a task to run `R CMD check`

Usage

```
check_task_spec(
  args = options::opt("check_args"),
  build_args = options::opt("check_build_args"),
  ...
)
```

Arguments

`args`	Character vector of arguments to pass to `R CMD check`. Pass each argument as a single element of this character vector (do not use spaces to delimit arguments like you would in the shell). For example, to skip running of examples and tests, use `args = c("--no-examples", "--no-tests")` and not `args = "--no-examples --no-tests"`. (Note that instead of the `--output` option you should use the `check_dir` argument, because `--output` cannot deal with spaces and other special characters on Windows.)
`build_args`	Character vector of arguments to pass to `R CMD build`. Pass each argument as a single element of this character vector (do not use spaces to delimit arguments like you would in the shell). For example, `build_args = c("--force", "--keep-empty-dirs")` is a correct usage and `build_args = "--force --keep-empty-dirs"` is incorrect.
`...`	Arguments passed on to `task_spec`

`alias` task alias which also serves as unique identifier of the task.

`package_spec` `package_spec` object

`env` environmental variables to be set in separate process running specific task.

See Also

Other tasks: `checked-task-df`, `custom_install_task_spec()`, `install_task_spec()`, `rev_dep_check_tasks_df()`, `revdep_check_task_spec()`, `source_check_tasks_df()`, `task_spec()`

`custom_install_task_spec` *Create a custom install task*

Description

Create a custom install task

Usage

```
custom_install_task_spec(...)
```

Arguments

`...` Arguments passed on to `install_task_spec`

`type` character, indicating the type of package to download and install. Will be `"source"` except on Windows and some macOS builds: see the section on 'Binary packages' for those.

`INSTALL_opts` an optional character vector of additional option(s) to be passed to `R CMD INSTALL` for a source package install. E.g., `c("--html", "--no-multiarch", "--no-test-load")`.
Can also be a named list of character vectors to be used as additional options, with names the respective package names.

See Also

Other tasks: `check_task_spec()`, `checked-task-df`, `install_task_spec()`, `rev_dep_check_tasks_df()`, `revdep_check_task_spec()`, `source_check_tasks_df()`, `task_spec()`

install_task_spec *Create a task to install a package and dependencies*

Description

Create a task to install a package and dependencies

Usage

```
install_task_spec(type = getOption("pkgType"), INSTALL_opts = NULL, ...)
```

Arguments

type	character, indicating the type of package to download and install. Will be "source" except on Windows and some macOS builds: see the section on 'Binary packages' for those.
INSTALL_opts	an optional character vector of additional option(s) to be passed to R CMD INSTALL for a source package install. E.g., c("--html", "--no-multiarch", "--no-test-load"). Can also be a named list of character vectors to be used as additional options, with names the respective package names.
...	Additional parameters passed to task_spec()

See Also

Other tasks: check_task_spec(), checked-task-df, custom_install_task_spec(), rev_dep_check_tasks_df(), revdep_check_task_spec(), source_check_tasks_df(), task_spec()

new_check_design *Creating new Check Design Objects*

Description

Instantiate a check design from a path or directory.

Usage

```
new_check_design(...)

new_rev_dep_check_design(x, ...)
```

Arguments

...	Additional arguments passed to new_check_design()
x	A file path, passed to rev_dep_check_tasks_df()

See Also

Other checks: check_design, check_dev_rev_deps(), check_dir(), check_pkgs(), check_rev_deps()

Other checks: check_design, check_dev_rev_deps(), check_dir(), check_pkgs(), check_rev_deps()

`options` *checked Options*

Description

Internally used, package-specific options. All options will prioritize R options() values, and fall back to environment variables if undefined. If neither the option nor the environment variable is set, a default value is used.

Checking Option Values

Option values specific to `checked` can be accessed by passing the package name to `env`.

```
options::opts(env = "checked")

options::opt(x, default, env = "checked")
```

Options

tty_tick_interval tty refresh interval when reporting results in milliseconds

default: `0.1`

option: checked.tty_tick_interval

envvar: R_CHECKED_TTY_TICK_INTERVAL (evaluated if possible, raw string otherwise)

results_error_on character vector indicating whether R error should be thrown when issues are discovered when generating results. "never" means that no errors are thrown. If "issues" then errors are emitted only on issues, whereas "potential issues" stands for error on both issues and potential issues.

default: `"never"`

option: checked.results_error_on

envvar: R_CHECKED_RESULTS_ERROR_ON (evaluated if possible, raw string otherwise)

results_keep character vector indicating which packages should be included in the results. "all" means that all packages are kept. If "issues" then only packages with issues identified, whereas "potential_issues" stands for keeping packages with both "issues" and "potential_issues".

default: `"all"`

option: checked.results_keep

envvar: R_CHECKED_RESULTS_KEEP (evaluated if possible, raw string otherwise)

restore `logical` indicating whether output directory should be unlinked before running checks. If `FALSE`, an attempt will me made to restore previous progress from the same `output`

default: `NA`

option: checked.restore

envvar: R_CHECKED_RESTORE (evaluated if possible, raw string otherwise)

check_envvars named `character` vector of environment variables to use during R CMD check.

default: `c(`_R_CHECK_FORCE_SUGGESTS_` = FALSE, `_R_CHECK_RD_XREFS_` = FALSE, `_R_CHECK_SYSTEM_CLOCK_` = FALSE, `_R_CHECK_SUGGESTS_ONLY_` = TRUE)`

option: checked.check_envvars

envvar: R_CHECKED_CHECK_ENVVARS (evaluated if possible, raw string otherwise)

check_build_args `character` vector of args passed to the R CMD build.

default: `c("--no-build-vignettes", "--no-manual")`

option: checked.check_build_args

envvar: R_CHECKED_CHECK_BUILD_ARGS (space-separated R CMD build flags)

check_args `character` vector of args passed to the R CMD check.

default: `c("--timings", "--ignore-vignettes", "--no-manual")`

option: checked.check_args

envvar: R_CHECKED_CHECK_ARGS (space-separated R CMD check flags)

See Also

options getOption Sys.setenv Sys.getenv

Other documentation: `options_params`

`options_params` *Checked Options*

Description

Checked Options

Arguments

`results_error_on` character vector indicating whether R error should be thrown when issues are discovered when generating results. "never" means that no errors are thrown. If "issues" then errors are emitted only on issues, whereas "potential issues" stands for error on both issues and potential issues. (Defaults to `"never"`, overwritable using option 'checked.results_error_on' or environment variable 'R_CHECKED_RESULTS_ERROR_ON')

`check_args` `character` vector of args passed to the R CMD check. (Defaults to `c("--timings", "--ignore-vignettes", "--no-manual")`, overwritable using option 'checked.check_args' or environment variable 'R_CHECKED_CHECK_ARGS')

`results_keep` character vector indicating which packages should be included in the results. "all" means that all packages are kept. If "issues" then only packages with issues identified, whereas "potential_issues" stands for keeping packages with both "issues" and "potential_issues". (Defaults to `"all"`, overwritable using option 'checked.results_keep' or environment variable 'R_CHECKED_RESULTS_KEEP')

`check_envvars` named `character` vector of environment variables to use during R CMD check. (Defaults to `c(`*R_CHECK_FORCE_SUGGESTS*`= FALSE,`*R_CHECK_RD_XREFS*`= FALSE, ;` *R_CHE* overwritable using option 'checked.check_envvars' or environment variable 'R_CHECKED_CHECK_EN

`tty_tick_interval`
tty refresh interval when reporting results in milliseconds (Defaults to `0.1`, overwritable using option 'checked.tty_tick_interval' or environment variable 'R_CHECKED_TTY_TICK_IN

`check_build_args`
`character` vector of args passed to the R CMD build. (Defaults to `c("--no-build-vignettes", "--no-manual")`, overwritable using option 'checked.check_build_args' or environment variable 'R_CHECKED_CHECK_BUILD_ARGS')

`restore` `logical` indicating whether output directory should be unlinked before running checks. If `FALSE`, an attempt will me made to restore previous progress from the same `output` (Defaults to `NA`, overwritable using option 'checked.restore' or environment variable 'R_CHECKED_RESTORE')

See Also

Other documentation: `options()`

`package_spec` *Package specification*

Description

Create package specification list which consists of all the details required to identify and acquire source of the package.

Usage

```
package_spec(name = NULL, repos = NULL)

package_spec_source(path = NULL, ...)

package_spec_archive_source(path = NULL, ...)
```

Arguments

`name` name of the package.

`repos` repository where package with given name should identified.

`path` path to the source of the package (either bundled or not). URLs are acceptable.

`...` parameters passed to downstream constructors

print.checked_results *Print checked results*

Description

Print checked results

Usage

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

## S3 method for class 'checked_results_check_task_spec'
print(x, keep = options::opt("results_keep"), ...)

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

Arguments

x	an object to be printed.
...	other parameters.
keep	character vector indicating which packages should be included in the results. "all" means that all packages are kept. If "issues" then only packages with issues identified, whereas "potential_issues" stands for keeping packages with both "issues" and "potential_issues". (Defaults to "all", overwritable using option 'checked.results_keep' or environment variable 'R_CHECKED_RESULTS_KEEP')

See Also

Other results: results(), results_to_file()

reporters *Check Design Runner Reporters*

Description

Reporters are used to configure how output is communicated while running a check_design. They range from glossy command-line tools intended for displaying progress in an interactive R session, to line-feed logs which may be better suited for automated execution, such as in continuous itegration.

Usage

```
reporter_ansi_tty()

reporter_basic_tty()

reporter_default()
```

Details

`reporter_default()`:
Automatically chooses an appropriate reporter based on the calling context.

`reporter_ansi_tty()`:
Highly dynamic output for fully capable terminals. Requires multi-line dynamic output, which may not be available in editors that that present a terminal as a web component.

`reporter_basic_tty()`:
A line-feed reporter presenting output one line at a time, providing a reporter with minimal assumptions about terminal capabilities.

---

results *Check results*

Description

Get R CMD check results

Usage

```
results(x, ...)

## S3 method for class 'check_design'
results(x, error_on = options::opt("results_error_on"), ...)
```

Arguments

x	`check_design` object.
...	other parameters.
error_on	character vector indicating whether R error should be thrown when issues are discovered when generating results. "never" means that no errors are thrown. If "issues" then errors are emitted only on issues, whereas "potential issues" stands for error on both issues and potential issues. (Defaults to "never", overwritable using option 'checked.results_error_on' or environment variable 'R_CHECKED_RESULTS_ERROR_ON')

See Also

Other results: `print.checked_results()`, `results_to_file()`

results_to_file *Results to file*

Description

Write checked_results object to the text file. When converting results to text, print.checked_results method is used.

Usage

```
results_to_file(results, file, keep = "all", ...)
```

Arguments

results	results object.
file	A connection or character path.
keep	character vector indicating which packages should be included in the results. "all" means that all packages are kept. If "issues" then only packages with issues identified, whereas "potential_issues" stands for keeping packages with both "issues" and "potential_issues". (Defaults to "all", overwritable using option 'checked.results_keep' or environment variable 'R_CHECKED_RESULTS_KEEP')
...	other parameters.

See Also

Other results: print.checked_results(), results()

revdep_check_task_spec *Create a task to run reverse dependency checks*

Description

Create a task to run reverse dependency checks

Usage

```
revdep_check_task_spec(revdep, ...)
```

Arguments

revdep	character indicating whether the task specification describes check associated with the development (new) or release (old) version of the for which reverse dependency check is run.
...	Additional parameters passed to task_spec()

See Also

Other tasks: check_task_spec(), checked-task-df, custom_install_task_spec(), install_task_spec(), rev_dep_check_tasks_df(), source_check_tasks_df(), task_spec()

---

rev_dep_check_tasks_df

Build Tasks for Reverse Dependency Checks Generates checks schedule data.frame appropriate for running reverse dependency check for certain source package. In such case path *parameter should point to the source of the development version of the package and* repos *should be a repository for which reverse dependencies should be identified.*

Description

Create data.frame which each row defines a package for which R CMD check should be run. Such data.frame is a prerequisite for generating check_design() which orchestrates all the processes including dependencies installation.

Usage

```
rev_dep_check_tasks_df(
  path,
  repos = getOption("repos"),
  versions = c("dev", "release"),
  ...
)
```

Arguments

path	path to the package source. Can be either a single source code directory or a directory containing multiple package source code directories.
repos	repository used to identify reverse dependencies.
versions	character vector indicating against which versions of the package reverse dependency should be checked. c("dev", "release") (default) stands for the classical reverse dependency check. "dev" checks only against development version of the package which is applicable mostly when checking whether adding new package would break tests of packages already in the repository and take the package as suggests dependency.
...	parameters passed to the task specs allowing to customize subprocesses.

Details

_tasks_df() functions generate check task data.frame for all source packages specified by the path. Therefore it accepts it to be a vector of an arbitrary length.

Value

The check schedule `data.frame` with the following columns:

- `alias`: The alias of the check to run. It also serves the purpose of providing a unique identifier and node name in the task graph.
- `version`: Version of the package to be checked.
- `package`: Object that inherits from `check_task_spec()`. Defines how package to be checked can be acquired.
- `custom`: Object that inherits from `custom_install_task_spec()`. Defines custom package, for instance only available from local source, that should be installed before checking the package.

See Also

Other tasks: `check_task_spec()`, `checked-task-df`, `custom_install_task_spec()`, `install_task_spec()`, `revdep_check_task_spec()`, `source_check_tasks_df()`, `task_spec()`

---

run *Run a Series of* `R CMD check`*s*

Description

`run()` provides a generic, and is the central interface for executing `check_design`s. If a path is provided, a new reverse dependency check plan is generated from the source code path. Otherwise a plan can be built separately and executed using `run()`.

Usage

```
run(design, ..., reporter = reporter_default())
```

Arguments

`design`	`character` or `check_design` If a `character` value is provided, it is first coerced into a `check_design` using `new_rev_dep_check_design()`.
`...`	Additional arguments passed to `new_rev_dep_check_design()`
`reporter`	A reporter to provide progress updates. Will default to the most expressive command-line reporter given your terminal capabilities.

`source_check_tasks_df` *Create a Task to Check a Package from Source*

Description

Create data.frame which each row defines a package for which R CMD check should be run. Such data.frame is a prerequisite for generating `check_design()` which orchestrates all the processes including dependencies installation.

Usage

```
source_check_tasks_df(path, ...)
```

Arguments

`path`	path to the package source. Can be either a single source code directory or a directory containing multiple package source code directories.
`...`	parameters passed to the task specs allowing to customize subprocesses.

Details

`_tasks_df()` functions generate check task `data.frame` for all source packages specified by the `path`. Therefore it accepts it to be a vector of an arbitrary length.

Value

The check schedule `data.frame` with the following columns:

- `alias`: The alias of the check to run. It also serves the purpose of providing a unique identifier and node name in the task graph.
- `version`: Version of the package to be checked.
- `package`: Object that inherits from `check_task_spec()`. Defines how package to be checked can be acquired.
- `custom`: Object that inherits from `custom_install_task_spec()`. Defines custom package, for instance only available from local source, that should be installed before checking the package.

See Also

Other tasks: `check_task_spec()`, `checked-task-df`, `custom_install_task_spec()`, `install_task_spec()`, `rev_dep_check_tasks_df()`, `revdep_check_task_spec()`, `task_spec()`

task_spec *Task specification*

Description

Create task specification list which consists of all the details required to run specific task.

Usage

```
task_spec(
  alias = NULL,
  package_spec = NULL,
  env = options::opt("check_envvars")
)
```

Arguments

alias	task alias which also serves as unique identifier of the task.
package_spec	package_spec object
env	environmental variables to be set in separate process running specific task.

See Also

Other tasks: check_task_spec(), checked-task-df, custom_install_task_spec(), install_task_spec(), rev_dep_check_tasks_df(), revdep_check_task_spec(), source_check_tasks_df()

Index