

Package ‘checked’

July 26, 2024

Title Systematically Run R CMD Checks

Version 0.2.0

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://github.com/Genentech/checked>

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

License MIT + file LICENSE

Encoding UTF-8

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

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

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 2024-07-26 02:20:02 UTC

Contents

checked-package	2
ansi	3
checks_capture	3
checks_df	4
checks_simplify	5

check_design	6
check_functions	8
devnull	10
package_spec	10
print.checked_results	11
results	12
run	12
silent_spinner	13
task_graph_create	13
task_graph_neighborhoods	14
task_graph_sort	14
task_graph_which_satisfied	15
task_spec	16
throttle	17

Index	18
--------------	-----------

checked-package	<i>checked: Systematically Run R CMD Checks</i>
-----------------	---

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.

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.

Author(s)

Maintainer: Szymon Maksymiuk <sz.maksymiuk@gmail.com> ([ORCID](#))

Authors:

- Doug Kelkhoff <doug.kelkhoff@gmail.com> ([ORCID](#))

Other contributors:

- F. Hoffmann-La Roche AG [copyright holder, funder]

See Also

Useful links:

- <https://github.com/Genentech/checked>
- Report bugs at <https://github.com/Genentech/checked/issues>

Useful links:

- <https://github.com/Genentech/checked>
- Report bugs at <https://github.com/Genentech/checked/issues>

`ansi`*Various utilities for formatting ANSI output*

Description

Various utilities for formatting ANSI output

Usage

```
ansi_line_erase(n = "")
```

```
ansi_move_line_rel(n)
```

Arguments

`n` The number of lines to move. Positive is up, negative is down.

Functions

- `ansi_line_erase()`: Erase the current line
- `ansi_move_line_rel()`: Offset the cursor by a relative number of lines

`checks_capture`*Parse R CMD checks from a partial check output string*

Description

Parse R CMD checks from a partial check output string

Usage

```
checks_capture(x)
```

Arguments

`x` A string, composed of any subsection of R CMD check console output

Value

A matrix of matches and capture groups "check" and "status" ("OK", "NONE", "NOTE", "WARNING" or "ERROR").

Examples

```

check_output <- "
* checking check one ... OK
* checking check two ... NOTE
* checking tests ...
  Running test_abc.R
  Running test_xyz.R
  NONE
* checking check three ... WARNING
* ch
"

checks_capture(check_output)

```

checks_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.

Usage

```

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

source_check_tasks_df(path)

```

Arguments

path	path to the package source. See Details .
repos	repository used to identify reverse dependencies.
versions	character vector indicating against which versions of the package reverse dependency should be checked. <code>c("dev", "release")</code> (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.

Details

rev_dep_check_tasks_df 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.

source_check_tasks_df generates checks schedule 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 has strict structure and consists of following columns:

- `alias` The alias of the check to run. It also serves the purpose of 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.

checks_simplify	<i>Simplify Captures into Vector</i>
-----------------	--------------------------------------

Description

Simplify Captures into Vector

Usage

```
checks_simplify(x)
```

Arguments

`x` Matrix of regex captures as produced by `checks_capture`.

Value

A vector of check status, with names indicating the check

check_design	<i>Check Design Object</i>
--------------	----------------------------

Description

Abstract object that drives all separate processes required to run R CMD check 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 data.frame which is the source of all the checks Created with [source_check_tasks_df](#)

`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 = TRUE,
  ...
)
```

Arguments:

`df` checks 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 be made to restore previous progress from the same output
... other parameters

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()`: Terminate 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.

Examples

```
## Not run:
library(checkeds)
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_functions

Check functions

Description

Set of functions to run orchestrated R CMD checks and automatically manage the dependencies installation. Each functions prepares the plan based on the supplied package source(s) which includes installing dependencies and running required R CMD checks. All the functions are parallelized through sperate processes

Usage

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

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



```

    ...
)

check_packages(
  path,
  n = 2L,
  output = tempfile(paste(utils::packageName(), Sys.Date(), sep = "-")),
  lib.loc = .libPaths(),
  repos = getOption("repos"),
  restore = TRUE,
  reporter = default_reporter(),
  ...
)

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

```

Arguments

path	path to the package source.
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 value, whether output directory should be unlinked before running checks. If FALSE, an attempt will be made to restore previous progress from the same output
reporter	A reporter to provide progress updates. Will default to the most expressive command-line reporter given your terminal capabilities.
...	other parameters

Details

`check_reverse_dependencies` runs classical reverse dependency check for the given source package. It first identifies reverse dependencies available in repos. Then, after installing all required dependencies, it runs the R CMD check twice for each package, one time with the release version of the package and the second time with the development version. Both R CMD checks are later compared to get the result.

`check_reverse_dependencies_development` works similarly to `check_reverse_dependencies` 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 repository breaks existing packages that possibly take said package as Suggests dependency.

`check_packages` Installs all dependencies and runs parallely R CMD checks for all source packages specified by path parameter

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

Value

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

<code>devnull</code>	<i>Reuse or Create A Null File Connection</i>
----------------------	---

Description

Reuse or Create A Null File Connection

Usage

```
devnull()
```

<code>package_spec</code>	<i>Package specification</i>
---------------------------	------------------------------

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 Plot checked results
```

Description

Plot checked results

Usage

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

## S3 method for class 'checked_results_check_task_spec'
print(
  x,
  keep = Sys.getenv("CHECKED_RESULTS_KEEP", c("all", "issues", "potential_issues")[1]),
  ...
)

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

Arguments

x	an object to be printed.
...	other parameters described below
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". Users can set the default value via env variable CHECKED_RESULTS_KEEP.

results	<i>Check results</i>
---------	----------------------

Description

Get R CMD check results

Usage

```
results(x, ...)

## S3 method for class 'check_design'
results(
  x,
  error_on = Sys.getenv("CHECKED_RESULTS_ERROR_ON", c("never", "issues",
    "potential_issues")[1]),
  ...
)
```

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. Users can set the default value via env variable CHECKED_RESULTS_ERROR_ON.

run	<i>Run Reverse-Dependency Checks</i>
-----	--------------------------------------

Description

Run Reverse-Dependency Checks

Usage

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

Arguments

design	A reverse-dependency plan, or an object coercible into a plan.
...	Additional arguments
reporter	A reporter to provide progress updates. Will default to the most expressive command-line reporter given your terminal capabilities.

silent_spinner	<i>Create a 'cli' Spinner With Suppressed Output</i>
----------------	--

Description

'cli' will implicitly push spinner output to various output streams, affecting the terminal cursor position. To allow for a terminal interface that has spinners above the last line, this function suppresses the output and simply returns its frame contents.

Usage

```
silent_spinner(..., stream = devnull())
```

Arguments

...	passed to cli::make_spinner
stream	passed to cli::make_spinner , defaults to a null file device

Value

A interface similar to a 'cli' spinner, but with suppressed output

task_graph_create	<i>Create Task Graph</i>
-------------------	--------------------------

Description

Create Task Graph

Usage

```
task_graph_create(df, repos = getOption("repos"))
```

Arguments

df	data.frame listing
repos	repositories which will be used to identify dependencies chain to run R CMD checks

Value

A dependency graph with vertex attributes "root" (a logical value indicating whether the package as one of the roots used to create the graph), "status" (installation status) and "order" (installation order).

task_graph_neighborhoods
Find Task Neighborhood

Description

Find Task Neighborhood

Usage

```
task_graph_neighborhoods(g, nodes)
```

Arguments

g	A task graph, as produced with task_graph_create()
nodes	Names or nodes objects of packages whose neighborhoods should be calculated.

task_graph_sort *Sort Task Graph by Strong Dependency Order*

Description

Sort Task Graph by Strong Dependency Order

Usage

```
task_graph_sort(g)
```

Arguments

g	A igraph::graph , expected to contain node attribute type.
---	--

Value

The [igraph::graph](#) g, with vertices sorted in preferred installation order.

Note

Cyclic dependencies are possible. Cyclic dependencies are disallowed for all hard dependencies on CRAN today, though there have been historical instances where they appeared on CRAN.

Installation priority is based on:

1. Total dependency footprint (low to high)
2. Topology (leaf nodes first)

`task_graph_which_satisfied`*Find the Next Packages Not Dependent on an Unavailable Package*

Description

While other packages are in progress, ensure that the next selected package already has its dependencies done.

Usage

```
task_graph_which_satisfied(  
  g,  
  v = igraph::V(g),  
  dependencies = TRUE,  
  status = STATUS$pending  
)  
  
task_graph_which_satisfied_strong(..., dependencies = "strong")  
  
task_graph_which_check_satisfied(  
  g,  
  ...,  
  dependencies = "all",  
  status = STATUS$pending  
)  
  
task_graph_which_install_satisfied(  
  g,  
  ...,  
  dependencies = "strong",  
  status = STATUS$pending  
)
```

Arguments

<code>g</code>	A dependency graph, as produced with <code>task_graph_create()</code> .
<code>v</code>	Names or nodes objects of packages whose satisfiability should be checked.
<code>dependencies</code>	Which dependencies types should be met for a node to be considered satisfied.
<code>status</code>	status name. Nodes in <code>v</code> will be filtered to consists only nodes with that status.
<code>...</code>	parametrs passed to down-stream functions.

Details

There are helpers defined for particular use cases that strictly rely on the `task_graph_which_satisfied`, they are:

task_graph_which_satisfied_strong - List vertices whose strong dependencies are satisfied.

task_graph_which_check_satisfied - List root vertices whose all dependencies are satisfied.

task_graph_which_install_satisfied - List install vertices whose dependencies are all satisfied

Value

The name of the next package to prioritize

task_spec	<i>Task specification</i>
-----------	---------------------------

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 = NULL)
```

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

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

```
check_task_spec(args = NULL, build_args = NULL, ...)
```

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

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.
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., <code>c("--html", "--no-multiarch", "--no-test-load")</code> . Can also be a named list of character vectors to be used as additional options, with names the respective package names.
...	parameters passed to downstream constructors

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 <code>args = c("--no-examples", "--no-tests")</code> and not <code>args = "--no-examples --no-tests"</code> . (Note that instead of the <code>--output</code> option you should use the <code>check_dir</code> argument, because <code>--output</code> 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, <code>build_args = c("--force", "--keep-empty-dirs")</code> is a correct usage and <code>build_args = "--force --keep-empty-dirs"</code> is incorrect.
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.

 throttle

Generate A Rate Limiting Throttle Function

Description

Generate A Rate Limiting Throttle Function

Usage

```
throttle(interval = 0.2)
```

Arguments

interval	An interval (in seconds) that is the minimum interval before <code>throttle</code> will return TRUE.
----------	--

Value

A throttling function with the provided interval. When called, returns a logical value indicating whether the throttle interval has passed (TRUE if the interval has not yet passed).

Index

ansi, 3
ansi_line_erase (ansi), 3
ansi_move_line_rel (ansi), 3

check_design, 4, 6, 7, 10, 12
check_dir (check_functions), 8
check_functions, 8
check_packages (check_functions), 8
check_reverse_dependencies
 (check_functions), 8
check_reverse_dependencies_development
 (check_functions), 8
check_task_spec, 5
check_task_spec (task_spec), 16
checked-package, 2
checks_capture, 3, 5
checks_df, 4
checks_simplify, 5
cli::make_spinner, 13
custom_install_task_spec, 5
custom_install_task_spec (task_spec), 16

devnull, 10

igraph::graph, 14
install_task_spec (task_spec), 16

package_spec, 10, 16
package_spec_archive_source
 (package_spec), 10
package_spec_source (package_spec), 10
print.checked_results, 11
print.checked_results_check_task_spec
 (print.checked_results), 11
print.checked_results_revdep_check_task_spec
 (print.checked_results), 11

results, 10, 12
rev_dep_check_tasks_df (checks_df), 4
revdep_check_task_spec (task_spec), 16
run, 12

silent_spinner, 13
source_check_tasks_df, 6
source_check_tasks_df (checks_df), 4

task_graph_create, 6, 13
task_graph_create(), 14, 15
task_graph_neighborhoods, 14
task_graph_sort, 14
task_graph_which_check_satisfied
 (task_graph_which_satisfied),
 15
task_graph_which_install_satisfied
 (task_graph_which_satisfied),
 15
task_graph_which_satisfied, 15
task_graph_which_satisfied_strong
 (task_graph_which_satisfied),
 15
task_spec, 16
throttle, 17, 17