

# Package ‘checked’

February 25, 2026

**Title** Systematically Run R CMD Checks

**Version** 0.5.1

**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

**Depends** R (>= 3.6.2)

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

**RoxygenNote** 7.3.3

**Suggests** remotes, testthat (>= 3.0.0), visNetwork, withr

**Config/Needs/website** r-lib/asciicast

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Szymon Maksymiuk [cre, aut] (ORCID:  
<<https://orcid.org/0000-0002-3120-1601>>),  
Doug Kelkhoff [aut] (ORCID: <<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** 2026-02-25 21:20:26 UTC

## Contents

checker	2
check_pkgs	5
check_rev_deps	6
check_task	7
install_task	8
lib_path	9
meta_task	9
new_checker	10
options	10
options_params	12
pkg_origin	13
plan_local_checks	14
plan_local_install	15
plan_rev_dep_checks	15
print.checked_results	16
reporters	17
results	17
results_to_df	18
run	19
STATUS	19
task	20
<b>Index</b>	<b>21</b>

---

checker *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()`

`plan` (`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:

- [checker\\$new\(\)](#)
- [checker\\$active\\_processes\(\)](#)
- [checker\\$failed\\_tasks\(\)](#)
- [checker\\$terminate\(\)](#)
- [checker\\$step\(\)](#)
- [checker\\$start\\_next\\_task\(\)](#)
- [checker\\$is\\_done\(\)](#)
- [checker\\$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:

```
checker$new(
  plan,
  n = 2L,
  output = file.path(tempdir(), paste(packageName(), Sys.Date(), sep = "-")),
  lib.loc = .libPaths(),
  repos = getOption("repos"),
  restore = options::opt("restore"),
  ...
)
```

#### Arguments:

plan plan 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.

... Additional arguments unused

Returns: [checker](#).

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

#### Usage:

```
checker$active_processes()
```

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

#### Usage:

```
checker$failed_tasks()
```

**Method** `terminate()`: Kill All Active Design Processes  
Immediately terminates all the active processes.

*Usage:*

```
checker$terminate()
```

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

*Usage:*

```
checker$step()
```

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

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

*Usage:*

```
checker$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:*

```
checker$is_done()
```

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

## See Also

Other checks: [STATUS](#), [check\\_pkgs\(\)](#), [check\\_rev\\_deps\(\)](#), [new\\_checker\(\)](#)

## Examples

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

orchestrator <- checker$new(
  plan,
  n = 10,
  repos = "https://cran.r-project.org/"
)
```

```

while (!orchestrator$is_done()) {
  orchestrator$start_next_task()
}

## End(Not run)

```

---

check\_pkgs

*Check packages*


---

### Description

Runs classical R CMD check for the given source package. It first identifies and installs, in parallel, all dependencies required to check the package. Then, it runs R CMD check for each specified package.

### Usage

```

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

```

### Arguments

package	A path to either package, directory with packages or name of the package (details)
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 <code>.libPaths()</code> .
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 be made to restore previous progress from the same output
...	Additional arguments passed to <code>run()</code>

**Value**

`checker()` 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: `STATUS`, `check_rev_deps()`, `checker`, `new_checker()`

---

check_rev_deps	<i>Check reverse dependencies</i>
----------------	-----------------------------------

---

**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 = TRUE,
  ...
)
```

**Arguments**

<code>path</code>	file path to the package source directory
<code>n</code>	integer value indicating maximum number of subprocesses that can be simultaneously spawned when executing tasks.
<code>output</code>	character value specifying path where the output should be stored.
<code>lib.loc</code>	character vector with libraries allowed to be used when checking packages, defaults to entire <code>.libPaths()</code> .
<code>repos</code>	character vector of repositories which will be used when generating task graph and later pulling dependencies.
<code>reverse_repos</code>	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.
<code>restore</code>	logical indicating whether output directory should be unlinked before running checks. If FALSE, an attempt will be made to restore previous progress from the same output
<code>...</code>	Additional arguments passed to <code>run()</code>

## 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

`checker()` 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: `STATUS`, `check_pkgs()`, `checker`, `new_checker()`

---

check_task	<i>Create a task to run R CMD check</i>
------------	---

---

## Description

Create a task to run R CMD check

## Usage

```
check_task(build_args = NULL, args = NULL, env = NULL, ...)
```

## Arguments

<code>build_args</code>	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.
<code>args</code>	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.)
<code>env</code>	A named character vector, extra environment variables to set in the check process.
<code>...</code>	Arguments passed on to <code>task</code> .subclass Additional subclasses.

**See Also**

Other tasks: [install\\_task\(\)](#), [meta\\_task\(\)](#), [task\(\)](#)

---

 install\_task

*Create a task to install a package and dependencies*


---

**Description**

Create a task to install a package and dependencies

**Usage**

```
install_task(
  origin,
  type = package_install_type(origin),
  INSTALL_opts = NULL,
  lib = lib_path(origin),
  ...
)
```

**Arguments**

origin	<a href="#">pkg_origin()</a> object.
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> or, for macOS, <code>"--dsym"</code> . Can also be a named list of character vectors to be used as additional options, with names the respective package names.
lib	Any object that can be passed to <a href="#">lib()</a> to generate a library path.
...	further arguments to be passed to <a href="#">download.file</a> , <a href="#">available.packages</a> , or to the functions for binary installs on macOS and Windows (which accept an argument "lock": see the section on 'Locking').

**See Also**

Other tasks: [check\\_task\(\)](#), [meta\\_task\(\)](#), [task\(\)](#)

---

lib_path	<i>Make a Library Location</i>
----------	--------------------------------

---

**Description**

A description of where packages should be installed. This object provides necessary information to determine where a package should be installed. `lib_path` method creates default path handlers for given pkg origin while `lib_path_x` creates an actual object.

**Usage**

```
lib_path(x, ..., .class = c())
```

```
lib_path_default(.class = c())
```

```
lib_path_isolated(.class = c())
```

**Arguments**

x	A <code>pkg_origin()</code> object used for default dispatch.
...	Additional values
.class	An optional subclass, used primarily for dispatch.

**See Also**

Other specs: [pkg\\_origin\(\)](#)

---

meta_task	<i>Construct a 'Meta' Task</i>
-----------	--------------------------------

---

**Description**

Meta tasks are tasks which are not intended to perform computation. They exist simply to provide relationships among computational tasks.

**Usage**

```
meta_task(..., .subclass = NULL)
```

**Arguments**

...	Objects passed to specified class functions
.subclass	character name of the subclass. It will be appended with "_meta" suffix.

**See Also**

Other tasks: [check\\_task\(\)](#), [install\\_task\(\)](#), [task\(\)](#)

---

new_checker	<i>Creating new Check Design Objects</i>
-------------	--

---

**Description**

Instantiate a check design from a path or directory.

**Usage**

```
new_checker(...)
```

```
new_rev_dep_checker(x, ...)
```

**Arguments**

...	Additional arguments passed to <a href="#">new_checker()</a>
x	A file path, passed to <a href="#">plan_rev_dep_checks()</a>

**See Also**

Other checks: [STATUS](#), [check\\_pkgs\(\)](#), [check\\_rev\\_deps\(\)](#), [checker](#)

Other checks: [STATUS](#), [check\\_pkgs\(\)](#), [check\\_rev\\_deps\(\)](#), [checker](#)

---

options	<i>checked Options</i>
---------	------------------------

---

**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)
- tty\_default\_height** default tty height used for the ANSI reporter. Used only if correct values could not be acquired with system('tput lines')  
**default:** 50  
**option:** checked.tty\_default\_height  
**envvar:** R\_CHECKED\_TTY\_DEFAULT\_HEIGHT (evaluated if possible, raw string otherwise)
- proactive\_gc** logical, indicating whether additional garbage collection should be performed before starting a new task, if at least one process recently finalized. This can cause the checker to orchestrate tasks slower but is recommended to be used for designs with many sub-processes required as native garbage collection can lag leading to memory issues. Disable only when maximum performance is required and memory is not the issue.  
**default:** TRUE  
**option:** checked.proactive\_gc  
**envvar:** R\_CHECKED\_PROACTIVE\_GC (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 be made to restore previous progress from the same output  
**default:** NA  
**option:** checked.restore  
**envvar:** R\_CHECKED\_RESTORE (evaluated if possible, raw string otherwise)
- add\_remotes** logical indicating whether origins inheriting from pkg\_origin\_local, should be scanned for packages in the remotes field and added while constructing a plan  
**task\_grap**

**default:** TRUE

**option:** checked.add\_remotes

**envvar:** R\_CHECKED\_ADD\_REMOTES (evaluated if possible, raw string otherwise)

**check\_envvars** named character vector of environment variables to use during the 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", `_R_CHECK_CRAN_INCOMING_` = "false")`

**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", "--as-cran")`

**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

`proactive_gc` logical, indicating whether additional garbage collection should be performed before starting a new task, if at least one process recently finalized. This can cause the checker to orchestrate tasks slower but is recommended to be used for designs with many sub-processes required as native garbage collection can lag leading to memory issues. Disable only when maximum performance is required and memory is not the issue. (Defaults to TRUE, overwritable using option 'checked.proactive\_gc' or environment variable 'R\_CHECKED\_PROACTIVE\_GC')

`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", "--as-cran"), 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')
add_remotes	logical indicating whether origins inheriting from pkg_origin_local, should be scanned for packages in the remotes field and added while constructing a plan task_grap (Defaults to TRUE, overwritable using option 'checked.add_remotes' or environment variable 'R_CHECKED_ADD_REMOTES')
check_envvars	named character vector of environment variables to use during the R CMD check. (Defaults to c(R_CHECK_FORCE_SUGGESTS= "false", R_CHECK_RD_XREFS= "false", overwritable using option 'checked.check_envvars' or environment variable 'R_CHECKED_CHECK_ENVVARS')
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_INTERVAL')
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 be made to restore previous progress from the same output (Defaults to NA, overwritable using option 'checked.restore' or environment variable 'R_CHECKED_RESTORE')
tty_default_height	default tty height used for the ANSI reporter. Used only if correct values could not be acquired with system('tput lines') (Defaults to 50, overwritable using option 'checked.tty_default_height' or environment variable 'R_CHECKED_TTY_DEFAULT_HEIGHT')

**See Also**

Other documentation: [options\(\)](#)

---

pkg\_origin

*Package specification*

---

**Description**

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

**Usage**

```

pkg_origin(package, ..., .class = c())

pkg_origin_repo(package, repos, ...)

pkg_origin_is_base(package, ...)

pkg_origin_base(package, ...)

pkg_origin_unknown(package, ...)

pkg_origin_local(path = NULL, ...)

pkg_origin_remote(remote = NULL, ...)

pkg_origin_archive(path = NULL, ...)

```

**Arguments**

package	name of the package.
...	parameters passed to downstream constructors.
.class	Additional subclasses.
repos	repository where package with given name should identified.
path	path to the source of the package (either bundled or not). URLs are acceptable.
remote	remote object from the remotes package used to identify non-standard packages.

**See Also**

Other specs: [lib\\_path\(\)](#)

---

plan\_local\_checks      *Plan R CMD Checks*

---

**Description**

Generates a plan for running R CMD check for a specified set of packages.

**Usage**

```
plan_local_checks(package, repos = getOption("repos"))
```

**Arguments**

package	A path to either package, directory with packages or name of the package (details)
repos	repository used to identify packages when name is provided.

**Details**

package parameter has two different allowed values:

- Package - checked looks for a DESCRIPTION file in the provided path, if found treats it like a source package.
- If the specified value does not correspond to a source package, the parameter is treated as the name and repos parameter is used to identify the source.

**See Also**

Other plan: [plan\\_local\\_install\(\)](#), [plan\\_rev\\_dep\\_checks\(\)](#)

---

plan\_local\_install      *Plan source package installation*

---

**Description**

Generates a plan for running installing a package from source.

**Usage**

```
plan_local_install(package, repos = getOption("repos"))
```

**Arguments**

package	A path to package source.
repos	repository used to identify packages when name is provided.

**See Also**

Other plan: [plan\\_local\\_checks\(\)](#), [plan\\_rev\\_dep\\_checks\(\)](#)

---

plan\_rev\_dep\_checks      *Plan Reverse Dependency Checks*

---

**Description**

Generates a plan for running reverse dependency check for certain source package. In such case path should be provided with a directory path to the development version of the package and repos should be a repository for which reverse dependencies should be identified.

**Usage**

```
plan_rev_dep_checks(path, repos = getOption("repos"))
```

**Arguments**

path            path to the package source.  
 repos          repository used to identify reverse dependencies.

**See Also**

Other plan: [plan\\_local\\_checks\(\)](#), [plan\\_local\\_install\(\)](#)

---

`print.checked_results` *Print checked results*

---

**Description**

Print checked results

**Usage**

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

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

## S3 method for class 'local_check_results'
print(x, ..., name = NULL, keep = options::opt("results_keep"))
```

**Arguments**

x              an object to be printed.  
 ...            other parameters.  
 name          character name of the rev\_dep\_dep package  
 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\\_df\(\)](#)

---

reporters

*Check checker Runner Reporters*

---

### Description

Reporters are used to configure how output is communicated while running a [checker](#). 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 integration.

### Usage

`reporter_ansi_tty()`

`reporter_ansi_tty2()`

`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 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 'checker'
results(x, error_on = options::opt("results_error_on"), ...)

## S3 method for class 'integer'
results(x, checker_obj, ...)

## S3 method for class 'igraph.vs'
results(x, ...)

## S3 method for class 'rev_dep_dep_meta_task'
results(x, checker_obj, ...)

## S3 method for class 'rev_dep_check_meta_task'
results(x, checker_obj, ...)

## S3 method for class 'local_check_meta_task'
results(x, checker_obj, ...)

```

**Arguments**

x	object which results should be presented.
...	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')
checker_obj	<a href="#">checker</a> object.

**See Also**

Other results: [print.checked\\_results\(\)](#), [results\\_to\\_df\(\)](#)

---

results\_to\_df

*Summarize checked results as data.frame*

---

**Description**

Turns checked results into a list of data.frames, one for each meta root task. Such form makes quick results analysis easier by providing a general overview of identified failures.

**Usage**

```
results_to_df(results, ...)
```

**Arguments**

`results` checked\_results object or any of the sub-objects.  
`...` other parameters passed to downstream functions.

**See Also**

Other results: [print.checked\\_results\(\)](#), [results\(\)](#)

---

run	<i>Run a Series of R CMD checks</i>
-----	-------------------------------------

---

**Description**

[run\(\)](#) provides a generic, and is the central interface for executing [checkers](#). 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(checker, ..., reporter = reporter_default())
```

**Arguments**

`checker` character or checker If a character value is provided, it is first coerced into a checker using [new\\_rev\\_dep\\_checker\(\)](#).  
`...` Additional arguments passed to [new\\_rev\\_dep\\_checker\(\)](#)  
`reporter` A reporter to provide progress updates. Will default to the most expressive command-line reporter given your terminal capabilities.

---

STATUS	<i>Check execution status categories</i>
--------	--

---

**Description**

Check execution status categories

**Usage**

```
STATUS
```

**Format**

An object of class `enum` (inherits from `factor`) of length 4.

**See Also**

Other checks: [check\\_pkgs\(\)](#), [check\\_rev\\_deps\(\)](#), [checker](#), [new\\_checker\(\)](#)

---

task

*Task specification*

---

**Description**

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

**Usage**

```
task(..., .subclass = NULL)
```

**Arguments**

...                    parameters passed to downstream constructors.  
.subclass            Additional subclasses.

**Details**

Tasks can be nested, representing either a singular task, or a set of related tasks.

**See Also**

Other tasks: [check\\_task\(\)](#), [install\\_task\(\)](#), [meta\\_task\(\)](#)

# Index

- \* **checks**
  - check\_pkgs, 5
  - check\_rev\_deps, 6
  - checker, 2
  - new\_checker, 10
  - STATUS, 19
- \* **datasets**
  - STATUS, 19
- \* **documentation**
  - options, 10
  - options\_params, 12
- \* **plan**
  - plan\_local\_checks, 14
  - plan\_local\_install, 15
  - plan\_rev\_dep\_checks, 15
- \* **reporters**
  - reporters, 17
- \* **results**
  - print.checked\_results, 16
  - results, 17
  - results\_to\_df, 18
- \* **specs**
  - lib\_path, 9
  - pkg\_origin, 13
- \* **tasks**
  - check\_task, 7
  - install\_task, 8
  - meta\_task, 9
  - task, 20
- .libPaths(), 5, 6
- available.packages, 8
- check\_pkgs, 4, 5, 7, 10, 20
- check\_rev\_deps, 4, 6, 6, 10, 20
- check\_task, 7, 8, 9, 20
- checker, 2, 3, 6, 7, 10, 17–20
- checker(), 6, 7
- download.file, 8
- install\_task, 8, 8, 9, 20
- lib(), 8
- lib\_path, 9, 14
- lib\_path\_default(lib\_path), 9
- lib\_path\_isolated(lib\_path), 9
- meta\_task, 8, 9, 20
- new\_checker, 4, 6, 7, 10, 20
- new\_checker(), 10
- new\_rev\_dep\_checker(new\_checker), 10
- new\_rev\_dep\_checker(), 19
- options, 10, 13
- options\_params, 12, 12
- pkg\_origin, 9, 13
- pkg\_origin(), 8, 9
- pkg\_origin\_archive(pkg\_origin), 13
- pkg\_origin\_base(pkg\_origin), 13
- pkg\_origin\_is\_base(pkg\_origin), 13
- pkg\_origin\_local(pkg\_origin), 13
- pkg\_origin\_remote(pkg\_origin), 13
- pkg\_origin\_repo(pkg\_origin), 13
- pkg\_origin\_unknown(pkg\_origin), 13
- plan\_local\_checks, 14, 15, 16
- plan\_local\_install, 15, 15, 16
- plan\_rev\_dep\_checks, 15, 15
- plan\_rev\_dep\_checks(), 10
- print.checked\_results, 16, 18, 19
- print.local\_check\_results
  - (print.checked\_results), 16
- print.rev\_dep\_dep\_results
  - (print.checked\_results), 16
- reporter\_ansi\_tty(reporters), 17
- reporter\_ansi\_tty(), 17
- reporter\_ansi\_tty2(reporters), 17
- reporter\_basic\_tty(reporters), 17
- reporter\_basic\_tty(), 17

reporter\_default (reporters), [17](#)  
reporter\_default(), [17](#)  
reporters, [17](#)  
results, [6](#), [7](#), [16](#), [17](#), [19](#)  
results\_to\_df, [16](#), [18](#), [18](#)  
run, [19](#)  
run(), [5](#), [6](#), [19](#)  
  
STATUS, [4](#), [6](#), [7](#), [10](#), [19](#)  
summary(), [6](#), [7](#)  
  
task, [7–9](#), [20](#)  
task\_graph(), [2](#)