

# Package: test.assessr (via r-universe)

September 16, 2026

**Title** Assessing Package Test Reliability and Quality

**Version** 2.1.3

**Description** A reliable and validated tool that calculates unit test coverage for R packages with standard testing frameworks and non-standard testing frameworks.

**License** GPL (>= 2)

**BugReports** <https://github.com/pharmaverse/test.assessr/issues>

**Depends** R (>= 4.1.0)

**Imports** callr, checkmate, covr (>= 3.6.0), dplyr, fs, jsonlite, pkgload, remotes, rlang, rmarkdown, RUnit, stringr (>= 1.5.0), testthat (>= 3.0.0), tinytest, tidyr, utils, withr

**Suggests** curl, devtools, DT, here, kableExtra (>= 1.4.0), knitr, methods, R6, S7, roxygen2, testit (>= 0.13), tidyselect, tools, mockery

**Encoding** UTF-8

**LazyData** true

**Config/testthat/edition** 3

**Config/build/clean-inst-doc** false

**VignetteBuilder** knitr

**Config/roxygen2/version** 8.0.0

**RoxygenNote** 8.0.0

**Config/pak/sysreqs** cmake git make libicu-dev libuv1-dev libssl-dev

**Repository** <https://pharmaverse.r-universe.dev>

**Date/Publication** 2026-08-13 12:29:31 UTC

**RemoteUrl** <https://github.com/pharmaverse/test.assessr>

**RemoteRef** HEAD

**RemoteSha** ff74ac15c36708ec5bfb55783f171dc6737326e9

Contents

|                                 |   |
|---------------------------------|---|
| extract_short_path . . . . .    | 2 |
| generate_test_report . . . . .  | 2 |
| get_package_coverage . . . . .  | 3 |
| get_pkg_name . . . . .          | 4 |
| install_package_local . . . . . | 5 |
| run_coverage . . . . .          | 6 |
| run_covr_modes . . . . .        | 7 |
| set_up_pkg . . . . .            | 9 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>10</b> |
|--------------|-----------|

---

|                    |                                                                                           |
|--------------------|-------------------------------------------------------------------------------------------|
| extract_short_path | <i>Helper to extract "R/&lt;file&gt;" from any path by taking the last two components</i> |
|--------------------|-------------------------------------------------------------------------------------------|

---

Description

Helper to extract "R/<file>" from any path by taking the last two components

Usage

extract\_short\_path(long\_file\_name)

Arguments

long\_file\_name A string containing the full file path (supports '/' or '\')

Value

A character string composed of the last two path components, e.g., "R/add.R"

---

|                      |                                                         |
|----------------------|---------------------------------------------------------|
| generate_test_report | <i>Generate HTML Report for Package Test Assessment</i> |
|----------------------|---------------------------------------------------------|

---

Description

Generates an HTML report for the package test assessment results using rmarkdown.

Usage

generate\_test\_report(test\_results, output\_dir = NULL)

**Arguments**

`test_results` List containing the results from `get_package_coverage` function.

`output_dir` (required) Character string path to an existing directory where the report will be saved.

**Value**

Path to the generated HTML report.

**Examples**

```
## Not run:
test_results <- get_package_coverage()
# Always provide a directory; tempdir() is convenient in examples.
generate_test_report(test_results, output_dir = tempdir())

## End(Not run)
```

---

`get_package_coverage`    *get package test coverage*

---

**Description**

simplified input to assess package for test coverage

**Usage**

```
get_package_coverage(path = NULL, package_installed = NULL)
```

**Arguments**

`path` (optional) path to locally stored package source code. If `package_installed = TRUE`, `path` must point to the unpacked local package directory (the package root containing files such as `DESCRIPTION`), not to a source tarball such as `.tar.gz`.

`package_installed` (optional) logical flag that controls local package installation. If `TRUE`, `install_package_local` is skipped and `path` is used directly. If `NULL` or `FALSE`, a local package installation is attempted.

**Value**

An object of class "coverage" as produced by `test.assessr::run_covr_modes()`. This object is a structured list containing detailed test coverage information, including:

- **File-level coverage:** percentage of lines covered in each file.
- **Line-by-line coverage:** number of executions for each line.
- **Overall coverage:** aggregated coverage across the package.

- **Metadata:** source file paths, expressions, and summary stats.

The resulting object can be printed, summarized, or passed to `test.assessr::generate_test_report()` to produce a human-readable test coverage report.

Returns NULL if the package cannot be installed, if `install_list$pkg_source_path` is missing/empty, or if the specified path does not exist.

### Examples

```
# Example STF package included in test.assessr
pkg_source_path <- system.file(
  "test-data",
  "test.package.0001_0.1.0.tar.gz",
  package = "test.assessr"
)

# Run get_package_coverage
pkg_test_coverage <- get_package_coverage(pkg_source_path)
```

---

|              |                                     |
|--------------|-------------------------------------|
| get_pkg_name | <i>get package name for display</i> |
|--------------|-------------------------------------|

---

### Description

get package name for display

### Usage

```
get_pkg_name(input_string)
```

### Arguments

`input_string` - string containing package name

### Value

`pkg_disp` - package name for display

### Examples

```
pkg_source_path <- "/home/user/R/test.package.0001_0.1.0.tar.gz"
pkg_disp_1 <- get_pkg_name(pkg_source_path)
print(pkg_disp_1)

pkg <- "TxDb.Dmelanogaster.UCSC.dm3.ensGene_3.2.2.tar.gz"
pkg_disp_2 <- get_pkg_name(pkg)
print(pkg_disp_2)
```

---

**install\_package\_local** *Install a Package from a Local Source Directory*

---

**Description**

Attempts to install an R package from a local source directory using `remotes::install_local()`. The function reports on whether installation succeeded, whether the package was already installed, or whether the provided source path does not exist.

**Usage**

```
install_package_local(pkg_source_path)
```

**Arguments**

`pkg_source_path`

Character string. Path to the local package source directory (e.g., an unpacked package or extracted tarball path).

**Details**

The display name of the package is derived from the input path using `get_pkg_name()`.

**Value**

A logical value indicating whether the package is installed after running the function.

The returned object is always of class `logical`:

- `TRUE` — The package is already installed or was successfully installed.
- `FALSE` — Installation failed or the path does not exist.

**Output Meaning**

`TRUE` does not necessarily imply that the installation occurred during this function call—it may also mean the package was already installed.

`FALSE` indicates a failure to install or an invalid path. All diagnostic messages are printed via `message()` for user visibility.

**Examples**

```
## Not run:
results <- install_package_local("pkg_source_path")
print(results)

## End(Not run)
```

run\_coverage

*Run Coverage and Return Structured Coverage Results***Description**

This function executes code coverage analysis for a package using the ‘covr’ framework. It is typically used after a package has been installed locally and test files are available. The function runs coverage in an isolated process (using ‘callr’) and returns a structured summary of overall and file-level coverage.

**Usage**

```
run_coverage(pkg_source_path, timeout = Inf)
```

**Arguments**

|                 |                                                                                                                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pkg_source_path | Character. Path to the installed package directory from which coverage should be computed.                                                                                                 |
| timeout         | Numeric. Timeout (in seconds) passed to <code>callr::r_safe()</code> when running coverage. This limits the maximum time allowed for executing the tests underlying the coverage analysis. |

**Details**

The function invokes covr’s coverage evaluation in a clean R session and extracts both:

- total coverage: percentage of lines covered across the package, and
- function/file-level coverage: coverage data for individual files.

It is used internally by higher-level functions such as `run_covr_modes()` and skip-aware coverage wrappers in the Standard Testing Framework (STF).

**Value**

A named list containing:

**total\_cov** Numeric. Aggregated coverage percentage for the package.

**res\_cov** A list containing file-level or function-level coverage results as returned by ‘covr’. This includes per-file coverage, errors, and diagnostic notes if present.

Returns NULL if coverage could not be computed.

**Examples**

```
# Save and adjust CRAN mirror for reproducibility
r <- getOption("repos")
old <- options(repos = r)
r["CRAN"] <- "http://cran.us.r-project.org"
options(repos = r)

# Example package from test.assessr
dp <- system.file(
  "test-data",
  "test.package.0001_0.1.0.tar.gz",
  package = "test.assessr"
)

# Set up package source directory
install_list <- set_up_pkg(dp)
pkg_source_path <- install_list$pkg_source_path

# Install locally to enable testing and coverage
package_installed <- install_package_local(pkg_source_path)

if (isTRUE(package_installed)) {
  coverage_results <- run_coverage(pkg_source_path)
}

# Restore user's original repository settings
options(old)
```

run\_covr\_modes

*Run Coverage Analysis with Test Detection***Description**

This function inspects the test configuration of an R package and runs code coverage analysis using any available testing framework, including ‘testthat’, ‘testit’, base R test scripts, or Bioconductor-style tests. If no recognised testing configuration is found, a default zero-coverage result is returned.

**Usage**

```
run_covr_modes(pkg_source_path, covr_timeout = 60)
```

**Arguments**

|                 |                                                                           |
|-----------------|---------------------------------------------------------------------------|
| pkg_source_path | Character. Path to the root directory of the package source.              |
| covr_timeout    | Numeric. Timeout in seconds for running coverage analysis. Default is 60. |

**Value**

A named list containing coverage results, package metadata, and test configuration details. The returned list includes (but is not limited to):

**pkg\_name** Character. Package name extracted from the DESCRIPTION file.

**pkg\_ver** Character. Package version.

**date\_time** Timestamp of when the analysis was run.

**executor** User or environment running the analysis.

**sysname, version, release, machine** System metadata.

**r\_version** R version used during analysis.

**test\_framework\_type** Character. Detected testing framework type.

**covr\_list** A nested list containing:

**total\_cov** Numeric. Aggregated coverage percentage.

**res\_cov** File-level and line-level coverage details.

**errors, notes** Any warnings or notes detected during testing.

**test\_pkg\_data** A list describing the test configuration of the package (presence of testthat, testit, base tests, snapshots, etc.).

This function always returns a value. When no supported testing framework is detected, a default object with zero coverage and diagnostic information is returned.

**Examples**

```
dp <- system.file("test-data",
  "test.package.0001_0.1.0.tar.gz",
  package = "test.assessr")

# set up package
install_list <- set_up_pkg(dp)

package_installed <- install_list$package_installed
pkg_source_path <- install_list$pkg_source_path

# install package locally to ensure test works
package_installed <- install_package_local(pkg_source_path)
package_installed <- TRUE

covr_mode_list <- run_covr_modes(pkg_source_path)
```

---

`set_up_pkg`*Create Information on Local Package Installation*

---

**Description**

This function unpacks a package tarball using `unpack_tarball()` and returns information about whether the unpacked directory exists. It is a lightweight preparatory step used before attempting a local package installation.

**Usage**

```
set_up_pkg(dp)
```

**Arguments**

`dp` Character string. Path to a package tarball or package directory.

**Value**

A named list with the following elements:

- `package_installed` — Logical. TRUE if the unpacked package directory exists, otherwise FALSE.
- `pkg_source_path` — Character string giving the unpacked package source directory, or "" if unpacking failed.

The returned object is always a base R list. It contains no side effects besides calling `unpack_tarball()` and checking filesystem paths.

**Output Meaning**

A value of `package_installed = TRUE` indicates that the unpacked directory exists on disk and can be used for local installation.

`package_installed = FALSE` indicates either:

- `unpack_tarball()` returned an empty result, or
- the unpacked directory does not exist on disk.

**Examples**

```
## Not run:  
set_up_pkg(path/to/package, "mypackage")  
  
## End(Not run)
```

# Index

`extract_short_path`, [2](#)  
`generate_test_report`, [2](#)  
`get_package_coverage`, [3](#)  
`get_pkg_name`, [4](#)  
`install_package_local`, [5](#)  
`run_coverage`, [6](#)  
`run_covr_modes`, [7](#)  
`set_up_pkg`, [9](#)