

# Package ‘covr’

October 12, 2022

**Encoding** UTF-8

**Title** Test Coverage for Packages

**Version** 3.6.1

**Description** Track and report code coverage for your package and (optionally) upload the results to a coverage service like 'Codecov' <<https://about.codecov.io>> or 'Coveralls' <<https://coveralls.io>>. Code coverage is a measure of the amount of code being exercised by a set of tests. It is an indirect measure of test quality and completeness. This package is compatible with any testing methodology or framework and tracks coverage of both R code and compiled C/C++/FORTRAN code.

**URL** <https://covr.r-lib.org>, <https://github.com/r-lib/covr>

**BugReports** <https://github.com/r-lib/covr/issues>

**Depends** R (>= 3.1.0), methods

**Imports** digest, stats, utils, jsonlite, rex, httr, crayon, withr (>= 1.0.2), yaml

**Suggests** R6, curl, knitr, rmarkdown, htmltools, DT (>= 0.2), testthat, rlang, rstudioapi (>= 0.2), xml2 (>= 1.0.0), parallel, memoise, mockery, covr

**License** MIT + file LICENSE

**VignetteBuilder** knitr

**RoxygenNote** 7.2.1

**NeedsCompilation** yes

**Author** Jim Hester [aut, cre],  
Willem Ligtenberg [ctb],  
Kirill Müller [ctb],  
Henrik Bengtsson [ctb],  
Steve Peak [ctb],  
Kirill Sevastyanenko [ctb],  
Jon Clayden [ctb],  
Robert Flight [ctb],  
Eric Brown [ctb],

Brodie Gaslam [ctb],  
 Will Beasley [ctb],  
 Robert Krzyzanowski [ctb],  
 Markus Wamser [ctb],  
 Karl Forner [ctb],  
 Gergely Daróczy [ctb],  
 Jouni Helske [ctb],  
 Kun Ren [ctb],  
 Jeroen Ooms [ctb],  
 Ken Williams [ctb],  
 Chris Campbell [ctb],  
 David Hugh-Jones [ctb],  
 Qin Wang [ctb],  
 Doug Kelkhoff [ctb],  
 Ivan Sagalaev [ctb, cph] (highlight.js library),  
 Mark Otto [ctb] (Bootstrap library),  
 Jacob Thornton [ctb] (Bootstrap library),  
 Bootstrap contributors [ctb] (Bootstrap library),  
 Twitter, Inc [cph] (Bootstrap library)

**Maintainer** Jim Hester <james.f.hester@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-08-26 20:24:34 UTC

## R topics documented:

covr-package . . . . .	3
as_coverage . . . . .	5
as_coverage_with_tests . . . . .	6
azure . . . . .	6
codecov . . . . .	7
code_coverage . . . . .	8
coverage_to_list . . . . .	9
coveralls . . . . .	9
covr.record_tests . . . . .	10
environment_coverage . . . . .	11
exclusions . . . . .	12
file_coverage . . . . .	13
file_report . . . . .	14
function_coverage . . . . .	15
gitlab . . . . .	15
has_srcref . . . . .	16
in_covr . . . . .	16
is_covr_count_call . . . . .	17
is_current_test_finished . . . . .	17
new_test_counter . . . . .	18
package_coverage . . . . .	18
percent_coverage . . . . .	20

print.coverage . . . . . 20  
 report . . . . . 21  
 tally\_coverage . . . . . 21  
 to\_cobertura . . . . . 22  
 to\_sonarqube . . . . . 22  
 truncate\_call . . . . . 23  
 value . . . . . 23  
 zero\_coverage . . . . . 24

**Index** 25

covr-package *covr: Test coverage for packages*

**Description**

covr tracks and reports code coverage for your package and (optionally) upload the results to a coverage service like 'Codecov' <https://about.codecov.io> or 'Coveralls' <https://coveralls.io>. Code coverage is a measure of the amount of code being exercised by a set of tests. It is an indirect measure of test quality and completeness. This package is compatible with any testing methodology or framework and tracks coverage of both R code and compiled C/C++/FORTRAN code.

**Details**

A coverage report can be used to inspect coverage for each line in your package. Using report() requires the additional dependencies DT and htmltools.

```
# If run with no arguments `report()` implicitly calls `package_coverage()`  
report()
```

**Package options**

covr uses the following options() to configure behaviour:

- covr.covrignore: A filename to use as an ignore file, listing glob-style wildcarded paths of files to ignore for coverage calculations. Defaults to the value of environment variable COVR\_COVRIGNORE, or ".covrignore" if the neither the option nor the environment variable are set.
- covr.exclude\_end: Used along with covr.exclude\_start, an optional regular expression which ends a line-exclusion region. For more details, see ?exclusions.
- covr.exclude\_pattern: An optional line-exclusion pattern. Lines which match the pattern will be excluded from coverage. For more details, see ?exclusions.
- covr.exclude\_start: Used along with covr.exclude\_end, an optional regular expression which starts a line-exclusion region. For more details, see ?exclusions.
- covr.filter\_non\_package: If TRUE (the default behavior), coverage of files outside the target package are filtered from coverage output.

- `covr.fix_parallel_mcxexit`:
- `covr.flags`:
- `covr.gcov`: If the appropriate gcov version is not on your path you can use this option to set the appropriate location. If set to "" it will turn off coverage of compiled code.
- `covr.gcov_additional_paths`:
- `covr.gcov_args`:
- `covr.icov`:
- `covr.icov_args`:
- `covr.icov_flags`:
- `covr.icov_prof`:
- `covr.rstudio_source_markers`: A logical value. If TRUE (the default behavior), source markers are displayed within the RStudio IDE when using `zero_coverage`.
- `covr.record_tests`: If TRUE (default NULL), record a listing of top level test expressions and associate tests with covr traces evaluated during the test's execution. For more details, see `?covr.record_tests`.
- `covr.showCfunctions`:

### Author(s)

**Maintainer:** Jim Hester <james.f.hester@gmail.com>

Other contributors:

- Willem Ligtenberg [contributor]
- Kirill Müller [contributor]
- Henrik Bengtsson [contributor]
- Steve Peak [contributor]
- Kirill Sevastyanenko [contributor]
- Jon Clayden [contributor]
- Robert Flight [contributor]
- Eric Brown [contributor]
- Brodie Gaslam [contributor]
- Will Beasley [contributor]
- Robert Krzyzanowski [contributor]
- Markus Wamser [contributor]
- Karl Forner [contributor]
- Gergely Daróczy [contributor]
- Jouni Helske [contributor]
- Kun Ren [contributor]
- Jeroen Ooms [contributor]
- Ken Williams [contributor]

- Chris Campbell [contributor]
- David Hugh-Jones [contributor]
- Qin Wang [contributor]
- Doug Kelkhoff [contributor]
- Ivan Sagalaev (highlight.js library) [contributor, copyright holder]
- Mark Otto (Bootstrap library) [contributor]
- Jacob Thornton (Bootstrap library) [contributor]
- Bootstrap contributors (Bootstrap library) [contributor]
- Twitter, Inc (Bootstrap library) [copyright holder]

### See Also

Useful links:

- <https://covr.r-lib.org>
- <https://github.com/r-lib/covr>
- Report bugs at <https://github.com/r-lib/covr/issues>

---

as\_coverage

*Convert a counters object to a coverage object*

---

### Description

Convert a counters object to a coverage object

### Usage

```
as_coverage(counters = NULL, ...)
```

### Arguments

counters	An environment of covr trace results to convert to a coverage object. If counters is not provided, the covr namespace value <code>.counters</code> is used.
...	Additional attributes to include with the coverage object.

---

 as\_coverage\_with\_tests

*Clean and restructure counter tests for a coverage object*


---

### Description

For tests produced with options(`covr.record_tests`), prune any unused records in the `$tests$stally` matrices of each trace and get rid of the wrapping `$tests` environment (reassigning with value of `$tests$stally`)

### Usage

```
as_coverage_with_tests(counters)
```

### Arguments

<code>counters</code>	An environment of <code>covr</code> trace results to convert to a coverage object. If <code>counters</code> is not provided, the <code>covr</code> namespace value <code>.counters</code> is used.
-----------------------	--

---

 azure

*Run covr on a package and output the result so it is available on Azure Pipelines*


---

### Description

Run `covr` on a package and output the result so it is available on Azure Pipelines

### Usage

```
azure(
  ...,
  coverage = package_coverage(..., quiet = quiet),
  filename = "coverage.xml",
  quiet = TRUE
)
```

### Arguments

<code>...</code>	arguments passed to <code>package_coverage()</code>
<code>coverage</code>	an existing coverage object to submit, if <code>NULL</code> , <code>package_coverage()</code> will be called with the arguments from <code>...</code>
<code>filename</code>	the name of the Cobertura XML file
<code>quiet</code>	if <code>FALSE</code> , print the coverage before submission.

---

 codecov

*Run covr on a package and upload the result to codecov.io*


---

## Description

Run covr on a package and upload the result to codecov.io

## Usage

```

codecov(
  ...,
  coverage = NULL,
  base_url = "https://codecov.io",
  token = NULL,
  commit = NULL,
  branch = NULL,
  pr = NULL,
  flags = NULL,
  quiet = TRUE
)

```

## Arguments

...	arguments passed to <a href="#">package_coverage()</a>
coverage	an existing coverage object to submit, if NULL, <a href="#">package_coverage()</a> will be called with the arguments from ...
base_url	Codecov url (change for Enterprise)
token	a codecov upload token, if NULL then following external sources will be checked in this order: <ol style="list-style-type: none"> <li>1. the environment variable 'CODECOV_TOKEN'. If it is empty, then</li> <li>2. package will look at directory of the package for a file <code>codecov.yml</code>. File must have <code>codecov</code> section where field <code>token</code> is set to a token that will be used.</li> </ol>
commit	explicitly set the commit this coverage result object corresponds to. Is looked up from the service or locally if it is NULL.
branch	explicitly set the branch this coverage result object corresponds to, this is looked up from the service or locally if it is NULL.
pr	explicitly set the pr this coverage result object corresponds to, this is looked up from the service if it is NULL.
flags	A flag to use for this coverage upload see <a href="https://docs.codecov.com/docs/flags">https://docs.codecov.com/docs/flags</a> for details.
quiet	if FALSE, print the coverage before submission.

## Examples

```
## Not run:  
codecov(path = "test")  
  
## End(Not run)
```

---

code_coverage	<i>Calculate coverage of code directly</i>
---------------	--

---

## Description

This function is useful for testing, and is a thin wrapper around [file\\_coverage\(\)](#) because parse-Data is not populated properly unless the functions are defined in a file.

## Usage

```
code_coverage(  
  source_code,  
  test_code,  
  line_exclusions = NULL,  
  function_exclusions = NULL,  
  ...  
)
```

## Arguments

source_code	A character vector of source code
test_code	A character vector of test code
line_exclusions	a named list of files with the lines to exclude from each file.
function_exclusions	a vector of regular expressions matching function names to exclude. Example <code>print\\.</code> to match print methods.
...	Additional arguments passed to <a href="#">file_coverage()</a>

## Examples

```
source <- "add <- function(x, y) { x + y }"  
test <- "add(1, 2) == 3"  
code_coverage(source, test)
```



---

coverage_to_list	<i>Convert a coverage dataset to a list</i>
------------------	---

---

**Description**

Convert a coverage dataset to a list

**Usage**

```
coverage_to_list(x = package_coverage())
```

**Arguments**

x a coverage dataset, defaults to running `package_coverage()`.

**Value**

A list containing coverage result for each individual file and the whole package

---

coveralls	<i>Run covr on a package and upload the result to coveralls</i>
-----------	---

---

**Description**

Run covr on a package and upload the result to coveralls

**Usage**

```
coveralls(
  ...,
  coverage = NULL,
  repo_token = Sys.getenv("COVERALLS_TOKEN"),
  service_name = Sys.getenv("CI_NAME", "travis-ci"),
  quiet = TRUE
)
```

**Arguments**

... arguments passed to `package_coverage()`

coverage an existing coverage object to submit, if NULL, `package_coverage()` will be called with the arguments from ...

repo\_token The secret repo token for your repository, found at the bottom of your repository's page on Coveralls. This is useful if your job is running on a service Coveralls doesn't support out-of-the-box. If set to NULL, it is assumed that the job is running on travis-ci

service_name	the CI service to use, if environment variable 'CI_NAME' is set that is used, otherwise 'travis-ci' is used.
quiet	if FALSE, print the coverage before submission.

---

covr.record\_tests      *Record Test Traces During Coverage Execution*

---

### Description

By setting `options(covr.record_tests = TRUE)`, the result of covr coverage collection functions will include additional data pertaining to the tests which are executed and an index of which tests, at what stack depth, trigger the execution of each trace.

### Details

This functionality requires that the package code and tests are installed and sourced with the source. For more details, refer to R options, `keep.source`, `keep.source.pkgs` and `keep.parse.data.pkgs`.

### Additional fields

Within the covr result, you can explore this information in two places:

- `attr("tests")`: A list of call stacks, which results in target code execution.
- `$<srcref>$tests`: For each `srcref` count in the coverage object, a `$tests` field is now included which contains a matrix with three columns, "test", "depth" and "i" which specify the test number (corresponding to the index of the test in `attr("tests")`), the stack depth into the target code where the trace was executed, and the order of execution for each test.

### Test traces

The content of test traces are dependent on the unit testing framework that is used by the target package. The behavior is contingent on the available information in the sources kept for the testing files.

Test traces are extracted by the following criteria:

1. If any `srcref` files are provided by a file within covr's temporary library, all calls from those files are kept as a test trace. This will collect traces from tests run with common testing frameworks such as `testthat` and `RUnit`.
2. Otherwise, as a conservative fallback in situations where no source references are found, or when none are from within the temporary directory, the entire call stack is collected.

These calls are subsequently subset for only those up until the call to covr's internal count function, and will always include the last call in the call stack prior to a call to `count`.

**Examples**

```

fcode <- '
f <- function(x) {
  if (x)
    f(!x)
  else
    FALSE
}'

options(covr.record_tests = TRUE)
cov <- code_coverage(fcode, "f(TRUE)")

# extract executed test code for the first test
tail(attr(cov, "tests")[[1L]], 1L)
# [[1]]
# f(TRUE)

# extract test itemization per trace
cov[[3]][c("srcref", "tests")]
# $srcref
# f(!x)
#
# $tests
#   test depth i
# [1,]   1   2 4

# reconstruct the code path of a test by ordering test traces by [, "i"]
lapply(cov, `[`, "tests")
# $`source.Ref2326138c55:4:6:4:10:6:10:4:4`
#   test depth i
# [1,]   1   1 2
#
# $`source.Ref2326138c55:3:8:3:8:8:8:3:3`
#   test depth i
# [1,]   1   1 1
# [2,]   1   2 3
#
# $`source.Ref2326138c55:6:6:6:10:6:10:6:6`
#   test depth i
# [1,]   1   2 4

```

---

environment\_coverage *Calculate coverage of an environment*

---

**Description**

Calculate coverage of an environment

**Usage**

```
environment_coverage(
  env = parent.frame(),
  test_files,
  line_exclusions = NULL,
  function_exclusions = NULL
)
```

**Arguments**

`env`                    The environment to be instrumented.

`test_files`            Character vector of test files with code to test the functions

`line_exclusions`        a named list of files with the lines to exclude from each file.

`function_exclusions`   a vector of regular expressions matching function names to exclude. Example `print\\.` to match print methods.

---

exclusions

*Exclusions*

---

**Description**

covr supports a couple of different ways of excluding some or all of a file.

**Line Exclusions**

The `line_exclusions` argument to `package_coverage()` can be used to exclude some or all of a file. This argument takes a list of filenames or named ranges to exclude.

**Function Exclusions**

Alternatively `function_exclusions` can be used to exclude R functions based on regular expression(s). For example `print\\.*` can be used to exclude all the print methods defined in a package from coverage.

**Exclusion Comments**

In addition you can exclude lines from the coverage by putting special comments in your source code. This can be done per line or by specifying a range. The patterns used can be specified by the `exclude_pattern`, `exclude_start`, `exclude_end` arguments to `package_coverage()` or by setting the global options `covr.exclude_pattern`, `covr.exclude_start`, `covr.exclude_end`.

**Examples**

```

## Not run:
# exclude whole file of R/test.R
package_coverage(exclusions = "R/test.R")

# exclude lines 1 to 10 and 15 from R/test.R
package_coverage(line_exclusions = list("R/test.R" = c(1:10, 15)))

# exclude lines 1 to 10 from R/test.R, all of R/test2.R
package_coverage(line_exclusions = list("R/test.R" = 1:10, "R/test2.R"))

# exclude all print and format methods from the package.
package_coverage(function_exclusions = c("print\\.", "format\\."))

# single line exclusions
f1 <- function(x) {
  x + 1 # nocov
}

# ranged exclusions
f2 <- function(x) { # nocov start
  x + 2
} # nocov end

## End(Not run)

```

---

file\_coverage

*Calculate test coverage for sets of files*


---

**Description**

The files in `source_files` are first sourced into a new environment to define functions to be checked. Then they are instrumented to track coverage and the files in `test_files` are sourced.

**Usage**

```

file_coverage(
  source_files,
  test_files,
  line_exclusions = NULL,
  function_exclusions = NULL,
  parent_env = parent.frame()
)

```

**Arguments**

`source_files` Character vector of source files with function definitions to measure coverage  
`test_files` Character vector of test files with code to test the functions

`line_exclusions` a named list of files with the lines to exclude from each file.

`function_exclusions` a vector of regular expressions matching function names to exclude. Example `print\\.` to match print methods.

`parent_env` The parent environment to use when sourcing the files.

### Examples

```
# For the purpose of this example, save code containing code and tests to files
cat("add <- function(x, y) { x + y }", file="add.R")
cat("add(1, 2) == 3", file="add_test.R")

# Use file_coverage() to calculate test coverage
file_coverage(source_files = "add.R", test_files = "add_test.R")

# cleanup
file.remove(c("add.R", "add_test.R"))
```

---

file\_report                      *A coverage report for a specific file*

---

### Description

A coverage report for a specific file

### Usage

```
file_report(
  x = package_coverage(),
  file = NULL,
  out_file = file.path(tempdir(), paste0(get_package_name(x), "-file-report.html")),
  browse = interactive()
)
```

### Arguments

`x` a coverage dataset, defaults to running `package_coverage()`.

`file` The file to report on, if NULL, use the first file in the coverage output.

`out_file` The output file

`browse` whether to open a browser to view the report.

---

function\_coverage      *Calculate test coverage for a specific function.*

---

**Description**

Calculate test coverage for a specific function.

**Usage**

```
function_coverage(fun, code = NULL, env = NULL, enc = parent.frame())
```

**Arguments**

fun	name of the function.
code	expressions to run.
env	environment the function is defined in.
enc	the enclosing environment which to run the expressions.

**Examples**

```
add <- function(x, y) { x + y }
function_coverage(fun = add, code = NULL) # 0% coverage
function_coverage(fun = add, code = add(1, 2) == 3) # 100% coverage
```

---

gitlab      *Run covr on package and create report for GitLab*

---

**Description**

Utilize internal GitLab static pages to publish package coverage. Creates local covr report in a package subdirectory. Uses the [pages](#) GitLab job to publish the report.

**Usage**

```
gitlab(..., coverage = NULL, file = "public/coverage.html", quiet = TRUE)
```

**Arguments**

...	arguments passed to <a href="#">package_coverage()</a>
coverage	an existing coverage object to submit, if NULL, <a href="#">package_coverage()</a> will be called with the arguments from ...
file	The report filename.
quiet	if FALSE, print the coverage before submission.

---

has_srcref	<i>Is the source bound to the expression</i>
------------	--

---

**Description**

Is the source bound to the expression

**Usage**

```
has_srcref(expr)
```

**Arguments**

expr            A language object which may have a srcref attribute

**Value**

A logical value indicating whether the language object has source

---

in_covr	<i>Determine if code is being run in covr</i>
---------	---

---

**Description**

covr functions set the environment variable R\_COVR when they are running. `in_covr()` returns TRUE if this environment variable is set and FALSE otherwise.

**Usage**

```
in_covr()
```

**Examples**

```
if (require(testthat)) {  
  testthat::skip_if(in_covr())  
}
```



---

is\_covr\_count\_call     *Is the expression a call to covr:::count*

---

**Description**

Is the expression a call to covr:::count

**Usage**

```
is_covr_count_call(expr)
```

**Arguments**

expr                    A language object

**Value**

A logical value indicating whether the object is a call to covr:::count.

---

is\_current\_test\_finished  
*Returns TRUE if we've moved on from test reflected in .current\_test*

---

**Description**

Quickly dismiss the need to update the current test if we can. To test if we're still in the last test, check if the same srcref (or call, if source is not kept) exists at the last recorded calling frame prior to entering a covr trace. If this has changed, do a more comprehensive test to see if any of the test call stack has changed, in which case we are onto a new test.

**Usage**

```
is_current_test_finished()
```

---

new_test_counter	<i>Initialize a new test counter for a coverage trace</i>
------------------	---

---

### Description

Initialize a test counter, a matrix used to tally tests, their stack depth and the execution order as the trace associated with key is hit. Each test trace is an environment, which allows assignment into a pre-allocated tests matrix with minimal reallocation.

### Usage

```
new_test_counter(key)
```

### Arguments

key                    generated with `key()`

### Details

The tests matrix has columns tests, depth and i, corresponding to the test index (the index of the associated test in `.counters$tests`), the stack depth when the trace is evaluated and the number of traces that have been hit so far during test evaluation.

---

package_coverage	<i>Calculate test coverage for a package</i>
------------------	--

---

### Description

This function calculates the test coverage for a development package on the path. By default it runs only the package tests, but it can also run vignette and example code.

### Usage

```
package_coverage(
  path = ".",
  type = c("tests", "vignettes", "examples", "all", "none"),
  combine_types = TRUE,
  relative_path = TRUE,
  quiet = TRUE,
  clean = TRUE,
  line_exclusions = NULL,
  function_exclusions = NULL,
  code = character(),
  install_path = temp_file("R_LIBS"),
  ...,
  exclusions,
  pre_clean = TRUE
)
```

**Arguments**

path	file path to the package.
type	run the package ‘tests’, ‘vignettes’, ‘examples’, ‘all’, or ‘none’. The default is ‘tests’.
combine_types	If TRUE (the default) the coverage for all types is simply summed into one coverage object. If FALSE separate objects are used for each type of coverage.
relative_path	whether to output the paths as relative or absolute paths. If a string, it is interpreted as a root path and all paths will be relative to that root.
quiet	whether to load and compile the package quietly, useful for debugging errors.
clean	whether to clean temporary output files after running, mainly useful for debugging errors.
line_exclusions	a named list of files with the lines to exclude from each file.
function_exclusions	a vector of regular expressions matching function names to exclude. Example <code>print\\.</code> to match print methods.
code	A character vector of additional test code to run.
install_path	The path the instrumented package will be installed to and tests run in. By default it is a path in the R sessions temporary directory. It can sometimes be useful to set this (along with <code>clean = FALSE</code> ) to help debug test failures.
...	Additional arguments passed to <code>tools::testInstalledPackage()</code> .
exclusions	‘Deprecated’, please use ‘line_exclusions’ instead.
pre_clean	whether to delete all objects present in the src directory before recompiling

**Details**

This function uses `tools::testInstalledPackage()` to run the code, if you would like to test your package in another way you can set `type = "none"` and pass the code to run as a character vector to the `code` parameter.

Parallelized code using `parallel`'s `mcpipeline()` needs to use a patched `parallel::mcexit`. This is done automatically if the package depends on `parallel`, but can also be explicitly set using the environment variable `COVR_FIX_PARALLEL_MCEXIT` or the global option `covr.fix_parallel_mcexit`.

**See Also**

`exclusions()` For details on excluding parts of the package from the coverage calculations.

---

percent_coverage	<i>Provide percent coverage of package</i>
------------------	--

---

**Description**

Calculate the total percent coverage from a coverage result object.

**Usage**

```
percent_coverage(x, ...)
```

**Arguments**

x	the coverage object returned from <code>package_coverage()</code>
...	additional arguments passed to <code>tally_coverage()</code>

**Value**

The total percentage as a `numeric(1)`.

---

print.coverage	<i>Print a coverage object</i>
----------------	--------------------------------

---

**Description**

Print a coverage object

**Usage**

```
## S3 method for class 'coverage'
print(x, group = c("filename", "functions"), by = "line", ...)
```

**Arguments**

x	the coverage object to be printed
group	whether to group coverage by filename or function
by	whether to count coverage by line or expression
...	additional arguments ignored

**Value**

The coverage object (invisibly).

---

report	<i>Display covr results using a standalone report</i>
--------	---

---

**Description**

Display covr results using a standalone report

**Usage**

```
report(  
  x = package_coverage(),  
  file = file.path(tempdir(), paste0(get_package_name(x), "-report.html")),  
  browse = interactive()  
)
```

**Arguments**

x	a coverage dataset, defaults to running <code>package_coverage()</code> .
file	The report filename.
browse	whether to open a browser to view the report.

**Examples**

```
## Not run:  
x <- package_coverage()  
report(x)  
  
## End(Not run)
```

---

tally_coverage	<i>Tally coverage by line or expression</i>
----------------	---

---

**Description**

Tally coverage by line or expression

**Usage**

```
tally_coverage(x, by = c("line", "expression"))
```

**Arguments**

x	the coverage object returned from <code>package_coverage()</code>
by	whether to tally coverage by line or expression

**Value**

a `data.frame` of coverage tallied by line or expression.

---

to_cobertura	<i>Create a Cobertura XML file</i>
--------------	------------------------------------

---

**Description**

This functionality requires the xml2 package be installed.

**Usage**

```
to_cobertura(cov, filename = "cobertura.xml")
```

**Arguments**

cov	the coverage object returned from <a href="#">package_coverage()</a>
filename	the name of the Cobertura XML file

**Author(s)**

Willem Ligtenberg

---

to_sonarqube	<i>Create a SonarQube Generic XML file for test coverage according to <a href="https://docs.sonarqube.org/latest/analysis/generic-test/">https://docs.sonarqube.org/latest/analysis/generic-test/</a> Based on cobertura.R</i>
--------------	--

---

**Description**

This functionality requires the xml2 package be installed.

**Usage**

```
to_sonarqube(cov, filename = "sonarqube.xml")
```

**Arguments**

cov	the coverage object returned from <a href="#">package_coverage()</a>
filename	the name of the SonarQube Generic XML file

**Author(s)**

Talkdesk Inc.

---

truncate_call	<i>Truncate call objects to limit the number of arguments</i>
---------------	---

---

**Description**

A helper to circumvent R errors when deserializing large call objects from Rds. Trims the number of arguments in a call object, and replaces the last argument with a <truncated> symbol.

**Usage**

```
truncate_call(call_obj, limit = 10000)
```

**Arguments**

call_obj	A (possibly large) call object
limit	A call length limit to impose

**Value**

The call\_obj with arguments trimmed

---

value	<i>Retrieve the value from an object</i>
-------	--

---

**Description**

Retrieve the value from an object

**Usage**

```
value(x, ...)
```

**Arguments**

x	object from which to retrieve the value
...	additional arguments passed to methods

---

zero_coverage	<i>Provide locations of zero coverage</i>
---------------	---

---

**Description**

When examining the test coverage of a package, it is useful to know if there are any locations where there is **0** test coverage.

**Usage**

```
zero_coverage(x, ...)
```

**Arguments**

x	a coverage object returned <a href="#">package_coverage()</a>
...	additional arguments passed to <a href="#">tally_coverage()</a>

**Details**

if used within RStudio this function outputs the results using the Marker API.

**Value**

A `data.frame` with coverage data where the coverage is 0.



# Index

`as_coverage`, 5  
`as_coverage_with_tests`, 6  
`azure`, 6

`code_coverage`, 8  
`codecov`, 7  
`coverage_to_list`, 9  
`coveralls`, 9  
`covr`, 10  
`covr (covr-package)`, 3  
`covr-package`, 3  
`covr.record_tests`, 10

`environment_coverage`, 11  
`exclusions`, 12  
`exclusions()`, 19

`file_coverage`, 13  
`file_coverage()`, 8  
`file_report`, 14  
`function_coverage`, 15

`gitlab`, 15

`has_srcref`, 16

`in_covr`, 16  
`in_covr()`, 16  
`is_covr_count_call`, 17  
`is_current_test_finished`, 17

`key()`, 18

`mcparallel()`, 19

`new_test_counter`, 18

`options()`, 3

`package_coverage`, 18  
`package_coverage()`, 6, 7, 9, 15, 20–22, 24  
`percent_coverage`, 20

`print.coverage`, 20

`report`, 21

`tally_coverage`, 21  
`tally_coverage()`, 20, 24  
`to_cobertura`, 22  
`to_sonarqube`, 22  
`tools::testInstalledPackage()`, 19  
`truncate_call`, 23

`value`, 23

`zero_coverage`, 24