Package ‘covr’

August 23, 2019

Encoding UTF-8

Title Test Coverage for Packages

Version 3.3.1

Description Track and report code coverage for your package and (optionally) upload the results to a coverage service like ‘Codecov’ <http://codecov.io> or ‘Coveralls’<http://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://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, knitr, rmarkdown, htmltools, DT (>= 0.2), testthat, rlang, rstudioapi (>= 0.2), xml2 (>= 1.0.0), parallel, memoise, mockery

License GPL-3

LazyData true

VignetteBuilder knitr

RoxygenNote 6.1.1

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 Daróczi [ctb],
Jouni Helske [ctb],
Kun Ren [ctb],
Jeroen Ooms [ctb],
Ken Williams [ctb],
Chris Campbell [ctb],
David Hugh-Jones [ctb],
Qin Wang [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. hyster@gmail.com>
Repository CRAN
Date/Publication 2019-08-23 21:20:02 UTC

R topics documented:

covr-package ................................................................. 3
azure ................................................................. 4
codecov ............................................................... 5
code_coverage ............................................................. 6
coverage_to_list .......................................................... 6
coveralls ............................................................ 7
environment_coverage .................................................. 7
exclusions .............................................................. 8
file_coverage ............................................................ 9
file_report .............................................................. 10
function_coverage ...................................................... 10
gitlab ............................................................... 11
package_coverage ......................................................... 11
percent_coverage .......................................................... 12
print.coverage ............................................................ 13
report ................................................................. 13
tally_coverage ............................................................ 14
to_cobertura .............................................................. 14
value ................................................................. 15
zero_coverage .............................................................. 15

Index 16
Description

covr tracks and reports code coverage for your package and (optionally) upload the results to a coverage service like `Codecov` [http://codecov.io](http://codecov.io) or `Coveralls` [http://coveralls.io](http://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()
```

Author(s)

**Maintainer:** Jim Hester <james.f.hestergmail.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 Daróczi [contributor]
- Jouni Helske [contributor]
- Kun Ren [contributor]
- Jeroen Ooms [contributor]
- Ken Williams [contributor]
Run covr on a package and output the result so it is available on Azure Pipelines

#### Usage

```r
azure(..., coverage = package_coverage(..., quiet = quiet),
    filename = "coverage.xml", 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...
- `filename`: the name of the Cobertura XML file
- `quiet`: if FALSE, print the coverage before submission.

#### Description

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

#### See Also

Useful links:
- [https://github.com/r-lib/covr](https://github.com/r-lib/covr)
- Report bugs at [https://github.com/r-lib/covr/issues](https://github.com/r-lib/covr/issues)
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, flags = NULL,
          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 ...

base_url Codecov url (change for Enterprise)

token a codecov upload token, if NULL then following external sources will be checked in this order:

1. the environment variable ‘CODECOV_TOKEN’. If it is empty, then
2. package will look at directory of the package for a file codecov.yml. File must have codecov section where field token is set to a token that will be used.

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.

flags A flag to use for this coverage upload see https://docs.codecov.io/docs/flags for details.

quiet if FALSE, print the coverage before submission.

Examples

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

## End(Not run)
```
code_coverage  Calculate coverage of code directly

Description
This function is useful for testing, and is a thin wrapper around file_coverage() because parseData 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 print\. to match print methods.
... Additional arguments passed to file_coverage()

coverage_to_list  Convert a coverage dataset to a list

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

Run covr on a package and upload the result to coveralls

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.

evironment_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.

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 expres-
sion(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

```r
## 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

```r
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.
file_report  

A coverage report for a specific file

Description

A coverage report for a specific file

Usage

```r
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

```r
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.
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

```r
gitlab(..., coverage = NULL, file = "public/coverage.html", 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 `...`
- `file` The report filename.
- `quiet` if FALSE, print the coverage before submission.

package_coverage  

Calculate test coverage for a package

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

```r
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(), ..., 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.
## percent_coverage

A function for calculating the total percent coverage from a coverage result object.

### Description

Calculate the total percent coverage from a coverage result object.

### Usage

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

### Arguments

- `x`  
  The coverage object returned from `package_coverage()`

- `...`  
  Additional arguments passed to `tally_coverage()`

### Value

The total percentage as a numeric(1).
**print.coverage**

*Print a coverage object*

**Description**

Print a coverage object

**Usage**

```r
## 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**

*Display covr results using a standalone report*

**Description**

Display covr results using a standalone report

**Usage**

```r
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 `package_coverage()`.
- **file** The report filename.
- **browse** whether to open a browser to view the report.
Examples

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

## End(Not run)
```

tally_coverage  

*Tally coverage by line or expression*

**Description**

Tally coverage by line or expression

**Usage**

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

**Arguments**

- `x`: the coverage object returned from `package_coverage()`
- `by`: whether to tally coverage by line or expression

**Value**

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

to_cobertura  

*Create a Cobertura XML file*

**Description**

This functionality requires the `xml2` package be installed.

**Usage**

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

**Arguments**

- `cov`: the coverage object returned from `package_coverage()`
- `filename`: the name of the Cobertura XML file

**Author(s)**

Willem Ligtenberg
value

Retrieve the value from an object

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
Provide locations of zero coverage

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 package_coverage()
... additional arguments passed to tally_coverage()

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

azure, 4

code_coverage, 6
codecov, 5
coverage_to_list, 6
coveralls, 7
covr (covr-package), 3
covr-package, 3

evironment_coverage, 7
exclusions, 8
exclusions(), 12

file_coverage, 9
file_coverage(), 6
file_report, 10
function_coverage, 10

gitlab, 11
mcparallel(), 12

package_coverage, 11
package_coverage(), 4, 5, 7, 11, 12, 14, 15
percent_coverage, 12
print.coverage, 13

report, 13

tally_coverage, 14
tally_coverage(), 12, 15
to_cobertura, 14
tools::testInstalledPackage(), 12

value, 15

zero_coverage, 15