Package 'ottr'

November 10, 2024

Title An R Autograding Extension for Otter-Grader

Version 1.5.1

Maintainer Christopher Pyles <cpyles@berkeley.edu>

Description An R autograding extension for Otter-Grader (<https://otter-grader.readthedocs.io>). It supports grading R scripts, R Markdown documents, and R Jupyter Notebooks.

License BSD_3_clause + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.0.0)

Imports jsonlite, testthat, tools, R6, zip, methods

Suggests IRdisplay, mockery, rmarkdown, stringr, withr, IRkernel

Config/testthat/edition 3

NeedsCompilation no

Author Christopher Pyles [aut, cre] (<https://orcid.org/0000-0001-8520-7593>), UC Berkeley Data Science Education Program [cph]

Repository CRAN

Date/Publication 2024-11-10 16:40:02 UTC

Contents

---

check *Run the test cases in a test file*

Description

Execute checks in a test suite and return the TestFileResult object from executing the test. Optionally prints results of the test to console.

Usage

```
check(test_file, test_env, show_results)
```

Arguments

`test_file`	Path to a test file
`test_env`	An environment against which to run tests
`show_results`	Whether to print the results to stdout

Value

The parsed test results for the suite

Examples

```
## Not run:
check("tests/q1.R")

## End(Not run)
```

CheckCollector *An R6 class for collecting TestFileResult objects during grading.*

Description

A collection of test file results created while grading an assignment

Public fields

`test_file_results` The TestFileResult objects created during grading

Methods

Public methods:

- `CheckCollector$new()`
- `CheckCollector$add_result()`
- `CheckCollector$get_results()`
- `CheckCollector$clone()`

Method `new()`**:** Create a CheckCollector. Add a TestFileResult to this collector.

Usage:

```
CheckCollector$new()
```

Method `add_result()`**:**

Usage:

```
CheckCollector$add_result(test_file_result)
```

Arguments:

`test_file_result` The TestFileResult to add Retrieve the list TestFileResult objects stored in this collector.

Method `get_results()`**:**

Usage:

```
CheckCollector$get_results()
```

Returns: The list of TestFileResult objects

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`collector_env` *An environment into which a collector will be initialized (so we don't need to update global variables).*

Description

An environment into which a collector will be initialized (so we don't need to update global variables).

Usage

```
collector_env
```

Format

An object of class `environment` of length 0.

`collector_varname` *The name of the active collector variable in collector_env*

Description

The name of the active collector variable in collector_env

Usage

```
collector_varname
```

Format

An object of class `character` of length 1.

execute_script *Generate an environment from an R script*

Description

Execute a string as an R script and return the environment from that execution.

Converts a string to an AST and executes that script in a dummy environment for running test cases against. Transforms all expressions of the form `. = ottr::check(...)` by replacing the `.` with an index into a list in the environment with name `check_results_{SECRET}` to collect the TestFileResult objects generated from those checks. (This helps to handle variable name collisions in tests when grading a script.)

Usage

```
execute_script(script, ignore_errors)
```

Arguments

`script`	The string to be executed
`ignore_errors`	Whether to ignore errors thrown while executing the script

Value

The global environment after executing the script

export *Export a submission to a zip file*

Description

Export a submission to a zip file for submitting. If indicated, a PDF of the submission is generated and included in the zip file. (PDF generation is only supported for Rmd and ipynb files.)

Usage

```
export(
  submission_path,
  export_path = NULL,
  display_link = TRUE,
  pdf = FALSE,
  force_save = FALSE,
  debug = FALSE
)
```

Arguments

submission_path	The path to the submission
export_path	The path at which to write the zip file (optional)
display_link	Whether to display a download link with IRdisplay
pdf	Whether to include a PDF of the submission (only works for Rmd and ipynb files)
force_save	Whether to attempt to force-save the notebook if running on Jupyter
debug	Whether to stop on PDF generation errors

Examples

```
## Not run:
export("hw01.ipynb")

# with pdf
export("hw01.ipynb", pdf = TRUE)

## End(Not run)
```

---

get_collector *Retrieve the global CheckCollector*

Description

Retrieve the global CheckCollector

Usage

```
get_collector()
```

---

grade_script *Grade an R script against a series of test files*

Description

Execute a script, parse check outputs, and run additional tests specified by the glob pattern `tests_glob` on the test environment.

Usage

```
grade_script(script_path, tests_glob, ignore_errors)
```

Arguments

`script_path`	The path to the script
`tests_glob`	The pattern to search for extra tests
`ignore_errors`	Whether to ignore errors thrown while executing the script

Value

The GradingResults object after executing tests referenced in the script and those specified by `tests_glob`

`GradingResults` *An R6 class representing a collection of test case results*

Description

A collection of test case results that correspond to a single test file.

Public fields

`test_file_results` The TestFileResult objects that make up this grading

Methods

Public methods:

- `GradingResults$new()`
- `GradingResults$to_list()`
- `GradingResults$to_json()`
- `GradingResults$clone()`

Method `new()`**:** Create a grading result.

Usage:

```
GradingResults$new(test_file_results)
```

Arguments:

`test_file_results` The TestFileResult objects that make up this grading result

Method `to_list()`**:** Convert these results to a JSON-like `list` that can be convert to a `GradingResults` object by Otter's Python library.
The returned list has the JSON format

```
{
  "test_file_results": [
    {
      // output of TestFileResult$to_list
    }
  ]
}
```

Usage:

```
GradingResults$to_list()
```

Returns: The generated list

Method `to_json()`**:** Export these results to a JSON string.

Usage:

```
GradingResults$to_json()
```

Returns: The JSON string

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`initialize_collector` *Create a new global CheckCollector*

Description

Create a new global CheckCollector

Usage

```
initialize_collector()
```

`load_test_cases` *Load test cases from a test file*

Description

Load test case data from a test file. Executes the file and grabs the global `test` variable, which should be a `list`.

Usage

```
load_test_cases(test_file)
```

Arguments

`test_file`	The path to the test file

Value

The test cases

running_on_jupyter *Determine whether this R session is running on Jupyter.*

Description

Determine whether this R session is running on Jupyter by checking for a CommManager in IRkernel.

Usage

```
running_on_jupyter()
```

Value

A boolean indicating whether IRkernel is running.

Examples

```
## Not run:
running_on_jupyter()

## End(Not run)
```

run_autograder *Grade an R script against test files in a directory*

Description

Run autograder in a Gradescope container and return the results as a properly-formatted JSON string.

Usage

```
run_autograder(script_path, ignore_errors, test_dir)
```

Arguments

script_path	The path to the script
ignore_errors	Whether to ignore errors thrown while executing the script
test_dir	A directory of tests to glob from

Value

The JSON string

Examples

```
## Not run:
run_autograder("hw01.R", "ABC123", TRUE, "tests")

## End(Not run)
```

`save_notebook` *Attempt to save the current notebook.*

Description

Attempt to save the notebook by displaying Javascript if running on Jupyter. This function waits until the modification time of the file has changed or until the specified timeout expires.

Usage

```
save_notebook(nb_path, timeout = 10)
```

Arguments

`nb_path`	The path to the notebook
`timeout`	Number of seconds to wait for save

Value

A boolean indicating whether the file was saved successfully. If Jupyter is not running, this function returns TRUE.

Examples

```
## Not run:
save_notebook("foo.ipynb")

## End(Not run)
```

`TestCase` *An R6 class representing a test case*

Description

A test case for Ottr. Contains configurations and code to be executed for the test.

Public fields

`name` The name of the test case

`code` The code to be executed as part of the test case

`points` The point value of the test case

`hidden` Whether the test case is hidden

`success_message` A message to show to students if the test passes

`failure_message` A message to show to students if the test fails

Methods

Public methods:

- `TestCase$new()`
- `TestCase$run()`
- `TestCase$to_list()`
- `TestCase$clone()`

Method `new()`**:** Create a test case.

Usage:

```
TestCase$new(
  name,
  code,
  points = 1,
  hidden = FALSE,
  success_message = NA,
  failure_message = NA
)
```

Arguments:

`name` The name of the test case

`code` The code to be executed as part of the test case

`points` The point value of the test case

`hidden` Whether the test case is hidden

`success_message` A message to show to students if the test passes

`failure_message` A message to show to students if the test fails

Method `run()`**:** Run the test case against the provided environment.

Usage:

```
TestCase$run(env)
```

Arguments:

`env` The environment to run the test case in

Method `to_list()`**:** Convert this test case to a JSON-compatible list with all of its fields.

Usage:

```
TestCase$to_list()
```

Returns: The list representation of this test case

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
tc = TestCase$new("q1", {
  testthat::assert_true(q1.ans)
})
env = new.env()
env$q1.ans = TRUE
tc$run(env)
```

`TestCaseResult` *An R6 representing the results of running a test case*

Description

Represents the results of running a test case against a global environment. Contains metadata about the passing/failing of the test case as well as a reference to the test case itself.

Public fields

`passed` Whether the test passed

`error` An error raised by executing the test, if any

`test_case` The TestCase that this result tracks

Methods

Public methods:

- `TestCaseResult$new()`
- `TestCaseResult$get_score()`
- `TestCaseResult$repr()`
- `TestCaseResult$to_list()`
- `TestCaseResult$get_message()`
- `TestCaseResult$clone()`

Method `new()`**:** Create a test case result.

Usage:

```
TestCaseResult$new(passed, error, test_case)
```

Arguments:

`passed` Whether the test passed

`error` An error raised by executing the test, if any

`test_case` The `TestCase` that this result tracks

Method `get_score()`**:** Get the score earned for this test case, accounting for whether the test passed or failed.

Usage:

```
TestCaseResult$get_score()
```

Returns: The score

Method `repr()`**:** Convert this result into a human-readable string for display.

Usage:

```
TestCaseResult$repr()
```

Returns: The string representation of this result

Method `to_list()`**:** Convert this result to a JSON-compatible list with all of its fields.

Usage:

```
TestCaseResult$to_list()
```

Returns: The list representation of this result

Method `get_message()`**:** Get the message to be displayed to the student based on whether the test case passed or failed, if any.

Usage:

```
TestCaseResult$get_message()
```

Returns: The message or `NA`

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`TestFileResult` *An R6 class representing a collection of test case results*

Description

A collection of test case results that correspond to a single test file.

Public fields

`test_case_results` The TestCaseResult objects that make up this test file

`filename` The name of the test file

`points` The point value of the test file or a list of test case point values

Methods

Public methods:

- TestFileResult$new()
- TestFileResult$get_basename()
- TestFileResult$get_score()
- TestFileResult$repr()
- TestFileResult$to_list()
- TestFileResult$clone()

Method `new()`**:** Create a test file result.

Usage:

```
TestFileResult$new(filename, test_case_results, points = NULL)
```

Arguments:

`filename` The name of the test file

`test_case_results` The TestCaseResult objects that make up this test file

`points` The point value of the test file or a list of test case point values

Method `get_basename()`**:** Get the basename of the file this result corresponds to.

Usage:

```
TestFileResult$get_basename()
```

Returns: The basename of the test file

Method `get_score()`**:** Get the total score earned for this test file as a percentage. Uses TestCaseResult$get_score() to determine the points earned for each test case.

Usage:

```
TestFileResult$get_score()
```

Returns: The score as a percentage.

Method `repr()`**:** Convert this result into a human-readable string for display.

Usage:

```
TestFileResult$repr()
```

Returns: The string representation of this result

Method `to_list()`**:** Convert this result to a JSON-compatible list with all of its fields.

Usage:

```
TestFileResult$to_list()
```

Returns: The list representation of this result

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

update_ast_check_calls *Collect results of calls to* ottr::check *in an AST*

Description

Traverse an AST (a list of expressions) and change calls of the form . = ottr::check(...) so that they are appended to a list with name list_name.

If list_name is check_results_XX, then . = ottr::check(...) becomes check_results_XX[[<int>]] = ottr::check(where <int> is an integer

Usage

```
update_ast_check_calls(tree, list_name)
```

Arguments

tree	The tree to traverse
list_name	The quoted name of the list

Value

The tree with substitutions made

valid_syntax *Check whether a string is valid R code*

Description

Determine whether a code snippet has any syntax errors.

Determine whether a code snippet has any syntax errors.

Usage

```
valid_syntax(script)
```

```
valid_syntax(script)
```

Arguments

script	The code snippet

Value

Whether the code snippet is valid (can be parsed with parse)

Whether the code snippet is valid (can be parsed with parse)

Examples

```
s = "
a = TRUE
b = c(1, 2, 3)
d = function(x) x ^ 2
f = d(b)
"
valid_syntax(s)
#> [1] TRUE

s = "
if (TRUE) {
  a = c(1, 2)
"
valid_syntax(s)
#> [1] FALSE
s = "
a = TRUE
b = c(1, 2, 3)
d = function(x) x ^ 2
f = d(b)
"
valid_syntax(s)
#> [1] TRUE

s = "
if (TRUE) {
  a = c(1, 2)
"
valid_syntax(s)
#> [1] FALSE
```

Index