

Package ‘ottr’

July 6, 2022

Title An R Autograding Extension for Otter-Grader

Version 1.2.0

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 7.2.0

Depends R (>= 4.0.0)

Imports testthat, jsonlite, R6, zip, methods

Suggests IRdisplay, mockery, stringr, withr

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 2022-07-05 22:20:02 UTC

R topics documented:

check	2
execute_script	3
export	3
grade_script	4
GradingResults	4
load_test_cases	6
make_secret	6
run_autograder	7
TestCase	7
TestCaseResult	9

TestFileResult	10
update_ast_check_calls	12
valid_expr_chars	12
valid_syntax	13

Index	14
--------------	-----------

check	<i>Run the test cases in a test file</i>
-------	--

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)
```

execute_script	<i>Generate an environment from an R script</i>
----------------	---

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, secret, ignore_errors)
```

Arguments

script	The string to be executed
secret	The string to be appended to the name <code>check_results_</code> as the list name to collect results
ignore_errors	Whether to ignore errors thrown while executing the script

Value

The global environment after executing the script

export	<i>Export a Jupyter Notebook to a zip file</i>
--------	--

Description

Export a Jupyter Notebook to a zip file for submission.

Usage

```
export(notebook_path, export_path = NULL, display_link = TRUE)
```

Arguments

notebook_path	The path to the notebook
export_path	The path at which to write the zip file (optional)
display_link	Whether to display a download link with <code>IRdisplay</code>

Examples

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

## End(Not run)
```

grade_script	<i>Grade an R script against a series of test files</i>
--------------	---

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, secret, ignore_errors)
```

Arguments

<code>script_path</code>	The path to the script
<code>tests_glob</code>	The pattern to search for extra tests
<code>secret</code>	The string to be appended to the name <code>check_results_</code> as the list name to collect results (optional)
<code>ignore_errors</code>	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	<i>An R6 class representing a collection of test case results</i>
----------------	---

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.

load_test_cases	<i>Load test cases from a test file</i>
-----------------	---

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

make_secret	<i>Generate a random string of characters</i>
-------------	---

Description

Randomly generate a string of n_chars sampled at random from valid_chars.

Usage

```
make_secret(n_chars, valid_chars)
```

Arguments

n_chars	The number of characters in the string; defaults to 6
valid_chars	A string of characters to choose from; defaults to all alphanumerals, ., and _

Value

The generated string

run_autograder	<i>Grade an R script against test files in a directory</i>
----------------	--

Description

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

Usage

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

Arguments

script_path	The path to the script
secret	The string to be appended to the name check_results_ as the list name to collect results (optional)
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)
```

TestCase	<i>An R6 class representing a test case</i>
----------	---

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

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)
```

Arguments:

`filename` The name of the test file

`test_case_results` The [TestCaseResult](#) objects that make up this test file

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

<code>tree</code>	The tree to traverse
<code>list_name</code>	The quoted name of the list

Value

The tree with substitutions made

```
valid_expr_chars
```

A string containing characters that can be made into a valid variable name. Does not include any digits because randomly sampling with them included could result in an invalid variable name.

Description

A string containing characters that can be made into a valid variable name. Does not include any digits because randomly sampling with them included could result in an invalid variable name.

Usage

```
valid_expr_chars
```

Format

An object of class character of length 1.

`valid_syntax`*Check whether a string is valid R code*

Description

Determine whether a code snippet has any syntax errors.

Usage

```
valid_syntax(script)
```

Arguments

`script` The code snippet

Value

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
```

Index

* datasets

valid_expr_chars, 12

check, 2

execute_script, 3

export, 3

grade_script, 4

GradingResults, 4, 4

load_test_cases, 6

make_secret, 6

run_autograder, 7

TestCase, 7, 9

TestCaseResult, 9, 10, 11

TestCaseResult\$get_score(), 11

TestFileResult, 2–5, 10

update_ast_check_calls, 12

valid_expr_chars, 12

valid_syntax, 13