Package: future.tests (via r-universe)

November 22, 2026

Title Test Suite for 'Future API' Backends

Version 1.0.0-9034

Description Backends implementing the 'Future' API <doi:10.32614/RJ-2021-048>, as defined by the 'future' package, should use the tests provided by this package to validate that they meet the minimal requirements of the 'Future' API. The tests can be performed easily from within R or from outside of R from the command line making it straightforward to include them in package tests and in Continuous Integration (CI) pipelines.

Imports future (>= 1.80.0), cli, crayon, prettyunits, sessioninfo, utils

Suggests commonmark, base64enc, tools, data.table, ff

VignetteBuilder future.tests

License Apache License (>= 2)

Encoding UTF-8

URL https://future.tests.futureverse.org, https://github.com/futureverse/future.tests

BugReports https://github.com/futureverse/future.tests/issues

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Repository https://futureverse-devel.r-universe.dev

Date/Publication 2026-11-22 00:50:13 UTC

RemoteUrl https://github.com/futureverse/future.tests

RemoteRef develop

RemoteSha b17a3a55671a812906f4a244496119b6d97d9255

Contents

add_test_plan . . . 2
along_test_plans . . . 3
check . . . 3
check_plan . . . 5
evaluate_expr . . . 6
load_tests . . . 6
make_test . . . 7
register_test . . . 8
run_test . . . 8
run_tests . . . 9
skip_test . . . 10
subset_tests . . . 10

Index **11**

`add_test_plan` *Add a Future Plan to Test Against*

Description

Add a Future Plan to Test Against

Usage

```
add_test_plan(expr, substitute = TRUE)
```

Arguments

`expr`	...
`substitute`	...

Value

(invisibly) returns current list of test plans.

along_test_plans *Evaluate an Expression Across A Set of Future Plans*

Description

Evaluate an Expression Across A Set of Future Plans

Usage

```
along_test_plans(
  expr,
  substitute = TRUE,
  envir = parent.frame(),
  local = TRUE,
  plans = test_plans()
)
```

Arguments

expr	An R expression.
substitute	...
envir	The environment where tests are run.
local	Should tests be evaluated in a local environment or not.
plans	A list of future plans.

Value

A list of results, one for each future plan tested against. If evaluating `expr` under a plan produces an error, then the corresponding element is the error base::condition, and evaluation continues with the next plan.

check *Run All or a Subset of the Tests Across Future Plans*

Description

Run All or a Subset of the Tests Across Future Plans

Usage

```
check(
  plan = NULL,
  tags = character(),
  index = NULL,
  timeout = NULL,
  settings = TRUE,
  session_info = FALSE,
  envir = parent.frame(),
  local = TRUE,
  debug = FALSE,
  exit_value = !interactive(),
  .args = commandArgs()
)
```

Arguments

plan	(character vector) One or more future strategy plans to be validated.
tags	(character vector; optional) Filter tests by tags. Only tests that have all of the specified tags are performed. If NULL, all tests are performed.
index	(integer vector; optional) Indices of the tests to run, after any filtering by tags. These are the test numbers reported in the output. If NULL, all tests are performed.
timeout	(numeric; optional) Maximum time (in seconds) each test may run before a timeout error is produced. If NULL, option future.tests.timeout is used, which defaults to 30 seconds.
settings	(logical) If TRUE, details on the settings are outputted before the tests start.
session_info	(logical) If TRUE, session information is outputted after the tests complete.
envir	The environment where tests are run.
local	Should tests be evaluated in a local environment or not.
debug	(logical) If TRUE, the raw test results are printed.
exit_value	(logical) If TRUE, and in a non-interactive session, then use base::quit() to quit R with an exit code of 0 (zero) if all tests passed with all OKs and otherwise 1 (one) if one or more tests failed.
.args	(character vector; optional) Command-line arguments.

Value

(list; invisible) A list of test results.

Command-line interface (CLI)

This function can be called from the shell. To specify an argument, use the format `--test-<arg_name>=<value>`. For example, `--test-timeout=600` will set argument `timeout=600`, and `--test-tags=lazy,rng`, or equivalently, `--test-tags=lazy --test-tags=rng` will set argument `tags=c("lazy", "rng")`.

Similarly, `--test-index=42` and `--test-index=2:13,14` select test 42 and tests 2, 3, ..., 13 and 14, respectively.

Here are some examples on how to call this function from the command line:

```
Rscript -e future.tests::check --args --test-plan=sequential
Rscript -e future.tests::check --args --test-plan=multicore,workers=2
Rscript -e future.tests::check --args --test-plan=sequential --test-plan=multicore,workers=2
Rscript -e future.tests::check --args --test-plan=future.callr::callr
Rscript -e future.tests::check --args --test-plan=future.batchtools::batchtools_local
```

The exit code will be 0 if all tests passed, otherwise 1. You can use for instance `exit_code=$?` to retrieve the exit code of the most recent call.

Examples

```
## Not run:
results <- future.tests::check(plan = "sequential", tags = c("rng"))
exit_code <- attr(results, "exit_code")
if (exit_code != 0) stop("One or more tests failed")

## End(Not run)
```

---

check_plan *Run All Tests*

Description

Run All Tests

Usage

```
check_plan(
  tests = test_db(),
  defaults = list(),
  timeout = getOption("future.tests.timeout", 30),
  envir = parent.frame(),
  local = TRUE
)
```

Arguments

`tests`	A list of tests to subset.
`defaults`	(optional) Named list with default argument values.
`timeout`	Maximum time allowed for evaluation before a timeout error is produced.
`envir`	The environment where tests are run.
`local`	Should tests be evaluated in a local environment or not.

Value

Nothing.

`evaluate_expr` *Evaluate an R Expression*

Description

Evaluate an R Expression

Usage

```
evaluate_expr(
  expr,
  envir = parent.frame(),
  local = TRUE,
  output = c("stdout+stderr", "stdout", "none"),
  timeout = +Inf
)
```

Arguments

`expr`	An expression
`envir`	The environment where tests are run.
`local`	Should tests be evaluated in a local environment or not.
`output`	Specifies whether standard output, standard error, or both should be captured or not.
`timeout`	Maximum time allowed for evaluation before a timeout error is produced.

Value

Value of test expression and benchmark information.

`load_tests` *Loads Future Tests*

Description

Loads Future Tests

Usage

```
load_tests(
  path = ".",
  recursive = TRUE,
  pattern = "[.]R$",
  root = getOption("future.tests.root", Sys.getenv("R_FUTURE_TESTS_ROOT",
    system.file("test-db", package = "future.tests", mustWork = TRUE)))
)
```

Arguments

path	A character string specifying a test script folder or file.
recursive	If TRUE, test-definition scripts are searched recursively.
pattern	Regular expression matching filenames to include.
root	(internal) An alternative file directory from where **future.tests** tests are sourced.

Value

(invisible) the value of `test_db()`.

---

make_test *Make a Test*

Description

Make a Test

Usage

```
make_test(
  expr,
  title = NA_character_,
  args = list(),
  tags = NULL,
  substitute = TRUE,
  reset_workers = FALSE,
  register = !("devel" %in% tags) || isTRUE(getOption("future.tests.devel"))
)
```

Arguments

expr, substitute	The expression to be tested and whether it is passed as an expression already or not.
title	(character) The title of the test.
args	(optional) Named arguments.

tags	(optional) Character vector of tags.
reset_workers	(optional) Specifies whether background workers should futures.
register	If TRUE, the test is registered in the test database, otherwise not. The default is to register the test, unless tags contains "devel".

Value

(invisibly) A Test.

`register_test` *Register a Test*

Description

Register a Test

Usage

```
register_test(test)
```

Arguments

test	A Test.

Value

(invisibly) The Test registered.

`run_test` *Run a Test*

Description

Run a Test

Usage

```
run_test(
  test,
  envir = parent.frame(),
  local = TRUE,
  args = list(),
  defaults = list(),
  output = "stdout+stderr",
  timeout = getOption("future.tests.timeout", 30)
)
```

Arguments

test	A Test.
envir	The environment where tests are run.
local	Should tests be evaluated in a local environment or not.
args	Arguments used in this test.
defaults	(optional) Named list with default argument values.
output	If TRUE, standard output is captured, otherwise not.
timeout	Maximum time allowed for evaluation before a timeout error is produced.

Value

Value of test expression and benchmark information.

---

run_tests *Run All Tests*

Description

Run All Tests

Usage

```
run_tests(
  tests = test_db(),
  envir = parent.frame(),
  local = TRUE,
  defaults = list(),
  output = "stdout+stderr",
  timeout = getOption("future.tests.timeout", 30)
)
```

Arguments

tests	A list of tests to subset.
envir	The environment where tests are run.
local	Should tests be evaluated in a local environment or not.
defaults	(optional) Named list with default argument values.
output	If TRUE, standard output is captured, otherwise not.
timeout	Maximum time allowed for evaluation before a timeout error is produced.

Details

Test arguments, declared by `args` of `make_test()`, that are neither specified via `defaults` nor exist in `envir`, are set to the first of the values declared by the test.

Value

List of test results.

`skip_test` *Skip The Current Test*

Description

Signals a `TestSkipped` condition.

Usage

```
skip_test(..., domain = NULL)
```

Arguments

`...`	zero or more objects which can be coerced to character (and which are pasted together with no separator) or (for `message` only) a single condition object.
`domain`	see `gettext`. If `NA`, messages will not be translated, see also the note in `stop`.

Value

(invisible) A base::condition of class `TestSkipped`.

`subset_tests` *Identify Subset of Tests with Specified Tags and that Support Specified Argument Settings*

Description

Identify Subset of Tests with Specified Tags and that Support Specified Argument Settings

Usage

```
subset_tests(tests = test_db(), tags = NULL, args = NULL, defaults = list())
```

Arguments

`tests`	A list of tests to subset.
`tags`	(optional) A character vector of tags that tests must have.
`args`	Named arguments with sets of values to test against.
`defaults`	(optional) Named list with default argument values.

Value

A list of tests that support specified arguments.

Index

add_test_plan, 2
along_test_plans, 3

base::condition, *3*, *10*
base::quit(), *4*

check, 3
check_plan, 5

evaluate_expr, 6

gettext, *10*

load_tests, 6

make_test, 7
make_test(), *9*

register_test, 8
run_test, 8
run_tests, 9

skip_test, 10
stop, *10*
subset_tests, 10