

Package ‘mutator’

October 21, 2026

Title Mutation Testing

Version 0.3.0

Description Performs mutation testing for 'R' packages to assess the effectiveness of a test suite. It mutates source files, runs package tests against each mutant, and reports which mutants were killed, survived, or timed out. Parallel execution is supported for larger code bases. The optional 'imputesrcref' package, available from <https://github.com/PRL-PRG/imputesrcref>, improves source locations when installed.

URL <https://github.com/PRL-PRG/mutator>,
<https://prl-prg.github.io/mutator/>

BugReports <https://github.com/PRL-PRG/mutator/issues>

License GPL (>= 3)

Encoding UTF-8

Imports jsonlite, httr, pkgload, testthat, future, furr, callr, covr,
R6

Suggests xml2, pkgbuild, cli, pbmcapply, knitr, rmarkdown, tinytest

Enhances imputesrcref

LinkingTo testthat

Config/testthat/edition 3

VignetteBuilder knitr

SystemRequirements C++17

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Pierre Donat-Bouillud [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0003-4455-1130>),
Assanali Amandykov [aut, cph]

Maintainer Pierre Donat-Bouillud <pierre.donat.bouillud@fit.cvut.cz>

Repository CRAN

Date/Publication 2026-10-21 16:50:07 UTC

Contents

get_openai_config	2
identify_equivalent_mutants	3
mutate_file	4
mutate_package	5
mutation_operators	9
reset_openai_config	10
set_openai_config	10
Index	12

get_openai_config	<i>Get OpenAI API configuration</i>
-------------------	-------------------------------------

Description

Resolves the API key, model and base URL used for equivalent-mutant detection. Each field is resolved independently, with this precedence (highest first):

Usage

```
get_openai_config(dir = getwd())
```

Arguments

dir	Directory to search for a .openai_config file. Only this directory is consulted (parent directories are not). Defaults to the current working directory; pass NULL to ignore config files.
-----	--

Details

1. values set with `set_openai_config()`;
2. a .openai_config file in dir (a human-readable "field: value" file that is parsed, never executed);
3. the environment variables OPENAI_API_KEY, OPENAI_MODEL, OPENAI_BASE_URL and OPENAI_MAX_PARALLEL_REQUESTS;
4. built-in defaults (model "gpt-4", the public OpenAI base URL).

Value

A list with elements api_key, model, base_url and max_parallel_requests (an integer cap on concurrent API requests, or NA for no cap).

Examples

```
config <- get_openai_config()
names(config)
```

`identify_equivalent_mutants`*Identify equivalent mutants using OpenAI API*

Description

Analyzes survived mutants to determine if they are functionally equivalent to the original code using OpenAI's language models.

Usage

```
identify_equivalent_mutants(  
  src_file,  
  survived_mutants,  
  api_config = NULL,  
  batch_size = 25,  
  workers = 1,  
  report = TRUE  
)
```

Arguments

<code>src_file</code>	Path to the original source file
<code>survived_mutants</code>	List of mutants that survived test execution
<code>api_config</code>	Optional API configuration (will be loaded if NULL)
<code>batch_size</code>	Maximum number of mutants sent in a single API request. Smaller batches keep each response short enough to avoid truncation (which silently drops verdicts) and let batches run concurrently. Defaults to 25.
<code>workers</code>	Number of API requests to run concurrently (requires a forking platform). Defaults to 1 (sequential).
<code>report</code>	Whether to print per-file progress and an equivalence summary to the console. Defaults to TRUE. <code>mutate_package()</code> sets this to FALSE when running many files in parallel, so it can print one aggregated summary (and a single progress bar) instead of one per batch.

Value

Updated list of survived mutants with equivalence information

Examples

```
src <- tempfile(fileext = ".R")  
writeLines("add <- function(x, y) x + y", src)  
survived <- list(mutant_001 = list(mutation_info = "x + y -> x - y"))  
suppressWarnings(identify_equivalent_mutants(  
  src_file = src,  
  survived_mutants = survived,  
  batch_size = 1,  
  workers = 1,  
  report = TRUE  
))
```

```

    src,
    survived,
    api_config = list(api_key = "", model = "gpt-4")
  ))

```

mutate_file

Generate Mutants for a Single R File

Description

Creates mutants for a single R source file by combining AST-based mutations from the C++ mutation engine with fallback line-deletion mutants.

Usage

```

mutate_file(
  src_file,
  out_dir,
  max_mutants = NULL,
  max_line_deletions = 5,
  operators = NULL
)

```

Arguments

src_file	Path to an R source file.
out_dir	Directory where mutant files are written. This argument is required so that the function never writes to the current working directory unless the caller explicitly requests it.
max_mutants	Optional cap on the number of returned mutants. If set, a random subset of generated mutants is returned.
max_line_deletions	Maximum number of line-deletion mutants generated per file (a random subset of deletable lines). These complement the AST-based statement deletions by also covering top-level / non-block lines. Use 0 to disable line-deletion mutants entirely. Defaults to 5.
operators	Mutation operators to apply: a character vector of operator ids and families, read left to right, where a - prefix removes (e.g. c("default", "seq_idiom", "-bool_flip")). NULL uses the option <code>mutator.operators</code> , or "default" if it is unset. See mutation_operators() .

Details

Mutants that cannot change behaviour are not written: those whose code is identical to the original or to another mutant, and those that simple rules show to be equivalent (for example `paste0` with a single argument). A message gives their number per file.

Value

A list of mutants. Each element contains:

`path` Path to the mutant file.

`info` Formatted mutation metadata (file, source range, and details).

`loc` Machine-readable location: a list with `file_path`, `start_line`, and `end_line` (the latter two NA when unavailable), and `operator_id`, the id of the mutation operator that produced the mutant (e.g. "rel_swap", "stmt_delete").

Examples

```
src <- tempfile(fileext = ".R")
writeLines("add <- function(x, y) x + y", src)
mutants <- mutate_file(src, out_dir = tempfile("mutations_"), max_mutants = 1)
length(mutants)
```

mutate_package	<i>Run Mutation Testing for an R Package</i>
----------------	--

Description

Mutates all .R files under a package's R/ directory, runs the package's tests against each mutant in parallel, and summarizes mutation outcomes.

Usage

```
mutate_package(
  pkg_dir,
  cores = max(1, parallel::detectCores() - 2),
  isFullLog = FALSE,
  detectEqMutants = FALSE,
  mutation_dir = NULL,
  max_mutants = NULL,
  timeout_seconds = NULL,
  config_dir = getwd(),
  max_line_deletions = 0,
  cran = TRUE,
  fail_fast = TRUE,
  isolate = FALSE,
  exclude_files = NULL,
  strategy = c("auto", "testthat", "tinytest", "tinytest-installed", "installed"),
  coverage_guided = TRUE,
  coverage_backend = c("record_tests", "per_file"),
  target_margin = NULL,
  confidence = 0.95,
  max_show = 50L,
```

```

    operators = NULL
  )

```

Arguments

pkg_dir	Path to the package directory.
cores	Number of parallel workers used for mutant test execution.
isFullLog	Logical; if TRUE, prints per-mutant logs, timeout info, and the mutation score per operator.
detectEqMutants	Logical; if TRUE, every generated mutant is analyzed for equivalence using the OpenAI-based workflow <i>before</i> the test suites are run. Mutants judged equivalent are recorded as survived without running their tests, as no test can kill an equivalent mutant; the remaining mutants are tested as usual.
mutation_dir	Optional directory to store generated mutant files. If NULL, a temporary directory is used.
max_mutants	Sample that number of mutants for testing. If NULL, all mutants are tested.
timeout_seconds	Optional timeout in seconds for each mutant run. If NULL, timeout is derived from baseline runtime with a small minimum floor. Still works with compiled native code.
config_dir	Directory searched for a .openai_config file when detectEqMutants = TRUE (see get_openai_config()). Defaults to the current working directory.
max_line_deletions	Maximum number of line-deletion mutants per .R file (passed to mutate_file()); 0 disables them. Defaults to 0, since line-deletion mutants are largely redundant with the AST block-deletion mutants generated by default.
cran	Logical; if TRUE (the default), tests run in "CRAN mode": the NOT_CRAN environment variable is set to "false" in the test subprocess so <code>testthat::skip_on_cran()</code> / <code>skip_if_offline()</code> guards take effect and the same tests CRAN would run are used (skipping network/slow tests the package marks). Set to FALSE to run the full suite (NOT_CRAN = "true"), as <code>devtools::test()</code> does.
fail_fast	Logical; if TRUE (the default), a mutant's test run stops at the first failing test rather than running the whole suite. A mutant is KILLED as soon as one test detects it, so the remainder of the suite is wasted work. Set to FALSE to run the full suite for every mutant. Applies to the <code>testthat</code> strategy; the <code>tinytest</code> strategy always runs the full set of selected test files, and the <code>installed-tests</code> fallback already stops at the first failing test file regardless of this flag.
isolate	Logical; if FALSE (the default), each mutant's package copy symlinks the unchanged directories of the original package (only the mutated R/ file is materialised), which is fast but makes those directories shared writable state across the parallel workers. If TRUE, the <code>src/</code> and <code>tests/</code> directories (or <code>src/</code> and <code>inst/</code> under the <code>tinytest</code> strategy) are deep-copied into every mutant copy instead. Use <code>isolate = TRUE</code> when a package has non-hermetic tests that write files into <code>tests/</code> (or <code>src/</code>) and parallel runs therefore produce spurious KILLED/HANG verdicts; it gives each worker its own copy at the cost of extra disk. Note that running with <code>cores = 1</code> avoids such contention without the copy cost.

exclude_files	Optional character vector of shell-style glob patterns (e.g. "import-standalone-*") matched against the base names of the .R files in R/. Matching files are skipped entirely before any mutants are generated. NULL (the default) mutates every file. This complements the in-source # mutator:ignore-file and # mutator:ignore-start / # mutator:ignore-end directives, which exclude a whole file or a line region from within the source itself, optionally only for the operators they list (e.g. # mutator:ignore-start seq_idiom, constants). Note that for operator mutations the engine only resolves positions to the enclosing top-level definition, so a region directive excludes that function's operator mutants as a group (line-deletion mutants are excluded line-precisely).
strategy	Test strategy to use. "auto" (the default) picks the testthat strategy when tests/testthat/ exists, the tinytest strategy when inst/tinytest/ exists, and the installed-tests strategy otherwise. "testthat" forces the in-process testthat::test_dir() path (requires tests/testthat/). "tinytest" forces the in-process tinytest::run_test_dir() path (requires inst/tinytest/). Both in-process strategies load the package with pkgload::load_all(), which does not dispatch S4 methods defined on ...-dispatching base generics such as seq(); a package that relies on those will fail the baseline, and the error points to "tinytest-installed". "tinytest-installed" runs the tinytest suite against an installed copy (R CMD INSTALL + tinytest::test_package(), requires inst/tinytest/): slower than dev-mode but matches an installed package (correct S4 dispatch) and still supports coverage guidance. "installed" forces the R CMD INSTALL --install-tests + tools::testInstalledPackage() path (requires tests/).
coverage_guided	Logical; if TRUE, only the tests that actually exercise a mutant's mutated line(s) are run for that mutant, instead of the whole suite. Coverage is measured once on the unmutated package with covr. A mutant on a line no test covers cannot be killed, so it is reported SURVIVED without running any test. Selection is at the test-file level; under the assumption that the suite deterministically exercises the code, it should not change a mutant's verdict, only which tests run. Defaults to TRUE. Coverage guidance is available under the testthat, tinytest, and tinytest-installed strategies; when the resolved strategy is the generic installed-tests fallback, mutator emits a warning and runs the full suite for every mutant. Pass FALSE to disable it (and silence that warning).
coverage_backend	How coverage_guided attributes coverage to tests under the testthat strategy (ignored when coverage_guided = FALSE and for the tinytest strategies, which have a single per-file driver). "record_tests" (the default) uses covr's record_tests in a single run; it relies only on covr's public output but, because covr credits a covered line to the deepest test-directory frame, code reached through a helper-*.R/setup-*.R wrapper is attributed to the helper rather than the originating test-*.R file, and such mutants conservatively run the whole suite. "per_file" instruments the package once and runs the suite a single time through a reporter that snapshots coverage per test file, giving exact file-level attribution (no helper fallback) at roughly the same cost; it depends on covr internals, so it is opt-in.
target_margin	Optional desired half-width of the confidence interval on the mutation score,

	as a proportion (e.g. 0.05 for +/-5 percentage points). When set, the number of mutants to sample is derived from it using worst-case ($p = 0.5$) sizing at confidence, finite-population corrected and capped at the number of mutants generated (if the requested precision needs more mutants than exist, all are tested). Mutually exclusive with <code>max_mutants</code> . The required sample size depends on the target precision, not on program size (Gopinath et al., ISSRE 2015).
<code>confidence</code>	Confidence level for <code>target_margin</code> sizing and for the Wilson confidence interval reported on a sampled mutation score. Default 0.95.
<code>max_show</code>	Maximum number of surviving mutants to print to the console; the remainder are summarised as "... and N more" but always remain in the returned <code>package_mutants</code> . Use <code>Inf</code> to print every survivor. Default 50.
<code>operators</code>	Mutation operators to apply: a character vector of operator ids and families, read left to right, where a - prefix removes (e.g. <code>c("default", "seq_idiom", "-bool_flip")</code>). NULL uses the option <code>mutator.operators</code> , or "default" if it is unset. See mutation_operators() .

Details

The example is not run during routine automated checks because it creates and mutation-tests a throwaway package, which is too slow for that context.

Test strategy is, by default, detected automatically:

- If `tests/testthat/` exists, the mutant is loaded in-process with `pkgload::load_all()` (no installation) and its tests are run the way the package's own `tests/testthat.R` harness runs them, i.e. with the same arguments (notably any filter) that the harness passes to `testthat::test_check()`, via `testthat::test_dir()`.
- Otherwise, if `inst/tinytest/` exists, the mutant is loaded in-process with `pkgload::load_all()` (no installation) and its tests are run with `tinytest::run_test_dir("inst/tinytest")`.
- Otherwise, if `tests/` exists, `mutator` installs the mutant package with `--install-tests` and runs `tools::testInstalledPackage()`.

Pass `strategy` to override this (for example to run a `testthat` or `tinytest` package through the slower `installed-tests` path for comparison).

Value

An invisible list with four components:

`package_mutants` Named list with mutant path, mutation info, status, and optional equivalence flags.

`test_results` Named list mapping mutant IDs to statuses: "KILLED", "SURVIVED", or "HANG".

`timing` Named list of phase durations in seconds: `baseline`, `generation`, `test_execution`, and `equivalence_detection`.

`summary` Named list with `generated`, `tested`, `killed`, `hanged`, `survived`, `mutation_score`, `mutation_score_ci` (a length-2 percentage vector, or NULL when no sampling occurred), `confidence`, and `by_operator`, a data frame with one row per mutation operator and columns `operator`, `tested`, `killed`, `hanged`, `survived` and `mutation_score`.

Examples

```
pkg <- file.path(tempdir(), "examplepkg")
dir.create(file.path(pkg, "R"), recursive = TRUE, showWarnings = FALSE)
dir.create(file.path(pkg, "tests", "testthat"), recursive = TRUE, showWarnings = FALSE)
writeLines(c(
  "Package: examplepkg",
  "Title: Example Package",
  "Version: 0.0.1",
  "Description: Minimal package for a mutator example.",
  "License: GPL-3",
  "Encoding: UTF-8"
), file.path(pkg, "DESCRIPTION"))
writeLines("export(add)", file.path(pkg, "NAMESPACE"))
writeLines("add <- function(x, y) x + y", file.path(pkg, "R", "add.R"))
writeLines(
  "testthat::expect_equal(add(1, 2), 3)",
  file.path(pkg, "tests", "testthat", "test-add.R")
)
result <- mutate_package(pkg, cores = 1, max_mutants = 1, timeout_seconds = 10)
names(result)
```

mutation_operators	<i>List the Mutation Operators</i>
--------------------	------------------------------------

Description

Returns the mutation operators that mutator knows about. Use their ids, or the names of their families, in the operators argument of [mutate_file\(\)](#) and [mutate_package\(\)](#).

Usage

```
mutation_operators()
```

Value

A data frame with one row per operator and columns id, family, default (whether it is part of the "default" group), description and example.

Selecting operators

operators is a character vector read from left to right. Each element is an operator id, a family name (e.g. "relational"), "default" or "all". An element prefixed with - removes operators instead of adding them. If the first element is a removal, the selection starts from "default". When operators is NULL, the option `mutator.operators` is used, and "default" if that is unset.

Examples

```
mutation_operators()

# Default set plus the constants family, without NA retyping:
src <- tempfile(fileext = ".R")
writeLines("f <- function(x) if (x > 0) x + 1 else NA", src)
mutants <- mutate_file(src, out_dir = tempfile("mutations_"),
                      operators = c("default", "constants", "-na_type_swap"))
```

```
reset_openai_config    Clear session OpenAI configuration
```

Description

Removes any values set with `set_openai_config()`, reverting to configuration taken from a `.openai_config` file or environment variables.

Usage

```
reset_openai_config()
```

Value

Invisibly NULL.

Examples

```
set_openai_config(model = "gpt-4o-mini")
reset_openai_config()
```

```
set_openai_config    Set OpenAI API configuration for the current session
```

Description

Overrides the API key, model and/or base URL used by the equivalent-mutant detection workflow. Values set here take precedence over a `.openai_config` file and over environment variables. Arguments left NULL are unchanged.

Usage

```
set_openai_config(
  api_key = NULL,
  model = NULL,
  base_url = NULL,
  max_parallel_requests = NULL
)
```

Arguments

api_key	API key string.
model	Model name (e.g. "gpt-4").
base_url	Base URL of an OpenAI-compatible Chat Completions API, such as "https://api.openai.com/v1" or another provider endpoint.
max_parallel_requests	Maximum number of equivalence-detection API requests to run concurrently. Use this to stay under a provider's per-key parallel-request limit (exceeding it returns HTTP 429). NA (the default) imposes no cap beyond the run's own cores.

Value

Invisibly, the resulting configuration (see [get_openai_config\(\)](#)).

Examples

```
set_openai_config(model = "gpt-4o-mini")
get_openai_config()$model
reset_openai_config()
```

Index

`get_openai_config`, [2](#)
`get_openai_config()`, [6](#), [11](#)

`identify_equivalent_mutants`, [3](#)

`mutate_file`, [4](#)
`mutate_file()`, [6](#), [9](#)
`mutate_package`, [5](#)
`mutate_package()`, [9](#)
`mutation_operators`, [9](#)
`mutation_operators()`, [4](#), [8](#)

`operators (mutation_operators)`, [9](#)

`reset_openai_config`, [10](#)

`set_openai_config`, [10](#)
`set_openai_config()`, [2](#), [10](#)