Package: mutator (via r-universe)

July 15, 2026

Title Mutation Testing

Version 0.1.1.9000

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

Roxygen list(markdown = TRUE)

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

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

Enhances imputesrcref

LinkingTo testthat

Config/testthat/edition 3

VignetteBuilder knitr

SystemRequirements C++17

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev libssl-dev

Repository https://prl-prg.r-universe.dev

Date/Publication 2026-07-15 19:32:48 UTC

RemoteUrl https://github.com/PRL-PRG/mutator

RemoteRef HEAD

RemoteSha 5b099b9920a91b489ba59cf6beb814cdd2ecf081

Contents

get_openai_config 2
identify_equivalent_mutants 3
mutate_file 4
mutate_package 5
operators 9
reset_openai_config 9
set_openai_config 10

Index **11**

`get_openai_config` *Get OpenAI API configuration*

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

`src_file`	Path to the original source file
`survived_mutants`	List of mutants that survived test execution
`api_config`	Optional API configuration (will be loaded if NULL)
`batch_size`	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.
`workers`	Number of API requests to run concurrently (requires a forking platform). Defaults to 1 (sequential).
`report`	Whether to print per-file progress and an equivalence summary to the console. Defaults to `TRUE`. `mutate_package()` 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,
    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 = "mutations",
  max_mutants = NULL,
  max_line_deletions = 5
)
```

Arguments

`src_file`	Path to an R source file.
`out_dir`	Directory where mutant files are written.
`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.

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).

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` *Run Mutation Testing for an R Package*

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", "installed"),
  coverage_guided = TRUE,
  coverage_backend = c("record_tests", "per_file"),
  target_margin = NULL,
  confidence = 0.95,
  max_show = 50L
)
```

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 and timeout info.

`detectEqMutants`
: Logical; if `TRUE`, every generated mutant is analyzed for equivalence using the OpenAI-based workflow *before* 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 `testthat::skip_on_cran()` / `skip_if_offline()` 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 `devtools::test()` 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 `testthat` strategy; the installed-tests 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 `src/` and `tests/` directories are deep-copied into every mutant copy instead. Use `isolate = TRUE` when a package has **non-hermetic tests** that write files into `tests/` (or `src/`) 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 `cores = 1` 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. 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 and the installed-tests strategy otherwise. `"testthat"` forces the in-process `testthat::test_dir()` path (requires `tests/testthat/`). `"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** (`options(covr.record_tests = TRUE)`). 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 (testthat filters by file); 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 only available under the `testthat` strategy; when the resolved strategy is the 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 (ignored when `coverage_guided = FALSE`). `"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 `max_mutants`. The required sample size depends on the target precision, not on program size (Gopinath et al., ISSRE 2015).

`confidence` Confidence level for `target_margin` sizing and for the Wilson confidence interval reported on a sampled mutation score. Default 0.95.

`max_show` Maximum number of surviving mutants to print to the console; the remainder are summarised as "... and N more" but always remain in the returned `package_mutants`. Use `Inf` to print every survivor. Default 50.

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 `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` 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), and `confidence`.

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)
```

operators *Mutation Operators Supported by mutator*

Description

This help page lists the mutation operators that the `mutator` package supports for mutation testing in R.

Details

The following mutation operators can be applied to R code:

- Arithmetic operators: `+` <-> `-`, and `*` <-> `/`
- Comparison operators: `==` <-> `!=`, `<` <-> `>`, and `<=` <-> `>=`
- Logical operators: `\&` <-> `|`, `&&` <-> `||`, removes `!`, and negates `if` / `while` conditions
- Assignment and call values: replaces assignment right-hand sides and ordinary function calls with `42`
- Scalar constants: replaces numeric zero with `42`, numeric non-zero values with `0`, constants with a typed `NA`, and constants with `NULL`
- Returns: replaces non-constant direct `return()` values with `NULL`, for example `return(x)` -> `return(NULL)`
- Deletions: removes statements inside `{ ... }` blocks and, as a fallback, valid source lines

Direct literal return values are not rewritten by the return-to-`NULL` mutation; for example, `return(1)` is left alone by that mutation.

Author(s)

Assanali Amandykov and Pierre Donat-Bouillud

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*, *10*

identify_equivalent_mutants, 3

mutate_file, 4
mutate_file(), *6*
mutate_package, 5

operators, 9

reset_openai_config, 9

set_openai_config, 10
set_openai_config(), *2*, *9*