Package 'resultcheck'

October 26, 2026

Title Result Stability Checks for Empirical R Projects

Version 0.3.3

Description Lightweight helpers for checking whether empirical results remain substantively unchanged across code revisions, platform differences, and package updates. The package supports regression-style testing of derived datasets, statistical model outputs, tables, and plots, helping researchers detect unintended result drift early and distinguish material from non-material changes in empirical workflows.

License MIT + file LICENSE

Encoding UTF-8

Imports withr, rprojroot, yaml

Suggests broom, knitr, rmarkdown, rstudioapi, testthat (>= 3.0.0), waldo

URL https://github.com/kv9898/resultcheck/, https://kv9898.github.io/resultcheck/

BugReports https://github.com/kv9898/resultcheck/issues

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Dianyi Yang [aut, cre, ctb] (ORCID: <https://orcid.org/0009-0004-4652-3429>)

Maintainer Dianyi Yang <dianyi.yang@politics.ox.ac.uk>

Repository CRAN

Date/Publication 2026-10-26 12:00:02 UTC

Contents

cleanup_sandbox 2
find_root 2
run_in_sandbox 3

setup_sandbox 4
snapshot 5
with_example 6

Index **8**

`cleanup_sandbox` *Clean Up a Sandbox Environment*

Description

Removes a sandbox directory and all its contents. This should be called after testing is complete to free up disk space.

Usage

```
cleanup_sandbox(sandbox = NULL, force = TRUE)
```

Arguments

`sandbox`	Optional. A sandbox object created by `setup_sandbox()`. If NULL (default), cleans up the most recently created sandbox.
`force`	Logical. If TRUE (default), removes directory even if it contains files.

Value

Logical indicating success (invisible).

Examples

```
with_example({
  sandbox <- setup_sandbox()
  cleanup_sandbox(sandbox)
})
```

`find_root` *Find Project Root Directory*

Description

Finds the root directory of the current R project using various heuristics. The function searches for markers like `_resultcheck.yml` (preferred), `resultcheck.yml` (legacy), `.Rproj` files, or a `.git` directory. When running inside a sandbox created by `setup_sandbox()`, it will search from the original working directory.

Usage

```
find_root(start_path = NULL)
```

Arguments

start_path	Optional. The directory to start searching from. If NULL (default), uses the current working directory or the stored original working directory if in a sandbox.

Value

The path to the project root directory.

Examples

```
with_example({
  root <- find_root()
  print(root)
})
```

---

run_in_sandbox *Run Code in a Sandbox Environment*

Description

Executes an R script within a sandbox directory, suppressing messages, warnings, and graphical output. This is useful for testing empirical analysis scripts without polluting the console or creating unwanted plots.

Usage

```
run_in_sandbox(
  script_path,
  sandbox = NULL,
  suppress_messages = TRUE,
  suppress_warnings = TRUE,
  capture_output = TRUE
)
```

Arguments

script_path	Path to the R script to execute.
sandbox	Optional. A sandbox object created by `setup_sandbox()`. If NULL (default), uses the most recently created sandbox.
suppress_messages	Logical. Whether to suppress messages (default: TRUE).
suppress_warnings	Logical. Whether to suppress warnings (default: TRUE).
capture_output	Logical. Whether to capture output (default: TRUE).

Value

Invisible TRUE on successful execution.

Examples

```
with_example({
  sandbox <- setup_sandbox()
  on.exit(cleanup_sandbox(sandbox), add = TRUE)
  run_in_sandbox("analysis.R", sandbox)
})
```

`setup_sandbox` *Setup a Sandbox Environment for Testing*

Description

Creates a temporary directory and copies specified files and/or directories into it while preserving their path structure. This is useful for testing empirical analysis scripts in isolation.

Usage

```
setup_sandbox(files = NULL, temp_base = NULL)
```

Arguments

`files`	Character vector of relative file or directory paths to copy to the sandbox. Leave as `NULL` (default) to create an empty sandbox. Paths are resolved relative to the project root (found using `find_root()`); if the project root cannot be determined the current working directory is used. When a path refers to a directory, the entire directory is copied recursively. Absolute paths and path traversal attempts (e.g., `..`) are rejected for security. Snapshot files do *not* need to be listed here: `snapshot()` always reads snapshots from the project root, not from the sandbox.
`temp_base`	Optional. Custom location for the temporary directory. If NULL (default), uses `tempfile()`.

Value

A list with class "resultcheck_sandbox" containing:

`path`	The path to the created temporary directory
`id`	A unique timestamp-based identifier for this sandbox

Examples

```
with_example({
  sandbox <- setup_sandbox()
  print(sandbox$path)
  cleanup_sandbox(sandbox)
})
```

snapshot *Interactive Snapshot Testing*

Description

Creates or updates a snapshot of an R object for interactive analysis. On first use, saves the object to a human-readable snapshot file (.md). On subsequent uses, compares the current object to the saved snapshot.

Usage

```
snapshot(value, name, script_name = NULL, method = NULL)
```

Arguments

value	The R object to snapshot (e.g., plot, table, model output).
name	Character. A descriptive name for this snapshot.
script_name	Optional. The name of the script creating the snapshot. If NULL, detects from the active Quarto document or knitr input, then the innermost `source()` or `sys.source()` input, call-stack source references, the active source editor, and finally `"interactive"`.
method	Optional function or non-empty list of functions used to serialize `value`. Functions are executed in order and each section header is taken from the method expression or list name. If omitted, method defaults are resolved in this order: `snapshot.method_by_class` (matched by object class), then `snapshot.method`, then `list(print = base::print, str = utils::str)`.

Details

In interactive mode (default), prompts the user to update if differences are found and emits a warning. In testing mode (inside `run_in_sandbox()`), throws an error if a snapshot doesn't exist or doesn't match. During a Quarto render, missing snapshots are created, while mismatches throw an error and stop rendering.

Snapshots are stored under `tests/_resultcheck_snaps/` by default, organized by script name, and configurable via `snapshot.dir` in `_resultcheck.yml`. Method defaults can be configured via `snapshot.method` and class-specific defaults via `snapshot.method_by_class`. Optional class defaults can also be loaded from an R file using `snapshot.method_defaults_file`. Method strings in config (for example `"print + str"` or `"stats::coef"`) are resolved to callable functions. In config expressions, `"+"` is treated as the method delimiter.

Built-in class defaults (loaded from inst/extdata/snapshot-method-defaults.R) use broom functions for many statistical model classes. The default method is typically broom::tidy, with broom::glance and/or broom::augment added where supported (per the broom available-methods table at https://broom.tidymodels.org/articl methods.html).

Set `snapshot.max_print` in the project's `_resultcheck.yml` (or legacy `resultcheck.yml`) to control base R printing during serialization. The default is 1000 entries per method, independent of

session `max.print`. Values must be whole numbers from 1 to 2147483647; omitted or null values use the default. The setting also applies when `method` is supplied. The caller's options are restored even if a method fails.

Entries are not rows or bytes: a 200-row, seven-column data frame contains 1400 entries and requires a higher limit, for example `max_print: 2000`. Base output beyond the limit can be truncated with an omission notice; changes in omitted values may not be detected. Increase the limit or select an appropriate summary method. Class-specific methods, including tibble printing, may use their own limits; custom methods may explicitly override this setting. Review and regenerate affected baselines when changing the limit or upgrading from session-dependent printing.

Value

Invisible TRUE if the snapshot matches, is created, or is updated. In testing mode, throws an error if the snapshot is missing or doesn't match. During a Quarto render, throws an error only on mismatch.

Examples

```
with_example({
  model <- stats::lm(mpg ~ wt, data = datasets::mtcars)

  # Create a snapshot, then check that the same result matches.
  snapshot(model, "model", script_name = "analysis")
  snapshot(model, "model", script_name = "analysis")

  # Supply a function to snapshot a particular part of the result.
  snapshot(model, "coefficients", script_name = "analysis", method = stats::coef)
})
```

`with_example` *Run Code Inside a Temporary Example Project*

Description

Creates a self-contained example project under `tempdir()`, including:

- `_resultcheck.yml` (project root marker)
- `analysis.R` with `snapshot(model, "model")`
- matching and mismatched snapshot files
- `tests/testthat/test-analysis.R`

then temporarily sets the working directory to that project while evaluating `code`.

Usage

```
with_example(code, mismatch = FALSE)
```

Arguments

code	Code to evaluate inside the temporary example project.
mismatch	Logical. If TRUE, replaces the active snapshot with a mismatched version before evaluating code.

Value

The value of code.

Examples

```
with_example({
  root <- find_root()
  print(root)
})
```

Index

cleanup_sandbox, 2

find_root, 2

run_in_sandbox, 3

setup_sandbox, 4
snapshot, 5

with_example, 6