Package: snapr (via r-universe)

June 3, 2026

Title Convenient Snapshot Testing Functions for Packages

Version 0.1.0.9000

Description Provides convenient snapshot testing functions for packages, including expect_snapshot_data() for data.frames and expect_snapshot_object() for any R object.

License MIT + file LICENSE

Language en-US

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL https://github.com/d-morrison/snapr, https://d-morrison.github.io/snapr/

BugReports https://github.com/d-morrison/snapr/issues

Depends R (>= 4.1.0)

Imports dplyr, jsonlite, readr, testthat, tidyselect, tools, waldo

Suggests altdoc, bslib, knitr, quarto, rmarkdown, spelling, withr

VignetteBuilder knitr, quarto

Config/testthat/edition 3

Config/Needs/website quarto

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev libx11-dev

Repository https://d-morrison.r-universe.dev

Date/Publication 2026-06-03 18:22:20 UTC

RemoteUrl https://github.com/d-morrison/snapr

RemoteRef HEAD

RemoteSha 004536127e40ec419f0794e81812ef671daab5b3

Contents

`snapr-package` *snapr: Convenient Snapshot Testing Functions for Packages*

Description

Provides convenient snapshot testing functions for packages, including expect_snapshot_data() for data.frames and expect_snapshot_object() for any R object.

Author(s)

Maintainer: Douglas Ezra Morrison <demorrison@ucdavis.edu>

Authors:

- Douglas Ezra Morrison <demorrison@ucdavis.edu>

See Also

Useful links:

- https://github.com/d-morrison/snapr
- https://d-morrison.github.io/snapr/
- Report bugs at https://github.com/d-morrison/snapr/issues

`compare_file_object` *Compare RDS files using waldo for better snapshot review*

Description

This comparison function loads RDS files and compares the deserialized R objects rather than raw bytes. This enables better visualization in snapshot_review() by comparing the actual object structure rather than binary serialization.

For meaningful diffs in testthat's snapshot_review(), this function compares the R objects after deserialization using `waldo::compare()`.

Usage

```
compare_file_object(old, new, print = FALSE, ...)
```

Arguments

old	Path to the old (reference) RDS file
new	Path to the new RDS file to compare
print	logical whether to print waldo::compare output to R console; can become very long for complex objects like lms
...	Arguments passed on to waldo::compare

x,y Objects to compare. x is treated as the reference object so messages describe how y is different to x.

x_arg,y_arg Name of x and y arguments, used when generated paths to internal components. These default to "old" and "new" since it's most natural to supply the previous value then the new value.

tolerance If non-NULL, used as threshold for ignoring small floating point difference when comparing numeric vectors. Using any non-NULL value will cause integer and double vectors to be compared based on their values, not their types, and will ignore the difference between NaN and NA_real_.
It uses the same algorithm as all.equal(), i.e., first we generate x_diff and y_diff by subsetting x and y to look only locations with differences. Then we check that mean(abs(x_diff - y_diff)) / mean(abs(y_diff)) (or just mean(abs(x_diff - y_diff)) if y_diff is small) is less than tolerance.

max_diffs Control the maximum number of differences shown. The default shows 10 differences when run interactively and all differences when run in CI. Set max_diffs = Inf to see all differences.

ignore_srcref Ignore differences in function srcrefs? TRUE by default since the srcref does not change the behaviour of a function, only its printed representation.

ignore_attr Ignore differences in specified attributes? Supply a character vector to ignore differences in named attributes. By default the "waldo_opts" attribute is listed in ignore_attr so that changes to it are not reported; if you customize ignore_attr, you will probably want to do this yourself.
For backward compatibility with all.equal(), you can also use TRUE, to all ignore differences in all attributes. This is not generally recommended as it is a blunt tool that will ignore many important functional differences.

ignore_encoding Ignore string encoding? TRUE by default, because this is R's default behaviour. Use FALSE when specifically concerned with the encoding, not just the value of the string.

ignore_function_env,ignore_formula_env Ignore the environments of functions and formulas, respectively? These are provided primarily for backward compatibility with all.equal() which always ignores these environments.

list_as_map Compare lists as if they are mappings between names and values. Concretely, this drops NULLs in both objects and sorts named components.

quote_strings Should strings be surrounded by quotes? If FALSE, only side-by-side and line-by-line comparisons will be used, and there's no way to distinguish between NA and "NA".

Value

logical TRUE if objects are identical, FALSE otherwise

Examples

```
old_obj <- list(a = 1, b = 2)
new_obj <- list(a = 1, b = 3)
old_path <- tempfile(fileext = ".rds")
new_path <- tempfile(fileext = ".rds")
saveRDS(old_obj, old_path)
saveRDS(new_obj, new_path)
compare_file_object(old_path, new_path)
```

---

darwin_variant *Get darwin snapshot variant for macOS*

Description

Returns "darwin" for macOS, NULL for other platforms (Linux/Windows).

Usage

```
darwin_variant()
```

Details

This is a specialized variant function for cases where only macOS produces different results while Linux and Windows are identical. Use this instead of system_os() when:

- macOS produces unique output (e.g., platform-specific math libraries)
- Linux and Windows produce identical results
- You want to maintain a single snapshot for Linux/Windows

Common use cases:

- JAGS MCMC output (macOS uses different floating-point arithmetic)
- Platform-specific numerical computations

Value

"darwin" on macOS, NULL on other platforms

Examples

```
darwin_variant()
```

`expect_snapshot_data` *Snapshot testing for data.frames*

Description

copied from https://github.com/bcgov/ssdtools with permission (https://github.com/bcgov/ssdtools/issues/379)

Usage

```
expect_snapshot_data(x, name, digits = 6, ...)
```

Arguments

`x`	a data.frame to snapshot
`name`	character snapshot name
`digits`	integer passed to `signif()` for numeric variables
`...`	Arguments passed on to `testthat::expect_snapshot_file`
	`binary` **[Deprecated]** Please use the `compare` argument instead.
	`cran` Should these expectations be verified on CRAN? By default, they are not, because snapshot tests tend to be fragile because they often rely on minor details of dependencies.
	`transform` Optionally, a function to scrub sensitive or stochastic text from the output. Should take a character vector of lines as input and return a modified character vector as output.
	`variant` If not-`NULL`, results will be saved in `_snaps/{variant}/{test}/{name}`. This allows you to create different snapshots for different scenarios, like different operating systems or different R versions. Note that there's no way to declare all possible variants up front which means that as soon as you start using variants, you are responsible for deleting snapshot variants that are no longer used. (testthat will still delete all variants if you delete the test.)
	`old,new` Paths to old and new snapshot files.

Value

NULL (from `testthat::expect_snapshot_file()`)

Examples

```
# expect_snapshot_data() must be called inside a test_that() block with
# testthat 3rd edition active. Outside a test suite, the snapshot is
# skipped because there is no reference file to compare against.
withr::with_tempdir({
  testthat::test_that("iris snapshot", {
    testthat::local_edition(3)
```

```
    expect_snapshot_data(iris, name = "iris")
  })
})
```

`expect_snapshot_object` *Snapshot testing for R objects*

Description

A flexible wrapper around `testthat::expect_snapshot_file()` that allows snapshotting any R object, not just data.frames. This function provides a convenient interface for snapshot testing with sensible defaults for serialization.

When using RDS format (the default), snapshots are compared using `waldo::compare()` which provides rich, visual diffs in `testthat::snapshot_review()`. This makes it much easier to review changes to complex R objects.

Usage

```
expect_snapshot_object(
  x,
  name,
  writer = save_rds,
  print = FALSE,
  tolerance = NULL,
  ...
)
```

Arguments

`x`	An R object to snapshot. Can be any R object including lists, models, data.frames, vectors, etc.
`name`	character snapshot name (file extension added automatically)
`writer`	function Function to write the object to a file. Default is `save_rds()`. Other options include `save_json()`, `save_deparse()`, `save_csv()`. Custom writer functions should accept `x` and return a file path.
`print`	logical whether to print waldo::compare output to R console; can become very long for complex objects like lms
`tolerance`	If non-`NULL`, used as threshold for ignoring small floating point differences when comparing numeric vectors. Only applies when `writer` produces an RDS file (the default); silently ignored for text-based formats (JSON, CSV, deparse, etc.). See `waldo::compare()` for full details.
`...`	Arguments passed on to `testthat::expect_snapshot_file`
	`binary` **[Deprecated]** Please use the `compare` argument instead.

`cran` Should these expectations be verified on CRAN? By default, they are not, because snapshot tests tend to be fragile because they often rely on minor details of dependencies.

`transform` Optionally, a function to scrub sensitive or stochastic text from the output. Should take a character vector of lines as input and return a modified character vector as output.

`variant` If not-`NULL`, results will be saved in `_snaps/{variant}/{test}/{name}`. This allows you to create different snapshots for different scenarios, like different operating systems or different R versions.
Note that there's no way to declare all possible variants up front which means that as soon as you start using variants, you are responsible for deleting snapshot variants that are no longer used. (testthat will still delete all variants if you delete the test.)

`old,new` Paths to old and new snapshot files.

Details

When using RDS format (the default), snapshots can vary across R versions and platforms even with fixed serialization versions. Consider using `variant = platform_variant()` for RDS snapshots to handle these differences, or use text-based formats like JSON or deparse for more stable snapshots across platforms and versions.

The RDS comparison uses `waldo::compare()` internally, which provides rich visual diffs in `testthat::snapshot_review(`. This is particularly useful for complex objects like models, nested lists, or data structures where byte-level comparison would be difficult to interpret.

Value

NULL (from `testthat::expect_snapshot_file()`)

Examples

```
# expect_snapshot_object() must be called inside a test_that() block with
# testthat 3rd edition active. Outside a test suite, the snapshot is
# skipped because there is no reference file to compare against.
withr::with_tempdir({
  testthat::test_that("snapshot examples", {
    testthat::local_edition(3)

    # Snapshot a list (RDS format with platform/version variant)
    expect_snapshot_object(
      list(a = 1, b = 2), name = "config", variant = platform_variant()
    )

    # Snapshot a model
    model <- lm(mpg ~ wt, data = mtcars)
    expect_snapshot_object(
      model, name = "model", variant = platform_variant()
    )

    # Snapshot with JSON format (for human-readable diffs)
```

```
    # Text formats don't need variants
    expect_snapshot_object(iris[1:5, ], name = "iris", writer = save_json)

    # Snapshot with deparse format
    expect_snapshot_object(
      list(x = 1:5), name = "simple_list", writer = save_deparse
    )
  })
})
```

`platform_variant` *Get platform variant including OS and R version*

Description

Returns a variant string combining OS and R major.minor version. This is useful for RDS snapshots that vary by both platform and R version.

Usage

```
platform_variant()
```

Details

RDS files can produce different binary output across R versions even when using the same serialization version. This function creates variant strings like "linux-4.4", "darwin-4.5", "windows-4.4" to handle these differences.

Use this function with `testthat::expect_snapshot_file()` when testing RDS files or other formats that vary by both OS and R version.

Value

A character string like "linux-4.4", "darwin-4.5", or "windows-4.4"

Examples

```
platform_variant()
```

`system_os` *Get the current operating system*

Description

Returns the name of the current operating system in lowercase. This is a simple wrapper around `Sys.info()[["sysname"]]`.

Usage

```
system_os()
```

Details

This function is used with testthat's `variant` parameter in `testthat::expect_snapshot_file()` to create OS-specific snapshots when file formats produce different output across platforms.

Common use cases:

- RDS files: Binary serialization format that can differ across OS
- Platform-specific numeric computations (e.g., MCMC algorithms)
- Files with platform-specific line endings or encodings

Value

A character string: "linux", "darwin" (macOS), or "windows"

Examples

```
system_os()
```

Index