Package: vdiffr (via r-universe)

October 26, 2024

Title Visual Regression Testing and Graphical Diffing

Version 1.0.7

Encoding UTF-8

Description An extension to the 'testthat' package that makes it easy to add graphical unit tests. It provides a Shiny application to manage the test cases.

License MIT + file LICENSE

ByteCompile true

Depends R (>= 3.2.0)

Imports diffobj, glue, grDevices, htmltools, lifecycle, rlang, testthat (>= 3.0.3), xml2 (>= 1.0.0)

Suggests covr, ggplot2 (>= 3.2.0), roxygen2, withr

LinkingTo cpp11

RoxygenNote 7.1.2

URL https://vdiffr.r-lib.org/, https://github.com/r-lib/vdiffr

BugReports https://github.com/r-lib/vdiffr/issues

SystemRequirements C++11, libpng

Config/Needs/website tidyverse/tidytemplate

NeedsCompilation yes

Author Lionel Henry [cre, aut], Thomas Lin Pedersen [aut] (<https://orcid.org/0000-0002-5147-4711>), Posit Software, PBC [cph, fnd], T Jake Luciani [aut] (svglite), Matthieu Decorde [aut] (svglite), Vaudor Lise [aut] (svglite), Tony Plate [ctb] (svglite: Early line dashing code), David Gohel [ctb] (svglite: Line dashing code and raster code), Yixuan Qiu [ctb] (svglite: Improved styles; polypath implementation), Håkon Malmedal [ctb] (svglite: Opacity code)

Maintainer Lionel Henry <lionel@posit.co>

Repository CRAN

Date/Publication 2023-09-22 07:20:02 UTC

Contents

`expect_doppelganger` *Does a figure look like its expected output?*

Description

`expect_doppelganger()` is a testthat expectation for graphical plots. It generates SVG snapshots that you can review graphically with `testthat::snapshot_review()`. You will find more information about snapshotting in the testthat snapshots vignette.

Usage

```
expect_doppelganger(
  title,
  fig,
  path = deprecated(),
  ...,
  writer = write_svg,
  cran = FALSE
)
```

Arguments

`title`	A brief description of what is being tested in the figure. For instance: "Points and lines overlap". If a ggplot2 figure doesn't have a title already, `title` is applied to the figure with `ggtitle()`. The title is also used as file name for storing SVG (in a sanitzed form, with special characters converted to `"-"`).
`fig`	A figure to test. This can be a ggplot object, a recordedplot, or more generally any object with a `print` method. If you need to test a plot with non-printable objects (e.g. base plots), `fig` can be a function that generates and prints the plot, e.g. `fig = function() plot(1:3)`.
`path, ...`	**[Deprecated]**.
`writer`	A function that takes the plot, a target SVG file, and an optional plot title. It should transform the plot to SVG in a deterministic way and write it to the target file. See `write_svg()` (the default) for an example.
`cran`	If `FALSE` (the default), mismatched snapshots only cause a failure when you run tests locally or in your CI (Github Actions or any platform that sets the `CI` environment variable). If `TRUE`, failures may also occur on CRAN machines.

Failures are disabled on CRAN by default because testing the appearance of a figure is inherently fragile. Changes in the R graphics engine or in ggplot2 may cause subtle differences in the aspect of a plot, such as a slightly smaller or larger margin. These changes will cause spurious failures because you need to update your snapshots to reflect the upstream changes.

It would be distracting for both you and the CRAN maintainers if such changes systematically caused failures on CRAN. This is why snapshot expectations do not fail on CRAN by default and should be treated as a monitoring tool that allows you to quickly check how the appearance of your figures changes over time, and to manually assess whether changes reflect actual problems in your package.

Internally, this argument is passed to `testthat::expect_snapshot_file()`.

Debugging

It is sometimes difficult to understand the cause of a failure. This usually indicates that the plot is not created deterministically. Potential culprits are:

- Some of the plot components depend on random variation. Try setting a seed.
- The plot depends on some system library. For instance sf plots depend on libraries like GEOS and GDAL. It might not be possible to test these plots with vdiffr.

To help you understand the causes of a failure, vdiffr automatically logs the SVG diff of all failures when run under R CMD check. The log is located in `tests/vdiffr.Rout.fail` and should be displayed on Travis.

You can also set the `VDIFFR_LOG_PATH` environment variable with `Sys.setenv()` to unconditionally (also interactively) log failures in the file pointed by the variable.

Examples

```
if (FALSE) {  # Not run

library("ggplot2")

test_that("plots have known output", {
  disp_hist_base <- function() hist(mtcars$disp)
  expect_doppelganger("disp-histogram-base", disp_hist_base)

  disp_hist_ggplot <- ggplot(mtcars, aes(disp)) + geom_histogram()
  expect_doppelganger("disp-histogram-ggplot", disp_hist_ggplot)
})

}
```

`write_svg` *Default SVG writer*

Description

This is the default SVG writer for vdiffr test cases. It uses embedded versions of svglite, harfbuzz, and the Liberation and Symbola fonts in order to create deterministic SVGs.

Usage

```
write_svg(plot, file, title = "")
```

Arguments

`plot`	A plot object to convert to SVG. Can be a ggplot2 object, a recorded plot, or any object with a print() method.
`file`	The file to write the SVG to.
`title`	An optional title for the test case.

Index