Package 'vdiffr'

January 2, 2019

Title Visual Regression Testing and Graphical Diffing

Version 0.3.0

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 GPL-3 | file LICENSE

LazyData true

ByteCompile true

Depends R (>= 3.2.0)

Imports devtools, diffobj, fontquiver (>= 0.2.0), freetypeharfbuzz (>= 0.2.5), gdtools, glue, grDevices, htmlwidgets (>= 0.6), htmltools, purrr (>= 0.2.0), rlang, R6, Rcpp, shiny, testthat (>= 1.0.0), usethis (>= 1.4.0), xml2 (>= 1.0.0)

Suggests crayon, ggplot2 (>= 3.0.0), roxygen2, rstudioapi, withr, yaml

LinkingTo freetypeharfbuzz, gdtools, Rcpp, BH

RoxygenNote 6.1.1

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

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

NeedsCompilation yes

Author Lionel Henry [cre, aut],
RStudio [cph],
Carl Sutherland [aut] (jg-imagediff library),
Humble Software [cph] (jg-imagediff library),
David Hong [aut] (TwoFace library),
jQuery Foundation [cph] (jQuery library),
jQuery contributors [ctb, cph] (jQuery library; authors listed in inst/htmlwidgets/lib/jquery-authors.txt),
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@rstudio.com>

Repository CRAN

Date/Publication 2019-01-02 19:30:02 UTC

R topics documented:

`vdiffr-package` *vdiffr: Visual Regression Testing and Graphical Diffing*

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.

Author(s)

Maintainer: Lionel Henry <lionel@rstudio.com>

Authors:

- Carl Sutherland (jg-imagediff library)
- David Hong (TwoFace library)
- T Jake Luciani <jake@apache.org> (svglite)
- Matthieu Decorde <matthieu.decorde@ens-lyon.fr> (svglite)
- Vaudor Lise <lise.vaudor@ens-lyon.fr> (svglite)

Other contributors:

- RStudio [copyright holder]

- Humble Software (jg-imagediff library) [copyright holder]
- jQuery Foundation (jQuery library) [copyright holder]
- jQuery contributors (jQuery library; authors listed in inst/htmlwidgets/lib/jquery-authors.txt) [contributor, copyright holder]
- Tony Plate (svglite: Early line dashing code) [contributor]
- David Gohel (svglite: Line dashing code and raster code) [contributor]
- Yixuan Qiu (svglite: Improved styles; polypath implementation) [contributor]
- Håkon Malmedal (svglite: Opacity code) [contributor]

See Also

Useful links:

- https://github.com/r-lib/vdiffr
- Report bugs at https://github.com/r-lib/vdiffr/issues

`collect_cases` *Collect cases*

Description

These functions are mainly intended for internal use by `manage_cases()`. They are useful to programmatically collect cases.

Usage

```
collect_cases(package = ".", filter = NULL)

collect_new_cases(package = ".")

collect_mismatched_cases(package = ".")

collect_orphaned_cases(package = ".")
```

Arguments

`package`	Package description, can be path or package name. See `devtools::as.package()` for more information.
`filter`	If not `NULL`, only tests with file names matching this regular expression will be executed. Matching will take on the file name after it has been stripped of `"test-"` and `".R"`.

Value

A `cases` object. `collect_new_cases()`, `collect_mismatched_cases()` and `collect_orphaned_cases()` return a filtered `cases` object.

See Also

manage_cases()

---

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

Description

expect_doppelganger() takes a figure to check visually.

- If the figure has yet to be validated, the test is skipped. Call manage_cases() to validate the new figure, so vdiffr knows what to compare against.
- If the test has been validated, fig is compared to the validated figure. If the plot differs, a failure is issued (except on CRAN, see section on regression testing below).

  Either fix the problem, or call manage_cases() to validate the new figure appearance.

Usage

```
expect_doppelganger(title, fig, path = NULL, ..., verbose = NULL,
  writer = write_svg)
```

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. For plots that can't be represented as printable objects, you can pass a function. This function must construct the plot and print it.
path	The path where the test case should be stored, relative to the tests/figs/ folder. If NULL (the default), the current testthat context is used to create a sub-folder. Supply an empty string "" if you want the figures to be stored in the root folder.
...	Additional arguments passed to testthat::compare() to control specifics of comparison.
verbose	Soft-deprecated. See the debugging section.
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.

Regression testing versus Unit testing

Failures to match a validated appearance are only reported when the tests are run locally, on Travis, Appveyor, or any environment where the `Sys.getenv("CI")` or `Sys.getenv("NOT_CRAN")` variables are set. Because vdiffr is more of a monitoring than a unit testing tool, it shouldn't cause R CMD check failures on the CRAN machines.

Checking the appearance of a figure is inherently fragile. It is similar to testing for errors by matching exact error messages: these messages are susceptible to change at any time. Similarly, the appearance of plots depends on a lot of upstream code, such as the way margins and spacing are computed. vdiffr uses a special ggplot2 theme that should change very rarely, but there are just too many upstream factors that could cause breakages. For this reason, figure mismatches are not necessarily representative of actual failures.

Visual testing is not an alternative to writing unit tests for the internal data transformations performed during the creation of your figure. It is more of 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.

If you need to override the default vdiffr behaviour on CRAN (not recommended) or Travis (for example to run the tests in a particular builds but not others), set the `VDIFFR_RUN_TESTS` environment variable to "true" or "false".

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 (which can still be used for manual inspection, add a `testthat::skip()` before the `expect_doppelganger()` call in that case).

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)
```

```
})

}
```

`htmlwidgets` *HTML Widgets for graphical comparison*

Description

These widgets can be used at the console and embedded in a R Markdown document or Shiny application.

Usage

```
widget_toggle_(before, after, ..., width = NULL, height = NULL)

widget_slide_(before, after, ..., width = NULL, height = NULL)

widget_diff_(before, after, ..., width = NULL, height = NULL)

widget_toggle(before, after, ..., width = NULL, height = NULL)

widget_slide(before, after, ..., width = NULL, height = NULL)

widget_diff(before, after, ..., width = NULL, height = NULL)
```

Arguments

`before`	The picture that is taken as reference.
`after`	The picture against which the reference is compared.
`...`	Unused. Meant for collecting unknown arguments and allow widget extensions.
`width`	Fixed width for widget (in css units). The default is `NULL`, which results in intelligent automatic sizing based on the widget's container.
`height`	Fixed height for widget (in css units). The default is `NULL`, which results in intelligent automatic sizing based on the widget's container.

Details

The regular versions take plots or functions as `before` and `after` arguments (see `expect_doppelganger()` for details). The versions suffixed with underscores take HTML image sources. These can be paths to SVG files or inlined SVG images. Currently, `widget_diff_()` is compatible only with inlined images.

Examples

```
p1 <- function() hist(mtcars$disp)
p2 <- function() hist(mtcars$drat)

# You can also call these functions in a R Markdown document or
# in a Shiny application:
widget_toggle(p1, p2)
widget_slide(p1, p2)
widget_diff(p1, p2)
```

---

last_collection_error *Print last error that occurred during collection*

Description

Print last error that occurred during collection

Usage

```
last_collection_error()
```

---

manage_cases *Manage visual test cases with a Shiny app*

Description

Manage visual test cases with a Shiny app

Usage

```
manage_cases(package = ".", filter = NULL, ..., options = list())
```

Arguments

package	Package description, can be path or package name. See `devtools::as.package()` for more information.
filter	If not `NULL`, only tests with file names matching this regular expression will be executed. Matching will take on the file name after it has been stripped of `"test-"` and `".R"`.
...	Unused.
options	Named options that should be passed to the `runApp` call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify `width` and `height` parameters which provide a hint to the embedding environment about the ideal height/width for the app.

See Also

vdiffrAddin(), collect_cases(), and validate_cases()

---

shinybindings *Shiny bindings for graphical comparison widgets*

Description

Output and render functions for using the Toggle, Slide and Diff widgets within Shiny applications and interactive Rmd documents. Used in manage_cases().

Usage

```
toggleOutput(outputId, width = "100%", height = "400px")

renderToggle(expr, env = parent.frame(), quoted = FALSE)

slideOutput(outputId, width = "100%", height = "400px")

slideTransition(expr, env = parent.frame(), quoted = FALSE)

diffOutput(outputId, width = "100%", height = "400px")

diffTransition(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	Output variable to read from.
width, height	Must be a valid CSS unit (like "100\%", "400px", "auto") or a number, which will be coerced to a string and have "px" appended.
expr	An expression that generates a comparison widget.
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

validate_cases *Cases validation*

Description

These functions are mainly intended for internal use by `manage_cases()`. They are useful to programmatically validate cases or delete orphaned cases.

Usage

```
validate_cases(cases = collect_new_cases())

delete_orphaned_cases(cases = collect_orphaned_cases())
```

Arguments

`cases`	A `cases` object returned by one of the collecting functions such as `collect_cases()`.

See Also

`manage_cases()`

vdiffrAddin *RStudio Addin for managing visual cases*

Description

The package is detected by looking for the currently active project, then for the current folder if no project is active.

Usage

```
vdiffrAddin()
```

See Also

`manage_cases()`, `collect_cases()`, and `validate_cases()`

`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