Package: odiffr (via r-universe)

October 9, 2026

Title Fast Pixel-by-Pixel Image Comparison Using 'odiff'

Version 0.6.0.9000

Description R bindings to 'odiff', a fast SIMD pixel-by-pixel image comparison tool <https://github.com/dmtrKovalenko/odiff>. Compares PNG, JPEG, WEBP, TIFF and BMP images, plots and PDF pages with configurable thresholds, antialiasing detection and ignore regions. Provides 'testthat' expectations and snapshot testing (including for 'shinytest2' screenshots), batch and directory comparison, HTML, Markdown and JUnit reports, baseline approval and audit records. Requires the 'odiff' binary, which can be downloaded with install_odiff().

SystemRequirements odiff (>= 4.1.1) - https://github.com/dmtrKovalenko/odiff

License MIT + file LICENSE

URL https://benwolst.github.io/odiffr/, https://github.com/BenWolst/odiffr

BugReports https://github.com/BenWolst/odiffr/issues

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports graphics, grDevices, grid, tools

Suggests base64enc, digest, ggplot2, jsonlite, knitr, lattice, magick, openssl, pdftools, png, ragg, rmarkdown, testthat (>= 3.1.7), tibble, withr, xml2

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Repository https://benwolst.r-universe.dev

Date/Publication 2026-10-09 09:57:54 UTC

RemoteUrl https://github.com/benwolst/odiffr

RemoteRef HEAD

RemoteSha 077176fc9fa1f5d33f3564765146ba84db1971a1

Contents

approve_changes . . . 3
audit_record . . . 5
batch_junit . . . 7
batch_markdown . . . 9
batch_report . . . 10
compare_dirs_report . . . 12
compare_file_odiff . . . 13
compare_image_dirs . . . 15
compare_images . . . 17
compare_images_batch . . . 20
compare_pdf_dirs . . . 21
compare_pdfs . . . 23
diff_image . . . 25
expect_images_match . . . 26
expect_snapshot_image . . . 29
failed_pairs . . . 32
find_odiff . . . 33
ignore_region . . . 34
install_odiff . . . 35
odiff_available . . . 36
odiff_info . . . 36
odiff_preset . . . 37
odiff_run . . . 38
odiff_version . . . 41
odiffr_cache_path . . . 42
odiffr_clear_cache . . . 42
odiffr_update . . . 43
passed_pairs . . . 44
plot.odiff_result . . . 45
plot_options . . . 46
snapshot_report . . . 47
summary.odiffr_batch . . . 49
use_odiffr_ci . . . 50

Index **52**

approve_changes *Approve Changes as New Baselines*

Description

Accept the current images of a batch comparison as the new baselines: for each selected failing pair, the current image (`img2`) is copied over the baseline image (`img1`). This is intended for the directory/batch workflow (`compare_image_dirs()`, `compare_images_batch()`); review the differences first (e.g. with `batch_report()`), then approve them.

Usage

```
approve_changes(
  object,
  which = NULL,
  reasons = c("pixel-diff", "layout-diff"),
  remove_missing = FALSE,
  dry_run = FALSE,
  backup_dir = NULL
)
```

Arguments

`object`	An `odiffr_batch` object returned by `compare_image_dirs()` or `compare_images_batch()`.
`which`	Optional selection of rows to approve. One of: • an integer/numeric vector of `pair_id`s, • a logical vector with one element per row of `object`, • a character vector of image names, matched against the base names of `img1`/`img2` or against the trailing part of their paths (e.g. `"subdir/page.png"`). When `NULL` (the default), all failing rows whose `reason` is in `reasons` are approved. An explicit selection takes precedence over `reasons`.
`reasons`	Character vector of failure reasons to approve when `which` is `NULL`. Defaults to `c("pixel-diff", "layout-diff")`. Add `"missing"` (together with `remove_missing = TRUE`) to also delete baselines whose current image no longer exists.
`remove_missing`	Logical; if `TRUE`, selected rows with `reason = "missing"` have their baseline file deleted (the screenshot was intentionally removed). If `FALSE` (the default), such rows are skipped.
`dry_run`	Logical; if `TRUE`, no files are changed and the returned actions describe what would happen. Default is `FALSE`.
`backup_dir`	Optional directory in which to save a copy of each baseline before it is overwritten or deleted. Paths relative to the common parent directory of the affected baselines are preserved; if there is no usable common directory, files are saved as `<pair_id>_<basename>`. Baselines that are already identical to the current image are not backed up.

Details

Rows with `reason = "match"` are never modified, and rows with `reason = "error"` cannot be approved (the comparison itself failed, so there is no valid current image to accept); they are reported as skipped. Rows whose `img1`/`img2` are not files (e.g. `"<magick-image>"` labels) are skipped as well. Rows with `reason = "within-tolerance"` passed within a tolerance (see `compare_images()`) and are skipped unless selected with `which` or listed in `reasons`.

Parent directories of baselines are created as needed. A failed copy or deletion is reported (action `"failed"`) and does not stop the remaining rows from being processed. Approving the same batch twice is harmless.

Value

Invisibly, a data.frame with one row per considered pair and columns:

pair_id Integer; the pair's `pair_id`.

baseline Character; path of the baseline image (`img1`).

current Character; path of the current image (`img2`).

action Character; one of `"updated"`, `"removed"`, `"skipped"`, `"failed"`, `"would update"`, or `"would remove"`.

detail Character; explanation (e.g. why a row was skipped), or `NA`.

A one-line summary of the actions is emitted as a message.

See Also

`compare_image_dirs()`, `batch_report()`, `failed_pairs()`

Examples

```
## Not run:
results <- compare_image_dirs("baseline/", "current/", diff_dir = "diffs/")
batch_report(results, "diffs/report.html")

# Preview, then accept all pixel and layout changes
approve_changes(results, dry_run = TRUE)
approve_changes(results, backup_dir = "baseline-backup/")

# Approve selected images only
approve_changes(results, which = c("home.png", "settings/profile.png"))

# Also delete baselines of screenshots that were removed
approve_changes(results, reasons = "missing", remove_missing = TRUE)

## End(Not run)
```

audit_record *Create an Audit Record of Image Comparisons*

Description

Builds a machine-readable evidence record of one or more image comparisons: which files were compared (with cryptographic hashes and sizes), the outcome of each comparison, the parameters used, and the software environment (odiffr and odiff versions, the odiff binary and its hash, R version, platform and user). The record can be written to a JSON or CSV file and kept alongside validation documentation. This supports audit trails in validated environments; it does not by itself make a process compliant with any regulation.

Usage

```
audit_record(
  x,
  file = NULL,
  format = c("json", "csv"),
  hash = c("sha256", "md5"),
  params = NULL
)
```

Arguments

x	The comparison result(s) to record: an odiff_result from odiff_run(), a data.frame/tibble from compare_images(), or an odiffr_batch from compare_images_batch() or compare_image_dirs().
file	Path of the file to write, or NULL (default) to only return the record.
format	Output file format, "json" (default) or "csv". Only used when file is given. JSON requires the jsonlite package.
hash	Hash algorithm for files: "sha256" (default) or "md5". SHA-256 requires the openssl or digest package; MD5 uses base R (tools::md5sum()).
params	Optional named list of the comparison parameters (e.g. list(threshold = 0.1, antialiasing = TRUE)). Results of odiff_run() carry their parameters, which are used when params is NULL. compare_images() and batch results do not, so pass the parameters used here to record them; otherwise they are recorded as unknown (null in JSON, NA in CSV). odiff_run() results never include the tolerance parameters (max_diff_percent, max_diff_pixels), which are applied by compare_images(); pass them in params to record them. A warning is given when comparisons passed within a tolerance but params records neither limit.

Details

Record structure. The returned list has two elements:

header, a named list:

schema Character; schema identifier, currently `"odiffr-audit/1"`.

created Character; creation time in UTC, ISO 8601 (`"YYYY-MM-DDTHH:MM:SSZ"`).

odiffr_version Character; version of the odiffr package.

odiff_version Character; version of the odiff binary, or `NA`.

odiff_path Character; path of the odiff binary that `find_odiff()` currently selects, or `NA`.

odiff_hash Character; hash of that binary, or `NA`.

hash_algorithm Character; `"sha256"` or `"md5"`.

r_version Character; R version, e.g. `"4.4.1"`.

platform Character; `R.version$platform`.

sysname, release, machine Character; from `Sys.info()`.

user Character; `Sys.info()[["user"]]`.

n_comparisons Integer; number of comparisons recorded.

params Named list of comparison parameters, or `NULL` if unknown.

`comparisons`, a data.frame with one row per comparison and columns:

pair_id Integer; the batch `pair_id`, or the row number.

img1, img2 Character; image paths as recorded in the result.

img1_hash, img2_hash Character; file hashes, `NA` if the file does not exist (e.g. `"missing"` rows) or is not a file (e.g. `"<magick-image>"`).

img1_size, img2_size Numeric; file sizes in bytes, or `NA`.

diff_output Character; diff image path, or `NA`.

diff_output_hash Character; hash of the diff image, or `NA`.

diff_output_size Numeric; size of the diff image, or `NA`.

match Logical; whether the images matched.

reason Character; `"match"`, `"pixel-diff"`, `"layout-diff"`, `"error"`, `"missing"` or `"within-tolerance"`.

diff_count Integer; number of different pixels, or `NA`.

diff_percentage Numeric; percentage of different pixels, or `NA`.

error Character; error message, or `NA`.

Files are hashed when `audit_record()` is called, so call it right after the comparison, before any of the files can change.

JSON files contain an object with `header` and `comparisons` (an array of row objects); missing values are written as `null`.

CSV files contain one row per comparison with the `comparisons` columns, preceded by the header fields repeated on every row (except `params`) and followed by one `param_<name>` column per parameter. The standard parameters (`threshold`, `antialiasing`, `fail_on_layout`, `ignore_regions`, `diff_mask`, `diff_overlay`, `diff_color`, `reduce_ram`, `enable_asm`, `max_diff_percent`, `max_diff_pixels`) always have a column (`NA` when unknown); other parameters passed in `params` are added after them.

Value

The record, a list with elements `header` and `comparisons` (see Details). If `file` is given, the record is written to it and the normalised file path is returned invisibly instead.

See Also

`odiff_info()`, `odiff_version()`

Examples

```
## Not run:
result <- odiff_run("baseline.png", "current.png", "diff.png",
                    threshold = 0.05)
rec <- audit_record(result)
rec$header$odiff_version
rec$comparisons$img1_hash

# Write a JSON evidence file
audit_record(result, file = "comparison-audit.json")

# Batch results: pass the parameters used, write CSV
results <- compare_image_dirs("baseline/", "current/", threshold = 0.05)
audit_record(results, file = "audit.csv", format = "csv",
             params = list(threshold = 0.05))

## End(Not run)
```

---

batch_junit *Write Batch Comparison Results as JUnit XML*

Description

Converts batch image comparison results to a JUnit XML report, the format understood by most CI systems (GitHub Actions test reporters, GitLab, Jenkins, Azure Pipelines, ...). Each comparison becomes one test case.

Usage

```
batch_junit(
  object,
  output_file = NULL,
  suite_name = "odiffr",
  include_passed = TRUE
)
```

Arguments

object	An odiffr_batch object from compare_images_batch() or compare_image_dirs().
output_file	Path to write the XML file. If NULL (default), the XML is returned as a character string. The file is written as UTF-8 and its parent directory is created if it does not exist.
suite_name	Name of the test suite, also used as the classname of every test case. Default: "odiffr".
include_passed	If TRUE (default), passing comparisons are included as successful test cases. If FALSE, only failures and errors are written.

Details

The report contains a single <testsuite> (inside a <testsuites> root) whose tests, failures and errors attributes count the test cases written. Test cases are named after the current image file (img2), or the baseline (img1) when img2 is not a file (for example "<magick-image>"), or "pair N" otherwise.

- Pixel and layout differences are reported as <failure> elements whose type is the comparison reason, e.g. message="pixel-diff: 1.26% (126 pixels)" type="pixel-diff".
- Baseline images without a current counterpart are failures of type "missing".
- Comparisons that could not be run (reason == "error") are reported as <error> elements carrying the error message.

The failure body lists the baseline, current and diff image paths. Text is XML-escaped and characters that are not allowed in XML 1.0 are removed.

Value

If output_file is NULL, the XML as a character string (invisibly); otherwise the file path (invisibly).

See Also

batch_markdown(), batch_report(), compare_image_dirs()

Examples

```
## Not run:
results <- compare_image_dirs("baseline/", "current/", diff_dir = "diffs/")
batch_junit(results, "odiffr-junit.xml")

## End(Not run)
```

batch_markdown *Write a Markdown Summary of Batch Comparison Results*

Description

Produces a GitHub-flavoured Markdown summary of batch image comparison results: a heading, a pass/fail line, a breakdown of failure reasons and a table of the worst offenders. On GitHub Actions the summary is appended to the job summary page by default.

Usage

```
batch_markdown(
  object,
  output_file = NULL,
  title = "odiffr comparison",
  n_worst = 10,
  append = TRUE
)
```

Arguments

object	An odiffr_batch object from compare_images_batch() or compare_image_dirs().
output_file	Path to write the Markdown to. If NULL (default), the file named by the GITHUB_STEP_SUMMARY environment variable is used when that variable is set; otherwise nothing is written.
title	Heading of the summary. Default: "odiffr comparison".
n_worst	Maximum number of failures listed in the table. Default: 10. Use 0 to omit the table.
append	If TRUE (default), append to output_file instead of overwriting it, as GitHub step summaries are built up by appending.

Details

The worst offenders table has columns Image, Reason, Diff %, Pixels and Error, ordered as in summary.odiffr_batch(). Cell text is escaped so that |, line breaks, Markdown emphasis characters and HTML-like text such as <magick-image> are shown literally. Files are written as UTF-8 and the parent directory is created if needed.

Value

If the summary is written to a file, the file path (invisibly); otherwise the Markdown as a character string (invisibly). Nothing is printed.

See Also

batch_junit(), batch_report(), summary.odiffr_batch()

Examples

```
## Not run:
results <- compare_image_dirs("baseline/", "current/", diff_dir = "diffs/")

# In a GitHub Actions step: appends to the job summary
batch_markdown(results)

# Anywhere else: get the Markdown as a string
md <- batch_markdown(results)
cat(md)

## End(Not run)
```

`batch_report` *Generate HTML Report for Batch Comparison Results*

Description

Creates a standalone HTML report summarizing batch image comparison results. Includes pass/fail statistics, failure reasons, diff statistics, and thumbnails of the worst offenders.

Usage

```
batch_report(
  object,
  output_file = NULL,
  title = "odiffr Comparison Report",
  embed = FALSE,
  relative_paths = FALSE,
  n_worst = 10,
  show_all = FALSE,
  images = c("diff", "all"),
  ...
)
```

Arguments

`object`	An `odiffr_batch` object from `compare_images_batch()` or `compare_image_dirs()`.
`output_file`	Path to write the HTML file. If NULL, returns HTML as a character string. The file is written as UTF-8 and its parent directory is created if it does not exist.
`title`	Report title. Default: "odiffr Comparison Report".
`embed`	If TRUE, embed diff images as base64 data URIs for a fully self-contained file. If FALSE (default), link to image files on disk using `file://` URIs.
`relative_paths`	If TRUE and `output_file` is specified, use paths relative to the report location for image `src` attributes. This makes reports portable without embedding. Paths are percent-encoded so that file names containing spaces, #, ? or % work. If no relative path can be built (e.g. different drives on Windows), a `file://` URI is used instead. Ignored when `embed = TRUE`. Default: FALSE.

n_worst	Number of worst offenders to display. Default: 10.
show_all	If TRUE, include a table of all comparisons. Default: FALSE.
images	Which images to show for each comparison: "diff" (default) shows only the diff image; "all" shows the baseline (img1), current (img2) and diff images side by side, each with a caption. Clicking a thumbnail shows the full-size image (a link to the file for linked reports, an in-page zoom for embedded ones).
...	Additional arguments passed to summary.odiffr_batch().

Details

Diff image thumbnails (or embedded images when embed = TRUE) are only shown for comparisons where a diff_output file was created. This requires using diff_dir in compare_images_batch() or compare_image_dirs(). Comparisons without diff images will show "No diff" in the preview column.

With images = "all", baseline and current images are linked or embedded in the same way as diff images (embed, relative_paths). Embedded images get a MIME type based on their file extension (PNG, JPEG, WebP, BMP or TIFF; note that most browsers cannot display TIFF). Images that are not files on disk (for example "<magick-image>" inputs) or that no longer exist are shown as a placeholder. The report stays a single HTML file with inline CSS and no JavaScript.

Failures without pixel statistics (layout differences, errors, or baseline images with no current counterpart) show "-" for the diff percentage and pixel count. If the results contain an error column, its message is shown in the Reason column. An empty batch produces a valid report with a pass rate of "-".

Value

If output_file is NULL, returns the HTML as a character string (invisibly). If output_file is specified, writes the file and returns the file path (invisibly).

See Also

compare_images_batch(), compare_image_dirs(), summary.odiffr_batch()

Examples

```
## Not run:
results <- compare_image_dirs("baseline/", "current/", diff_dir = "diffs/")

# Generate report file
batch_report(results, output_file = "report.html")

# Self-contained report with embedded images
batch_report(results, output_file = "report.html", embed = TRUE)

# Baseline, current and diff images side by side
batch_report(results, output_file = "report.html", images = "all")

# Get HTML as string
html <- batch_report(results)
```

```
## End(Not run)
```

`compare_dirs_report` *Compare Directories and Generate HTML Report*

Description

Convenience function that compares all images in two directories and generates an HTML report in one step.

Usage

```
compare_dirs_report(
  baseline_dir,
  current_dir,
  diff_dir = "diffs",
  output_file = file.path(diff_dir, "report.html"),
  parallel = FALSE,
  title = "odiffr Comparison Report",
  embed = FALSE,
  relative_paths = FALSE,
  n_worst = 10,
  show_all = FALSE,
  images = c("diff", "all"),
  ...
)
```

Arguments

`baseline_dir`	Path to the directory containing baseline images.
`current_dir`	Path to the directory containing current images to compare against baseline.
`diff_dir`	Directory to save diff images. If `NULL`, no diff images are created.
`output_file`	Path for the HTML report. Defaults to `file.path(diff_dir, "report.html")`.
`parallel`	Logical; if `TRUE`, compare images in parallel. See `compare_images_batch()` for details.
`title`	Title for the HTML report.
`embed`	Logical; if `TRUE`, embed images as base64 data URIs for a self-contained report. If `FALSE` (default), link to image files.
`relative_paths`	Logical; if `TRUE`, use relative paths for images in the HTML report. Makes reports portable without embedding. Ignored when `embed = TRUE`. Default: `FALSE`.
`n_worst`	Number of worst offenders to display in the report.
`show_all`	Logical; if `TRUE`, show all comparisons in the report, not just failures.
`images`	Which images to show in the report: `"diff"` (default) for the diff image only, or `"all"` for baseline, current and diff images side by side. See `batch_report()`.
`...`	Additional arguments passed to `compare_image_dirs()` (e.g. `threshold`, `antialiasing`, `pattern`, `recursive`, or `max_diff_percent` for an image-level tolerance).

Value

The `odiffr_batch` results (invisibly). The HTML report is written to `output_file` as a side effect.

See Also

`compare_image_dirs()`, `batch_report()`

Examples

```
## Not run:
# One-liner for QA workflow
compare_dirs_report("baseline/", "current/")
# -> Creates diffs/ directory with diff images and report.html

# With parallel processing and embedded images
compare_dirs_report("baseline/", "current/", parallel = TRUE, embed = TRUE)

# Pass comparison options via ...
compare_dirs_report("baseline/", "current/", threshold = 0.1, antialiasing = TRUE)

## End(Not run)
```

`compare_file_odiff` *Compare Files with odiff (for testthat and shinytest2 Snapshots)*

Description

A function factory that returns a comparison function suitable for the `compare` argument of `testthat::expect_snapshot_fi` and of `shinytest2::AppDriver$expect_screenshot()`. The returned function takes the paths of the old (snapshot) and new image files and returns `TRUE` if odiff considers them a match. When they do not match, it writes a diff image highlighting the changed pixels and reports where it is. `expect_snapshot_image()` uses it internally; use it directly when you write image files yourself or take screenshots with shinytest2.

Usage

```
compare_file_odiff(
  threshold = 0.1,
  antialiasing = FALSE,
  ignore_regions = NULL,
  fail_on_layout = TRUE,
  ...,
  max_diff_percent = NULL,
  max_diff_pixels = NULL,
  preset = NULL,
  diff_dir = getOption("odiffr.snapshot_diff_dir")
)
```

Arguments

threshold	Numeric; colour difference threshold between 0.0 and 1.0. Default is 0.1 (or the value from preset).
antialiasing	Logical; if TRUE, ignore antialiased pixels. Default is FALSE (or the value from preset).
ignore_regions	List of regions to ignore during comparison. Use ignore_region() to create regions, or pass a data.frame with columns x1, y1, x2, y2.
fail_on_layout	Logical; if TRUE (the default), odiff stops at the size check when images have different dimensions. Such images fail either way; FALSE adds pixel counts and a diff image where odiff can (odiff >= 4.3.5).
...	Additional arguments passed to odiff_run().
max_diff_percent, max_diff_pixels	Image-level tolerance: the largest percentage (0 to 100) and/or number of differing pixels to accept. NULL (the default) accepts no difference. A comparison within the tolerance passes and keeps the original snapshot, so differences cannot accumulate. Images of different sizes always fail. See the "Tolerance" section of compare_images() before using this.
preset	Optional name of a comparison preset, see odiff_preset(): "strict", "default", "screenshot" or "cross_platform". The preset supplies threshold and antialiasing; values given explicitly for those arguments take precedence. NULL (the default) uses the argument defaults.
diff_dir	Directory for diff images of failed comparisons. NULL (the default, unless the odiffr.snapshot_diff_dir option is set) chooses a directory automatically, see "Diff images". FALSE disables diff images. The directory is resolved each time the comparison runs.

Value

A function with arguments old and new (file paths) that returns a single TRUE or FALSE. If odiff cannot compare the files (reason == "error"), the function gives a warning with odiff's error message and returns FALSE.

Diff images

testthat removes unrecognised files from tests/testthat/_snaps/, so diff images are written elsewhere. With diff_dir = NULL the directory is, in order of preference:

1. the odiffr.diff_dir option (shared with expect_images_match());
2. tests/testthat/_odiffr/ when running tests (or when called from a package root that has a tests/testthat directory);
3. a directory in tempdir() otherwise.

Setting options(odiffr.save_diff = FALSE) disables automatic diff images (an explicit diff_dir still wins).

Inside that directory, the layout below _snaps/ is mirrored and the file is named after the snapshot: the diff for _snaps/<variant>/<test-file>/<name>.png is <diff_dir>/<variant>/<test-file>/<name>_diff.png.

Re-running a failing test overwrites the diff; a passing comparison removes a stale one. Add `tests/testthat/_odiffr/` to `.gitignore` and `.Rbuildignore`.

When a comparison fails, a message (not a warning, so it does not add a warning to the test results) such as `odiff: 1.26% pixels differ (126 px) in 'plot.png'; diff image: <path>` is shown.

See Also

`expect_snapshot_image()`, `odiff_preset()`, `snapshot_report()` for reviewing failed snapshots on CI.

Examples

```
## Not run:
test_that("exported chart is stable", {
  path <- tempfile(fileext = ".png")
  save_my_chart(path)
  testthat::expect_snapshot_file(
    path,
    name = "chart.png",
    compare = compare_file_odiff(threshold = 0.2, antialiasing = TRUE)
  )
})

# shinytest2: tolerate anti-aliasing noise in browser screenshots
test_that("app looks right", {
  app <- shinytest2::AppDriver$new()
  app$expect_screenshot(compare = compare_file_odiff(preset = "screenshot"))
})

## End(Not run)
```

---

compare_image_dirs *Compare Images in Two Directories*

Description

Compare all images in a baseline directory against corresponding images in a current directory. Files are matched by relative path (including subdirectories when `recursive = TRUE`).

Usage

```
compare_image_dirs(
  baseline_dir,
  current_dir,
  pattern = "\\.(png|jpe?g|webp|tiff|bmp)$",
  recursive = FALSE,
  diff_dir = NULL,
```

```
  parallel = FALSE,
  ...
)
```

Arguments

baseline_dir	Path to the directory containing baseline images.
current_dir	Path to the directory containing current images to compare against baseline.
pattern	Regular expression pattern to match image files (matched case-insensitively). The default matches the file extensions accepted by odiff: .png, .jpg, .jpeg, .webp, .tiff and .bmp. Note that odiff does not accept the .tif extension; such files matched by a custom pattern are reported with reason = "error".
recursive	Logical; if TRUE, search subdirectories recursively. Default is FALSE.
diff_dir	Directory to save diff images. If NULL, no diff images are created.
parallel	Logical; if TRUE, compare images in parallel. See compare_images_batch() for details.
...	Additional arguments passed to compare_images_batch() (and on to compare_images()), e.g. max_diff_percent.

Details

The baseline directory is the source of truth. For each image found in baseline_dir matching pattern:

- If a corresponding file exists in current_dir (same relative path), the two images are compared.
- If the file is missing from current_dir, a warning is issued and the file is included in the results as a failed row with match = FALSE, reason = "missing", NA diff statistics and diff_output, img2 set to the expected (nonexistent) path, and an explanatory error message. This ensures that a disappearing screenshot fails the comparison. If every file is missing, all rows are "missing".

An error is raised if baseline_dir contains no images matching pattern.

Files that exist only in current_dir (not in baseline_dir) are not compared, but a message is emitted noting how many such files were found.

Value

A tibble (if available) or data.frame with class odiffr_batch, with one row per baseline image (in baseline file order), containing all columns from compare_images() (including error) plus a leading pair_id column.

See Also

compare_images_batch() for comparing explicit pairs, compare_images() for single comparisons.

Examples

```
## Not run:
# Compare all images in two directories
results <- compare_image_dirs("baseline/", "current/")

# Only compare PNG files
results <- compare_image_dirs("baseline/", "current/", pattern = "\\.png$")

# Include subdirectories and save diff images
results <- compare_image_dirs(
  "baseline/",
  "current/",
  recursive = TRUE,
  diff_dir = "diffs/"
)

# Check which comparisons failed (including missing files)
results[!results$match, ]

## End(Not run)
```

---

compare_images *Compare Two Images*

Description

High-level function for comparing images with convenient output. Returns a tibble if the tibble package is available, otherwise a data.frame. Accepts file paths, magick-image objects, and plots (ggplot objects, functions that draw a plot, or recorded plots), which are rendered to a temporary PNG file before comparison.

Usage

```
compare_images(
  img1,
  img2,
  diff_output = NULL,
  threshold = 0.1,
  antialiasing = FALSE,
  fail_on_layout = FALSE,
  ignore_regions = NULL,
  plot_options = NULL,
  max_diff_percent = NULL,
  max_diff_pixels = NULL,
  ...
)
```

Arguments

`img1`	Path to the first image, a magick-image object, or a plot: a ggplot object, a function of no arguments that draws a plot (base or grid graphics) or returns a ggplot, lattice or grid object when called, or a recorded plot (`grDevices::recordPlot()`).
`img2`	Path to the second image, a magick-image object, or a plot (see `img1`).
`diff_output`	Path for the diff output image (PNG only). Use `NULL` for no diff output, or `TRUE` to auto-generate a temporary file path.
`threshold`	Numeric; colour difference threshold between 0.0 and 1.0. Default is 0.1.
`antialiasing`	Logical; if `TRUE`, ignore antialiased pixels. Default is `FALSE`.
`fail_on_layout`	Logical; if `TRUE`, fail immediately if images have different dimensions. Default is `FALSE`. Images of different dimensions never match and are reported as `"layout-diff"` either way; `TRUE` stops odiff at the size check, while `FALSE` also gives pixel counts and a diff image where odiff can (odiff >= 4.3.5).
`ignore_regions`	List of regions to ignore during comparison. Use `ignore_region()` to create regions, or pass a data.frame with columns `x1`, `y1`, `x2`, `y2`.
`plot_options`	Options for rendering plot inputs, created with `plot_options()`. `NULL` (the default) uses `plot_options()`: 7 x 5 inches at 96 dpi on a white background. Ignored for file and magick-image inputs.
`max_diff_percent, max_diff_pixels`	Image-level tolerance. `NULL` (the default) for none, or the largest percentage of differing pixels (0 to 100, on the scale of `diff_percentage`) and/or the largest number of differing pixels to accept. Each limit is inclusive; if both are given, both must hold. See "Tolerance".
`...`	Additional arguments passed to `odiff_run()`.

Value

A tibble (if available) or data.frame with columns:

match Logical; `TRUE` if images match, or differ within the tolerance (see `max_diff_percent`).

reason Character; `"match"`, `"pixel-diff"`, `"layout-diff"`, `"error"`, or `"within-tolerance"` for a pixel difference accepted by `max_diff_percent`/`max_diff_pixels`.

diff_count Integer; number of different pixels.

diff_percentage Numeric; percentage of different pixels.

diff_output Character; path to diff image, or `NA`.

img1 Character; path to first image (`"<magick-image>"` or `"<plot>"` for magick-image and plot inputs).

img2 Character; path to second image (labelled as `img1`).

error Character; the error message reported by odiff when `reason` is `"error"` (e.g. an image could not be loaded or has an unsupported format), otherwise `NA`. This is always the last column.

Tolerance

By default any differing pixel (after `threshold`, `antialiasing` and `ignore_regions`) makes a comparison fail. `max_diff_percent` and `max_diff_pixels` accept a pixel difference up to a limit instead: the result then has `match = TRUE` and `reason = "within-tolerance"`, and `diff_count`, `diff_percentage` and `diff_output` still describe the difference. A tolerance never applies to images of different sizes (even with `fail_on_layout = FALSE`), to errors, or when the image dimensions cannot be read (non-PNG images without the magick package).

Use a tolerance with care. Font rendering differs between systems, and for plots it can change a few percent of pixels, while real regressions are often tiny: a dropped data point or an edited axis label can be well under 0.1% of a plot. A tolerance large enough to absorb font differences hides such changes, so it is best kept for comparisons across environments, with exact matching elsewhere.

See Also

`odiff_run()` for the low-level interface, `ignore_region()` for creating ignore regions.

Examples

```
## Not run:
# Compare two image files
result <- compare_images("baseline.png", "current.png")
result$match

# With diff output
result <- compare_images("baseline.png", "current.png", diff_output = TRUE)
result$diff_output

# Compare magick-image objects (requires magick package)
library(magick)
img1 <- image_read("baseline.png")
img2 <- image_read("current.png")
result <- compare_images(img1, img2)

# Compare a ggplot against a baseline PNG (rendered with ragg if
# installed, otherwise grDevices::png())
library(ggplot2)
p <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
result <- compare_images("baseline_plot.png", p,
                         plot_options = plot_options(width = 6, height = 4))

# Base graphics: pass a function that draws the plot
result <- compare_images("baseline_hist.png", function() hist(mtcars$mpg))

# Ignore specific regions
result <- compare_images("baseline.png", "current.png",
                         ignore_regions = list(
                           ignore_region(0, 0, 100, 50),    # Header
                           ignore_region(0, 500, 800, 600)  # Footer
                         ))

## End(Not run)
```

`compare_images_batch` *Compare Multiple Image Pairs*

Description

Compare multiple pairs of images in batch. Useful for visual regression testing across many screenshots.

Usage

```
compare_images_batch(pairs, diff_dir = NULL, parallel = FALSE, ...)
```

Arguments

`pairs`	A data.frame with columns `img1` and `img2` containing file paths, or a list of named lists with `img1` and `img2` elements.
`diff_dir`	Directory to save diff images. If `NULL`, no diff images are created. If provided, diff images are named based on the input file names.
`parallel`	Logical; if `TRUE`, compare images in parallel using multiple CPU cores. Uses `parallel::mclapply` on Unix systems (macOS, Linux) and falls back to sequential processing on Windows. Default is `FALSE`. The number of cores is taken from `getOption("mc.cores")` (or detected), capped at the number of pairs, and limited to 2 when the `_R_CHECK_LIMIT_CORES_` environment variable is `"TRUE"`/`"true"` or `"warn"` (as during `R CMD check --as-cran`).
`...`	Additional arguments passed to `compare_images()`, e.g. `threshold` or `max_diff_percent`.

Details

A failure while comparing one pair (for example, a file that does not exist, cannot be read, or has an unsupported format) does not abort the batch. Instead, that pair is reported as a row with `match = FALSE`, `reason = "error"`, and the error message in the `error` column. The same applies to failed worker processes when `parallel = TRUE`.

Value

A tibble (if available) or data.frame with class `odiffr_batch`, containing one row per comparison with all columns from `compare_images()` (including `error`) plus a leading `pair_id` column. Use `summary()` to get aggregate statistics. If `pairs` is empty (a zero-row data.frame or an empty list), an empty `odiffr_batch` with the same columns is returned.

See Also

`summary.odiffr_batch()` for summarizing batch results, `compare_image_dirs()` for directory-based comparison.

Examples

```
## Not run:
# Create a data frame of image pairs
pairs <- data.frame(
  img1 = c("baseline/page1.png", "baseline/page2.png"),
  img2 = c("current/page1.png", "current/page2.png")
)

# Compare all pairs
results <- compare_images_batch(pairs, diff_dir = "diffs/")

# Compare in parallel (Unix only)
results <- compare_images_batch(pairs, parallel = TRUE)

# Check which comparisons failed
results[!results$match, ]

# Inspect errors (e.g. unreadable or missing files)
results[results$reason == "error", c("img1", "img2", "error")]

## End(Not run)
```

---

compare_pdf_dirs *Compare PDF Files in Two Directories*

Description

Compare every PDF file in a baseline directory with the PDF file of the same relative path in a current directory, page by page, using `compare_pdfs()`. Useful for re-running a full set of outputs (for example all tables, listings and figures of a study) after an R or package upgrade and checking that nothing changed visually.

Usage

```
compare_pdf_dirs(
  baseline_dir,
  current_dir,
  pattern = "\\.pdf$",
  recursive = FALSE,
  pages = NULL,
  dpi = 150,
  diff_dir = NULL,
  parallel = FALSE,
  ...
)
```

Arguments

baseline_dir	Path to the directory containing the baseline PDFs.
current_dir	Path to the directory containing the current PDFs.
pattern	Regular expression matched (case-insensitively) against file names. Default matches .pdf files.
recursive	Logical; if TRUE, search subdirectories recursively. Default is FALSE.
pages, dpi	Passed to compare_pdfs() for every file.
diff_dir	Directory to save diff images and rendered pages. Each PDF gets its own subdirectory, named after its relative path. If NULL (default), no diff images are created.
parallel	Logical; if TRUE, compare the pages of each file in parallel. See compare_images_batch() for details.
...	Additional arguments passed to compare_pdfs() (e.g. threshold, antialiasing, ignore_regions, fail_on_layout). Use max_diff_percent or max_diff_pixels for an image-level tolerance, see compare_images().

Details

The baseline directory is the source of truth, as in compare_image_dirs():

- A PDF missing from current_dir triggers a warning and is reported as a single row with match = FALSE, reason = "missing", page = NA and the error "File not found in current_dir".
- A PDF that cannot be read (e.g. corrupt or password-protected) is reported as a single row with reason = "error", page = NA and the error message, instead of aborting the whole comparison.
- PDFs that exist only in current_dir are not compared, but a message notes how many were found.

Differences in page counts are reported per file as described in compare_pdfs().

An error is raised if baseline_dir contains no files matching pattern.

Value

A tibble (if available) or data.frame with class odiffr_batch, combining the results for all files (in baseline file order) with a sequential pair_id, the columns of compare_pdfs() (including page) and a trailing file column giving the relative path of the PDF.

See Also

compare_pdfs(), compare_image_dirs(), batch_report()

Examples

```
## Not run:
res <- compare_pdf_dirs("outputs-r4.3/", "outputs-r4.4/",
                        diff_dir = "pdf-diffs", recursive = TRUE)
summary(res)
```

```
unique(res$file[!res$match])
batch_report(res, output_file = "pdf-diffs/report.html")

## End(Not run)
```

`compare_pdfs` *Compare Two PDF Files Page by Page*

Description

Render each page of two PDF files to PNG and compare them page by page with odiff. This is useful for checking that rendered documents (for example clinical tables, listings and figures, or Quarto/R Markdown documents rendered to PDF) still look the same after a code, package or R upgrade.

Usage

```
compare_pdfs(
  baseline,
  current,
  pages = NULL,
  dpi = 150,
  diff_dir = NULL,
  threshold = 0.1,
  antialiasing = FALSE,
  ignore_regions = NULL,
  parallel = FALSE,
  ...
)
```

Arguments

`baseline`	Path to the baseline (reference) PDF file.
`current`	Path to the current PDF file to compare against `baseline`.
`pages`	Integer vector of page numbers to compare, or `NULL` (default) to compare all pages of both files (the union of their page ranges). Pages that do not exist in either file raise an error.
`dpi`	Resolution, in dots per inch, used to render the pages. Default is 150. Higher values detect smaller differences but take longer and produce larger images.
`diff_dir`	Directory to save diff images. If `NULL` (default), no diff images are created. When given, the rendered pages are also kept under `file.path(diff_dir, "pages")` (in `baseline/` and `current/` subdirectories).
`threshold`	Numeric; colour difference threshold between 0.0 and 1.0. Default is 0.1.
`antialiasing`	Logical; if `TRUE`, ignore antialiased pixels. Default is `FALSE`.

ignore_regions	Regions to ignore on every compared page, as accepted by compare_images() (see ignore_region()). Coordinates are in pixels of the rendered page, so they depend on dpi: a point at x inches from the left edge is at pixel x * dpi.
parallel	Logical; if TRUE, compare pages in parallel. See compare_images_batch() for details.
...	Additional arguments passed to compare_images() via compare_images_batch(), e.g. fail_on_layout = TRUE. Use max_diff_percent or max_diff_pixels for an image-level tolerance, see compare_images().

Details

Requires the **pdftools** package (and therefore the poppler library) for rendering. Only PDF input is supported: convert RTF or DOCX outputs to PDF first, for example with LibreOffice (soffice --headless --convert-to p

Pages are rendered with pdftools::pdf_convert() to files named <pdf name>_p001.png, <pdf name>_p002.png, ... in separate baseline/ and current/ directories. If diff_dir is NULL, they are written to a new directory inside tempdir(), which persists until the R session ends, so that reports showing the rendered pages can still be created after compare_pdfs() returns.

When the two files have a different number of pages, a message is emitted and the pages present in only one file are reported using the existing reasons, so that summaries and reports work unchanged:

- A page present in baseline but not in current is reported with match = FALSE, reason = "missing" and the error "Page N not present in current PDF" (like a file missing from the current directory in compare_image_dirs()).
- A page present in current but not in baseline is reported with match = FALSE, reason = "error" and the error "Page N not present in baseline PDF".

In both cases img1/img2 point to the rendered page that exists and to the (nonexistent) path the other page would have had.

Pages with different sizes (e.g. portrait versus landscape) are reported with reason = "layout-diff". By default (fail_on_layout = FALSE) they also get pixel counts and a diff image where odiff can provide them (odiff >= 4.3.5).

An error is raised if a file does not exist or cannot be read as a PDF (for example a corrupt or password-protected file).

Value

A tibble (if available) or data.frame with class odiffr_batch, with one row per compared page, containing the same columns as compare_images_batch() (including error) plus a trailing integer page column. img1 and img2 are the paths of the rendered page images. The result can be passed to summary(), batch_report() and the other functions that accept an odiffr_batch.

See Also

compare_pdf_dirs() for comparing directories of PDF files, compare_images_batch(), batch_report().

Examples

```
## Not run:
res <- compare_pdfs("qc/t_demog.pdf", "prod/t_demog.pdf",
                    diff_dir = "pdf-diffs")
summary(res)
res[!res$match, c("page", "reason", "diff_percentage")]

# Only the first two pages, at a higher resolution
compare_pdfs("old.pdf", "new.pdf", pages = 1:2, dpi = 300)

# Ignore a run-date footer: bottom 0.5 inch of a US Letter page at 150 dpi
compare_pdfs("old.pdf", "new.pdf", dpi = 150,
             ignore_regions = ignore_region(0, 1575, 1275, 1650))

## End(Not run)
```

`diff_image` *Get the Diff Image of a Comparison*

Description

Read the diff image produced by a comparison.

Usage

```
diff_image(x, as = c("magick", "raster"))
```

Arguments

`x`	An `odiff_result` from `odiff_run()`, a one-row data.frame from `compare_images()`, a single row of an `odiffr_batch` (e.g. `results[2, ]`), or the path to a diff image file.
`as`	Output type: `"magick"` (default) returns a `magick-image` (requires the magick package); `"raster"` returns a `raster` object (see `grDevices::as.raster()`) read with `png::readPNG()`, so magick is not required.

Details

A diff image only exists when the comparison was run with `diff_output` (or `diff_dir` for batches) and the images differ. An error is raised otherwise.

Value

A `magick-image` or a `raster` object.

See Also

`plot.odiff_result()`

Examples

```
## Not run:
result <- compare_images("baseline.png", "current.png", diff_output = TRUE)
img <- diff_image(result)

# Without magick
plot(diff_image(result, as = "raster"))

# One row of a batch
results <- compare_image_dirs("baseline/", "current/", diff_dir = "diffs/")
diff_image(failed_pairs(results)[1, ])

## End(Not run)
```

---

expect_images_match *testthat Expectations for Image Comparison*

Description

Assert that images match or differ using odiff. These expectations are designed for visual regression testing in testthat test suites.

Usage

```
expect_images_match(
  actual,
  expected,
  threshold = 0.1,
  antialiasing = FALSE,
  fail_on_layout = TRUE,
  ignore_regions = NULL,
  max_diff_percent = NULL,
  max_diff_pixels = NULL,
  ...,
  info = NULL,
  label = NULL
)

expect_images_differ(
  img1,
  img2,
  threshold = 0.1,
  antialiasing = FALSE,
  ...,
  info = NULL,
  label = NULL
)
```

Arguments

actual	Path to the actual/current image, or a magick-image object.
expected	Path to the expected/baseline image, or a magick-image object.
threshold	Numeric; colour difference threshold between 0.0 and 1.0. Default is 0.1.
antialiasing	Logical; if TRUE, ignore antialiased pixels. Default is FALSE.
fail_on_layout	Logical; if TRUE (the default for tests), odiff stops at the size check when images have different dimensions. Such images fail either way; FALSE adds pixel counts and a diff image where odiff can (odiff >= 4.3.5).
ignore_regions	List of regions to ignore during comparison. Use ignore_region() to create regions, or pass a data.frame with columns x1, y1, x2, y2.
max_diff_percent, max_diff_pixels	Image-level tolerance. NULL (the default) for none, or the largest percentage of differing pixels (0 to 100, on the scale of diff_percentage) and/or the largest number of differing pixels to accept. Each limit is inclusive; if both are given, both must hold. See "Tolerance".
...	Additional arguments passed to odiff_run().
info	Extra information to be included in the failure message (useful for providing context about what was being tested).
label	Optional custom label for the actual image in failure messages. If not provided, uses the deparsed expression.
img1, img2	Paths to images being compared (for expect_images_differ).

Details

expect_images_match() asserts that two images are visually identical (within the specified threshold). On failure, a diff image is saved to tests/testthat/_odiffr/ by default, which can be controlled via options(odiffr.save_diff = FALSE) or options(odiffr.diff_dir = "path"). Diff file names are deterministic, so re-running a failing test overwrites the previous diff rather than accumulating files: for file paths the name is <actual>_vs_<expected>.png (basenames without extension); for magick-image objects it is built from the label/expressions passed (e.g. img_new_vs_img_old.png). If two different comparisons in the same session would produce the same name (e.g. identical basenames in different directories), the parent directory names, or else a numeric suffix, are added. A stale diff image left by a previous failing run is removed when the expectation is run again (odiff writes no diff image when the images match). With max_diff_percent or max_diff_pixels, a small pixel difference passes (see the "Tolerance" section of compare_images()); no diff image is kept for it.

expect_images_differ() asserts that two images are visually different. No diff image is saved since there's nothing to debug when images match unexpectedly. It also accepts max_diff_percent and max_diff_pixels (through ...), and then requires the images to differ by more than the tolerance.

If odiff cannot compare the images (reason == "error", e.g. a file that cannot be loaded or has an unsupported format), both expectations fail and the failure message includes odiff's error message. In particular, an error does not count as the images differing for expect_images_differ().

Both expectations will skip (not fail) if the odiff binary is not available, making tests portable across environments.

Value

Invisibly returns the comparison result (a data.frame/tibble with match, reason, diff_count, diff_percentage, error, etc.), allowing further inspection if needed.

Comparison with vdiffr

odiffr expectations are designed for **pixel-based** comparison of screenshots, rendered images, and bitmap files. For **SVG-based** comparison of ggplot2 and grid graphics, consider using the vdiffr package instead. The two approaches are complementary.

See Also

`compare_images()` for the underlying comparison function, `ignore_region()` for excluding regions from comparison.

Examples

```
## Not run:
# Basic visual regression test
test_that("login page renders correctly", {
  skip_if_no_odiff()

  expect_images_match(
    "screenshots/login_current.png",
    "screenshots/login_baseline.png"
  )
})

# With tolerance for minor differences
test_that("chart renders correctly", {
  skip_if_no_odiff()

  expect_images_match(
    "actual_chart.png",
    "expected_chart.png",
    threshold = 0.2,
    antialiasing = TRUE,
    ignore_regions = list(
      ignore_region(0, 0, 100, 30)  # Ignore timestamp
    )
  )
})

# Assert images are different
test_that("button changes on hover", {
  skip_if_no_odiff()

  expect_images_differ(
    "button_normal.png",
    "button_hover.png"
  )
```

```
})

## End(Not run)
```

`expect_snapshot_image` *Snapshot Testing for Images*

Description

`expect_snapshot_image()` is a testthat snapshot expectation for images that compares the image with its stored snapshot using odiff, rather than requiring the files to be byte-for-byte identical. Snapshots are managed with testthat's usual tools (`testthat::snapshot_review()`, `testthat::snapshot_accept()`).

Usage

```
expect_snapshot_image(
  x,
  name = NULL,
  threshold = 0.1,
  antialiasing = FALSE,
  ignore_regions = NULL,
  fail_on_layout = TRUE,
  plot_options = NULL,
  variant = NULL,
  ...,
  max_diff_percent = NULL,
  max_diff_pixels = NULL,
  preset = NULL,
  diff_dir = getOption("odiffr.snapshot_diff_dir")
)
```

Arguments

`x`	The image to snapshot: a path to an image file (PNG; other formats are converted to PNG with magick), a magick-image object, a ggplot object, a function of no arguments that draws a plot (base or grid graphics) or returns a ggplot, lattice or grid object when called, or a recorded plot (`grDevices::recordPlot()`). Plots are rendered to PNG using `plot_options`.
`name`	Snapshot file name. A `.png` extension is added if missing. If `NULL` (the default), the name is the base name of `x` when `x` is a file path, or the variable name when `x` is a simple variable (e.g. `expect_snapshot_image(p)` uses `"p.png"`). For any other expression (e.g. an inline function), `name` must be supplied. Names must be unique within a test file.
`threshold`	Numeric; colour difference threshold between 0.0 and 1.0. Default is 0.1 (or the value from `preset`).
`antialiasing`	Logical; if `TRUE`, ignore antialiased pixels. Default is `FALSE` (or the value from `preset`).

`ignore_regions` List of regions to ignore during comparison. Use `ignore_region()` to create regions, or pass a data.frame with columns `x1`, `y1`, `x2`, `y2`.

`fail_on_layout` Logical; if `TRUE` (the default), odiff stops at the size check when images have different dimensions. Such images fail either way; `FALSE` adds pixel counts and a diff image where odiff can (odiff >= 4.3.5).

`plot_options` Options for rendering plot inputs, created with `plot_options()`. `NULL` uses the defaults of `plot_options()`.

`variant` If not `NULL`, the snapshot is stored in a variant-specific subdirectory (`_snaps/<variant>/<file>/`). See the section on platform differences.

`...` Additional arguments passed to `odiff_run()` (via `compare_file_odiff()`).

`max_diff_percent`, `max_diff_pixels`
Image-level tolerance: the largest percentage (0 to 100) and/or number of differing pixels to accept. `NULL` (the default) accepts no difference. A comparison within the tolerance passes and keeps the original snapshot, so differences cannot accumulate. Images of different sizes always fail. See the "Tolerance" section of `compare_images()` before using this.

`preset` Optional name of a comparison preset, see `odiff_preset()`: `"strict"`, `"default"`, `"screenshot"` or `"cross_platform"`. The preset supplies `threshold` and `antialiasing`; values given explicitly for those arguments take precedence. `NULL` (the default) uses the argument defaults.

`diff_dir` Where to write a diff image when the comparison fails. See the section "Diff images" in `compare_file_odiff()`. `NULL` (the default, unless the `odiffr.snapshot_diff_dir` option is set) uses `tests/testthat/_odiffr/`; `FALSE` disables diff images.

Details

On the first run, the image is saved as the snapshot `tests/testthat/_snaps/<test-file>/<name>.png` and testthat emits an "Adding new file snapshot" warning. On subsequent runs the image is compared with the snapshot using odiff and the expectation fails if they differ (beyond `threshold`, after `ignore_regions` and, optionally, antialiasing are taken into account). On failure, the new image is saved next to the snapshot as `<name>.new.png`, and a diff image highlighting the changed pixels is written to `tests/testthat/_odiffr/` (outside `_snaps/`, see `compare_file_odiff()`); a message gives its path.

Like `testthat::expect_snapshot_file()`, on which it is built:

- It requires the third edition of testthat.
- It is skipped on CRAN (when `NOT_CRAN` is not `"true"`), because snapshots are not shipped reliably and image rendering differs between machines.
- Snapshots must be committed to version control: depending on the testthat version, a missing snapshot may be reported as a failure rather than created when running on CI (the `CI` environment variable is `"true"`).

The expectation is skipped if the odiff binary is not available.

If odiff cannot compare the images (e.g. the stored snapshot is not a valid image), a warning with odiff's error message is given and the expectation fails, so the new image can still be reviewed and accepted.

Value

Invisibly returns NULL, like other testthat snapshot expectations.

Reviewing changes

When a snapshot changes, run testthat::snapshot_review() to compare the old and new images side by side in an interactive viewer, then accept the new image with testthat::snapshot_accept() (or from the viewer) if the change is intended. Unwanted .new.png files are removed on the next successful run.

Platform differences

Rendered plots can differ slightly between operating systems, graphics devices and installed fonts. To reduce spurious failures:

- Install the ragg package: plots are then rendered with ragg::agg_png(), which gives consistent output across platforms.
- Use a tolerant comparison (preset = "screenshot" or preset = "cross_platform", or threshold, antialiasing = TRUE, ignore_regions) for anti-aliasing and sub-pixel differences.
- Store separate snapshots per platform with variant, e.g. variant = Sys.info()[["sysname"]]. This is the reliable answer to different fonts, which no colour threshold absorbs.
- If one baseline must be checked on machines with different fonts, accept a small share of differing pixels with max_diff_percent. Choose the limit from measurements, and keep in mind that it also hides real changes of that size (see the "Tolerance" section of compare_images()).

Comparison with vdiffr

vdiffr snapshots plots as SVG and compares the SVG text. odiffr compares rendered pixels, which also works for images that are not plots (e.g. screenshots or magick images) and tolerates small rendering differences.

See Also

compare_file_odiff() for the comparison function, expect_images_match() for comparing against a baseline file that you manage yourself, testthat::expect_snapshot_file().

Examples

```
## Not run:
# tests/testthat/test-plots.R
test_that("scatter plot is stable", {
  p <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) +
    ggplot2::geom_point()
  expect_snapshot_image(p)  # snapshot: _snaps/plots/p.png
})

test_that("base graphics histogram is stable", {
  expect_snapshot_image(
    function() hist(mtcars$mpg),
```

```
    name = "mpg-histogram",
    plot_options = plot_options(width = 5, height = 4),
    antialiasing = TRUE
  )
})

test_that("screenshot is stable on each OS", {
  expect_snapshot_image(
    "output/screenshot.png",
    preset = "screenshot",
    ignore_regions = list(ignore_region(0, 0, 200, 40)),  # timestamp
    variant = Sys.info()[["sysname"]]
  )
})

# After an intended change, review and accept the new snapshots:
testthat::snapshot_review("plots")
testthat::snapshot_accept("plots")

## End(Not run)
```

`failed_pairs` *Get Failed Comparisons from Batch Results*

Description

Extract only the failed (non-matching) comparisons from batch results.

Usage

```
failed_pairs(object)
```

Arguments

`object`	An `odiffr_batch` object from `compare_images_batch()` or `compare_image_dirs()`.

Value

A tibble or data.frame containing only rows where `match` is `FALSE`.

See Also

`compare_images_batch()`, `compare_image_dirs()`, `passed_pairs()`

Examples

```
## Not run:
results <- compare_image_dirs("baseline/", "current/")
failed <- failed_pairs(results)
nrow(failed)  # Number of failures

## End(Not run)
```

`find_odiff` *Find the odiff Binary*

Description

Locates the odiff executable using a priority-based search:

1. User-specified path via `options(odiffr.path = "...")`
2. System PATH (`Sys.which("odiff")`)
3. Cached binary from `install_odiff()` (or `odiffr_update()`)

Usage

```
find_odiff()
```

Details

Installing odiff. If odiff cannot be found and the R session is interactive, `find_odiff()` (and so `compare_images()`, `odiff_run()` and the other functions that need the binary) offers once per session to download the latest odiff release to the user cache with `install_odiff()`. Nothing is ever downloaded without asking: the offer is never made in non-interactive sessions, while running tests with testthat, while knitting, or during `R CMD check`. Set `options(odiffr.ask_install = FALSE)` to turn the offer off. If the offer is declined, `find_odiff()` signals an error explaining how to install odiff. `odiff_available()` never makes the offer.

npm installs. Since odiff 4.4, `npm install -g odiff-bin` puts a small Node.js launcher script on the PATH, which starts Node and then spawns the native binary shipped in an `@odiff/<platform>-<arch>` package. Starting Node adds tens of milliseconds to every comparison, so when the odiff found on the PATH is such a launcher (a `#!` script that runs `node`, or an npm `.cmd`/`.ps1` wrapper on Windows), `find_odiff()` looks for the native binary inside the npm installation and returns it instead. For older `odiff-bin` releases, the native `bin/odiff.exe` inside the package is used. If no native binary can be found, the launcher itself is returned, so a working setup is never broken. The lookup is cached per launcher path and redone when the launcher file changes.

Set `options(odiffr.resolve_npm = FALSE)` to disable this and always use the PATH entry as-is. A path given via `options(odiffr.path = ...)` is always used exactly as specified and is never resolved.

Value

Character string with the absolute path to the odiff executable.

See Also

odiff_info() to see which binary is used and whether it was resolved from an npm launcher.

Examples

```
## Not run:
find_odiff()

# Use the npm launcher on the PATH as-is
options(odiffr.resolve_npm = FALSE)
find_odiff()

## End(Not run)
```

---

ignore_region *Create an Ignore Region*

Description

Helper function to create a region specification for use with odiff_run() and compare_images().

Usage

```
ignore_region(x1, y1, x2, y2)
```

Arguments

x1	Integer; x-coordinate of the top-left corner.
y1	Integer; y-coordinate of the top-left corner.
x2	Integer; x-coordinate of the bottom-right corner.
y2	Integer; y-coordinate of the bottom-right corner.

Value

A list with components x1, y1, x2, y2.

Examples

```
# Create a region to ignore
region <- ignore_region(10, 10, 100, 50)

# Use with odiff_run
## Not run:
result <- odiff_run("img1.png", "img2.png",
                    ignore_regions = list(region))

## End(Not run)
```

`install_odiff` *Install odiff*

Description

Downloads the odiff binary for your platform from the odiff GitHub releases to the odiffr user cache (`odiffr_cache_path()`), where `find_odiff()` finds it. This is the easiest way to install odiff: it needs neither Node.js nor npm, nor administrator rights.

Usage

```
install_odiff(version = "latest", force = FALSE)
```

Arguments

`version`	Character string specifying the version to download. Use `"latest"` (default) to download the most recent release, or specify a version like `"v4.1.2"` or `"4.1.2"` (the v prefix is optional).
`force`	Logical; if `TRUE`, re-download even if the binary already exists in the cache. Default is `FALSE`.

Details

`install_odiff()` is a user-facing wrapper around `odiffr_update()`: it downloads the binary (see `odiffr_update()` for details, e.g. on GitHub API rate limits), clears odiffr's cached binary lookups so that `find_odiff()` and `odiff_version()` use the new binary straight away, and reports the installed version and path.

The cached binary is used only when no binary is set via `options(odiffr.path)` and no odiff is found on the PATH; a message says so when another binary takes precedence.

odiffr never downloads anything on its own: the binary is only downloaded when `install_odiff()` (or `odiffr_update()`) is called, or when you accept the offer `find_odiff()` makes in interactive sessions.

Value

The path to the installed binary (invisibly).

See Also

`odiffr_update()`, `odiffr_clear_cache()`, `odiff_info()`

Examples

```
## Not run:
install_odiff()

# A specific version
install_odiff(version = "4.5.0", force = TRUE)

## End(Not run)
```

`odiff_available` *Check if odiff is Available*

Description

A silent check: unlike `find_odiff()`, it never offers to install odiff, so it is safe to use in skip conditions and scripts.

Usage

```
odiff_available()
```

Value

Logical `TRUE` if odiff is found and executable, `FALSE` otherwise.

Examples

```
odiff_available()
```

`odiff_info` *Display odiff Configuration Information*

Description

Display odiff Configuration Information

Usage

```
odiff_info()
```

Value

A list with components:

os Operating system (darwin, linux, windows)

arch Architecture (arm64, x64)

path Path to the odiff binary

version odiff version string

source Source of the binary (option, system, cached)

shim Path of the npm launcher script found on the PATH that `path` was resolved from, or `NA` if the binary is used directly. See `find_odiff()`.

Examples

```
## Not run:
odiff_info()

## End(Not run)
```

`odiff_preset` *Comparison Presets*

Description

Named sets of comparison settings (`threshold` and `antialiasing`) for common situations. Pass the name as `preset` to `compare_file_odiff()`, `expect_snapshot_image()` or `snapshot_report()`, or splice the values into other functions with `do.call()`.

Usage

```
odiff_preset(name = c("strict", "default", "screenshot", "cross_platform"))
```

Arguments

`name`	One of `"strict"`, `"default"`, `"screenshot"` or `"cross_platform"`.

Details

`"strict"` `threshold` = `0`, `antialiasing` = `FALSE`. Any change to any pixel fails.

`"default"` `threshold` = `0.1`, `antialiasing` = `FALSE`. The defaults of `compare_images()` and `expect_snapshot_image()`.

`"screenshot"` `threshold` = `0.1`, `antialiasing` = `TRUE`. For browser screenshots (shinytest2, webshot2, chromote) and plots compared on the same platform. Browsers render the edges of rounded corners, circles and thin borders with slightly different anti-aliasing from run to run; odiff's anti-aliasing detection ignores those pixels. The colour threshold stays at 0.1 so that real colour changes are still caught.

`"cross_platform"` `threshold = 0.2`, `antialiasing = TRUE`. For baselines shared across machines that render the same way apart from anti-aliasing, e.g. the same fonts and graphics device on another operating system or CPU. Also tolerates edges that move by up to about half a pixel, thicker anti-aliased borders and small colour or gamma shifts. The price: changes between colours of similar brightness (e.g. a blue element turning green), darkening or lightening without a change of hue (black turning to dark grey, `#303030`) and very faint elements (light grey on white) can go unnoticed. It does little for different fonts or text rendering, which change the shape and position of glyphs rather than their colour. In one calibration of 640 plots rendered on five Linux distributions, raising the threshold from 0.1 to 0.2 left the 95th percentile of differing pixels almost unchanged (2.78% and 2.75%). Use snapshot variants or `ignore_regions` for font differences or, when one baseline must be checked across environments, an image-level tolerance (`max_diff_percent`, see the "Tolerance" section of `compare_images()`).

The values were calibrated with images rendered by ragg: shapes with rounded corners and 1px borders drawn at sub-pixel offsets (0.25 and 0.5 px), and with a different anti-aliasing rasteriser (cairo), should pass, while a new 10 x 10 pixel element, or a 10 x 10 pixel patch recoloured from blue to green, must fail. With the `"default"` settings the anti-aliasing-only differences fail (5 to 439 differing pixels); with `"screenshot"` they pass, except a 0.5 px shift of a bordered shape (2 pixels), which `"cross_platform"` passes too. A blue to green recolouring is detected up to a threshold of 0.14, which is why `"screenshot"` keeps 0.1; a black patch turning `#303030` is caught by `"screenshot"` and missed by `"cross_platform"`. The calibration is part of the package's tests.

Value

A named list with elements `threshold` and `antialiasing`.

See Also

`compare_file_odiff()`, `expect_snapshot_image()`

Examples

```
odiff_preset("screenshot")

## Not run:
# Use with any comparison function
do.call(compare_images, c(list("before.png", "after.png"),
                          odiff_preset("cross_platform")))

## End(Not run)
```

`odiff_run` *Run odiff Command (Low-Level)*

Description

Direct wrapper around the odiff CLI with zero external dependencies. Returns a structured list with comparison results.

Usage

```
odiff_run(
  img1,
  img2,
  diff_output = NULL,
  threshold = 0.1,
  antialiasing = FALSE,
  fail_on_layout = FALSE,
  diff_mask = FALSE,
  diff_overlay = NULL,
  diff_color = NULL,
  diff_lines = FALSE,
  reduce_ram = FALSE,
  enable_asm = FALSE,
  ignore_regions = NULL,
  timeout = 60,
  diff_cols = FALSE
)
```

Arguments

img1	Character; path to the first (baseline) image file.
img2	Character; path to the second (comparison) image file.
diff_output	Character or NULL; optional path for the diff output image. odiff only writes PNG: a path with a different extension has it replaced by .png, and a path with no extension gets .png appended (both with a warning). If NULL, no diff image is created. No diff image is written when the images match.
threshold	Numeric; colour difference threshold between 0.0 and 1.0. Lower values are more precise. Default is 0.1.
antialiasing	Logical; if TRUE, ignore antialiased pixels. Default is FALSE.
fail_on_layout	Logical; if TRUE, fail immediately if images have different dimensions. Default is FALSE. Images of different dimensions never match and are reported as "layout-diff" either way; TRUE stops odiff at the size check, while FALSE also gives pixel counts and a diff image where odiff can (odiff >= 4.3.5).
diff_mask	Logical; if TRUE, output only the changed pixels in the diff image. Default is FALSE.
diff_overlay	Logical or numeric; if TRUE or a number between 0 and 1, add a white shaded overlay to the diff image for easier reading. Default is NULL (no overlay).
diff_color	Character; hex color for highlighting differences, in the form "#RRGGBB" or "RRGGBB" (e.g., "#FF0000"). Default is NULL (uses odiff default, red).
diff_lines	Logical; if TRUE, include line numbers containing different pixels in the output. Default is FALSE.
reduce_ram	Logical; if TRUE, use less memory but run slower. Useful for very large images. Default is FALSE.

`enable_asm` Logical; if `TRUE`, pass `--enable-asm` to the underlying `odiff` binary to enable assembly-optimised code paths (e.g. AVX-512) on supported CPUs. Requires odiff >= 4.1.1. Default is `FALSE`.

`ignore_regions` A list of regions to ignore during comparison. Each region should be a list with `x1`, `y1`, `x2`, `y2` components, or use `ignore_region()` to create them. Can also be a data.frame with these columns.

`timeout` Numeric; timeout in seconds for the odiff process. Default is 60. Positive values below one second are rounded up to one second (the resolution of `system2()`); `0` or `Inf` means no timeout. If the timeout is reached, the result has `reason = "error"` and an `error` message.

`diff_cols` Logical; if `TRUE`, include column numbers containing different pixels in the output (`--output-diff-cols`). Requires odiff >= 4.5.0; ignored with a warning for older versions. Default is `FALSE`.

Details

The `enable_asm` option is an advanced, platform-specific optimisation flag. For odiff < 4.1.1, odiffr ignores `enable_asm` with a warning. Behaviour on unsupported CPUs is determined by odiff itself.

odiff is always invoked with `--parsable-stdout`, and its machine-readable output is parsed to fill `diff_count`, `diff_lines` and `diff_cols`. `diff_percentage` is computed from `diff_count` and the image dimensions, since odiff rounds it to 2 decimal places.

Images of different dimensions are always reported as `"layout-diff"`, whatever the odiff version. With `fail_on_layout = FALSE`, odiff >= 4.3.5 counts the pixels outside the shared area as different, and those counts are kept; odiff < 4.3.5 compares only the shared area, so its counts are discarded (`NA`).

Value

A list with the following components:

match Logical; `TRUE` if images match, `FALSE` otherwise.

reason Character; one of `"match"`, `"pixel-diff"`, `"layout-diff"`, or `"error"`.

diff_count Integer; number of different pixels (`0` for a match), or `NA` if unknown (an error, or a layout difference that odiff did not count; see `fail_on_layout`).

diff_percentage Numeric; percentage of different pixels (`0` for a match), or `NA` if unknown.

diff_lines Integer vector of line numbers with differences, or `NULL`.

exit_code Integer; odiff exit code (0 = match, 21 = layout diff, 22 = pixel diff, other values = error).

stdout Character; raw stdout output (odiff's parsable output).

stderr Character; raw stderr output.

error Character; `NA` if no error occurred, otherwise the error message reported by odiff (or by odiffr, e.g. on timeout).

img1 Character; path to first image.

img2 Character; path to second image.

diff_output Character or `NULL`; path to diff image if created.

duration Numeric; time elapsed in seconds.

diff_cols Integer vector of column numbers with differences, or NULL. Only present when diff_cols = TRUE.

params Named list of the effective comparison parameters (after version guards): threshold, antialiasing, fail_on_layout, ignore_regions (formatted as "x1:y1-x2:y2,..."), or NA), diff_mask, diff_overlay (NA if unset), diff_color (NA if unset), reduce_ram and enable_asm. Used by audit_record().

See Also

compare_images() for a higher-level interface, ignore_region() for creating ignore regions.

Examples

```
## Not run:
# Basic comparison
result <- odiff_run("baseline.png", "current.png")
result$match

# With diff output
result <- odiff_run("baseline.png", "current.png", "diff.png")

# With threshold and antialiasing
result <- odiff_run("baseline.png", "current.png",
                    threshold = 0.05, antialiasing = TRUE)

# Ignoring specific regions
result <- odiff_run("baseline.png", "current.png",
                    ignore_regions = list(
                      ignore_region(10, 10, 100, 50),
                      ignore_region(200, 200, 300, 300)
                    ))

## End(Not run)
```

---

odiff_version *Get odiff Version*

Description

The version is cached per binary: repeated calls do not spawn odiff --version again unless the binary path (or the file itself) changes.

Usage

```
odiff_version()
```

Value

Character string with the odiff version, or `NA_character_` if unavailable.

Examples

```
## Not run:
odiff_version()

## End(Not run)
```

`odiffr_cache_path` *Get Cache Directory Path*

Description

Returns the path to the odiffr cache directory where downloaded binaries are stored.

Usage

```
odiffr_cache_path()
```

Value

Character string with the path to the cache directory.

Examples

```
odiffr_cache_path()
```

`odiffr_clear_cache` *Clear the odiffr Cache*

Description

Removes all cached binaries downloaded by `odiffr_update()`.

Usage

```
odiffr_clear_cache()
```

Value

Invisibly returns `TRUE` if successful, `FALSE` otherwise.

Examples

```
## Not run:
odiffr_clear_cache()

## End(Not run)
```

`odiffr_update` *Download Latest odiff Binary*

Description

Downloads the odiff binary from GitHub releases to the user's cache directory. The downloaded binary will be used by `find_odiff()` if no system-wide installation or user-specified path is found. `install_odiff()` is the recommended, user-facing way to do this.

Usage

```
odiffr_update(version = "latest", force = FALSE)
```

Arguments

`version`	Character string specifying the version to download. Use `"latest"` (default) to download the most recent release, or specify a version like `"v4.1.2"` or `"4.1.2"` (the v prefix is optional).
`force`	Logical; if `TRUE`, re-download even if the binary already exists in the cache. Default is `FALSE`.

Details

The latest release is looked up via the GitHub API. If the `GITHUB_PAT` or `GITHUB_TOKEN` environment variable is set, it is sent as a bearer token to avoid API rate limits.

The binary is downloaded to a temporary file next to its final location and only moved into place once the download has succeeded, so an interrupted download never leaves a broken binary behind. The download timeout (`getOption("timeout")`) is temporarily raised to at least 300 seconds. The downloaded version is recorded in a `CACHED_VERSION` file next to the binary.

Note that some odiff releases were published without binary assets; if the download fails with HTTP 404, try another version.

Value

Character string with the path to the downloaded binary.

Examples

```
## Not run:
# Download latest version
odiffr_update()

# Download specific version
odiffr_update(version = "v4.1.2")

# Force re-download
odiffr_update(force = TRUE)

## End(Not run)
```

`passed_pairs` *Get Passed Comparisons from Batch Results*

Description

Extract only the passed (matching) comparisons from batch results.

Usage

```
passed_pairs(object)
```

Arguments

`object`	An `odiffr_batch` object from `compare_images_batch()` or `compare_image_dirs()`.

Value

A tibble or data.frame containing only rows where `match` is `TRUE`.

See Also

`compare_images_batch()`, `compare_image_dirs()`, `failed_pairs()`

Examples

```
## Not run:
results <- compare_image_dirs("baseline/", "current/")
passed <- passed_pairs(results)
nrow(passed)  # Number of passing comparisons

## End(Not run)
```

`plot.odiff_result` *Plot an odiff Comparison*

Description

Display the baseline, current and diff images of an `odiff_run()` result in the current graphics device, using base graphics.

Usage

```
## S3 method for class 'odiff_result'
plot(x, which = c("all", "diff", "baseline", "current"), ...)
```

Arguments

`x`	An `odiff_result` object returned by `odiff_run()`.
`which`	Which image(s) to show: `"all"` (default) shows the baseline, current and diff images side by side; `"diff"`, `"baseline"` or `"current"` show a single image.
`...`	Currently unused.

Details

PNG images are read with `png::readPNG()` (the png package must be installed). Other formats (JPEG, WEBP, TIFF, ...) are read with the magick package if it is installed.

A diff image is only available when the comparison was run with `diff_output` and the images differ; otherwise the diff panel says "No diff image".

The graphical parameters (`par()`) are restored on exit.

For data-frame results of `compare_images()` or a row of a batch, use `diff_image()` to get the diff image.

Value

`x`, invisibly.

See Also

`diff_image()`, `odiff_run()`

Examples

```
## Not run:
result <- odiff_run("baseline.png", "current.png", diff_output = "diff.png")
plot(result)
plot(result, which = "diff")

## End(Not run)
```

plot_options *Plot Rendering Options*

Description

Create a set of options controlling how plot inputs (ggplot objects, plotting functions and recorded plots) are rendered to PNG before being compared by `compare_images()`, `expect_images_match()`, `expect_images_differ()` and `expect_snapshot_image()`.

Usage

```
plot_options(width = 7, height = 5, units = "in", res = 96, bg = "white")
```

Arguments

`width, height`	Plot dimensions, in `units`. Default is 7 x 5 inches.
`units`	Units for `width` and `height`: one of `"in"`, `"cm"`, `"mm"` or `"px"`. Default is `"in"`.
`res`	Resolution in pixels per inch. Default is 96.
`bg`	Background colour. Default is `"white"`.

Details

Plots are rendered with `ragg::agg_png()` when the ragg package is installed (recommended: its output is consistent across platforms), and with `grDevices::png()` otherwise. When comparing a plot against a stored baseline PNG, render it with the same options (and the same graphics device) that were used to create the baseline, or the comparison will fail with a layout difference.

Value

A list of class `odiffr_plot_options`, to be passed as the `plot_options` argument.

See Also

`compare_images()`, `expect_snapshot_image()`

Examples

```
plot_options(width = 4, height = 3, res = 72)
```

snapshot_report *Report Failing Image Snapshots*

Description

Finds the image snapshots that changed in the last test run (the <name>.new.png files testthat leaves next to <name>.png in tests/testthat/_snaps/), compares each with its baseline using odiff and writes an HTML, Markdown or JUnit XML report. Unlike testthat::snapshot_review(), this works non-interactively, so it is suited to continuous integration.

Usage

```
snapshot_report(
  path = "tests/testthat/_snaps",
  output_file = NULL,
  format = c("html", "markdown", "junit"),
  threshold = 0.1,
  antialiasing = FALSE,
  ...,
  images = "all",
  embed = TRUE,
  title = "Image snapshot changes",
  diff_dir = NULL,
  preset = NULL
)
```

Arguments

path	Path to the snapshot directory. Searched recursively, so variant subdirectories (_snaps/<variant>/<test-file>/) are included. Default: "tests/testthat/_snaps".
output_file	Path of the report file. Required for "html" and "junit". For "markdown", NULL (the default) appends to the GitHub Actions job summary when the GITHUB_STEP_SUMMARY environment variable is set, and otherwise writes nothing (see batch_markdown()).
format	Report format: "html" (batch_report()), "markdown" (batch_markdown()) or "junit" (batch_junit()).
threshold	Numeric; colour difference threshold between 0.0 and 1.0. Default is 0.1 (or the value from preset). Use the settings of the tests that produced the snapshots to get the same verdicts.
antialiasing	Logical; if TRUE, ignore antialiased pixels. Default is FALSE (or the value from preset).
...	Additional arguments passed to compare_images() (and from there to odiff_run()), e.g. ignore_regions or max_diff_percent. Use the settings of the tests that produced the snapshots.
images	For "html": which images to show per snapshot, "all" (default: baseline, new and diff side by side) or "diff".

embed	For "html": if TRUE (default), images are embedded so the report is a single self-contained file, e.g. for a CI artifact.
title	Report title. Default: "Image snapshot changes".
diff_dir	Directory for the diff images. NULL (default) uses a snapshot-diffs directory next to output_file, or a directory in tempdir() when there is no output_file. It must not be inside path, because testthat deletes unknown files in _snaps/. Diff images left there by an earlier report (NNN_<name>_diff.png) are removed first.
preset	Optional comparison preset, see odiff_preset(). It supplies threshold and antialiasing unless those are given.

Details

Each <name>.new.png is paired with <name>.png in the same directory and compared with compare_images_batch(). The returned batch has an extra column snapshot with the snapshot's path relative to path (e.g. "linux/plots/scatter.png"). Reports label rows by this relative snapshot path.

A .new.png file without a baseline cannot be compared; it is reported with reason = "error" and the error "no baseline snapshot", so it is not silently missed.

If no .new.png files are found, an empty batch is returned. The HTML and JUnit reports are still written (with no comparisons); the Markdown report contains "No image snapshot changes.".

Snapshots whose new image matches the baseline under the given settings are reported as passing: testthat compares more strictly (or with other settings) than the report.

Value

The comparison results, an odiffr_batch (invisibly).

Continuous integration

Run the tests without stopping at the first failure, then write the report. In GitHub Actions:

```
- name: Test
  run: |
    res <- testthat::test_local(stop_on_failure = FALSE)
    odiffr::snapshot_report(format = "markdown")
    odiffr::snapshot_report(output_file = "snapshot-report.html")
    if (any(as.data.frame(res)$failed > 0)) stop("Tests failed")
  shell: Rscript {0}

- name: Upload snapshot report
  if: always()
  uses: actions/upload-artifact@v4
  with:
    name: snapshot-report
    path: snapshot-report.html
```

See Also

expect_snapshot_image(), compare_file_odiff(), batch_report(), batch_markdown(), batch_junit()

Examples

```
## Not run:
# After a test run with failing image snapshots
snapshot_report(output_file = "snapshot-report.html")

# GitHub Actions job summary
snapshot_report(format = "markdown")

# JUnit XML for a CI test reporter
snapshot_report(output_file = "snapshots.xml", format = "junit")

## End(Not run)
```

---

summary.odiffr_batch *Summarize Batch Comparison Results*

Description

Generate a summary of batch image comparison results, including pass/fail statistics, failure reasons, and worst offenders.

Usage

```
## S3 method for class 'odiffr_batch'
summary(object, n_worst = 5, ...)

## S3 method for class 'odiffr_batch_summary'
print(x, ...)
```

Arguments

object	An odiffr_batch object returned by compare_images_batch() or compare_image_dirs().
n_worst	Integer; number of worst offenders to include in the summary. Default is 5.
...	Additional arguments (currently unused).
x	An odiffr_batch_summary object.

Details

The summary method expects the standard output of compare_images_batch(), which includes columns: match, reason, diff_percentage, diff_count, pair_id, and img2.

Value

An `odiffr_batch_summary` object with the following components:

total Total number of comparisons.

passed Number of matching image pairs.

failed Number of non-matching image pairs.

tolerated Number of passing comparisons that differ within a tolerance (`reason = "within-tolerance"`, see `compare_images()`).

pass_rate Proportion of passing comparisons (0 to 1), or `NA` for an empty batch (zero comparisons).

reason_counts Table of failure reasons (NULL if no failures).

diff_stats List with min, median, mean, max diff percentages (NULL if no failures with diff data).

worst Data frame of worst offenders, ordered by diff percentage (descending). Failures without a diff percentage (e.g. layout differences, errors, or missing files) are listed after those with one. NULL if no failures.

See Also

`compare_images_batch()`, `compare_image_dirs()`

Examples

```
## Not run:
# Compare image pairs and summarize
pairs <- data.frame(
  img1 = c("baseline/a.png", "baseline/b.png", "baseline/c.png"),
  img2 = c("current/a.png", "current/b.png", "current/c.png")
)
results <- compare_images_batch(pairs)
summary(results)

# Get summary with more worst offenders
summary(results, n_worst = 10)

## End(Not run)
```

---

use_odiffr_ci *Add a GitHub Actions Workflow for Visual Tests*

Description

Writes a ready-to-use GitHub Actions workflow that runs your package's testthat tests, including image snapshot tests, with odiff. The workflow installs odiff with `install_odiff()` (no Node.js needed), adds a `snapshot_report()` summary of changed image snapshots to the job summary and, when tests fail, uploads the new snapshots (`*.new.png`) and diff images (`tests/testthat/_odiffr/`) as an artifact.

Usage

```
use_odiffr_ci(
  path = ".github/workflows/odiffr.yaml",
  overwrite = FALSE,
  open = FALSE
)
```

Arguments

path	Path of the workflow file, relative to the working directory (which should be the root of your package). Default: ".github/workflows/odiffr.yaml". Parent directories are created as needed.
overwrite	Logical; if TRUE, replace an existing file at path. Default is FALSE, in which case an existing file is an error.
open	Logical; if TRUE and the session is interactive, open the new file with utils::file.edit(). Default is FALSE.

Details

The workflow is written only when you call this function; odiffr never writes it by itself. It runs on every push and pull request on ubuntu-latest, so record image snapshots on Linux or use a tolerant preset (see odiff_preset()). The template is installed with odiffr, see system.file("templates", "odiffr-ci.yaml", package = "odiffr"); edit the written file as needed.

The tests run with NOT_CRAN=true, as snapshot tests are skipped on CRAN.

Value

The path of the written file (invisibly).

See Also

install_odiff(), snapshot_report(), expect_snapshot_image()

Examples

```
## Not run:
# From the root of your package
use_odiffr_ci()

## End(Not run)

# In a temporary directory
dir <- tempfile()
dir.create(dir)
path <- use_odiffr_ci(file.path(dir, ".github", "workflows", "odiffr.yaml"))
unlink(dir, recursive = TRUE)
```

Index

approve_changes, 3
audit_record, 5
audit_record(), *41*

batch_junit, 7
batch_junit(), *9*, *47*, *49*
batch_markdown, 9
batch_markdown(), *8*, *47*, *49*
batch_report, 10
batch_report(), *3, 4*, *8, 9*, *12, 13*, *22*, *24*, *47*, *49*

compare_dirs_report, 12
compare_file_odiff, 13
compare_file_odiff(), *30, 31*, *37, 38*, *49*
compare_image_dirs, 15
compare_image_dirs(), *3–5*, *8–13*, *20*, *22*, *24*, *32*, *44*, *49, 50*
compare_images, 17
compare_images(), *4, 5*, *14*, *16*, *20*, *22*, *24, 25*, *27, 28*, *30, 31*, *33, 34*, *37, 38*, *41*, *45–47*, *50*
compare_images_batch, 20
compare_images_batch(), *3*, *5*, *8–12*, *16*, *22*, *24*, *32*, *44*, *48–50*
compare_pdf_dirs, 21
compare_pdf_dirs(), *24*
compare_pdfs, 23
compare_pdfs(), *21, 22*

diff_image, 25
diff_image(), *45*
do.call(), *37*

expect_images_differ *(*expect_images_match*)*, 26
expect_images_differ(), *46*
expect_images_match, 26
expect_images_match(), *14*, *31*, *46*
expect_snapshot_image, 29
expect_snapshot_image(), *13*, *15*, *37, 38*, *46*, *49*, *51*

failed_pairs, 32
failed_pairs(), *4*, *44*
find_odiff, 33
find_odiff(), *6*, *35–37*

grDevices::as.raster(), *25*
grDevices::png(), *46*
grDevices::recordPlot(), *18*, *29*

ignore_region, 34
ignore_region(), *14*, *18, 19*, *24*, *27, 28*, *30*, *40, 41*
install_odiff, 35
install_odiff(), *33*, *43*, *50, 51*

odiff_available, 36
odiff_available(), *33*
odiff_info, 36
odiff_info(), *7*, *34, 35*
odiff_preset, 37
odiff_preset(), *14, 15*, *30*, *48*, *51*
odiff_run, 38
odiff_run(), *5*, *14*, *18, 19*, *25*, *27*, *30*, *33, 34*, *45*, *47*
odiff_version, 41
odiff_version(), *7*, *35*
odiffr_cache_path, 42
odiffr_cache_path(), *35*
odiffr_clear_cache, 42
odiffr_clear_cache(), *35*
odiffr_update, 43
odiffr_update(), *35*

passed_pairs, 44
passed_pairs(), *32*
pdftools::pdf_convert(), *24*
plot.odiff_result, 45
plot.odiff_result(), *25*

plot_options, 46
plot_options(), *18*, *30*
png::readPNG(), *25*, *45*
print.odiffr_batch_summary
        *(*summary.odiffr_batch*)*, 49

ragg::agg_png(), *31*, *46*

snapshot_report, 47
snapshot_report(), *15*, *37*, *50*, *51*
summary(), *20*, *24*
summary.odiffr_batch, 49
summary.odiffr_batch(), *9*, *11*, *20*
Sys.info(), *6*
system2(), *40*

tempdir(), *14*, *24*, *48*
testthat::expect_snapshot_file(), *13*,
        30, *31*
testthat::snapshot_accept(), *29*, *31*
testthat::snapshot_review(), *29*, *31*, *47*
tools::md5sum(), *5*

use_odiffr_ci, 50
utils::file.edit(), *51*