Package 'odiffr'

May 9, 2026

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

Version 0.5.1

Description R bindings to 'odiff', a blazing-fast pixel-by-pixel image comparison tool <https://github.com/dmtrKovalenko/odiff>. Supports PNG, JPEG, WEBP, and TIFF with configurable thresholds, antialiasing detection, and region ignoring. Requires system installation of 'odiff'. Ideal for visual regression testing in automated workflows.

SystemRequirements odiff (>= 3.0.0) - https://github.com/dmtrKovalenko/odiff

License MIT + file LICENSE

URL https://github.com/BenWolst/odiffr

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

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports tools

Suggests knitr, magick, png, rmarkdown, testthat (>= 3.1.0), tibble, withr

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Ben Wolstenholme [aut, cre]

Maintainer Ben Wolstenholme <odiffr@benwolst.dev>

Repository CRAN

Date/Publication 2025-12-09 18:40:03 UTC

Contents

batch_report . . . 2
compare_dirs_report . . . 4
compare_images . . . 5
compare_images_batch . . . 7
compare_image_dirs . . . 8
expect_images_match . . . 9
failed_pairs . . . 12
find_odiff . . . 13
ignore_region . . . 13
odiffr_cache_path . . . 14
odiffr_clear_cache . . . 14
odiffr_update . . . 15
odiff_available . . . 16
odiff_info . . . 16
odiff_run . . . 17
odiff_version . . . 19
passed_pairs . . . 19
summary.odiffr_batch . . . 20

Index **22**

`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,
  ...
)
```

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.
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.
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. 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.
...	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.

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)

# 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,
  ...
)
```

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.
...	Additional arguments passed to compare_image_dirs() (e.g. threshold, antialiasing, pattern, recursive).

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_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 or magick-image objects.

Usage

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

Arguments

img1	Path to the first image, or a magick-image object.
img2	Path to the second image, or a magick-image object.
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; color 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 if images have different dimensions. Default is `FALSE`.
`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`.
`...`	Additional arguments passed to `odiff_run()`.

Value

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

match Logical; `TRUE` if images match.

reason Character; comparison result reason.

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.

img2 Character; path to second image.

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)

# 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`.
`...`	Additional arguments passed to `compare_images()`.

Value

A tibble (if available) or data.frame with class `odiffr_batch`, containing one row per comparison with all columns from `compare_images()` plus a `pair_id` column. Use `summary()` to get aggregate statistics.

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, ]

## 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?)$",
  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. Default matches common image formats (PNG, JPEG, WEBP, TIFF).
`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()`.

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), it is included in the comparison.

- If the file is missing from `current_dir`, a warning is issued and the file is excluded from results.

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 one row per comparison, containing all columns from `compare_images()` plus a `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
results[!results$match, ]

## 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,
  ...,
  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; color 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 if images have different dimensions. Default is TRUE for tests (stricter than compare_images()).
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.
...	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").

`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.

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, 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)
```

`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 `odiffr_update()`

Usage

```
find_odiff()
```

Value

Character string with the absolute path to the odiff executable.

Examples

```
## Not run:
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)
```

---

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.

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 tag like `"v4.1.2"`.
force	Logical; if TRUE, re-download even if the binary already exists in the cache. Default is FALSE.

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)
```

odiff_available *Check if odiff is Available*

Description

Check if odiff is Available

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)

Examples

```
## Not run:
odiff_info()

## 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,
  ignore_regions = NULL,
  timeout = 60
)
```

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. Must have .png extension. If NULL, no diff image is created.
threshold	Numeric; color 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.
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 (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`.
`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.

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, or `NA`.

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

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).

stdout Character; raw stdout output.

stderr Character; raw stderr output.

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.

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

Get odiff Version

Usage

```
odiff_version()
```

Value

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

Examples

```
## Not run:
odiff_version()

## 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)
```

`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.

pass_rate Proportion of passing comparisons (0 to 1).

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 by diff percentage (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)
```

Index

batch_report, 2
batch_report(), *5*

compare_dirs_report, 4
compare_image_dirs, 8
compare_image_dirs(), *3–5*, *7*, *12*, *19–21*
compare_images, 5
compare_images(), *7*, *9–11*, *13*, *18*
compare_images_batch, 7
compare_images_batch(), *3, 4*, *8, 9*, *12*, *19–21*

expect_images_differ *(*expect_images_match*)*, 9
expect_images_match, 9

failed_pairs, 12
failed_pairs(), *19*
find_odiff, 13

ignore_region, 13
ignore_region(), *6*, *10, 11*, *18*

odiff_available, 16
odiff_info, 16
odiff_run, 17
odiff_run(), *6*, *10*, *13*
odiff_version, 19
odiffr_cache_path, 14
odiffr_clear_cache, 14
odiffr_update, 15

passed_pairs, 19
passed_pairs(), *12*
print.odiffr_batch_summary *(*summary.odiffr_batch*)*, 20

summary(), *7*
summary.odiffr_batch, 20
summary.odiffr_batch(), *3*, *7*