Package 'EQUALCompareImages'

May 7, 2026

Title Comparison of Images for Researchers Without Coding Skills

Version 0.1.0

Date 2025-08-29

Author Kurinchi Gurusamy [aut, cre]

Maintainer Kurinchi Gurusamy <k.gurusamy@ucl.ac.uk>

Depends cowplot, ggplot2, magick, knitr, zip

Imports shinybusy

Description Support functions for R-based ``EQUALCompareImages - Compare similarity between and within images'' shiny application which allow researchers without coding skills or expertise in image comparison algorithms to compare images. Gurusamy,K (2025)<doi:10.5281/zenodo.16994047>.

License GPL (>= 3)

Encoding UTF-8

URL https://sites.google.com/view/equal-group/home

NeedsCompilation no

Repository CRAN

Date/Publication 2025-09-09 13:00:11 UTC

Contents

compare_similarities_same_image *Compare similarities within sections of the same image*

Description

Divides the images into a number of sections (determined by the user) and uses **magick** to compare the similarities between the different sections of the images.It computes the absolute error and the perceptual hash of each pairwise comparison. Measures such as fuzz, root mean squared error are also available as optional measures.

Usage

```
compare_similarities_same_image(rv)
```

Arguments

rv	a list provided by user input containing the path to the image, the number of sections into which the width and height must be split, and the number of resizes.

Value

A csv file containing the results

Note

This is part of a suite of functions required to compare images.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

pairwise_comparison() compare_similarities_two_images()

Examples

```
# Create a plot
filename <- tempfile(fileext = ".png")
png(filename, width = 300, height = 450, units = "px")
plot.new()
hist(rnorm(100))
recordPlot()
dev.off()
```

```
# Create a list to simulate the uploads of shiny app
rv <- {list(
  file_upload_image_1 = cbind.data.frame(datapath = ""),
  file_upload_image_2 = cbind.data.frame(datapath = ""),
  stretch_images = "No",
  number_of_resizes_between = 1,
  extra_parameters_between = c("fuzz"),
  file_upload_image = cbind.data.frame(datapath = ""),
  number_of_splits = 3,
  number_of_resizes_within = 1,
  extra_parameters_within = c("fuzz")
)}
# Allocate the file paths
rv$file_upload_image$datapath <- filename
# Perform the function
library(magick)
library("ggplot2")
library("cowplot")
library("knitr")
library("zip")

# Not run for CRAN checks because of time it takes to run
if (interactive()) {
results <- compare_similarities_same_image(rv)
}
```

`compare_similarities_two_images` *Compare similarities between two images*

Description

Uses **magick** to compare the similarities between the different sections of the images.It computes the absolute error and the perceptual hash of each pairwise comparison. Measures such as fuzz, root mean squared error are also available as optional measures.

Usage

```
compare_similarities_two_images(rv)
```

Arguments

`rv`	a list provided by user input containing the paths to the two images being compared, whether the images must be streteched to the maximum width and height of the two images, the number of sections into which the width and height must be split, and the number of resizes.

Value

A csv file containing the results

Note

This is part of a suite of functions required to compare images.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

pairwise_comparison() compare_similarities_same_image()

Examples

```
# Create a plot and save this as file
filename_1 <- tempfile(fileext = ".png")
png(filename_1, width = 300, height = 450, units = "px")
plot.new()
hist(rnorm(100))
recordPlot()
dev.off()
# One more plot for comparison
filename_2 <- tempfile(fileext = ".png")
png(filename_2, width = 300, height = 450, units = "px")
plot.new()
boxplot(rnorm(100))
recordPlot()
dev.off()

# Create a list to simulate the uploads of shiny app
rv <- {list(
  file_upload_image_1 = cbind.data.frame(datapath = ""),
  file_upload_image_2 = cbind.data.frame(datapath = ""),
  stretch_images = "No",
  number_of_resizes_between = 1,
  extra_parameters_between = c("fuzz"),
  file_upload_image = cbind.data.frame(datapath = ""),
  number_of_splits = 3,
  number_of_resizes_within = 1,
  extra_parameters_within = c("fuzz")
)}
# Allocate the file paths
rv$file_upload_image_1$datapath <- filename_1
rv$file_upload_image_2$datapath <- filename_2
```

```
# Perform the function
library(magick)
results <- compare_similarities_two_images(rv)
```

`find_dominant_colour` *Find dominant colour in an image*

Description

This function finds the dominant colour in an image. This is used when the option "remove_dominant_colour" argument in `pairwise_comparison()` is TRUE.

Usage

```
find_dominant_colour(image)
```

Arguments

`image`	The image in which the dominant colour must be determined

Value

Dominant colour

Note

This is part of a suite of functions required to compare images.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

`pairwise_comparison()`

Examples

```
# Create a plot and save this as file
filename <- tempfile(fileext = ".png")
png(filename)
plot.new()
hist(rnorm(100))
recordPlot()
dev.off()
```

```
# Read image
library(magick)
image <- image_read(filename)
# Perform the function
results <- find_dominant_colour(image)
```

`pairwise_comparison` *Compare a pair of images*

Description

This is the main function of this package and supports both the `compare_similarities_two_images()` function and the `compare_similarities_same_image()` function.

Usage

```
pairwise_comparison(image_1, image_2,
include_dimension_insensitive_measures = FALSE, remove_dominant_colour = FALSE,
extra_parameters = "")
```

Arguments

`image_1`	The first image of the pair of images for which the comparison must be made.
`image_2`	The second image of the pair of images for which the comparison must be made.
`include_dimension_insensitive_measures`	Logical (default value is FALSE) indicating whether dimension insensitive measures such as perceptual hash must be calculated. These are necessary only for the comparison of images without any modification as these measures are expected to find similarities between the images even if they are rotated or resized.
`remove_dominant_colour`	Logical (default value is TRUE) indicating whether the dominant colour must be removed. When indicated as TRUE, this is performed only for the comparison of images without any modification as some of the algorithms to find similarities between the images even if they are rotated or resized rely on the background colour.
`extra_parameters`	By default, only absolute error and perceptual hash are calculated and reported. Measures such as fuzz, root mean squared error can also be calculated optionally.

Value

a data frame containing the pixels_compared, absolute_error, perceptual_hash, and any optional measures.

Note

This is part of a suite of functions required to compare images.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

compare_similarities_two_images() compare_similarities_same_image() find_dominant_colour()

Examples

```
# Create a plot and save this as file
filename_1 <- tempfile(fileext = ".png")
png(filename_1)
plot.new()
hist(rnorm(100))
recordPlot()
dev.off()
# One more plot for comparison
filename_2 <- tempfile(fileext = ".png")
png(filename_2)
plot.new()
boxplot(rnorm(100))
recordPlot()
dev.off()

# Read images
library(magick)
image_1 <- image_read(filename_1)
image_2 <- image_read(filename_2)

# Perform the function
results <- pairwise_comparison(image_1 = image_1, image_2 = image_2,
                               include_dimension_insensitive_measures = TRUE)
```

---

stretch_images *Stretch images to the maximum width and height of two images*

Description

This stretches the images to the maximum width and height of two images.

Usage

```
stretch_images(image_1, image_2)
```

Arguments

image_1	The first image of the pair of images for which the comparison must be made.
image_2	The second image of the pair of images for which the comparison must be made.

Value

image_1	The first image stretched to the maximum width and height of the two images being compared
image_2	The second image stretched to the maximum width and height of the two images being compared

Note

This is part of a suite of functions required to compare images.

Author(s)

Kurinchi Gurusamy

References

https://sites.google.com/view/equal-group/home

See Also

compare_similarities_two_images()

Examples

```
# Create a plot and save this as file - width less than height
filename_1 <- tempfile(fileext = ".png")
png(filename_1, width = 300, height = 450, units = "px")
plot.new()
hist(rnorm(100))
recordPlot()
dev.off()
# One more plot for comparison - width more than height
filename_2 <- tempfile(fileext = ".png")
png(filename_2, width = 450, height = 300, units = "px")
plot.new()
boxplot(rnorm(100))
recordPlot()
dev.off()

# Read images
library(magick)
image_1 <- image_read(filename_1)
image_2 <- image_read(filename_2)

# Perform the function
```

```
results <- stretch_images(image_1 = image_1, image_2 = image_2)
# Plots are stretched to the maximum height and maximum width among the two images
plot(results[[1]])
plot(results[[2]])
```

Index