

# Package ‘pliman’

November 10, 2021

**Title** Tools for Plant Image Analysis

**Version** 1.0.0

**Description** Provides tools for image manipulation that will help you to quantify plant leaf area, disease severity, number of disease lesions, and obtain statistics of image objects such as grains, pods, pollen, leaves, and more. Tools for segment images and create binary images using the method of automatic threshold selection method proposed by Otsu (1979) <[doi:10.1109/tsmc.1979.4310076](https://doi.org/10.1109/tsmc.1979.4310076)> are also provided.

**License** GPL (>= 3)

**URL** <https://github.com/TiagoOlivoto/pliman>

**BugReports** <https://github.com/TiagoOlivoto/pliman/issues>

**Depends** R (>= 4.1)

**Suggests** BiocManager, EBImage, knitr, rmarkdown

**VignetteBuilder** knitr

**biocViews**

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.1.2

**Imports** lattice

**NeedsCompilation** no

**Author** Tiago Olivoto [aut, cre] (<<https://orcid.org/0000-0002-0241-9636>>)

**Maintainer** Tiago Olivoto <[tiagoolivoto@gmail.com](mailto:tiagoolivoto@gmail.com)>

**Repository** CRAN

**Date/Publication** 2021-11-10 00:20:05 UTC

## R topics documented:

analyze_objects	2
count_lesions	8
count_objects	12
image_binary	15
image_combine	17
image_index	18
image_palette	20
image_segment	21
image_to_mat	24
leaf_area	25
measure_disease	27
objects_rgb	31
pliman_images	34
prop_segmented	35
rgb_to_hsv	37
sad	37
symptomatic_area	39
tune_tolerance	41
utils_dpi	43
utils_file	45
utils_image	47
utils_measures	48
utils_objects	51
utils_polygon	53
utils_transform	56
<b>Index</b>	<b>62</b>

---

analyze_objects	<i>Analyzes objects in an image</i>
-----------------	-------------------------------------

---

### Description

- `analyze_objects()` provides tools for counting and extracting object features (e.g., area, perimeter, radius, pixel intensity) in an image. See more at **Details** section.
- `plot.anal_obj()` Produces an histogram for the R, G, and B values when argument `object_index` is used in the function `analyze_objects()`.

### Usage

```
analyze_objects(
  img,
  foreground = NULL,
  background = NULL,
  pattern = NULL,
  img_pattern = NULL,
```

```
parallel = FALSE,
workers = NULL,
watershed = TRUE,
resize = FALSE,
trim = FALSE,
fill_hull = FALSE,
filter = FALSE,
invert = FALSE,
object_size = "medium",
index = "NB",
my_index = NULL,
object_index = NULL,
threshold = "Otsu",
tolerance = NULL,
extension = NULL,
lower_size = NULL,
upper_size = NULL,
topn_lower = NULL,
topn_upper = NULL,
lower_eccent = NULL,
upper_eccent = NULL,
randomize = TRUE,
nrows = 2000,
show_image = TRUE,
show_original = TRUE,
show_chull = FALSE,
show_contour = TRUE,
contour_col = "red",
contour_size = 1,
show_background = TRUE,
show_segmentation = FALSE,
col_foreground = NULL,
col_background = NULL,
marker = FALSE,
marker_col = NULL,
marker_size = NULL,
save_image = FALSE,
prefix = "proc_",
dir_original = NULL,
dir_processed = NULL,
verbose = TRUE
)

## S3 method for class 'anal_obj'
plot(x, facet = FALSE, ...)
```

### Arguments

`img`                    The image to be analyzed.

foreground	A color palette of the foreground (optional).
background	A color palette of the background (optional).
pattern	A pattern of file name used to identify images to be imported. For example, if <code>pattern = "im"</code> all images in the current working directory that the name matches the pattern (e.g., <code>img1.-</code> , <code>image1.-</code> , <code>im2.-</code> ) will be imported as a list. Providing any number as pattern (e.g., <code>pattern = "1"</code> ) will select images that are named as <code>1.-</code> , <code>2.-</code> , and so on. An error will be returned if the pattern matches any file that is not supported (e.g., <code>img1.pdf</code> ).
img_pattern	Deprecated. Use <code>pattern</code> instead.
parallel	If TRUE processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when <code>pattern</code> is used is informed. When <code>object_index</code> is informed, multiple sections will be used to extract the RGB values for each object in the image. This may significantly speed up processing time when an image has lots of objects (say >1000).
workers	A positive numeric scalar or a function specifying the number of parallel processes that can be active at the same time. By default, the number of sections is set up to 50% of available cores.
watershed	If TRUE (default) performs watershed-based object detection. This will detect objects even when they are touching one other. If FALSE, all pixels for each connected set of foreground pixels are set to a unique object. This is faster but is not able to segment touching objects.
resize	Resize the image before processing? Defaults to FALSE. Use a numeric value of range 0-100 (proportion of the size of the original image).
trim	Number of pixels removed from edges in the analysis. The edges of images are often shaded, which can affect image analysis. The edges of images can be removed by specifying the number of pixels. Defaults to FALSE (no trimmed edges).
fill_hull	Fill holes in the binary image? Defaults to FALSE. This is useful to fill holes in objects that have portions with a color similar to the background. <b>IMPORTANT:</b> Objects touching each other can be combined into one single object, which may underestimate the number of objects in an image.
filter	Performs median filtering after image processing? defaults to FALSE. See more at <a href="#">image_filter()</a> .
invert	Inverts the binary image, if desired. This is useful to process images with black background. Defaults to FALSE.
object_size	The size of the object. Used to automatically set up tolerance and extension parameters. One of the following. "small" (e.g, wheat grains), "medium" (e.g, soybean grains), "large"(e.g, peanut grains), and "elarge" (e.g, soybean pods)‘.
index, my_index	A character value specifying the target mode for conversion to binary image when foreground and background are not declared. Defaults to "NB" (normalized blue). See <a href="#">image_index()</a> for more details.

object_index	Defaults to FALSE. If an index is informed, the average value for each object is returned. It can be the R, G, and B values or any operation involving them, e.g., <code>object_index = "R/B"</code> . In this case, it will return for each object in the image, the average value of the R/B ratio. Use <code>pliman_indexes_eq()</code> to see the equations of available indexes.
threshold	A numeric value for the segmentation threshold. By default, a threshold value based on Otsu's method is used to reduce the grayscale image to a binary image.
tolerance	The minimum height of the object in the units of image intensity between its highest point (seed) and the point where it contacts another object (checked for every contact pixel). If the height is smaller than the tolerance, the object will be combined with one of its neighbors, which is the highest.
extension	Radius of the neighborhood in pixels for the detection of neighboring objects. Higher value smooths out small objects.
lower_size, upper_size	Lower and upper limits for size for the image analysis. Plant images often contain dirt and dust. To prevent dust from affecting the image analysis, objects with lesser than 10% of the mean of all objects are removed. Upper limit is set to NULL, i.e., no upper limit used. One can set a known area or use <code>lower_limit = 0</code> to select all objects (not advised). Objects that matches the size of a given range of sizes can be selected by setting up the two arguments. For example, if <code>lower_size = 120</code> and <code>upper_size = 140</code> , objects with size greater than or equal 120 and less than or equal 140 will be considered.
topn_lower, topn_upper	Select the top n objects based on its area. <code>topn_lower</code> selects the n elements with the smallest area whereas <code>topn_upper</code> selects the n objects with the largest area.
lower_eccent, upper_eccent	Lower and upper limit for object eccentricity for the image analysis. Users may use these arguments to remove objects such as square papers for scale (low eccentricity) or cut petioles (high eccentricity) from the images. Defaults to NULL (i.e., no lower and upper limits).
randomize	Randomize the lines before training the model?
nrows	The number of lines to be used in training step. Defaults to 2000.
show_image	Show image after processing?
show_original	Show the count objects in the original image?
show_chull	Show the convex hull around the objects? Defaults to FALSE.
show_contour	Show a contour line around the objects? Defaults to TRUE.
contour_col, contour_size	The color and size for the contour line around objects. Defaults to <code>contour_col = "red"</code> and <code>contour_size = 1</code> .
show_background	Show the background? Defaults to TRUE. A white background is shown by default when <code>show_original = FALSE</code> .
show_segmentation	Shows the object segmentation colored with random permutations. Defaults to FALSE.

col_foreground, col_background	Foreground and background color after image processing. Defaults to NULL, in which "black", and "white" are used, respectively.
marker, marker_col, marker_size	The type, color and size of the object marker. Defaults to NULL, which plots the object id. Use marker = "point" to show a point in each object or marker = FALSE to omit object marker.
save_image	Save the image after processing? The image is saved in the current working directory named as proc_* where * is the image name given in img.
prefix	The prefix to be included in the processed images. Defaults to "proc_".
dir_original, dir_processed	The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image img in the current working directory. After processing, when save_image = TRUE, the processed image will be also saved in such a directory. It can be either a full path, e.g., "C:/Desktop/imgs", or a subfolder within the current working directory, e.g., "/imgs".
verbose	If TRUE (default) a summary is shown in the console.
x	An object of class anal_obj.
facet	Create a facet plot for each object. Defaults to FALSE.
...	Currently not used

## Details

A binary image is first generated to segment the foreground and background. The argument `index` is useful to choose a proper index to segment the image (see [image\\_binary\(\)](#) for more details). Then, the number of objects in the foreground is counted. By setting up arguments such as `lower_size`, `upper_size` it is possible to set a threshold for lower and upper sizes of the objects, respectively. The argument `object_size` can be used to set up pre-defined values of tolerance and extension depending on the image resolution. This will influence the watershed-based object segmentation. Users can also tune-up tolerance and extension explicitly to a better precision of watershed segmentation.

If `watershed = FALSE` is used, all pixels for each connected set of foreground pixels in `img` are set to a unique object. This is faster (specially for a large number of objects) but is not able to segment touching objects.

If color palettes samples are provided, a general linear model (binomial family) fitted to the RGB values is used to segment fore- and background.

By using `pattern` it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in `dir_original`. To speed up the computation time, one can set `parallel = TRUE`.

## Value

`analyze_objects()` returns a list with the following objects:

- `results` A data frame with the following variables for each object in the image:

- id: object identification.
- x,y: x and y coordinates for the center of mass of the object.
- area: area of the object (in pixels).
- area\_ch: the area of the convex hull around object (in pixels).
- perimeter: perimeter (in pixels).
- radius\_min, radius\_mean, and radius\_max: The minimum, mean, and maximum radius (in pixels), respectively.
- radius\_sd: standard deviation of the mean radius (in pixels).
- radius\_ratio: radius ratio given by  $\text{radius\_max} / \text{radius\_min}$ .
- diam\_min, diam\_mean, and diam\_max: The minimum, mean, and maximum diameter (in pixels), respectively.
- major\_axis, minor\_axis: elliptical fit for major and minor axes (in pixels).
- eccentricity: elliptical eccentricity defined by  $\sqrt{1 - \text{minoraxis}^2 / \text{majoraxis}^2}$ . Circle eccentricity is 0 and straight line eccentricity is 1.
- theta: object angle (in radians).
- solidity: object solidity given by  $\text{area} / \text{area\_ch}$ .
- circularity: the object circularity given by  $4 * \pi * (\text{area} / \text{perimeter}^2)$ .
- statistics: A data frame with the summary statistics for the area of the objects.
- count: If pattern is used, shows the number of objects in each image.
- object\_rgb: If object\_index is used, returns the R, G, and B values for each pixel of each object.
- object\_index: If object\_index is used, returns the index computed for each object.

plot.anal\_obj() returns a trellis object containing the distribution of the pixels, optionally for each object when facet = TRUE is used.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

### References

- Gupta, S., Rosenthal, D. M., Stinchcombe, J. R., & Baucom, R. S. (2020). The remarkable morphological diversity of leaf shape in sweet potato (*Ipomoea batatas*): the influence of genetics, environment, and G×E. *New Phytologist*, 225(5), 2183–2195. doi: [10.1111/NPH.16286](https://doi.org/10.1111/NPH.16286)
- Lee, Y., & Lim, W. (2017). Shoelace Formula: Connecting the Area of a Polygon and the Vector Cross Product. *The Mathematics Teacher*, 110(8), 631–636. doi: [10.5951/mathteacher.110.8.0631](https://doi.org/10.5951/mathteacher.110.8.0631)

### Examples

```
library(pliman)
img <- image_pliman("soybean_touch.jpg")
obj <- analyze_objects(img)
obj$statistics
```

```

# Enumerate the objects in the original image
# Return the top-5 grains with the largest area

top <-
  analyze_objects(img,
                 marker = "id",
                 topn_upper = 5)
top$results

library(pliman)

img <- image_pliman("soy_green.jpg")
# Segment the foreground (grains) using the normalized blue index (NB)
# Shows the average value of the blue index in each object

rgb <-
  analyze_objects(img,
                 marker = "id",
                 index = "NB", # default
                 object_index = "B")

plot(rgb)
plot(rgb, facet = TRUE)

```

---

count\_lesions

*Counts the number of lesions*


---

## Description

Counts the number of lesions in a sample or entire leaf based on provided color palettes samples. A general linear model (binomial family) fitted to the RGB values is used to segment the lesions from the healthy leaf. If a pallet of background is provided, the function takes care of the details to isolate it before computing the number and area of lesions. By using `img_pattern` it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in `dir_original`.

## Usage

```

count_lesions(
  img,
  img_healthy = NULL,
  img_lesion = NULL,
  img_background = NULL,
  img_pattern = NULL,
  parallel = FALSE,
  workers = NULL,
  resize = FALSE,

```



```

fill_hull = TRUE,
invert = FALSE,
index = "NB",
my_index = NULL,
lower_size = NULL,
upper_size = NULL,
randomize = TRUE,
nrows = 3000,
lesion_size = "medium",
segment = TRUE,
tolerance = NULL,
extension = NULL,
show_segmentation = TRUE,
show_image = FALSE,
show_original = TRUE,
show_background = TRUE,
col_leaf = NULL,
col_lesions = NULL,
col_background = NULL,
marker = NULL,
marker_col = NULL,
marker_size = NULL,
save_image = FALSE,
prefix = "proc_",
dir_original = NULL,
dir_processed = NULL,
verbose = TRUE
)

```

### Arguments

<code>img</code>	The image to be analyzed.
<code>img_healthy</code>	A color palette of healthy areas.
<code>img_lesion</code>	A color palette of lesioned areas.
<code>img_background</code>	An optional color palette of the image background.
<code>img_pattern</code>	A pattern of file name used to identify images to be processed. For example, if <code>img_pattern = "im"</code> all images that the name matches the pattern (e.g., <code>img1.-</code> , <code>image1.-</code> , <code>im2.-</code> ) will be analyzed. Providing any number as pattern (e.g., <code>img_pattern = "1"</code> ) will select images that are named as <code>1.-</code> , <code>2.-</code> , and so on.
<code>parallel</code>	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when <code>img_pattern</code> is used is informed. The number of sections is set up to 70% of available cores.
<code>workers</code>	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
<code>resize</code>	Resize the image before processing? Defaults to <code>FALSE</code> . Use a numeric value of range 0-100 (proportion of the size of the original image).

fill_hull	Fill holes in the image? Defaults to TRUE. This is useful to fill holes in leaves, e.g., those caused by insect attack, ensuring the hole area will be accounted for the leaf, not background.
invert	Inverts the binary image, if desired. This is useful to process images with black background. Defaults to FALSE.
index, my_index	A character value specifying the target mode for conversion to binary image when <code>img_healthy</code> and <code>img_lesion</code> are not declared. Defaults to "NB" (normalized blue). See <code>image_index()</code> for more details.
lower_size	Lower limit for size for the image analysis. Leaf images often contain dirt and dust. To prevent dust from affecting the image analysis, the lower limit of analyzed size is set to 0.1, i.e., objects with lesser than 10% of the mean of all objects are removed. One can set a known area or use <code>lower_limit = 0</code> to select all objects (not advised).
upper_size	Upper limit for size for the image analysis. Defaults to NULL, i.e., no upper limit used.
randomize	Randomize the lines before training the model?
nrows	The number of lines to be used in training step.
lesion_size	The size of the lesion. Used to automatically set up tolerance and extension parameters. One of the following. "small" (2-5 mm in diameter, e.g. rust pustules), "medium" (0.5-1.0 cm in diameter, e.g. wheat leaf spot), "large" (1-2 cm in diameter, and "elarge" (2-3 cm in diameter, e.g. target spot of soybean).
segment	If TRUE (Default) implements the Watershed Algorithm to segment lesions connected by a fairly few pixels that could be considered as two distinct lesions. If FALSE, lesions that are connected by any pixel are considered unique lesions. For more details see <code>EImage::watershed()</code> .
tolerance	The minimum height of the object in the units of image intensity between its highest point (seed) and the point where it contacts another object (checked for every contact pixel). If the height is smaller than the tolerance, the object will be combined with one of its neighbors, which is the highest. Defaults to NULL, i.e., starting values are set up according to the argument <code>lesion_size</code> .
extension	Radius of the neighborhood in pixels for the detection of neighboring objects. Defaults to 20. Higher value smooths out small objects.
show_segmentation	Shows the object segmentation colored with random permutations. Defaults to TRUE.
show_image	Show image after processing?
show_original	Show the symptoms in the original image?
show_background	Show the background? Defaults to TRUE. A white background is shown by default when <code>show_original = FALSE</code> .
col_leaf	Leaf color after image processing. Defaults to "green"
col_lesions	Symptoms color after image processing. Defaults to "red".
col_background	Background color after image processing. Defaults to "NULL".

marker, marker_col, marker_size	The type, color and size of the object marker. Defaults to NULL, which shows a red point when show_segmentation = FALSE. To force a marker to be used with segmented objects, set up to marker = "point" (to show a point) or marker = "text" to enumerate the objects.
save_image	Save the image after processing? The image is saved in the current working directory named as proc_* where * is the image name given in img.
prefix	The prefix to be included in the processed images. Defaults to "proc_".
dir_original, dir_processed	The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image img in the current working directory. After processing, when save_image = TRUE, the processed image will be also saved in such a directory.
verbose	If TRUE (default) a summary is shown in the console.

### Value

A list with the following objects:

- results A data frame with the results (area, perimeter, radius) for object.
- statistics A data frame with the summary statistics for the image.
- count (If img\_pattern is used), summarizing the count number for each image.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

### Examples

```
library(pliman)
img <- image_pliman("sev_leaf_nb.jpg")
healthy <- image_pliman("sev_healthy.jpg")
lesions <- image_pliman("sev_symppt.jpg")
image_combine(img, healthy, lesions, ncol = 3)
a <-
count_lesions(img = img,
              img_healthy = healthy,
              img_lesion = lesions,
              lesion_size = "elarge", # extra large lesions
              show_image = TRUE,
              show_segmentation = FALSE,
              marker = "text")
```

---

`count_objects`*Computes number of objects in an image*

---

**Description**

Counts the number of objects in an image. See more at details.

**Usage**

```
count_objects(  
    img,  
    foreground = NULL,  
    background = NULL,  
    img_pattern = NULL,  
    parallel = FALSE,  
    workers = NULL,  
    resize = FALSE,  
    fill_hull = FALSE,  
    filter = FALSE,  
    invert = FALSE,  
    index = "NB",  
    my_index = NULL,  
    object_size = "medium",  
    tolerance = NULL,  
    extension = NULL,  
    lower_size = NULL,  
    upper_size = NULL,  
    topn_lower = NULL,  
    topn_upper = NULL,  
    randomize = TRUE,  
    nrows = 10000,  
    show_image = TRUE,  
    show_original = TRUE,  
    show_background = TRUE,  
    show_segmentation = TRUE,  
    col_foreground = NULL,  
    col_background = NULL,  
    marker = NULL,  
    marker_col = NULL,  
    marker_size = NULL,  
    save_image = FALSE,  
    prefix = "proc_",  
    dir_original = NULL,  
    dir_processed = NULL,  
    verbose = TRUE  
)
```

**Arguments**

img	The image to be analyzed.
foreground	A color palette of the foreground (optional).
background	A color palette of the background (optional).
img_pattern	A pattern of file name used to identify images to be processed. For example, if <code>img_pattern = "im"</code> all images that the name matches the pattern (e.g., <code>img1.-</code> , <code>image1.-</code> , <code>im2.-</code> ) will be analyzed. Providing any number as pattern (e.g., <code>img_pattern = "1"</code> ) will select images that are named as <code>1.-</code> , <code>2.-</code> , and so on.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when <code>img_pattern</code> is used is informed. The number of sections is set up to 70% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
resize	Resize the image before processing? Defaults to FALSE. Use a numeric value of range 0-100 (proportion of the size of the original image).
fill_hull	Fill holes in the binary image? Defaults to FALSE. This is useful to fill holes in objects that have portions with a color similar to the background. <b>IMPORTANT:</b> Objects touching each other can be combined into one single object, which may underestimate the number of objects in an image.
filter	Performs median filtering after image processing? defaults to FALSE. See more at <a href="#">image_filter()</a> .
invert	Inverts the binary image, if desired. This is useful to process images with black background. Defaults to FALSE.
index, my_index	A character value specifying the target mode for conversion to binary image when foreground and background are not declared. Defaults to "NB" (normalized blue). See <a href="#">image_index()</a> for more details.
object_size	The size of the object. Used to automatically set up tolerance and extension parameters. One of the following. "small" (e.g, wheat grains), "medium" (e.g, soybean grains), "large"(e.g, peanut grains), and "elarge" (e.g, soybean pods)‘.
tolerance	The minimum height of the object in the units of image intensity between its highest point (seed) and the point where it contacts another object (checked for every contact pixel). If the height is smaller than the tolerance, the object will be combined with one of its neighbors, which is the highest.
extension	Radius of the neighborhood in pixels for the detection of neighboring objects. Defaults to 20. Higher value smooths out small objects.
lower_size, upper_size	Lower and upper limits for size for the image analysis. Plant images often contain dirt and dust. To prevent dust from affecting the image analysis, objects with lesser than 10% of the mean of all objects are removed. Upper limit is set to NULL, i.e., no upper limit used. One can set a known area or use <code>lower_limit = 0</code> to select all objects (not advised). Objects that matches the size of a given

range of sizes can be selected by setting up the two arguments. For example, if `lower_size = 120` and `upper_size = 140`, objects with size greater than or equal 120 and less than or equal 140 will be considered.

<code>topn_lower, topn_upper</code>	Select the top n objects based on its area. <code>topn_lower</code> selects the n elements with the smallest area whereas <code>topn_upper</code> selects the n objects with the largest area.
<code>randomize</code>	Randomize the lines before training the model?
<code>nrows</code>	The number of lines to be used in training step.
<code>show_image</code>	Show image after processing?
<code>show_original</code>	Show the count objects in the original image?
<code>show_background</code>	Show the background? Defaults to TRUE. A white background is shown by default when <code>show_original = FALSE</code> .
<code>show_segmentation</code>	Shows the object segmentation colored with random permutations. Defaults to TRUE.
<code>col_foreground, col_background</code>	Foreground and background color after image processing. Defaults to NULL, in which "black", and "white" are used, respectively.
<code>marker, marker_col, marker_size</code>	The type, color and size of the object marker. Defaults to NULL, which shows a red point when <code>show_segmentation = FALSE</code> . To force a marker to be used with segmented objects, set up to <code>marker = "point"</code> (to show a point) or <code>marker = "text"</code> to enumerate the objects.
<code>save_image</code>	Save the image after processing? The image is saved in the current working directory named as <code>proc_*</code> where * is the image name given in <code>img</code> .
<code>prefix</code>	The prefix to be included in the processed images. Defaults to "proc_".
<code>dir_original, dir_processed</code>	The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image <code>img</code> in the current working directory. After processing, when <code>save_image = TRUE</code> , the processed image will be also saved in such a directory.
<code>verbose</code>	If TRUE (default) a summary is shown in the console.

## Details

Counts the number of objects in an image. A binary image is first generated to segment the foreground and background. The argument `index` is useful to choose a proper index to segment the image (see `image_binary()` for more details). Then, the number of objects in the foreground is counted. By setting up arguments such as `lower_size`, `upper_size` is possible to set a threshold for lower and upper sizes of the objects, respectively. Change tolerance and extension values to better set up watershed-based object detection. If color palettes samples are provided, a general linear model (binomial family) fitted to the RGB values is used to segment fore- and background.

By using `img_pattern` it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in `dir_original`. To speed up the computation time, one can set `parallel = TRUE`.

**Value**

A list with the following objects:

- `results` A data frame with the results (area, perimeter, radius) for object.
- `statistics` A data frame with the summary statistics for the image.
- `count` (If `img_pattern` is used), summarizing the count number for each image.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**Examples**

```
library(pliman)
##### deprecated use analyze_objects() instead #####
img <- image_pliman("soybean_touch.jpg")
analyze_objects(img)
```

---

image\_binary

*Creates a binary image*

---

**Description**

Reduce a color, color near-infrared, or grayscale images to a binary image using a given color channel (red, green blue) or even color indexes. The Otsu's thresholding method (Otsu, 1979) is used to automatically perform clustering-based image thresholding.

**Usage**

```
image_binary(  
  image,  
  index = NULL,  
  my_index = NULL,  
  threshold = "Otsu",  
  resize = 30,  
  fill_hull = FALSE,  
  re = NULL,  
  nir = NULL,  
  invert = FALSE,  
  show_image = TRUE,  
  nrow = NULL,  
  ncol = NULL,  
  parallel = FALSE,
```

```

workers = NULL,
verbose = TRUE
)

```

### Arguments

image	An image object.
index	A character value (or a vector of characters) specifying the target mode for conversion to binary image. See the available indexes with <code>pliman_indexes()</code> and <code>image_index()</code> for more details.
my_index	User can calculate a different index using the band names, e.g. <code>my_index = "R+B/G"</code> .
threshold	By default ( <code>threshold = "Otsu"</code> ), a threshold value based on Otsu's method is used to reduce the grayscale image to a binary image. If a numeric value is informed, this value will be used as a threshold. Inform any non-numeric value different than "Otsu" to iteratively chosen the threshold based on a raster plot showing pixel intensity of the index.
resize	Resize the image before processing? Defaults to FALSE. Use a numeric value as the percentage of desired resizing. For example, if <code>resize = 30</code> , the resized image will have 30% of the size of original image.
fill_hull	Fill holes in the objects? Defaults to FALSE.
re	Respective position of the red-edge band at the original image file.
nir	Respective position of the near-infrared band at the original image file.
invert	Inverts the binary image, if desired.
show_image	Show image after processing?
nrow, ncol	The number of rows or columns in the plot grid. Defaults to NULL, i.e., a square grid is produced.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 70% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.

### Value

A list containing binary images. The length will depend on the number of indexes used.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

### References

Nobuyuki Otsu, "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9 (1): 62-66. 1979. doi: [10.1109/TSMC.1979.4310076](https://doi.org/10.1109/TSMC.1979.4310076)



## Examples

```
library(pliman)
img <- image_pliman("soybean_touch.jpg")
image_binary(img, index = c("R", "G"))
```

---

image_combine	<i>Combines images to a grid</i>
---------------	----------------------------------

---

## Description

Combines several images to a grid

## Usage

```
image_combine(
  ...,
  labels = NULL,
  nrow = NULL,
  ncol = NULL,
  col = "black",
  verbose = TRUE
)
```

## Arguments

...	a comma-separated name of image objects or a list containing image objects.
labels	A character vector with the same length of the number of objects in ... to indicate the plot labels.
nrow, ncol	The number of rows or columns in the plot grid. Defaults to NULL, i.e., a square grid is produced.
col	The color for the plot labels. Defaults to col = "black".
verbose	Shows the name of objects declared in ... or a numeric sequence if a list with no names is provided. Set to FALSE to suppress the text.

## Value

A grid with the images in ...

## Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

## Examples

```
library(pliman)
img1 <- image_pliman("sev_leaf.jpg")
img2 <- image_pliman("sev_leaf_nb.jpg")
image_combine(img1, img2)
```

---

 image\_index

*Image indexes*


---

### Description

image\_index() Builds image indexes using Red, Green, Blue, Red-Edge, and NIR bands.  
 plot.image\_index() produces a raster (type = "raster", default) or a density (type = "density") plot of the index values computed with image\_index().

### Usage

```
image_index(
  image,
  index = NULL,
  my_index = NULL,
  resize = FALSE,
  re = NULL,
  nir = NULL,
  show_image = TRUE,
  nrow = NULL,
  ncol = NULL,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)

## S3 method for class 'image_index'
plot(x, type = "raster", nrow = NULL, ncol = NULL, ...)
```

### Arguments

image	An image object.
index	A character value (or a vector of characters) specifying the target mode for conversion to binary image. Use <a href="#">pliman_indexes()</a> or the details section to see the available indexes. Defaults to NULL ((normalized) Red, Green and Blue). One can also use "RGB" for RGB only, "NRGB" for normalized RGB, or "all" for all indexes.
my_index	User can calculate a different index using the bands names, e.g. my_index = "R+B/G".
resize	Resize the image before processing? Defaults to 30, which resizes the image to 30% of the original size to speed up image processing. Set resize = FALSE to keep the original size of the image.
re	Respective position of the red-edge band at the original image file.
nir	Respective position of the near-infrared band at the original image file.
show_image	Show image after processing?

nrow, ncol	The number of rows or columns in the plot grid. Defaults to NULL, i.e., a square grid is produced.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 70% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.
x	An object of class image_index.
type	The type of plot. Use type = "raster" (default) to produce a raster plot showing the intensity of the pixels for each image index or type = "density" to produce a density plot with the pixels' intensity.
...	Currently not used

## Details

The following indexes are available in pliman.

- R red
- G green
- B blue
- NR normalized red  $R/(R+G+B)$ .
- NG normalized green  $G/(R+G+B)$
- NB normalized blue  $B/(R+G+B)$
- GB green blue ratio  $G/B$
- RB red blue ratio  $R/B$
- GR green red ratio  $G/R$
- BI brightness Index  $\sqrt{(R^2+G^2+B^2)/3}$
- BIM brightness Index 2  $\sqrt{(R*2+G*2+B*2)/3}$
- SCI Soil Colour Index  $(R-G)/(R+G)$
- GLI Green leaf index Vis Louhaichi et al. (2001)  $(2*G-R-B)/(2*G+R+B)$
- HI Primary colours Hue Index  $(2*R-G-B)/(G-B)$
- NDGRI Normalized green red difference index (Tucker, 1979)  $(G-R)/(G+R)$
- NDGBI Normalized green blue difference index  $(G-B)/(G+B)$
- NDRBI Normalized red blue difference index  $(R-B)/(R+B)$
- I  $R+G+B$
- S  $((R+G+B)-3*B)/(R+G+B)$
- L  $R+G+B/3$
- VARI A Visible Atmospherically Resistant Index  $(G-R)/(G+R-B)$
- HUE Overall Hue Index  $\text{atan}(2*(B-G-R)/30.5*(G-R))$

- HUE2  $\text{atan}(2*(R-G-R)/30.5*(G-B))$
- BGI  $B/G$
- GRAY  $0.299*R + 0.587*G + 0.114*B$
- GLAI  $(25*(G-R)/(G+R-B)+1.25)$
- CI Coloration Index  $(R-B)/R$
- SAT Overhall Saturation Index  $(\max(R,G,B) - \min(R,G,B)) / \max(R,G,B)$
- SHP Shape Index  $2*(R-G-B)/(G-B)$
- RI Redness Index  $R**2/(B*G**3)$

### Value

A list containing Grayscale images. The length will depend on the number of indexes used.

A trellis object containing the distribution of the pixels for each index.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Tiago Olivoto <tiagoolivoto@gmail.com>

### References

Nobuyuki Otsu, "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9 (1): 62-66. 1979. doi: [10.1109/TSMC.1979.4310076](https://doi.org/10.1109/TSMC.1979.4310076)

### Examples

```
library(pliman)
img <- image_pliman("soybean_touch.jpg")
image_index(img, index = c("R, NR"))
library(pliman)
img <- image_pliman("sev_leaf.jpg")

# resize the image to 30% of the original size
ind <- image_index(img, resize = 30, show_image = FALSE)
plot(ind)
```

---

image\_palette

*Create an image palette*

---

### Description

Creates image palettes by applying the k-means algorithm to the RGB values.

**Usage**

```
image_palette(  
  image,  
  npal,  
  filter = TRUE,  
  blur = FALSE,  
  parallel = FALSE,  
  workers = NULL,  
  verbose = TRUE  
)
```

**Arguments**

image	An image object.
npal	The number of color palettes.
filter	Performs median filtering. This can be useful to reduce the noise in produced palettes. Defaults to TRUE. See more at <a href="#">image_filter()</a> .
blur	Performs blurring filter of palettes? Defaults to FALSE. See more at <a href="#">image_blur()</a> .
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 70% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.

**Value**

A list with npal color palettes of class Image.

**Examples**

```
library(pliman)  
img <- image_pliman("sev_leaf_nb.jpg")  
pal <- image_palette(img, npal = 4)  
  
image_combine(pal)
```

**Description**

- `image_segment()` reduces a color, color near-infrared, or grayscale images to a segmented image using a given color channel (red, green blue) or even color indexes (See [image\\_index\(\)](#) for more details). The Otsu's thresholding method (Otsu, 1979) is used to automatically perform clustering-based image thresholding.
- `image_segment_iter()` Provides an iterative image segmentation, returning the proportions of segmented pixels.

**Usage**

```
image_segment(
  image,
  index = NULL,
  my_index = NULL,
  threshold = "Otsu",
  fill_hull = FALSE,
  re = NULL,
  nir = NULL,
  invert = FALSE,
  show_image = TRUE,
  nrow = NULL,
  ncol = NULL,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE
)
```

```
image_segment_iter(
  image,
  nseg = 1,
  index = NULL,
  invert = NULL,
  threshold = NULL,
  show_image = TRUE,
  verbose = TRUE,
  nrow = NULL,
  ncol = NULL,
  parallel = FALSE,
  workers = NULL,
  ...
)
```

**Arguments**

- |                    |   |
|--------------------|---|
| <code>image</code> | An image object or a list of image objects.   |
| <code>index</code> | <ul style="list-style-type: none"> <li>• For <code>image_segment()</code>, a character value (or a vector of characters) specifying the target mode for conversion to binary image. See the available indexes with <a href="#">pliman_indexes()</a>. See <a href="#">image_index()</a> for more details.</li> </ul> |

	<ul style="list-style-type: none"> <li>For <code>image_segment_iter()</code> a character or a vector of characters with the same length of <code>nseg</code>. It can be either an available index (described above) or any operation involving the RGB values (e.g., "B/R+G").</li> </ul>
<code>my_index</code>	User can calculate a different index using the bands names, e.g. <code>my_index = "R+B/G"</code> .
<code>threshold</code>	By default ( <code>threshold = "Otsu"</code> ), a threshold value based on Otsu's method is used to reduce the grayscale image to a binary image. If a numeric value is informed, this value will be used as a threshold. Inform any non-numeric value different than "Otsu" to iteratively chosen the threshold based on a raster plot showing pixel intensity of the index. For <code>image_segmentation_iter()</code> , use a vector (allows a mixed (numeric and character) type) with the same length of <code>nseg</code> .
<code>fill_hull</code>	Fill holes in the objects? Defaults to FALSE.
<code>re</code>	Respective position of the red-edge band at the original image file.
<code>nir</code>	Respective position of the near-infrared band at the original image file.
<code>invert</code>	Inverts the binary image, if desired. For <code>image_segmentation_iter()</code> use a vector with the same length of <code>nseg</code> .
<code>show_image</code>	Show image after processing?
<code>nrow, ncol</code>	The number of rows or columns in the plot grid. Defaults to NULL, i.e., a square grid is produced.
<code>parallel</code>	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when <code>image</code> is a list. The number of sections is set up to 70% of available cores.
<code>workers</code>	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
<code>verbose</code>	If TRUE (default) a summary is shown in the console.
<code>nseg</code>	The number of iterative segmentation steps to be performed.
<code>...</code>	Additional arguments passed on to <code>image_segment()</code> .

### Value

- `image_segment()` returns list containing `n` objects where `n` is the number of indexes used. Each objects contains:
  - `image` an image with the RGB bands (layers) for the segmented object.
  - `mask` A mask with logical values of 0 and 1 for the segmented image.
- `image_segment_iter()` returns a list with (1) a data frame with the proportion of pixels in the segmented images and (2) the segmented images.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

### References

Nobuyuki Otsu, "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9 (1): 62-66. 1979. doi: [10.1109/TSMC.1979.4310076](https://doi.org/10.1109/TSMC.1979.4310076)

## Examples

```
library(pliman)
img <- image_pliman("soybean_touch.jpg", plot = TRUE)
image_segment(img, index = c("R, G, B"))
```

---

image\_to\_mat

*Convert an image to numerical matrices*

---

## Description

Given an object image, converts it into three matrices (RGB) and a data frame where each column corresponds to the RGB values.

## Usage

```
image_to_mat(image, parallel = FALSE, workers = NULL, verbose = TRUE)
```

## Arguments

image	An image object.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 70% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.

## Value

A list containing three matrices (R, G, and B), and a data frame containing four columns: the name of the image in image and the R, G, B values.

## Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

## Examples

```
library(pliman)
img <- image_pliman("sev_leaf.jpg")
dim(img)
mat <- image_to_mat(img)
dim(mat[[1]])
```



---

leaf_area	<i>Calculates the leaf area</i>
-----------	---------------------------------

---

### Description

- leaf\_area () is deprecated. use [analyze\\_objects\(\)](#) instead.

### Usage

```
leaf_area(  
  img,  
  img_leaf,  
  img_background,  
  img_template,  
  area_template,  
  resize = FALSE,  
  parallel = FALSE,  
  workers = NULL,  
  img_pattern = NULL,  
  lower_size = NULL,  
  upper_size = NULL,  
  randomize = TRUE,  
  nrows = 10000,  
  show_image = TRUE,  
  show_original = TRUE,  
  show_background = TRUE,  
  col_background = NULL,  
  col_leaf = "green",  
  text_col = "black",  
  text_size = 1,  
  text_digits = 2,  
  save_image = FALSE,  
  dir_original = NULL,  
  dir_processed = NULL,  
  verbose = TRUE  
)
```

### Arguments

img	The image to be analyzed.
img_leaf	A color palette of the leaves.
img_background	A color palette of background area.
img_template	A color palette of the template areas.
area_template	The known area of the template. The leaf area will be given in the same unit as area_template.

resize	Resize the image before processing? Defaults to FALSE. Use a numeric value of range 0-100 (proportion of the size of the original image).
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when <code>img_pattern</code> is used is informed. The number of sections is set up to 70% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
img_pattern	A pattern of file name used to identify images to be processed. For example, if <code>img_pattern = "im"</code> all images that the name matches the pattern (e.g., <code>img1.-</code> , <code>image1.-</code> , <code>im2.-</code> ) will be analyzed. Providing any number as pattern (e.g., <code>img_pattern = "1"</code> ) will select images that are named as <code>1.-</code> , <code>2.-</code> , and so on.
lower_size	Lower limit for size for the image analysis. Leaf images often contain dirt and dust. To prevent dust from affecting the image analysis, the lower limit of analyzed size is set to 0.1, i.e., objects with lesser than 10% of the mean of all objects are removed. One can set a known area or use <code>lower_limit = 0</code> to select all objects (not advised).
upper_size	Upper limit for size for the image analysis. Defaults to NULL, i.e., no upper limit used.
randomize	Randomize the lines before training the model?
nrows	The number of lines to be used in training step.
show_image	Show image after processing?
show_original	Show the symptoms in the original image?
show_background	Show the background? Defaults to TRUE. A white background is shown by default when <code>show_original = FALSE</code> .
col_background	Background color after image processing.
col_leaf	Leaf color after image processing when <code>show_original = FALSE</code> . Defaults to "green".
text_col, text_size, text_digits	The color, size and significant digits used in the text. The shows the pattern <code>o a</code> , where <code>o</code> and <code>a</code> are the object id and its area, respectively.
save_image	Save the image after processing? The image is saved in the current working directory named as <code>proc_*</code> where <code>*</code> is the image name given in <code>img</code> .
dir_original, dir_processed	The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image <code>img</code> in the current working directory. After processing, when <code>save_image = TRUE</code> , the processed image will be also saved in such a directory.
verbose	If TRUE (default) a summary is shown in the console.

**Value**

A data frame with the results for each image.

## Examples

```
# use analyze_objects() to compute the leaf area.

library(pliman)
img <- image_pliman("la_leaves.jpg")
lef_area <- analyze_objects(img, marker = "id")

# correct pixel to metric units using the image dpi (~84)
# the object 6 (leaf square) has a known area of ~ 4 cm2.
get_measures(lef_area, dpi = 84)
```

---

measure_disease	<i>Performs plant disease measurements</i>
-----------------	--

---

## Description

Computes the percentage of symptomatic leaf area and (optionally) counts and compute shapes (area, perimeter, radius, etc.) of lesions in a sample or entire leaf. See more at **Details**.

## Usage

```
measure_disease(  
  img,  
  img_healthy = NULL,  
  img_symptoms = NULL,  
  img_background = NULL,  
  pattern = NULL,  
  parallel = FALSE,  
  workers = NULL,  
  resize = FALSE,  
  fill_hull = TRUE,  
  index_lb = NULL,  
  index_dh = "GLI",  
  threshold = NULL,  
  invert = FALSE,  
  lower_size = NULL,  
  upper_size = NULL,  
  topn_lower = NULL,  
  topn_upper = NULL,  
  randomize = TRUE,  
  nrows = "deprecated",  
  nsample = 3000,  
  watershed = FALSE,  
  lesion_size = "medium",
```

```

tolerance = NULL,
extension = NULL,
show_features = FALSE,
show_segmentation = FALSE,
show_image = TRUE,
show_original = TRUE,
show_background = TRUE,
show_contour = TRUE,
contour_col = "white",
contour_size = 1,
col_leaf = NULL,
col_lesions = NULL,
col_background = NULL,
marker = FALSE,
marker_col = NULL,
marker_size = NULL,
save_image = FALSE,
prefix = "proc_",
dir_original = NULL,
dir_processed = NULL,
verbose = TRUE
)

```

### Arguments

<code>img</code>	The image to be analyzed.
<code>img_healthy</code>	A color palette of healthy areas.
<code>img_symptoms</code>	A color palette of lesioned areas.
<code>img_background</code>	An optional color palette of the image background.
<code>pattern</code>	A pattern of file name used to identify images to be processed. For example, if <code>pattern = "im"</code> all images that the name matches the pattern (e.g., <code>img1.-</code> , <code>image1.-</code> , <code>im2.-</code> ) will be analyzed. Providing any number as pattern (e.g., <code>pattern = "1"</code> ) will select images that are named as <code>1.-</code> , <code>2.-</code> , and so on.
<code>parallel</code>	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when <code>pattern</code> is used is informed. The number of sections is set up to 70% of available cores.
<code>workers</code>	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
<code>resize</code>	Resize the image before processing? Defaults to <code>FALSE</code> . Use a numeric value of range 0-100 (proportion of the size of the original image).
<code>fill_hull</code>	Fill holes in the image? Defaults to <code>TRUE</code> . This is useful to fill holes in leaves, e.g., those caused by insect attack, ensuring the hole area will be accounted for the leaf, not background.
<code>index_lb</code>	The index used to segment the foreground (e.g., leaf) from the background. If not declared, the entire image area (pixels) will be considered in the computation of the severity.

index_dh	The index used to segment diseased from healthy tissues when <code>img_healthy</code> and <code>img_symptoms</code> are not declared. Defaults to "GLI". See <a href="#">image_index()</a> for more details.
threshold	By default ( <code>threshold = NULL</code> ), a threshold value based on Otsu's method is used to reduce the grayscale image to a binary image. If a numeric value is informed, this value will be used as a threshold. Inform any non-numeric value different than "Otsu" to iteratively choose the threshold based on a raster plot showing pixel intensity of the index. Must be a vector of length 2 to indicate the threshold for <code>index_lb</code> and <code>index_dh</code> , respectively.
invert	Inverts the binary image if desired. This is useful to process images with black background. Defaults to <code>FALSE</code> .
lower_size	Lower limit for size for the image analysis. Leaf images often contain dirt and dust. To prevent dust from affecting the image analysis, the lower limit of analyzed size is set to 0.1, i.e., objects with lesser than 10% of the mean of all objects are removed. One can set a known area or use <code>lower_limit = 0</code> to select all objects (not advised).
upper_size	Upper limit for size for the image analysis. Defaults to <code>NULL</code> , i.e., no upper limit used.
topn_lower, topn_upper	Select the top n lesions based on its area. <code>topn_lower</code> selects the n lesions with the smallest area whereas <code>topn_upper</code> selects the n lesions with the largest area.
randomize	Randomize the lines before training the model? Defaults to <code>TRUE</code> .
nrows	Deprecated. Use <code>nsamples</code> instead.
nsample	The number of sample pixels to be used in training step. Defaults to 3000.
watershed	If <code>TRUE</code> (Default) implements the Watershed Algorithm to segment lesions connected by a fairly few pixels that could be considered as two distinct lesions. If <code>FALSE</code> , lesions that are connected by any pixel are considered unique lesions. For more details see <a href="#">EBImage::watershed()</a> .
lesion_size	The size of the lesion. Used to automatically tune tolerance and extension parameters. One of the following. "small" (2-5 mm in diameter, e.g. rust pustules), "medium" (0.5-1.0 cm in diameter, e.g. wheat leaf spot), "large" (1-2 cm in diameter, and "elarge" (2-3 cm in diameter, e.g. target spot of soybean).
tolerance	The minimum height of the object in the units of image intensity between its highest point (seed) and the point where it contacts another object (checked for every contact pixel). If the height is smaller than the tolerance, the object will be combined with one of its neighbors, which is the highest. Defaults to <code>NULL</code> , i.e., starting values are set up according to the argument <code>lesion_size</code> .
extension	Radius of the neighborhood in pixels for the detection of neighboring objects. Defaults to 20. Higher value smooths out small objects.
show_features	If <code>TRUE</code> returns the lesion features such as number, area, perimeter, and radius. Defaults to <code>FALSE</code> .
show_segmentation	Shows the object segmentation colored with random permutations. Defaults to <code>TRUE</code> .
show_image	Show image after processing? Defaults to <code>TRUE</code> .

show_original	Show the symptoms in the original image?
show_background	Show the background? Defaults to TRUE. A white background is shown by default when show_original = FALSE.
show_contour	Show a contour line around the lesions? Defaults to TRUE.
contour_col, contour_size	The color and size for the contour line around objects. Defaults to contour_col = "white" and contour_size = 1.
col_leaf	Leaf color after image processing. Defaults to "green"
col_lesions	Symptoms color after image processing. Defaults to "red".
col_background	Background color after image processing. Defaults to "NULL".
marker, marker_col, marker_size	The type, color and size of the object marker. Defaults to NULL, which shows nothing. Use marker = "point" to show a point in each lesion or marker = "*" where "*" is any variable name of the shape data frame returned by the function.
save_image	Save the image after processing? The image is saved in the current working directory named as proc_* where * is the image name given in img.
prefix	The prefix to be included in the processed images. Defaults to "proc_".
dir_original, dir_processed	The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image img in the current working directory. After processing, when save_image = TRUE, the processed image will be also saved in such a directory. It can be either a full path, e.g., "C:/Desktop/imgs", or a subfolder within the current working directory, e.g., "/imgs".
verbose	If TRUE (default) a summary is shown in the console.

## Details

A general linear model (binomial family) fitted to the RGB values is used to segment the lesions from the healthy leaf. If a pallet of background is provided, the function takes care of the details to isolate it before computing the number and area of lesions. By using `pattern` it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in `dir_original`.

If `img_healthy` and `img_symptoms` are not declared, RGB-based phenotyping of foliar disease severity is performed using the index informed in `index_lb` to first segment leaf from background and `index_dh` to segment diseased from healthy tissues.

## Value

A list with the following objects:

- `severity` A data frame with the percentage of healthy and symptomatic areas.
- `shape_statistics` If `show_features = TRUE` is used, returns the shape (area, perimeter, etc.) for each lesion and a summary statistic of the results.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**Examples**

```
library(pliman)
img <- image_pliman("sev_leaf_nb.jpg")
healthy <- image_pliman("sev_healthy.jpg")
lesions <- image_pliman("sev_symp.jpg")
image_combine(img, healthy, lesions, ncol = 3)

sev <-
  measure_disease(img = img,
                 img_healthy = healthy,
                 img_symptoms = lesions,
                 lesion_size = "large",
                 show_image = TRUE)
```

---

objects\_rgb

*Get Red Green and Blue for image objects*

---

**Description**

Get the Red Green and Blue (RGB) for objects in an image. This function will be deprecated in a near future. Use [analyze\\_objects\(\)](#) with the argument `object_index` instead.

**Usage**

```
objects_rgb(
  img,
  foreground = NULL,
  background = NULL,
  img_pattern = NULL,
  parallel = FALSE,
  workers = NULL,
  resize = FALSE,
  fill_hull = FALSE,
  invert = FALSE,
  index = "NB",
  my_index = NULL,
  object_index = "B",
  object_size = "large",
  tolerance = NULL,
  extension = NULL,
  lower_size = NULL,
```

```

upper_size = NULL,
topn_lower = NULL,
topn_upper = NULL,
nrows = 10000,
show_image = TRUE,
save_image = FALSE,
prefix = "proc_",
marker = NULL,
marker_col = NULL,
marker_size = NULL,
marker_digits = NULL,
dir_original = NULL,
dir_processed = NULL,
verbose = TRUE
)

```

### Arguments

<code>img</code>	The image to be analyzed.
<code>foreground</code>	A color palette of the foreground (optional).
<code>background</code>	A color palette of the background (optional).
<code>img_pattern</code>	A pattern of file name used to identify images to be processed. For example, if <code>img_pattern = "im"</code> all images that the name matches the pattern (e.g., <code>img1.-</code> , <code>image1.-</code> , <code>im2.-</code> ) will be analyzed. Providing any number as pattern (e.g., <code>img_pattern = "1"</code> ) will select images that are named as <code>1.-</code> , <code>2.-</code> , and so on.
<code>parallel</code>	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when <code>img_pattern</code> is used is informed. The number of sections is set up to 70% of available cores.
<code>workers</code>	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
<code>resize</code>	Resize the image before processing? Defaults to FALSE. Use a numeric value of range 0-100 (proportion of the size of the original image).
<code>fill_hull</code>	Fill holes in the binary image? Defaults to FALSE. This is useful to fill holes in objects that have portions with a color similar to the background. <b>IMPORTANT:</b> Objects touching each other can be combined into one single object, which may underestimate the number of objects in an image.
<code>invert</code>	Inverts the binary image, if desired. This is useful to process images with black background. Defaults to FALSE.
<code>index, my_index</code>	A character value specifying the target mode for conversion to binary image when <code>foreground</code> and <code>background</code> are not declared. Defaults to "NB" (normalized blue). See <code>image_index()</code> for more details.
<code>object_index</code>	The same as <code>index</code> , used to compute the index for each object in the image.



object_size	The size of the object. Used to automatically set up tolerance and extension parameters. One of the following. "small" (e.g, wheat grains), "medium" (e.g, soybean grains), "large"(e.g, peanut grains), and "elarge" (e.g, soybean pods)‘.
tolerance	The minimum height of the object in the units of image intensity between its highest point (seed) and the point where it contacts another object (checked for every contact pixel). If the height is smaller than the tolerance, the object will be combined with one of its neighbors, which is the highest.
extension	Radius of the neighborhood in pixels for the detection of neighboring objects. Defaults to 20. Higher value smooths out small objects.
lower_size, upper_size	Lower and upper limits for size for the image analysis. Plant images often contain dirt and dust. To prevent dust from affecting the image analysis, objects with lesser than 10% of the mean of all objects are removed. Upper limit is set to NULL, i.e., no upper limit used. One can set a known area or use lower_limit = 0 to select all objects (not advised). Objects that matches the size of a given range of sizes can be selected by setting up the two arguments. For example, if lower_size = 120 and upper_size = 140, objects with size greater than or equal 120 and less than or equal 140 will be considered.
topn_lower, topn_upper	Select the top n objects based on its area. topn_lower selects the n elements with the smallest area whereas topn_upper selects the n objects with the largest area.
nrows	The number of lines to be used in training step.
show_image	Show image after processing? Defaults to TRUE.
save_image	Save the image after processing? The image is saved in the current working directory named as proc_* where * is the image name given in img.
prefix	The prefix to be included in the processed images. Defaults to "proc_".
marker, marker_col, marker_size, marker_digits	The marker, color, size and significant digits of the object marker. Defaults to marker = "index", which shows the object index. Set to marker = "id" to show the object id.
dir_original, dir_processed	The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image img in the current working directory. After processing, when save_image = TRUE, the processed image will be also saved in such a directory.
verbose	If FALSE, runs the code silently.

## Details

A binary image is first generated to segment the foreground and background. The argument index is useful to choose a proper index to segment the image (see [image\\_binary\(\)](#) for more details). Then, the number of objects in the foreground is counted. Change tolerance and extension values to better set up watershed-based object detection. If color palettes samples are provided, a general linear model (binomial family) fitted to the RGB values is used to segment fore- and background.

For each segmented object, the RGB values are obtained with. Users can also compute an index for each object using the argument `object_index`, useful to classify objects based on its RGB values.

By using `img_pattern` it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in `dir_original`'. To speed up the computation time, one can set `parallel = TRUE`.

### Value

A list with the following objects.

- `objects` A data frame with the measures for each object.
- `rgb` A data frame with the Red, Green and Blue values for each object
- `indexes` A data frame with the index computed according to the argument `object_index`.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

### Examples

```
# objects_rgb() will be deprecated in the future.
# Use analyze_objects() with the argument `object_index` instead.
```

---

pliman\_images

*Sample images*

---

### Description

Sample images installed with the **pliman** package

### Format

\*.jpg format

- `la_back.jpg` A cyan palette representing the background of images `la_pattern`, `la_leaves`, and `soybean_touch`.
- `la_leaf.jpg` A sample of the leaves in `la_leaves`
- `la_leaves.jpg` Tree leaves with a sample of known area.
- `objects_300dpi.jpg` An image with 300 dpi resolution.
- `potato_leaves.jpg` Three potato leaves, which were gathered from Gupta et al. (2020).
- `sev_leaf.jpg` A soybean leaf with a blue background.
- `sev_leaf_nb.jpg` A soybean leaf without background.
- `sev_back.jpg` A blue palette representing the background of `sev_leaf`.
- `sev_healthy.jpg` Healthy area of `sev_leaf`.

- sev\_sympt.jpg The symptomatic area sev\_leaf.
- soy\_green.jpg Soybean grains with a white background.
- soybean\_grain.jpg A sample palette of the grains in soy\_green.
- soybean\_touch.jpg Soybean grains with a cyan background touching one each other.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**Source**

Personal data, Gupta et al. (2020).

**References**

Gupta, S., Rosenthal, D. M., Stinchcombe, J. R., & Baucom, R. S. (2020). The remarkable morphological diversity of leaf shape in sweet potato (*Ipomoea batatas*): the influence of genetics, environment, and G×E. *New Phytologist*, 225(5), 2183–2195. doi: [10.1111/NPH.16286](https://doi.org/10.1111/NPH.16286)

---

prop\_segmented

*Image segmentation with pixels proportion*

---

**Description**

Provides (iterative) image segmentation and get the proportion of pixels. This can be used as an alternative way to compute disease severity without using color palettes.

**Usage**

```
prop_segmented(  
  image,  
  nseg = 1,  
  index = NULL,  
  threshold = "Otsu",  
  fill_hull = FALSE,  
  filter = FALSE,  
  parallel = FALSE,  
  workers = NULL,  
  show_image = TRUE,  
  ncol = NULL,  
  nrow = NULL,  
  verbose = TRUE  
)
```

**Arguments**

<code>image</code>	An object (or a list of objects) of class <code>Image</code> .
<code>nseg</code>	The number of iterative segmentation steps to be performed.
<code>index</code>	The index to be used in the image segmentation. See <code>image_index()</code> for more details.
<code>threshold</code>	By default ( <code>threshold = "Otsu"</code> ), a threshold value based on Otsu's method is used to reduce the grayscale image to a binary image. If a numeric value is informed, this value will be used as a threshold. Inform any non-numeric value different than "Otsu" to iteratively chosen the threshold based on a raster plot showing pixel intensity of the index.
<code>fill_hull</code>	Fill holes in the objects? Defaults to <code>FALSE</code> .
<code>filter</code>	Performs median filtering after image processing? defaults to <code>FALSE</code> . See more at <code>image_filter()</code> .
<code>parallel</code>	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when <code>image</code> is a list. The number of sections is set up to 70% of available cores.
<code>workers</code>	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
<code>show_image</code>	Show the image results? Defaults to <code>TRUE</code> .
<code>nrow, ncol</code>	Arguments passed on to <code>EImage::combine()</code> . The number of rows or columns in the plot grid. Defaults to <code>NULL</code> , i.e., a square grid is produced.
<code>verbose</code>	If <code>TRUE</code> (default) a summary is shown in the console.

**Value**

A list with the following objects

- `results` A data frame with the number of pixels and proportion of pixels in relation to the previous segmentation.
- `images` A list of segmented images.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**Examples**

```
img <- image_pliman("sev_leaf.jpg")
plot(img)

# prop_segmented() will be deprecated in the future. Use image_segment_iter() instead.
image_segment_iter(img,
  nseg = 2,
  index = c("G", "GLI"),
  ncol = 3)
```

---

rgb\_to\_hsv

*Color spaces*

---

### Description

Convert RGB to LAB color space.

### Usage

```
rgb_to_hsv(image)
```

### Arguments

image            An image object.

### Value

A list containing the image in the new color space.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

### Examples

```
library(pliman)
img <- image_pliman("sev_leaf.jpg")
img2 <- rgb_to_hsv(img)
image_combine(img, img2)
```

---

sad

*Produces Santandard Area Diagrams*

---

### Description

Given an object computed with `measure_disease()` a Standard Area Diagram (SAD) with `n` images are returned with the respective severity values.

### Usage

```
sad(
  object,
  n,
  show_original = FALSE,
  show_contour = FALSE,
  nrow = NULL,
  ncol = NULL,
  ...
)
```

## Arguments

object	An object computed with <code>measure_disease()</code> .
n	The number of leaves in the Standard Area Diagram.
show_original	Show original images? Defaults to FALSE, i.e., a mask is returned.
show_contour	Show original images? Defaults to FALSE, i.e., a mask is returned.
nrow, ncol	The number of rows and columns in the plot. See <code>[image_combine()]</code> <code>[image_combine()]</code> : <code>R:image_combine()</code>
...	Other arguments passed on to <code>measure_disease()</code> .

## Details

The leaves with the smallest and highest severity will always be in the SAD. If  $n = 1$ , the leaf with the smallest severity will be returned. The others are sampled sequentially to achieve the  $n$  images after severity has been ordered in an ascending order. For example, if there are 30 leaves and  $n$  is set to 3, the leaves sampled will be the 1st, 15th, and 30th with the smallest severity values.

The SAD can be only computed if an image pattern name is used in argument `pattern` of `measure_disease()`. If the images are saved, the  $n$  images will be retrieved from `dir_processed` directory. Otherwise, the severity will be computed again to generate the images.

## Value

A data frame with the severity values for the  $n$  sampled leaves. A plot with the standard area diagram can be saved by wrapping `sad()` with `png()`.

## References

Del Ponte EM, Pethybridge SJ, Bock CH, et al (2017) Standard area diagrams for aiding severity estimation: Scientometrics, pathosystems, and methodological trends in the last 25 years. *Phytopathology* 107:1161–1174. doi: [10.1094/PHYTO02170069FI](https://doi.org/10.1094/PHYTO02170069FI)

## Examples

```
## Not run:
library(pliman)
sev <-
measure_disease(pattern = "sev_leaf",
                img_healthy = "sev_healthy",
                img_symptoms = "sev_symp",
                img_background = "sev_back",
                show_image = FALSE,
                save_image = TRUE,
                show_original = FALSE,
                dir_original = image_pliman(),
                dir_processed = tempdir())

sad(sev, n = 2)

## End(Not run)
```

---

symptomatic_area	<i>Calculates the percentage of symptomatic area</i>
------------------	--

---

### Description

Calculates the percentage of symptomatic leaf area in sample or entire leaf based on provided color palettes samples. A general linear model (binomial family) fitted to the RGB values is used to segment the lesions from the healthy leaf. If a pallet of background is provided, the function takes care of the details to isolate it before computing the number and area of lesions. By using `pattern` it is possible to process several images with common pattern names that are stored in the current working directory or in the subdirectory informed in `dir_original`.

### Usage

```
symptomatic_area(  
  img,  
  img_healthy,  
  img_symptoms,  
  img_background = NULL,  
  pattern = NULL,  
  img_pattern = NULL,  
  resize = FALSE,  
  fill_hull = TRUE,  
  parallel = FALSE,  
  workers = NULL,  
  nrows = 3000,  
  show_image = FALSE,  
  show_original = TRUE,  
  show_contour = TRUE,  
  show_background = TRUE,  
  col_leaf = "green",  
  col_symptoms = "red",  
  col_background = NULL,  
  save_image = FALSE,  
  prefix = "proc_",  
  dir_original = NULL,  
  dir_processed = NULL,  
  verbose = TRUE  
)
```

### Arguments

<code>img</code>	The image to be analyzed.
<code>img_healthy</code>	A color palette of healthy areas.
<code>img_symptoms</code>	A color palette of symptomatic areas.
<code>img_background</code>	A color palette of areas with symptoms.

pattern	A pattern of file name used to identify images to be imported. For example, if pattern = "im" all images in the current working directory that the name matches the pattern (e.g., img1.-, image1.-, im2.-) will be imported as a list. Providing any number as pattern (e.g., pattern = "1") will select images that are named as 1.-, 2.-, and so on. An error will be returned if the pattern matches any file that is not supported (e.g., img1.pdf).
img_pattern	Deprecated. Use pattern instead.
resize	Resize the image before processing? Defaults to FALSE. Use a numeric value of range 0-100 (proportion of the size of the original image).
fill_hull	Fill holes in the image? Defaults to TRUE. This is useful to fill holes in leaves, e.g., those caused by insect attack, ensuring the hole area will be accounted for the leaf, not background.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time, especially when img_pattern is used is informed. The number of sections is set up to 70% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time. Defaults to 50% of available cores.
nrows	The number of lines to be used in training step. Defaults to 3000.
show_image	Show image after processing?
show_original	Show the symptoms in the original image? Defaults to TRUE.
show_contour	Show a contour line around the symptomatic area? Defaults to TRUE. If false, the symptomatic area will be filled with the color informed in col_symptoms argument.
show_background	Show the background? Defaults to TRUE. A white background is shown by default when show_original = FALSE.
col_leaf	Leaf color after image processing. Defaults to "green"
col_symptoms	Symptoms color after image processing. Defaults to "red".
col_background	Background color after image processing. Defaults to "NULL".
save_image	Save the image after processing? The image is saved in the current working directory named with the prefix provided in proc_* where * is the image name given in img.
prefix	The prefix to be included in the processed images. Defaults to "proc_".
dir_original, dir_processed	The directory containing the original and processed images. Defaults to NULL. In this case, the function will search for the image img in the current working directory. After processing, when save_image = TRUE, the processed image will be also saved in such a directory.
verbose	If TRUE (default) a summary is shown in the console.

### Value

A data frame with the results (healthy and symptomatic area) for each image.



## Examples

```
library(pliman)
img <- image_pliman("sev_leaf.jpg")
healthy <- image_pliman("sev_healthy.jpg")
symptoms <- image_pliman("sev_sympt.jpg")
background <- image_pliman("sev_back.jpg")
image_combine(img, healthy, symptoms, background)
symptomatic_area(img = img,
                 img_healthy = healthy,
                 img_symptoms = symptoms,
                 img_background = background,
                 show_image = TRUE)
```

---

tune_tolerance	<i>Tune tolerance parameter</i>
----------------	---------------------------------

---

## Description

Provides options for tuning tolerance parameter utilized in `[analyze_objects()]` in two ways:

- Declaring the actual argument, an iterative algorithm will compute the first analysis and sequentially increase the parameter tolerance if the computed number of objects is greater than actual or reduce the parameter tolerance if the computed number of objects is less than actual. If the algorithm did not converge up to `maxiter` is reached, users can change the default extension value.
- The second way is to create a grid with tolerance and extension values. When grid is informed, all combinations (made by `base::expand.grid()`) are tested and the residual from actual value is plotted. Users can then find a better combination of parameters to use in `analyze_objects()`.

## Usage

```
tune_tolerance(  
  img,  
  actual,  
  start_tol = NULL,  
  extension = NULL,  
  grid = NULL,  
  maxiter = 200,  
  index = "NB",  
  my_index = NULL,  
  plot = TRUE,  
  fill_hull = FALSE,  
  filter = FALSE,  
  invert = FALSE,
```

```

workers = NULL,
verbose = TRUE
)

```

### Arguments

<code>img</code>	The image to be analyzed.
<code>actual</code>	The actual number of objects.
<code>start_tol</code>	An starting value for tolerance. Defaults to 1.
<code>extension</code>	The extension value. Defaults to 1.
<code>grid</code>	A list with a numeric sequence for tolerance and extension values. When grid is informed, all combinations are tested and the residual from actual value is plotted.
<code>maxiter</code>	The maximum number of iterations. Default to 200.
<code>index, my_index</code>	A character value specifying the target mode for conversion to binary image when foreground and background are not declared. Defaults to "NB" (normalized blue). See <a href="#">image_index()</a> for more details.
<code>plot</code>	Logical. If TRUE (default) generates a plot showing the results.
<code>fill_hull</code>	Fill holes in the binary image? Defaults to FALSE. This is useful to fill holes in objects that have portions with a color similar to the background. IMPORTANT: Objects touching each other can be combined into one single object, which may underestimate the number of objects in an image.
<code>filter</code>	Performs median filtering after image processing? defaults to FALSE. See more at <a href="#">image_filter()</a> .
<code>invert</code>	Inverts the binary image, if desired. This is useful to process images with black background. Defaults to FALSE.
<code>workers</code>	The number of multiple sections to be used in the computation.
<code>verbose</code>	If TRUE (default) a summary is shown in the console.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

### Examples

```

library(pliman)
if(FALSE){
img <- image_pliman("soybean_touch.jpg")

# starts with tolerance = 1
tune_tolerance(img, actual = 30)

# Using a grid of tolerance and extension values
tune2 <-
  tune_tolerance(img,
                actual = 30,

```

```
        grid = list(tolerance = seq(0, 5, by = 0.1),
                    extension = 1:4), plot = TRUE)
    }
```

---

utils\_dpi

*Utilities for image resolution*

---

## Description

Provides useful conversions between size (cm), number of pixels (px) and dots per inch (dpi).

- `dpi_to_cm()` converts a known dpi value to centimeters.
- `cm_to_dpi()` converts a known centimeter values to dpi.
- `pixels_to_cm()` converts the number of pixels to centimeters, given a known resolution (dpi).
- `cm_to_pixels()` converts a distance (cm) to number of pixels, given a known resolution (dpi).
- `distance()` Computes the distance between two points in an image based on the Pythagorean theorem.
- `dpi()` An interactive function to compute the image resolution given a known distance informed by the user. See more information in the **Details** section.
- `npixels()` returns the number of pixels of an image.

## Usage

```
dpi_to_cm(dpi)
```

```
cm_to_dpi(cm)
```

```
pixels_to_cm(px, dpi)
```

```
cm_to_pixels(cm, dpi)
```

```
npixels(image)
```

```
dpi(image, plot = TRUE)
```

```
distance(image, plot = TRUE)
```

## Arguments

<code>dpi</code>	The image resolution in dots per inch.
<code>cm</code>	The size in centimeters.
<code>px</code>	The number of pixels.
<code>image</code>	An image object.
<code>plot</code>	Call a new plot to image? Defaults to TRUE.

## Details

`dpi()` only run in an interactive section. To compute the image resolution (dpi) the user must use the left button mouse to create a line of known distance. This can be done, for example, using a template with known distance in the image (e.g., `la_leaves.jpg`).

## Value

- `dpi_to_cm()`, `cm_to_dpi()`, `pixels_to_cm()`, and `cm_to_pixels()` return a numeric value or a vector of numeric values if the input data is a vector.
- `dpi()` returns the computed dpi (dots per inch) given the known distance informed in the plot.

## Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

## Examples

```
library(pliman)
# Convert dots per inch to centimeter
dpi_to_cm(c(1, 2, 3))

# Convert centimeters to dots per inch
cm_to_dpi(c(1, 2, 3))

# Convert centimeters to number of pixels with resolution of 96 dpi.
cm_to_pixels(c(1, 2, 3), 96)

# Convert number of pixels to cm with resolution of 96 dpi.
pixels_to_cm(c(1, 2, 3), 96)

if(isTRUE(interactive())){
#### compute the dpi (dots per inch) resolution ####
# only works in an interactive section
# objects_300dpi.jpg has a known resolution of 300 dpi
img <- image_pliman("objects_300dpi.jpg")
# Higher square: 10 x 10 cm
# 1) Run the function dpi()
# 2) Use the left mouse button to create a line in the higher square
# 3) Declare a known distance (10 cm)
# 4) See the computed dpi
dpi(img)

img2 <- image_pliman("la_leaves.jpg")
# square leaf sample (2 x 2 cm)
dpi(img2)
}
```

**Description**

- `file_extension()` Get the extension of a file.
- `file_name()` Get the name of a file.
- `file_dir()` Get or directory of a file
- `manipulate_files()` Manipulate files in a directory with options to rename (insert prefix or suffix) and save the new files to the same or other provided directory.
- `pliman_indexes()` Get the indexes available in pliman.
- `pliman_indexes_eq()` Get the equation of the indexes available in pliman.

**Usage**

```
file_extension(file)
```

```
file_name(file)
```

```
file_dir(file)
```

```
manipulate_files(  
    pattern,  
    dir = NULL,  
    prefix = NULL,  
    name = NULL,  
    suffix = NULL,  
    extension = NULL,  
    sep = "",  
    save_to = NULL,  
    overwrite = FALSE,  
    remove_original = FALSE,  
    verbose = TRUE  
)
```

```
pliman_indexes()
```

```
pliman_indexes_eq()
```

**Arguments**

- |                      |  |
|----------------------|--|
| <code>file</code>    | The file name.   |
| <code>pattern</code> | A file name pattern.   |
| <code>dir</code>     | The working directory containing the files to be manipulated. Defaults to the current working directory. |

prefix, suffix	A prefix or suffix to be added in the new file names. Defaults to NULL (no prefix or suffix).
name	The name of the new files. Defaults to NULL (original names). name can be either a single value or a character vector of the same length as the number of files manipulated. If one value is informed, a sequential vector of names will be created as "name_1", "name_2", and so on.
extension	The new extension of the file. If not declared (default), the original extensions will be used.
sep	An optional separator. Defaults to "".
save_to	The directory to save the new files. Defaults to the current working directory. If the file name of a file is not changed, nothing will occur. If save_to refers to a subfolder in the current working directory, the files will be saved to the given folder. In case of the folder doesn't exist, it will be created. By default, the files will not be overwritten. Set overwrite = TRUE to overwrite the files.
overwrite	Overwrite the files? Defaults to FALSE.
remove_original	Remove original files after manipulation? defaults to FALSE. If TRUE the files in pattern will be removed.
verbose	If FALSE, the code is run silently.

### Value

- file\_extension(), file\_name(), and file\_dir() return a character string.
- manipulate\_files() No return value. If verbose == TRUE, a message is printed indicating which operation succeeded (or not) for each of the files attempted.

### Examples

```
library(pliman)
# get file name, directory and extension
file <- "E:/my_folder/my_subfolder/image1.png"
file_dir(file)
file_name(file)
file_extension(file)

# manipulate files
dir <- tempdir()
list.files(dir)
file.create(paste0(dir, "/test.txt"))
list.files(dir)
manipulate_files("test",
                 dir = paste0(dir, "\\"),
                 prefix = "chang_",
                 save_to = paste0(dir, "\\"),
                 overwrite = TRUE)
list.files(dir)
```

---

 utils\_image                      *Import and export images*


---

## Description

Import images from files and URLs and write images to files, possibly with batch processing.

## Usage

```
image_import(
  image,
  ...,
  pattern = NULL,
  path = NULL,
  plot = FALSE,
  nrow = NULL,
  ncol = NULL,
  img_pattern = NULL
)
```

```
image_export(image, name, extension = NULL, subfolder = NULL, ...)
```

```
image_show(image)
```

```
image_pliman(image, plot = FALSE)
```

## Arguments

image	<ul style="list-style-type: none"> <li>• For <code>image_import()</code>, a character vector of file names or URLs.</li> <li>• For <code>image_export()</code>, an Image object, an array or a list of images.</li> <li>• For <code>image_pliman()</code>, a character value specifying the image example. See <code>?pliman_images</code> for more details.</li> </ul>
...	Alternative arguments passed to the corresponding functions from the <code>jpeg</code> , <code>png</code> , and <code>tiff</code> packages.
pattern	A pattern of file name used to identify images to be imported. For example, if <code>pattern = "im"</code> all images in the current working directory that the name matches the pattern (e.g., <code>img1.-</code> , <code>image1.-</code> , <code>im2.-</code> ) will be imported as a list. Providing any number as pattern (e.g., <code>pattern = "1"</code> ) will select images that are named as <code>1.-</code> , <code>2.-</code> , and so on. An error will be returned if the pattern matches any file that is not supported (e.g., <code>img1.pdf</code> ).
path	A character vector of full path names; the default corresponds to the working directory, <code>getwd()</code> . It will overwrite (if given) the path informed in <code>image</code> argument.
plot	Plots the image after importing? Defaults to <code>FALSE</code> .
nrow, ncol	Passed on to <code>image_combine()</code> . The number of rows and columns to use in the composite image when <code>plot = TRUE</code> .

img_pattern	Deprecated. Use pattern instead.
name	An string specifying the name of the image. It can be either a character with the image name (e.g., "img1") or name and extension (e.g., "img1.jpg"). If none file extension is provided, the image will be saved as a *.jpg file.
extension	When image is a list, extension can be used to define the extension of exported files. This will overwrite the file extensions given in image.
subfolder	Optional character string indicating a subfolder within the current working directory to save the image(s). If the folder doesn't exist, it will be created.

### Value

- `image_import()` returns a new Image object.
- `image_export()` returns an invisible vector of file names.
- `image_pliman()` returns a new Image object with the example image required. If an empty call is used, the path to the `tmp_images` directory installed with the package is returned.

### Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

### Examples

```
library(pliman)
folder <- image_pliman()
full_path <- paste0(folder, "/sev_leaf.jpg")
(path <- file_dir(full_path))
(file <- basename(full_path))
image_import(image = full_path)
image_import(image = file, path = path)
```

---

utils\_measures

*Utilities for object measures*

---

### Description

- `get_measures()` computes object measures (area, perimeter, radius) by using either a known resolution (dpi) or an object with known measurements.
- `plot_measures()` draws the object measures given in an object to the current plot. The object identification ("id") is drawn by default.



**Usage**

```

get_measures(
    object,
    id = NULL,
    measure = NULL,
    dpi = NULL,
    sep = "\\_|-",
    verbose = TRUE,
    digits = 3
)

plot_measures(
    object,
    measure = "id",
    hjust = NULL,
    vjust = NULL,
    digits = 2,
    size = 0.9,
    col = "white",
    ...
)

```

**Arguments**

object	An object computed with <code>analyze_objects()</code> .
id	An object in the image to indicate a known value.
measure	For <code>plot_measures()</code> , a character string; for <code>get_measures()</code> , a two-sided formula, e.g., <code>measure = area ~ 100</code> indicating the known value of object <code>id</code> . The right-hand side is the known value and the left-hand side can be one of the following. <ul style="list-style-type: none"> <li>• <code>area</code> The known area of the object.</li> <li>• <code>perimeter</code> The known perimeter of the object.</li> <li>• <code>radius_mean</code> The known radius of the object.</li> <li>• <code>radius_min</code> The known minimum radius of the object. If the object is a square, then the <code>radius_min</code> of such object will be <math>L/2</math> where <math>L</math> is the length of the square side.</li> <li>• <code>radius_max</code> The known maximum radius of the object. If the object is a square, then the <code>radius_max</code> of such object according to the Pythagorean theorem will be <math>L \times \sqrt{2} / 2</math> where <math>L</math> is the length of the square side.</li> </ul>
dpi	A known resolution of the image in DPI (dots per inch).
sep	Regular expression to manage file names. The function combines in the merge object the object measures (sum of area and mean of all the other measures) of all images that share the same filename prefix, defined as the part of the filename preceding the first hyphen (-) or underscore (_) (no hyphen or underscore is required). For example, the measures of images named <code>L1-1.jpeg</code> , <code>L1-2.jpeg</code> , and <code>L1-3.jpeg</code> would be combined into a single image information ( <code>L1</code> ). This

	feature allows the user to treat multiple images as belonging to a single sample, if desired. Defaults to <code>sep = "\\_ -"</code> .
<code>verbose</code>	If <code>FALSE</code> , runs the code silently.
<code>digits</code>	The number of significant figures. Defaults to 2.
<code>hjust, vjust</code>	A numeric value to adjust the labels horizontally and vertically. Positive values will move labels to right ( <code>hjust</code> ) and top ( <code>vjust</code> ). Negative values will move the labels to left and bottom, respectively.
<code>size</code>	The size of the text. Defaults to 0.9.
<code>col</code>	The color of the text. Defaults to "white".
<code>...</code>	Further arguments passed on to <code>graphics::text()</code> .

### Value

- For `get_measures()`, if `measure` is informed, the pixel values will be corrected by the value of the known object, given in the unit of the right-hand side of `measure`. If `dpi` is informed, then all the measures will be adjusted to the known `dpi`.
  - If applied to an object of class `anal_obj`, returns a data frame with the object `id` and the (corrected) measures.
  - If applied to an object of class `anal_obj_ls`, returns a list of class `measures_ls`, with two objects: (i) `results`, a data frame containing the identification of each image (`img`) and object within each image (`id`); and (ii) `summary` a data frame containing the values for each image. If more than one object is detected in a given image, the number of objects (`n`), total area (`area_sum`), mean area (`area_mean`) and the standard deviation of the area (`area_sd`) will be computed. For the other measures (perimeter and radius), the mean values are presented.
- `plot_measures()` returns a `NULL` object, drawing the text according to the `x` and `y` coordinates of the objects in `object`.

### Examples

```
library(pliman)
img <- image_pliman("objects_300dpi.jpg")
plot(img)
# Image with four objects with a known resolution of 300 dpi
# Higher square: 10 x 10 cm
# Lower square: 5 x 5 cm
# Rectangle: 4 x 2 cm
# Circle: 3 cm in diameter

# Count the objects using the blue band to segment the image
results <-
  analyze_objects(img,
                 index = "B")
plot_measures(results, measure = "id")

# Get object measures by declaring the known resolution in dots per inch
(measures <- get_measures(results, dpi = 300))
```

```
# Calculated diagonal of the object 1
# 10 * sqrt(2) = 14.14

# Observed diagonal of the object 1
measures[1, "radius_max"] * 2

# Get object measures by declaring the known area of object 1
get_measures(results,
             id = 1,
             area ~ 100)
```

---

utils\_objects

*Utilities for working with image objects*

---

## Description

- `object_id()` get the object identification in an image.
- `object_coord()` get the object coordinates and (optionally) draw a bounding rectangle around multiple objects in an image.
- `object_contour()` returns the coordinates (x and y) for the contours of each object in the image.
- `object_isolate()` isolates an object from an image.

## Usage

```
object_coord(
  image,
  id = NULL,
  index = "NB",
  watershed = TRUE,
  invert = FALSE,
  fill_hull = FALSE,
  edge = 2,
  extension = NULL,
  tolerance = NULL,
  object_size = "medium",
  parallel = FALSE,
  workers = NULL,
  show_image = TRUE
)
```

```
object_contour(
  image,
  index = "NB",
```

```

invert = FALSE,
fill_hull = FALSE,
watershed = TRUE,
extension = NULL,
tolerance = NULL,
object_size = "medium",
parallel = FALSE,
workers = NULL,
show_image = TRUE
)

```

```
object_isolate(image, id = NULL, parallel = FALSE, workers = NULL, ...)
```

```
object_id(image, parallel = FALSE, workers = NULL, ...)
```

### Arguments

image	An image of class Image or a list of Image objects.
id	<ul style="list-style-type: none"> <li>For <code>object_coord()</code>, a vector (or scalar) of object id to compute the bounding rectangle. Object ids can be obtained with <code>object_id()</code>. Set <code>id = "all"</code> to compute the coordinates for all objects in the image. If <code>id = NULL</code> (default) a bounding rectangle is drawn including all the objects.</li> <li>For <code>object_isolate()</code>, a scalar that identifies the object to be extracted.</li> </ul>
index	The index to produce a binary image used to compute bounding rectangle coordinates. See <code>image_binary()</code> for more details.
watershed	If TRUE (default) performs watershed-based object detection. This will detect objects even when they are touching one other. If FALSE, all pixels for each connected set of foreground pixels are set to a unique object. This is faster but is not able to segment touching objects.
invert	Inverts the binary image, if desired. Defaults to FALSE.
fill_hull	Fill holes in the objects? Defaults to FALSE.
edge	The number of pixels in the edge of the bounding rectangle. Defaults to 2.
extension, tolerance, object_size	Controls the watershed segmentation of objects in the image. See <code>analyze_objects()</code> for more details.
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 50% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
show_image	Shows the image with bounding rectangles? Defaults to TRUE.
...	<ul style="list-style-type: none"> <li>For <code>object_isolate()</code>, further arguments passed on to <code>object_coord()</code>.</li> <li>For <code>object_id()</code>, further arguments passed on to <code>analyze_objects()</code>.</li> </ul>

**Value**

- `object_id()` An image of class "Image" containing the object's identification.
- `object_coord()` A list with the coordinates for the bounding rectangles. If `id = "all"` or a numeric vector, a list with a vector of coordinates is returned.
- `object_isolate()` An image of class "Image" containing the isolated object.

**Examples**

```
library(pliman)
img <- image_pliman("la_leaves.jpg")
# Get the object's (leaves) identification
object_id(img)

# Get the coordinates and draw a bounding rectangle around leaves 1 and 3
object_coord(img, id = c(1, 3))

# Isolate leaf 3
isolated <- object_isolate(img, id = 3)
plot(isolated)
```

---

utils\_polygon

*Utilities for Polygons*

---

**Description**

- `conv_hull()` Compute convex hull of a set of points.
- `poly_area()` Compute the area of a polygon given by the vertices in the vectors `x` and `y`.
- `poly_mass()` Compute the center of mass of a polygon given by the vertices in the vectors `x` and `y`.
- `poly_spline()` Smooths a polygon contour.
- `plot_contour()` Plot contour lines.
- `plot_ellipse()` Plots an ellipse that fits the major and minor axis for each object.

**Usage**

```
conv_hull(x, y = NULL, closed = TRUE)

poly_area(x, y = NULL)

poly_mass(x, y = NULL)

poly_spline(x, y = NULL, vertices = 100, k = 2, ...)
```

```

plot_contour(x, y = NULL, id = NULL, col = "black", lwd = 1, ...)

plot_mass(
  x,
  y = NULL,
  id = NULL,
  arrow = TRUE,
  col = "black",
  cex = 1,
  lwd = 1
)

plot_ellipse(object, id = NULL, col = "black", lwd = 1)

```

### Arguments

<code>x, y</code>	Coordinate vectors of points. This can be specified as two vectors ( <code>x</code> and <code>y</code> ), or a 2-column matrix <code>x</code> . If <code>x</code> is a list of vector coordinates the function will be applied to each element using <code>base::lapply()</code> .
<code>closed</code>	If <code>TRUE</code> (default) returns the vector of points of a closed polygon, i.e., the first point is replicated as the last one.
<code>vertices</code>	The number of spline vertices to create.
<code>k</code>	The number of points to wrap around the ends to obtain a smooth periodic spline.
<code>...</code>	<ul style="list-style-type: none"> <li>For <code>plot_contour()</code> and <code>plot_ellipse()</code> further arguments passed on to <code>graphics::lines()</code>.</li> <li>For <code>plot_mass()</code>, further arguments passed on to <code>graphics::points()</code>.</li> </ul>
<code>id</code>	The object identification (numeric) to plot the contour/ellipse. By default ( <code>id = NULL</code> ), the contour is plotted to all objects
<code>col, lwd, cex</code>	The color, width of the lines, and size of point, respectively.
<code>arrow</code>	If <code>TRUE</code> (default) plots two arrows connecting the center of mass to the minimum and maximum radius.
<code>object</code>	An object computed with <code>analyze_objects()</code> .

### Details

`poly_area()` computes the area of a polygon given a set of `x` and `y` coordinates using the Shoelace formula, as follows (Lee and Lim, 2017).

$$A = \frac{1}{2} \left| \sum_{i=1}^n (x_i y_{i+1} - x_{i+1} y_i) \right|$$

, where `x` and `y` are the coordinates which form the corners of a polygon, and `n` is the number of coordinates.

**Value**

- `conv_hull()` and `poly_spline()` returns a matrix with x and y coordinates for the convex hull/smooth line in clockwise order. If x is a list, a list of points is returned.
- `poly_area()` returns a double, or a list if x is a list of vector points.
- `poly_mass()` returns a data.frame containing the coordinates for the center of mass, as well as for the maximum and minimum distance from contour to the center of mass.
- `plot_contour()`, `plot_mass()`, and `plot_ellipse()` return a NULL object.

**References**

Lee, Y., & Lim, W. (2017). Shoelace Formula: Connecting the Area of a Polygon and the Vector Cross Product. *The Mathematics Teacher*, 110(8), 631–636. doi: [10.5951/mathteacher.110.8.0631](https://doi.org/10.5951/mathteacher.110.8.0631)

**Examples**

```
library(pliman)
# A 2 x 2 square
x <- c(0, 0, 2, 2, 0)
y <- c(0, 2, 2, 0, 0)
df <- data.frame(x = x, y = y)
plot(df)
with(df, polygon(x, y, col = "red"))

poly_area(x, y)
poly_area(df)

# center of mass of the square
cm <- poly_mass(df)
plot_mass(cm)

# The convex hull will be the vertices of the square
(conv_square <- conv_hull(df))
plot_contour(conv_square,
             col = "blue",
             lwd = 6)
poly_area(conv_square)

##### Example with a polygon#####
x <- c(0, 1, 2, 3, 5, 2, -1, 0, 0)
y <- c(5, 6.5, 7, 3, 1, 1, 0, 2, 5)
df_poly <- data.frame(x = x, y = y)

# area of the polygon
poly_area(df_poly)
plot(df_poly, pch = 19, col = "red")
with(df_poly, polygon(x, y, col = "red"))

# center of mass of polygon
# arrows from center of mass to maximum and minimum radius
```

```
cm <- poly_mass(df_poly)
plot_mass(cm, arrow = TRUE, col = "blue")

# vertices of the convex hull
(conv_poly <- conv_hull(df_poly))

# area of the convex hull
poly_area(conv_poly)

with(conv_poly,
      polygon(x, y,
              col = rgb(1, 0, 0, 0.2)))
```

---

utils\_transform

*Spatial transformations*

---

## Description

Performs image rotation and reflection

- `image_autocrop()` Crops automatically an image to the area of objects.
- `image_crop()` Crops an image to the desired area.
- `image_trim()` Remove pixels from the edges of an image (20 by default).
- `image_dimension()` Gives the dimension (width and height) of an image.
- `image_rotate()` Rotates the image clockwise by the given angle.
- `image_horizontal()` Converts (if needed) an image to a horizontal image.
- `image_vertical()` Converts (if needed) an image to a vertical image.
- `image_hreflect()` Performs horizontal reflection of the image.
- `image_vreflect()` Performs vertical reflection of the image.
- `image_resize()` Resize the image. See more at [EBImage::resize\(\)](#).
- `image_contrast()` Improve contrast locally by performing adaptive histogram equalization. See more at [EBImage::clahe\(\)](#).
- `image_dilate()` Performs image dilatation. See more at [EBImage::dilate\(\)](#).
- `image_erode()` Performs image erosion. See more at [EBImage::erode\(\)](#).
- `image_opening()` Performs an erosion followed by a dilation. See more at [EBImage::opening\(\)](#).
- `image_closing()` Performs a dilation followed by an erosion. See more at [EBImage::closing\(\)](#).
- `image_filter()` Performs median filtering in constant time. See more at [EBImage::medianFilter\(\)](#).
- `image_blur()` Performs blurring filter of images. See more at [EBImage::gblur\(\)](#).
- `image_skeleton()` Performs image skeletonization.



**Usage**

```
image_autocrop(  
  image,  
  index = "NB",  
  edge = 5,  
  parallel = FALSE,  
  workers = NULL,  
  verbose = TRUE,  
  plot = FALSE  
)  
  
image_crop(  
  image,  
  width = NULL,  
  height = NULL,  
  parallel = FALSE,  
  workers = NULL,  
  verbose = TRUE,  
  plot = FALSE  
)  
  
image_dimension(image, parallel = FALSE, workers = NULL, verbose = TRUE)  
  
image_rotate(  
  image,  
  angle,  
  bg_col = "white",  
  parallel = FALSE,  
  workers = NULL,  
  verbose = TRUE,  
  plot = TRUE  
)  
  
image_horizontal(  
  image,  
  parallel = FALSE,  
  workers = NULL,  
  verbose = TRUE,  
  plot = FALSE  
)  
  
image_vertical(  
  image,  
  parallel = FALSE,  
  workers = NULL,  
  verbose = TRUE,  
  plot = FALSE  
)
```

```
image_hreflect(  
    image,  
    parallel = FALSE,  
    workers = NULL,  
    verbose = TRUE,  
    plot = FALSE  
)
```

```
image_vreflect(  
    image,  
    parallel = FALSE,  
    workers = NULL,  
    verbose = TRUE,  
    plot = FALSE  
)
```

```
image_resize(  
    image,  
    rel_size = 100,  
    width,  
    height,  
    parallel = FALSE,  
    workers = NULL,  
    verbose = TRUE,  
    plot = FALSE  
)
```

```
image_trim(  
    image,  
    edge = NULL,  
    top = NULL,  
    bottom = NULL,  
    left = NULL,  
    right = NULL,  
    parallel = FALSE,  
    workers = NULL,  
    verbose = TRUE,  
    plot = FALSE  
)
```

```
image_dilate(  
    image,  
    kern = NULL,  
    size = NULL,  
    shape = "disc",  
    parallel = FALSE,  
    workers = NULL,
```

```
    verbose = TRUE,  
    plot = FALSE  
  )  
  
  image_erode(  
    image,  
    kern = NULL,  
    size = NULL,  
    shape = "disc",  
    parallel = FALSE,  
    workers = NULL,  
    verbose = TRUE,  
    plot = FALSE  
  )  
  
  image_opening(  
    image,  
    kern = NULL,  
    size = NULL,  
    shape = "disc",  
    parallel = FALSE,  
    workers = NULL,  
    verbose = TRUE,  
    plot = FALSE  
  )  
  
  image_closing(  
    image,  
    kern = NULL,  
    size = NULL,  
    shape = "disc",  
    parallel = FALSE,  
    workers = NULL,  
    verbose = TRUE,  
    plot = FALSE  
  )  
  
  image_skeleton(  
    image,  
    kern = NULL,  
    parallel = FALSE,  
    workers = NULL,  
    verbose = TRUE,  
    plot = FALSE,  
    ...  
  )  
  
  image_filter(  
    image,  
    kernel = NULL,  
    mode = "multiply",  
    parallel = FALSE,  
    workers = NULL,  
    verbose = TRUE,  
    plot = FALSE  
  )  
}
```

```

    image,
    size = 2,
    cache = 512,
    parallel = FALSE,
    workers = NULL,
    verbose = TRUE,
    plot = FALSE
  )

```

```

image_blur(
  image,
  sigma = 3,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE,
  plot = FALSE
)

```

```

image_contrast(
  image,
  parallel = FALSE,
  workers = NULL,
  verbose = TRUE,
  plot = FALSE
)

```

## Arguments

image	An image or a list of images of class Image.
index	The index to segment the image. See <code>image_index()</code> for more details. Defaults to "NB" (normalized blue).
edge	<ul style="list-style-type: none"> <li>for <code>image_autocrop()</code> the number of pixels in the edge of the cropped image. If <code>edge = 0</code> the image will be cropped to create a bounding rectangle (x and y coordinates) around the image objects.</li> <li>for <code>image_trim()</code>, the number of pixels removed from the edges. By default, 20 pixels are removed from all the edges.</li> </ul>
parallel	Processes the images asynchronously (in parallel) in separate R sessions running in the background on the same machine. It may speed up the processing time when image is a list. The number of sections is set up to 70% of available cores.
workers	A positive numeric scalar or a function specifying the maximum number of parallel processes that can be active at the same time.
verbose	If TRUE (default) a summary is shown in the console.
plot	If TRUE plots the modified image. Defaults to FALSE.
width, height	<ul style="list-style-type: none"> <li>For <code>image_resize()</code> the Width and height of the resized image. These arguments can be missing. In this case, the image is resized according to the relative size informed in <code>rel_size</code>.</li> </ul>

- For `image_crop()` a numeric vector indicating the pixel range (x and y, respectively) that will be maintained in the cropped image, e.g., `width = 100:200`

<code>angle</code>	The rotation angle in degrees.
<code>bg_col</code>	Color used to fill the background pixels, defaults to "white".
<code>rel_size</code>	The relative size of the resized image. Defaults to 100. For example, setting <code>rel_size = 50</code> to an image of width 1280 x 720, the new image will have a size of 640 x 360.
<code>top, bottom, left, right</code>	The number of pixels removed from top, bottom, left, and right when using <code>image_trim()</code> .
<code>kern</code>	An Image object or an array, containing the structuring element. Defaults to a brush generated with <code>EBImage::makeBrush()</code> .
<code>size</code>	<ul style="list-style-type: none"> <li>• For <code>image_filter()</code> is the median filter radius (integer). Defaults to 3.</li> <li>• For <code>image_dilate()</code> and <code>image_erode()</code> is an odd number containing the size of the brush in pixels. Even numbers are rounded to the next odd one. The default depends on the image resolution and is computed as the image resolution (megapixels) times 20.</li> </ul>
<code>shape</code>	A character vector indicating the shape of the brush. Can be box, disc, diamond, Gaussian or line. Default is disc.
<code>...</code>	Additional arguments passed on to <code>image_binary()</code> .
<code>cache</code>	The the L2 cache size of the system CPU in kB (integer). Defaults to 512.
<code>sigma</code>	A numeric denoting the standard deviation of the Gaussian filter used for blurring. Defaults to 3.

**Value**

- `image_skeleton()` returns a binary Image object.
- All other functions returns a modified version of image depending on the `image_*`() function used.
- If image is a list, a list of the same length will be returned.

**Author(s)**

Tiago Olivoto <tiagoolivoto@gmail.com>

**Examples**

```
library(pliman)
img <- image_pliman("sev_leaf.jpg")
plot(img)
img <- image_resize(img, 50)
img1 <- image_rotate(img, 45)
img2 <- image_hreflect(img)
img3 <- image_vreflect(img)
img4 <- image_vertical(img)
image_combine(img1, img2, img3, img4)
```

# Index

- \* **images**
  - pliman\_images, 34
- analyze\_objects, 2
- analyze\_objects(), 2, 25, 31, 49, 52, 54
  
- base::expand.grid(), 41
- base::lapply(), 54
  
- cm\_to\_dpi (utils\_dpi), 43
- cm\_to\_dpi(), 43, 44
- cm\_to\_pixels (utils\_dpi), 43
- cm\_to\_pixels(), 43, 44
- conv\_hull (utils\_polygon), 53
- count\_lesions, 8
- count\_objects, 12
  
- distance (utils\_dpi), 43
- distance(), 43
- dpi (utils\_dpi), 43
- dpi(), 43, 44
- dpi\_to\_cm (utils\_dpi), 43
- dpi\_to\_cm(), 43, 44
  
- EImage::clahe(), 56
- EImage::closing(), 56
- EImage::combine(), 36
- EImage::dilate(), 56
- EImage::erode(), 56
- EImage::gblur(), 56
- EImage::makeBrush(), 61
- EImage::medianFilter(), 56
- EImage::opening(), 56
- EImage::resize(), 56
- EImage::watershed(), 10, 29
  
- file\_dir (utils\_file), 45
- file\_extension (utils\_file), 45
- file\_name (utils\_file), 45
  
- get\_measures (utils\_measures), 48
  
- getwd(), 47
- graphics::lines(), 54
- graphics::points(), 54
- graphics::text(), 50
  
- image\_autocrop (utils\_transform), 56
- image\_autocrop(), 60
- image\_binary, 15
- image\_binary(), 6, 14, 33, 52, 61
- image\_blur (utils\_transform), 56
- image\_blur(), 21
- image\_closing (utils\_transform), 56
- image\_combine, 17
- image\_combine(), 47
- image\_contrast (utils\_transform), 56
- image\_crop (utils\_transform), 56
- image\_dilate (utils\_transform), 56
- image\_dimension (utils\_transform), 56
- image\_erode (utils\_transform), 56
- image\_export (utils\_image), 47
- image\_filter (utils\_transform), 56
- image\_filter(), 4, 13, 21, 36, 42
- image\_horizontal (utils\_transform), 56
- image\_hreflect (utils\_transform), 56
- image\_import (utils\_image), 47
- image\_index, 18
- image\_index(), 4, 10, 13, 16, 22, 29, 32, 36, 42, 60
- image\_opening (utils\_transform), 56
- image\_palette, 20
- image\_pliman (utils\_image), 47
- image\_resize (utils\_transform), 56
- image\_rotate (utils\_transform), 56
- image\_segment, 21
- image\_segment\_iter (image\_segment), 21
- image\_show (utils\_image), 47
- image\_skeleton (utils\_transform), 56
- image\_to\_mat, 24
- image\_trim (utils\_transform), 56
- image\_trim(), 60, 61

image\_vertical (utils\_transform), 56  
image\_vreflect (utils\_transform), 56  
leaf\_area, 25  
manipulate\_files (utils\_file), 45  
measure\_disease, 27  
measure\_disease(), 37, 38  
npixels (utils\_dpi), 43  
npixels(), 43  
object\_contour (utils\_objects), 51  
object\_coord (utils\_objects), 51  
object\_coord(), 52  
object\_id (utils\_objects), 51  
object\_id(), 52  
object\_isolate (utils\_objects), 51  
objects\_rgb, 31  
pixels\_to\_cm (utils\_dpi), 43  
pixels\_to\_cm(), 43, 44  
pliman\_images, 34  
pliman\_indexes (utils\_file), 45  
pliman\_indexes(), 16, 18, 22  
pliman\_indexes\_eq (utils\_file), 45  
pliman\_indexes\_eq(), 5  
plot.anal\_obj (analyze\_objects), 2  
plot.anal\_obj(), 2  
plot.image\_index (image\_index), 18  
plot\_contour (utils\_polygon), 53  
plot\_ellipse (utils\_polygon), 53  
plot\_mass (utils\_polygon), 53  
plot\_measures (utils\_measures), 48  
png(), 38  
poly\_area (utils\_polygon), 53  
poly\_mass (utils\_polygon), 53  
poly\_spline (utils\_polygon), 53  
prop\_segmented, 35  
rgb\_to\_hsv, 37  
sad, 37  
sad(), 38  
symptomatic\_area, 39  
tune\_tolerance, 41  
utils\_dpi, 43  
utils\_file, 45  
utils\_image, 47  
utils\_measures, 48  
utils\_objects, 51  
utils\_polygon, 53  
utils\_transform, 56