Package 'pliman'

October 14, 2022

Title Tools for Plant Image Analysis

Version 1.1.0

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 to segment images and create binary images using the method of automatic threshold selection proposed by Otsu (1979) <doi: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>

Repository CRAN

Date/Publication 2021-12-10 08:30:02 UTC

R topics documented:

`analyze_objects` *Analyzes objects in an image*

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,
  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,
  lower_circ = NULL,
  upper_circ = 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,
  which = "measure",
  measure = "area",
  type = "density",
  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 `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).
`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 `pattern` is used is informed. When `object_index` 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. IMPORTANT: 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 `image_filter()`.
`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 `image_index()` 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., `object_index = "R/B"`. In this case, it will return for each object in the image, the average value of the R/B ratio. Use `pliman_indexes_eq()` to see the equations of available indexes.

`threshold`
: By default (`threshold = "Otsu"`), 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.

`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 `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.

`lower_eccent, upper_eccent, lower_circ, upper_circ`
: Lower and upper limit for object eccentricity/circularity 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 `contour_col = "red"` and `contour_size = 1`.

`show_background`
: Show the background? Defaults to `TRUE`. A white background is shown by default when `show_original = FALSE`.

`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`.

`which`
: Which to plot. Either 'measure' (object measures) or 'index' (object index). Defaults to `"measure"`.

`measure`
: The measure to plot. Defaults to `"area"`.

`type`
: The type of plot. Either `"hist"` or `"density"`. Partial matches are recognized.

`facet`
: Create a facet plot for each object when `which = "index"` is used?. Defaults to `FALSE`.

`...`
: Further argument passed on to `lattice::histogram()` or `lattice::densityplot()`

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 radius_max / 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-minoraxis^2/majoraxis^2). Circle eccentricity is 0 and straight line eccentricity is 1.
  - theta: object angle (in radians).
  - solidity: object solidity given by area / area_ch.
  - circularity: the object circularity given by $4 * pi * (area/perimeter^2)$.
- 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

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

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, default)
# Shows the average value of the blue index in each object

rgb <-
   analyze_objects(img,
                   marker = "id",
                   object_index = "B")
# density of area
plot(rgb)

# histogram of perimeter
plot(rgb, measure = "perimeter", type = "histogram") # or 'hist'

# density of the blue (B) index
plot(rgb, which = "index")
```

`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 pliman_indexes() and image_index() for more details.
my_index	User can calculate a different index using the band names, e.g. my_index = "R+B/G".
threshold	By default (threshold = "Otsu"), 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 resize = 30, 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

Examples

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

---

image_combine *Combines images to a grid*

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 supress 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 pliman_indexes() 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 atan(2*(B-G-R)/30.5*(G-R))
- HUE2 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

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_segment` *Image segmentation*

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

image	An image object or a list of image objects.
index	• For image_segment(), a character value (or a vector of characters) specifying the target mode for conversion to binary image. See the available indexes with pliman_indexes(). See image_index() for more details. • For image_segment_iter() a character or a vector of characters with the same length of nseg. It can be either an available index (described above) or any operation involving the RGB values (e.g., "B/R+G").
my_index	User can calculate a different index using the bands names, e.g. my_index = "R+B/G".
threshold	By default (threshold = "Otsu"), 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 image_segmentation_iter(), use a vector (allows a mixed (numeric and character) type) with the same length of nseg.
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. For image_segmentation_iter() use a vector with the same length of nseg.
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.
nseg	The number of iterative segmentation steps to be performed.
...	Additional arguments passed on to image_segment().

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

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]])
```

---

measure_disease *Performs plant disease measurements*

Description

- measure_disease() 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 using color palettes. See more at **Details**.
- measure_disease_iter() provides an iterative section for measure_disease(), where the user picks up samples in the image to create the needed color palettes.

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,
  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
)

measure_disease_iter(img, has_background = TRUE, r = 5, ...)
```

Arguments

`img`	The image to be analyzed.
`img_healthy`	A color palette of healthy areas.
`img_symptoms`	A color palette of lesioned areas.
`img_background`	An optional color palette of the image background.
`pattern`	A pattern of file name used to identify images to be processed. For example, if `pattern = "im"` all images that the name matches the pattern (e.g., img1.-, image1.-, im2.-) will be analyzed. Providing any number as pattern (e.g., `pattern = "1"`) will select images that are named as 1.-, 2.-, 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 `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.
`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.

`index_lb` 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 `img_healthy` and `img_symptoms` are not declared. Defaults to `"GLI"`. See `image_index()` for more details.

`threshold` By default (`threshold = NULL`), 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 `index_lb` and `index_dh`, respectively.

`invert` Inverts the binary image if desired. This is useful to process images with black background. Defaults to `FALSE`.

`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 `lower_limit = 0` 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.

`topn_lower, topn_upper`
Select the top `n` lesions based on its area. `topn_lower` selects the `n` lesions with the smallest area whereas `topn_upper` selects the `n` lesions with the largest area.

`randomize` Randomize the lines before training the model? Defaults to `TRUE`.

`nsample` The number of sample pixels to be used in training step. Defaults to `3000`.

`watershed` 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 `EBImage::watershed()`.

`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 `NULL`, i.e., starting values are set up according to the argument `lesion_size`.

`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 TRUE returnS the lesion features such as number, area, perimeter, and radius. Defaults to FALSE.

show_segmentation
Shows the object segmentation colored with random permutations. Defaults to TRUE.

show_image Show image after processing? Defaults to TRUE.

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.

has_background A logical indicating if the image has a background to be segmented before processing.

r The radius of neighborhood pixels. Defaults to 5. A square is drawn indicating the selected pixels.

... Further parameters passed on to measure_disease().

Details

In measure_disease(), 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.

measure_disease_iter() only run in an interactive section. In this function, users will be able to pick up samples of images to iteratively create the needed color palettes. This process calls pick_palette() internally. If has_background is TRUE (default) the color palette for the background is first created. The sample of colors is performed in each left-button mouse click and continues until the user press Esc. Then, a new sampling process is performed to sample the color of healthy tissues and then diseased tissues. The generated palettes are then passed on to measure_disease(). All the arguments of such function can be passed using the ... (three dots).

Value

- measure_disease() returns 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.
- measure_disease_iter() returns a list with the following objects:
  - results A list with the objects returned by measure_disease().
  - leaf The color palettes for the healthy leaf.
  - disease The color palettes for the diseased leaf.
  - background The color palettes for the background.

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_sympt.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)

# an interactive section
measure_disease_iter(img)
```

palettes *Create image palettes*

Description

`image_palette()` 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 `image_filter()`.
`blur`	Performs blurring filter of palettes? Defaults to `FALSE`. See more at `image_blur()`.
`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

- `image_palette()` returns 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)
```

```
# runs only in an iterative section
if(FALSE){
image_palette_pick(img)
}
```

---

pipe *Forward-pipe operator*

Description

Pipe an object forward into a function or call expression.

Usage

```
lhs %>% rhs
```

Arguments

lhs	The result you are piping.
rhs	Where you are piping the result to.

Author(s)

Nathan Eastwood <nathan.eastwood@icloud.com> and Antoine Fabri <antoine.fabri@gmail.com>. The code was obtained from poorman package at https://github.com/nathaneastwood/poorman/blob/master/R/pipe.R

Examples

```
library(pliman)

# Basic use:
 iris %>% head()

# use to apply several functions to an image
img <- image_pliman("la_leaves.jpg")

img %>%
 image_resize(50) %>%        # resize to 50% of the original size
 object_isolate(id = 1) %>%  # isolate object 1
 image_filter() %>%          # apply a median filter
 plot()                      # plot
```

`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

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 measure_disease().
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 [image_combine())] [image_combine())]: R:image_combine())
...	Other arguments passed on to measure_disease().

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 retrevied 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

Examples

```
## Not run:
library(pliman)
sev <-
measure_disease(pattern = "sev_leaf",
                img_healthy = "sev_healthy",
                img_symptoms = "sev_sympt",
                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)
```

summary_index *Summary an object index*

Description

Performs a report of the index between and within objects when `object_index` argument is used in `analyze_objects()`. By using a cut point, the number and proportion of objects with mean value of `index` bellow and above `cut_point` are returned. Additionaly, the number and proportion of pixels bellow and above the cutpoint is shown for each object (id).

Usage

```
summary_index(object, index, cut_point, select_higher = FALSE)
```

Arguments

`object`	An object computed with `analyze_objects()`.
`index`	The index desired, e.g., `"B"`. Note that these value must match the index(es) used in the argument `object_index` of `analyze_objects()`.
`cut_point`	The cut point.
`select_higher`	If `FALSE` (default) selects the objects with `index` smaller than the `cut_point`. Use `select_higher = TRUE` to select the objects with `index` higher than `cut_point`.

Value

A list with the following elements:

- `ids` The identification of selected objects.
- `between_id` A data frame with the following columns
  - `n` The number of objects.
  - `nsel` The number of selected objects.
  - `prop` The proportion of objects selected.
  - `mean_index_sel`, and `mean_index_nsel` The mean value of `index` for the selected and non-selected objects, respectively.
- `within_id` A data frame with the following columns
  - `id` The object identification
  - `n_less` The number of pixels with values lesser than or equal to `cut_point`.
  - `n_greater` The number of pixels with values greater than `cut_point`.
  - `less_ratio` The proportion of pixels with values lesser than or equal to `cut_point`.
  - `greater_ratio` The proportion of pixels with values greater than `cut_point`.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

```
library(pliman)
soy <- image_pliman("soy_green.jpg")
anal <- analyze_objects(soy, object_index = "G")
plot_measures(anal, measure = "G")

summary_index(anal, index = "G", cut_point = 0.5)
```

`tune_tolerance` *Tune tolerance parameter*

Description

Provides options for tunning `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 than 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

img	The image to be analyzed.
actual	The actual number of objects.
start_tol	An starting value for tolerance. Defaults to 1.
extension	The extension value. Defaults to 1.
grid	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.
maxiter	The maximum number of iterations. Default to 200.
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 image_index() for more details.
plot	Logical. If TRUE (default) generates a plot showing the results.
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. IMPORTANT: 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 image_filter().
invert	Inverts the binary image, if desired. This is useful to process images with black background. Defaults to FALSE.
workers	The number of multiple sections to be used in the computation.
verbose	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

`dpi`	The image resolution in dots per inch.
`cm`	The size in centimeters.
`px`	The number of pixels.
`image`	An image object.
`plot`	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)
}
```

---

utils_file *Utilities for file manipulation*

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

`file`	The file name.
`pattern`	A file name pattern.
`dir`	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
)

image_export(image, name, prefix = "", extension = NULL, subfolder = NULL, ...)

image_pliman(image, plot = FALSE)
```

Arguments

`image`	• For `image_import()`, a character vector of file names or URLs. • For `image_export()`, an Image object, an array or a list of images. • For `image_pliman()`, a charactere value specifying the image example. See `?pliman_images` for more details.
`...`	Alternative arguments passed to the corresponding functions from the `jpeg`, `png`, and `tiff` packages.
`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).
`path`	A character vector of full path names; the default corresponds to the working directory, `getwd()`. It will overwrite (if given) the path informed in `image` argument.
`plot`	Plots the image after importing? Defaults to `FALSE`.
`nrow, ncol`	Passed on to `image_combine()`. The number of rows and columns to use in the composite image when `plot = TRUE`.
`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.

prefix	A prefix to include in the image name when exporting a list of images. Defaults to "", i.e., no prefix.
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,
  id = NULL,
  measure = "id",
  hjust = NULL,
  vjust = NULL,
  digits = 2,
  size = 0.9,
  col = "white",
  ...
)
```

Arguments

`object`	An object computed with `analyze_objects()`.
`id`	An object in the image to indicate a known value.
`measure`	For `plot_measures()`, a character string; for `get_measures()`, a two-sided formula, e.g., `measure = area ~ 100` indicating the known value of object `id`. The right-hand side is the known value and the left-hand side can be one of the following. • `area` The known area of the object. • `perimeter` The known perimeter of the object. • `radius_mean` The known radius of the object. • `radius_min` The known minimum radius of the object. If the object is a square, then the `radius_min` of such object will be `L/2` where `L` is the length of the square side. • `radius_max` The known maximum radius of the object. If the object is a square, then the `radius_max` of such object according to the Pythagorean theorem will be `L x sqrt(2) / 2` where `L` is the length of the square side.
`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 L1-1.jpeg, L1-2.jpeg, and L1-3.jpeg would be combined into a single image information (L1). This feature allows the user to treat multiple images as belonging to a single sample, if desired. Defaults to `sep = "\\_|-"`.
`verbose`	If `FALSE`, runs the code silently.
`digits`	The number of significant figures. Defaults to `2`.
`hjust, vjust`	A numeric value to adjust the labels horizontally and vertically. Positive values will move labels to right (hjust) and top (vjust). Negative values will move the labels to left and bottom, respectively.

size	The size of the text. Defaults to 0.9.
col	The color of the text. Defaults to "white".
...	Further arguments passed on to graphics::text().

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.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

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,
  threshold = "Otsu",
  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,
  threshold = "Otsu",
  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	• For object_coord(), a vector (or scalar) of object id to compute the bounding rectangle. Object ids can be obtained with object_id(). Set id = "all" to compute the coordinates for all objects in the image. If id = NULL (default) a bounding rectangle is drawn including all the objects. • For object_isolate(), a scalar that identifies the object to be extracted.
index	The index to produce a binary image used to compute bounding rectangle coordinates. See image_binary() 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.
threshold	By default (threshold = "Otsu"), 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.
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 analyze_objects() 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.
...	• For object_isolate(), further arguments passed on to object_coord(). • For object_id(), further arguments passed on to analyze_objects().

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_pick` *Picking up points in an image*

Description

- `pick_count()` opens an interactive section where the user will be able to click in the image to count objects (points) manually. In each mouse click, a point is drawn and an upward counter is shown in the console. After `n` counts or after the user press Esc, the interactive process is terminated and a data.frame with the `x` and `y` coordinates for each point is returned.
- `pick_palette()` creates an image palette by picking up color point(s) from the image.
- `pick_rgb()` Picks up the RGB values from selected point(s) in the image.

Usage

```
pick_count(
  image,
  n = Inf,
  col = "red",
  size = 0.8,
  plot = TRUE,
  verbose = TRUE
)
```

```
pick_rgb(image, n = Inf, col = "red", size = 0.8, plot = TRUE, verbose = TRUE)

pick_palette(
  image,
  n = Inf,
  r = 3,
  shape = "box",
  random = TRUE,
  width = 100,
  height = 100,
  col = "red",
  size = 0.8,
  plot = TRUE,
  palette = TRUE,
  verbose = TRUE
)
```

Arguments

image	An Image object.
n	The number of points of the pick_* function. Defaults to Inf. This means that picking will run until the user press Esc.
col, size	The color and size for the marker point.
plot	Call a new plot(image) before processing? Defaults to TRUE.
verbose	If TRUE (default) shows a counter in the console.
r	The radius of neighborhood pixels. Defaults to 3.
shape	A character vector indicating the shape of the brush around the selected pixel. It can be "box", "disc", "diamond", "Gaussian" or "line". Defaults to "box". In this case, if 'r = 1', all the 8 surrounding pixels are sampled. Setting to "disc" and increasing the radius (r) will select surrounding pixels towards the format of a sphere around the selected pixel.
random	Randomize the selected pixels? Defaults to TRUE.
width, height	The width and height of the generated palette. Defaults to 100 for both, i.e., a square image of 100 x 100.
palette	Plot the generated palette? Defaults to TRUE.

Value

- pick_count() returns data.frame with the x and y coordinates of the selected point(x).
- pick_rgb() returns a data.frame with the R, G, and B values of the selected point(s).
- pick_palette() returns an object of class Image.

Author(s)

Tiago Olivoto <tiagoolivoto@gmail.com>

Examples

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

# start a counting process
pick_count(img)

# get rgb from point(s)
pick_rgb(img)

# create a palette from point(s)
pick_palette(img)
}
```

`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

x, y	Coordinate vectors of points. This can be specified as two vectors (x and y), or a 2-column matrix x. If x is a list of vector coordinates the function will be applied to each element using base::lapply().
closed	If TRUE (default) returns the vector of points of a closed polygon, i.e., the first point is replicated as the last one.
vertices	The number of spline vertices to create.
k	The number of points to wrap around the ends to obtain a smooth periodic spline.
...	• For plot_contour() and plot_ellipse() further arguments passed on to graphics::lines(). • For plot_mass(), further arguments passed on to graphics::points().
id	The object identification (numeric) to plot the contour/ellipse. By default (id = NULL), the contour is plotted to all objects
col, lwd, cex	The color, width of the lines, and size of point, respectively.
arrow	If TRUE (default) plots two arrows connecting the center of mass to the minimum and maximum radius.
object	An object computed with analyze_objects().

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

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,
  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 image_index() for more details. Defaults to "NB" (normalized blue).
edge	• for image_autocrop() the number of pixels in the edge of the cropped image. If edge = 0 the image will be cropped to create a bounding rectangle (x and y coordinates) around the image objects. • for image_trim(), the number of pixels removed from the edges. By default, 20 pixels are removed from all the edges.
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	• For image_resize() 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 rel_size. • 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
angle	The rotation angle in degrees.
bg_col	Color used to fill the background pixels, defaults to "white".
rel_size	The relative size of the resized image. Defaults to 100. For example, setting rel_size = 50 to an image of width 1280 x 720, the new image will have a size of 640 x 360.
top, bottom, left, right	The number of pixels removed from top, bottom, left, and right when using image_trim().

kern	An Image object or an array, containing the structuring element. Defaults to a brushe generated with EBImage::makeBrush().
size	• For image_filter() is the median filter radius (integer). Defaults to 3. • For image_dilate() and image_erode() 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.
shape	A character vector indicating the shape of the brush. Can be box, disc, diamond, Gaussian or line. Default is disc.
...	Additional arguments passed on to image_binary().
cache	The the L2 cache size of the system CPU in kB (integer). Defaults to 512.
sigma	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