Package ‘colordistance’

March 20, 2021

Title Distance Metrics for Image Color Similarity

Date 2021-03-19

Version 1.1.2

Description Loads and displays images, selectively masks specified background colors, bins pixels by color using either data-dependent or automatically generated color bins, quantitatively measures color similarity among images using one of several distance metrics for comparing pixel color clusters, and clusters images by object color similarity. Uses CIELAB, RGB, or HSV color spaces. Originally written for use with organism coloration (reef fish color diversity, butterfly mimicry, etc), but easily applicable for any image set.

Imports jpeg, png, stats, clue, ape, mgcv, emdist, scatterplot3d, plotly, gplots, abind, magrittr, scales, qpdf, spatstat.geom

Depends R (>= 3.4.0)

License GPL-3

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Hannah Weller [aut, cre]

Maintainer Hannah Weller <hannahiweller@gmail.com>

Repository CRAN

Date/Publication 2021-03-20 20:40:23 UTC

R topics documented:

chisqDistance ....................................................... 2
colorDistance ....................................................... 3
chisqDistance

Chi-square distance between vectors

Description

Computes the chi-squared distance between each element of a pair of vectors which must be of the same length. Good for comparing color histograms if you don’t want to weight by color similarity. Probably hugely redundant; alas.

Usage

chisqDistance(a, b)

Arguments

a         Numeric vector.
b         Numeric vector; must be the same length as a.
Value

Chi-squared distance, \((a - b)^2 / (a + b)\), between vectors a and b. If one or both elements are NA/NaN, contribution is counted as a 0.

Examples

colordistance:::chisqDistance(rnorm(10), rnorm(10))

---

colorDistance

Sum of Euclidean distances between color clusters

Description

Calculates the Euclidean distance between each pair of points in two dataframes as returned by extractClusters or getImageHist and returns the sum of the distances.

Usage

colorDistance(T1, T2)

Arguments

T1
- Dataframe (especially a dataframe as returned by extractClusters() or getImageHist(), but first three columns must be coordinates).

T2
- Another dataframe like T1.

Value

Sum of Euclidean distances between each pair of points (rows) in the provided dataframes.

Examples

```r
## Not run: cluster.list <- colordistance::getHistList(system.file("extdata", "Heliconius/Heliconius_B", package="colordistance"), lower=rep(0.8, 3), upper=rep(1, 3))
colordistance:::colorDistance(cluster.list[[1]], cluster.list[[2]])
```

## End(Not run)
combineClusters  

*Average 3D color histograms by subdirectory*

**Description**

Calculates color histograms for images in immediate subdirectories of a folder, and averages histograms for images in the same subdirectory.

**Usage**

```
combineClusters(folder, method = "mean", ...)
```

**Arguments**

- **folder**  
  Path to the folder containing subdirectories of images. Must be a character vector.

- **method**  
  Method for combining color histograms. Default is "mean", but other generic functions ("median", "sum", etc) will work. String is evaluated using "eval" so any appropriate R function is accepted.

- **...**  
  Additional arguments passed to `getHistList`, including number of bins, HSV flag, etc.

**Examples**

```
combined_clusters <- colordistance::combineClusters(system.file("extdata", "Heliconius", package="colordistance"), method="median", bins=2, lower=rep(0.8, 3), upper=rep(1, 3))
```

combineList  

*Combine a list of cluster features into a single cluster set*

**Description**

Combine a list of cluster features as returned by `getHistList` according to the specified method.

**Usage**

```
combineList(hist_list, method = "mean")
```

**Arguments**

- **hist_list**  
  A list of cluster dataframes as returned by `getHistList`.

- **method**  
  Method for combining color histograms. Default is "mean", but other generic functions ("median", "sum", etc) will work. String is evaluated using "eval" so any appropriate R function is accepted.
Note

While the function can also accept clusters generated using kmeans (getKMeansList followed by extractClusters), this is not recommended, as kmeans does not provide explicit analogous pairs of clusters, and clusters are combined by row number (all row 1 clusters are treated as analogous, etc). Color histograms are appropriate because the bins are defined the same way for each image.

Examples

```r
hist_list <- getHistList(system.file("extdata", "Heliconius/Heliconius_A", package="colordistance"), lower=rep(0.8, 3), upper=rep(1, 3))
median_clusters <- combineList(hist_list, method="median")
```

convertColorSpace  
Convert between color spaces

Description

Wrapper for convertColor that builds in random sampling, error messages, and removes default illuminant (D65) to enforce manual specification of a reference white.

Usage

```r
convertColorSpace(
  color.coordinate.matrix,
  from = "sRGB",
  to = "Lab",
  sample.size = 1e+05,
  from.ref.white, 
  to.ref.white 
)
```

Arguments

color.coordinate.matrix
  A color coordinate matrix with rows as colors and channels as columns. If a color histogram (e.g. as returned by getImageHist) is passed, the 'Pct' column is ignored.

from, to
  Input and output color spaces, passed to convertColor. See details.

sample.size
  Number of pixels to be randomly sampled from filtered pixel array for conversion. If not numeric or larger than number of colors provided (i.e. cluster matrix), all colors are converted. See details.

from.ref.white, to.ref.white
  Reference whites passed to convertColor. Unlike convertColor, no default is provided. See details for explanation of different reference whites.
convertColorSpace

Details

Color spaces are all passed to `convertColor`, and can be any of: "XYZ", "sRGB", "Apple RGB", "CIE RGB", "Lab", or "Luv".

Lab and Luv color spaces are approximately perceptually uniform, meaning they usually do the best job of reflecting intuitive color distances without the non-linearity problems of more familiar RGB spaces. However, because they describe object colors, they require a reference 'white light' color (dimly and brightly lit photographs of the same object will have very different RGB palettes, but similar Lab palettes if appropriate white references are used). The idea here is that the apparent colors in an image depend not just on the "absolute" color of an object, but also on the available light in the scene. There are seven CIE standardized illuminants available in colordistance (A, B, C, E, and D50, D55, and D65), but the most common are:

- "A": Standard incandescent lightbulb
- "D65": Average daylight
- "D50": Direct sunlight

Color conversions will be highly dependent on the reference white used, which is why no default is provided. Users should look into standard illuminants to choose an appropriate reference for a dataset.

The conversion from RGB to a standardized color space (XYZ, Lab, or Luv) is approximate, non-linear, and relatively time-consuming. Converting a large number of pixels can be computationally expensive, so `convertColorSpace` will randomly sample a specified number of rows to reduce the time. The default sample size, 100,000 rows, takes about 5 seconds convert from sRGB to Lab space on an early 2015 Macbook with 8 GB of RAM. Time scales about linearly with number of rows converted.

Value

A 3- or 4-column matrix depending on whether `color.coordinate.matrix` included a 'Pct' column (as from `getImageHist`), with one column per channel.

Examples

```r
# Convert a single RGB triplet and then back convert it
rgb_color <- c(0, 1, 0)
lab_color <- colordistance::convertColorSpace(rgb_color,
   from="sRGB", to="Lab", to.ref.white="D65")
rgb_again <- colordistance::convertColorSpace(lab_color,
   from="Lab", to="sRGB", from.ref.white="D65")

# Convert pixels from loadImage() function
img <- colordistance::loadImage(system.file("extdata",
   "Heliconius/Heliconius_B/Heliconius_07.jpeg", package="colordistance"))
lab_pixels <- colordistance::convertColorSpace(img$filtered.rgb.2d,
   from="sRGB", to="XYZ", sample.size=5000)

# Convert clusters
img <- colordistance::loadImage(system.file("extdata",
   "Heliconius/Heliconius_B/Heliconius_07.jpeg", package="colordistance"))
```
EMDistance

Earth mover’s distance between two sets of color clusters

Description

Calculates the Earth mover’s distance (briefly, the amount of work required to move the data from one distribution to resemble the other distribution, or the amount of "dirt" you have to shovel weighted by how far you have to shovel it). Accounts for both color disparity and size disparity. Recommended unless `binAvg` is off for histogram generation.

Usage

EMDistance(T1, T2)

Arguments

T1 Dataframe (especially a dataframe as returned by `extractClusters` or `getImageHist`, but first three columns must be coordinates).

T2 Another dataframe like T1.

Value

Earth mover’s distance between the two dataframes (metric of overall bin similarity for a pair of 3-dimensional histograms).

Examples

```r
## Not run:
cluster.list <- colordistance::getHistList(system.file("extdata", "Heliconius/Heliconius_B", package="colordistance"), lower=rep(0.8, 3), upper=rep(1, 3))
colordistance::EMDistance(cluster.list[[1]], cluster.list[[2]])

## End(Not run)
```
exportTree

Export a distance matrix as a tree object

Description

Converts a symmetrical distance matrix to a tree and saves it in newick format. Uses hclust to form clusters.

Usage

exportTree(getColorDistanceMatrixObject, file, return.tree = FALSE)

Arguments

getColorDistanceMatrixObject

A distance matrix, especially as returned by getColorDistanceMatrix, but any numeric symmetrical matrix will work.

df

Character vector of desired filename for saving tree. Should end in ".newick".

return.tree

Logical. Should the tree object be returned to the working environment in addition to being saved as a file?

Value

Newick tree saved in specified location and as.phylo tree object if return.tree=TRUE.

Examples

```r
# Not run:
clusterList <- colordistance::getHistList(dir(system.file("extdata", "Heliconius/", package="colordistance"), full.names=TRUE), lower=rep(0.8, 3), upper=rep(1, 3))
CDM <- colordistance::getColorDistanceMatrix(clusterList, method="emd", plotting=FALSE)

# Tree is both saved in current working directory and stored in
# heliconius_tree variable
heliconius_tree <- colordistance::exportTree(CDM, "./HeliconiusColorTree.newick", return.tree=TRUE)
```

## End(Not run)
extractClusters

Extract cluster values and sizes from kmeans fit objects

Description

Extract a list of dataframes with the same format as those returned by getHistList, where each dataframe has 3 color attributes (R, G, B or H, S, V) and a size attribute (Pct) for every cluster.

Usage

extractClusters(getKMeansListObject, ordering = TRUE, normalize = FALSE)

Arguments

getKMeansListObject
  A list of kmeans fit objects (especially as returned by getKMeansList).

ordering
  Logical. Should clusters be reordered by color similarity? If TRUE, the Hungarian algorithm via solve_LSAP is applied to find the minimum sum of Euclidean distances between color pairs for every pair of cluster objects and colors are reordered accordingly.

normalize
  Logical. Should each cluster be normalized to show R:G:B or H:S:V ratios rather than absolute values? Can be helpful for inconsistent lighting, but reduces variation. See normalizeRGB.

Value

A list of dataframes (same length as input list), each with 4 columns: R, G, B (or H, S, V) and Pct (cluster size), with one row per cluster.

Note

Names are inherited from the list passed to the function.

Examples

clusterList <- colordistance::getKMeansList(system.file("extdata", "Heliconius/Heliconius_A", package="colordistance"), bins=3)

colordistance::extractClusters(clusterList)
getColorDistanceMatrix

Distance matrix for a list of color cluster sets

Description

Calculates a distance matrix for a list of color cluster sets as returned by `extractClusters` or `getHistList` based on the specified distance metric.

Usage

```r
getColorDistanceMatrix(
  cluster.list,
  method = "emd",
  ordering = "default",
  size.weight = 0.5,
  color.weight = 0.5,
  plotting = TRUE,
  ...
)
```

Arguments

- **cluster.list**: A list of identically sized dataframes with 4 columns each (R, G, B, Pct or H, S, V, Pct) as output by `extractClusters` or `getHistList`.
- **method**: One of four possible comparison methods for calculating the color distances: "emd" (uses `EMDistance`, recommended), "chisq" (uses `chisqDistance`), "color.dist" (uses `colorDistance`; not appropriate if binAvg=F), or "weighted.pairs" (uses `weightedPairsDistance`).
- **ordering**: Logical if not left as "default". Should the color clusters in the list be reordered to minimize the distances between the pairs? If left as default, ordering depends on distance method: "emd" and "chisq" do not order clusters ("emd" orders on a case-by-case in the `EMDistance` function itself and reordering by size similarity would make chi-squared meaningless); "color.dist" and "weighted.pairs" use ordering. To override defaults, set to either `T` (for ordering) or `F` (for no ordering).
- **size.weight**: Same as in `weightedPairsDistance`.
- **color.weight**: Same as in `weightedPairsDistance`.
- **plotting**: Logical. Should a heatmap of the distance matrix be displayed once the function finishes running?
- **...**: Additional arguments passed on to `heatmapColorDistance`.
**getColorDistanceMatrix**

**Details**

Each cell represents the distance between a pair of color cluster sets as measured using either chi-squared distance (cluster size only), earth mover’s distance (size and color), weighted pairs (size and color with user-specified weights for each), or color distance (Euclidean distance between clusters as 3-dimensional - RGB or HSV - color coordinates).

Earth mover’s distance is recommended unless `binAvg` is set to false during cluster list generation (in which case all paired bins will have the same colors across datasets), in which case chi-squared is recommended. Weighted pairs or color distance may be appropriate depending on the question, but generally give poorer results.

**Value**

A distance matrix of image distance scores (the scales vary depending on the distance metric chosen, but for all four methods, higher scores = more different).

**Examples**

```r
## Not run:
cluster.list <- colordistance::getHistList(c(system.file("extdata", "Heliconius/Heliconius_A", package="colordistance"), system.file("extdata", "Heliconius/Heliconius_B", package="colordistance")), lower=rep(0.8, 3), upper=rep(1, 3))
# Default values - recommended!
colordistance::getColorDistanceMatrix(cluster.list, main="EMD")
# Without plotting
colordistance::getColorDistanceMatrix(cluster.list, plotting=FALSE)
# Use chi-squared instead
colordistance::getColorDistanceMatrix(cluster.list, method="chisq", main="Chi-squared")
# Override ordering (throws a warning if you're trying to do this with # chisq!)
colordistance::getColorDistanceMatrix(cluster.list, method="chisq", ordering=TRUE, main="Chi-squared w/ ordering")
# Specify high size weight/low color weight for weighted pairs
colordistance::getColorDistanceMatrix(cluster.list, method="weighted.pairs", color.weight=0.1, size.weight=0.9, main="Weighted pairs")
# Color distance only
colordistance::getColorDistanceMatrix(cluster.list, method="color.dist", ordering=TRUE, main="Color distance only")
## End(Not run)
```
**getHistColors**  
*Vector of hex colors for histogram bin coloration*

**Description**

Gets a vector of colors for plotting histograms from `getImageHist` in helpful ways.

**Usage**

```r
getHistColors(bins, hsv = FALSE)
```

**Arguments**

- `bins` Number of bins for each channel OR a vector of length 3 with bins for each channel. Bins = 3 will result in 3^3 = 27 bins; bins = c(2, 2, 3) will result in 2 * 2 * 3 = 12 bins (2 red, 2 green, 3 blue), etc.
- `hsv` Logical. Should HSV be used instead of RGB?

**Value**

A vector of hex codes for bin colors.

**Examples**

```r
colordistance:::getHistColors(bins = 3)
colordistance:::getHistColors(bins = c(8, 3, 3), hsv = TRUE)
```

---

**getHistList**  
*Generate a list of cluster sets for multiple images*

**Description**

Applies `getImageHist` to every image in a provided set of image paths and/or directories containing images.

**Usage**

```r
ggetHistList(
  images,
  bins = 3,
  bin.avg = TRUE,
  lower = c(0, 0.55, 0),
  upper = c(0.24, 1, 0.24),
  alpha.channel = TRUE,
  norm.pix = FALSE,
  plotting = FALSE,
```
getHistList

pausing = TRUE,
hsb = FALSE,
title = "path",
img.type = FALSE,
bounds = c(0, 1)
)

Arguments

images Character vector of directories, image paths, or both.

bins Number of bins for each channel OR a vector of length 3 with bins for each channel. Bins=3 will result in 3^3 = 27 bins; bins=c(2, 2, 3) will result in 2*2*3=12 bins (2 red, 2 green, 3 blue), etc.

bin.avg Logical. Should the returned color clusters be the average of the pixels in that bin (bin.avg=TRUE) or the center of the bin (FALSE)? If a bin is empty, the center of the bin is returned as the cluster color regardless.

lower RGB or HSV triplet specifying the lower bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]).

upper RGB or HSV triplet specifying the upper bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]). Determining these bounds may take some trial and error, but the following bounds may work for certain common background colors:

- Black: lower=c(0, 0, 0); upper=c(0.1, 0.1, 0.1)
- White: lower=c(0.8, 0.8, 0.8); upper=c(1, 1, 1)
- Green: lower=c(0, 0.55, 0); upper=c(0.24, 1, 0.24)
- Blue: lower=c(0, 0, 0.55); upper=c(0.24, 0.24, 1)

If no background filtering is needed, set bounds to some non-numeric value (NULL, FALSE, "off", etc); any non-numeric value is interpreted as NULL.

alpha.channel Logical. If available, should alpha channel transparency be used to mask background? See removeBackground for more details.

norm.pix Logical. Should RGB or HSV cluster values be normalized using normalizeRGB?

plotting Logical. Should the histogram generated for each image be displayed?

pausing Logical. If plotting=TRUE, should the function pause between graphing and wait for user to hit [enter] before continuing? Useful for data/histogram inspection.

hsb Logical. Should HSV be used instead of RGB?

title String for what the title the plots if plotting is on; defaults to the image name.

img.type Logical. Should the file extension for the images be retained when naming the output list elements? If FALSE, just the image name is used (so "Heliconius_01.png" becomes "Heliconius_01").

bounds Upper and lower limits for the channels; R reads in images with intensities on a 0-1 scale, but 0-255 is common.
Value

A list of `getImageHist` dataframes, 1 per image, named by image name.

Note

For every image, the pixels are binned according to the specified bin breaks. By providing the bounds for the bins rather than letting an algorithm select centers (as in `getKMeansList`), clusters of nearly redundant colors are avoided. So you don’t end up with, say, 3 nearly-identical yellow clusters which are treated as unrelated just because there’s a lot of yellow in your image; you just get a very large yellow cluster and empty non-yellow bins.

Examples

```r
## Not run:
# Takes >10 seconds if you run all examples
clusterList <- colordistance::getHistList(system.file("extdata", "Heliconius/Heliconius_B", package="colordistance"), upper = rep(1, 3), lower = rep(0.8, 3))

clusterList <- colordistance::getHistList(c(system.file("extdata", "Heliconius/Heliconius_B", package="colordistance"), system.file("extdata", "Heliconius/Heliconius_A", package="colordistance")), pausing = FALSE, upper = rep(1, 3), lower = rep(0.8, 3))

clusterList <- colordistance::getHistList(system.file("extdata", "Heliconius/Heliconius_B", package = "colordistance"), plotting = TRUE, upper = rep(1, 3), lower = rep(0.8, 3))

## End(Not run)
```

---

`getImageHist`  
*Generate a 3D histogram based on color distribution in an image*

Description

Computes a histogram in either RGB or HSV colorspace by sorting pixels into a specified number of bins.

Usage

```r
getImageHist(
  image,
  bins = 3,
  bin.avg = TRUE,
  defaultClusters = NULL,
  lower = c(0, 0.55, 0),
)```

getImageHist

```r
upper = c(0.24, 1, 0.24),
as.vec = FALSE,
alpha.channel = TRUE,
norm.pix = FALSE,
plotting = TRUE,
hsv = FALSE,
title = "path",
bounds = c(0, 1),
...)
```

**Arguments**

- **image**: Path to a valid image (PNG or JPG) or a `loadImage` object.
- **bins**: Number of bins for each channel OR a vector of length 3 with bins for each channel. Bins=3 will result in $3^3 = 27$ bins; bins=c(2, 2, 3) will result in $2*2*3=12$ bins (2 red, 2 green, 3 blue), etc.
- **bin.avg**: Logical. Should the returned color clusters be the average of the pixels in that bin (bin.avg=TRUE) or the center of the bin (FALSE)? If a bin is empty, the center of the bin is returned as the cluster color regardless.
- **defaultClusters**: Optional dataframe of default color clusters to be returned when a bin is empty. If NULL, the geometric centers of the bins are used.
- **lower**: RGB or HSV triplet specifying the lower bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]).
- **upper**: RGB or HSV triplet specifying the upper bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]). Determining these bounds may take some trial and error, but the following bounds may work for certain common background colors:
  - Black: lower=c(0, 0, 0); upper=c(0.1, 0.1, 0.1)
  - White: lower=c(0.8, 0.8, 0.8); upper=c(1, 1, 1)
  - Green: lower=c(0, 0.55, 0); upper=c(0.24, 1, 0.24)
  - Blue: lower=c(0, 0, 0.55); upper=c(0.24, 0.24, 1)
If no background filtering is needed, set bounds to some non-numeric value (NULL, FALSE, “off”, etc); any non-numeric value is interpreted as NULL.
- **as.vec**: Logical. Should the bin sizes just be returned as a vector? Much faster if only using `chisqDistance` for comparison metric.
- **alpha.channel**: Logical. If available, should alpha channel transparency be used to mask background? See `removeBackground` for more details.
- **norm.pix**: Logical. Should RGB or HSV cluster values be normalized using `normalizeRGB`?
- **plotting**: Logical. Should a histogram of the bin colors and sizes be plotted?
- **hsv**: Logical. Should HSV be used instead of RGB?
getImagePaths

getStringPaths

Fetch paths to all valid images in a given directory

Description

Find all valid image paths (PNG and JPG) in a directory (does not search subdirectories). Will recover any image ending in .PNG, JPG, or JPEG, case-insensitive.

Usage

getImagePaths(path)
getKMeanColors

Arguments

path Path to directory in which to search for images. Absolute or relative filepaths are fine.

Value

A vector of absolute filepaths to JPG and PNG images in the given directory.

Note

In the event that no compatible images are found in the directory, it returns a message to that effect instead of an empty vector.

Examples

im.dir <- colordistance::getImagePaths(system.file("extdata", "Heliconius/Heliconius_A", package="colordistance"))
## Not run:
im.dir <- colordistance::getImagePaths("some/nonexistent/directory")
## End(Not run)
im.dir <- colordistance::getImagePaths(getwd())

documented
getKMeanColors

Arguments

path
Path to an image (JPG or PNG).

n
Number of KMeans clusters to fit. Unlike `getImageHist`, this represents the actual final number of bins, rather than the number of breaks in each channel.

sample.size
Number of pixels to be randomly sampled from filtered pixel array for performing fit. If set to `FALSE`, all pixels are fit, but this can be time-consuming, especially for large images.

plotting
Logical. Should the results of the KMeans fit (original image + histogram of colors and bin sizes) be plotted?

lower
RGB triplet specifying the lower bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]).

upper
RGB triplet specifying the upper bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]). Determining these bounds may take some trial and error, but the following bounds may work for certain common background colors:

- **Black**: `lower=c(0, 0, 0); upper=c(0.1, 0.1, 0.1)`
- **White**: `lower=c(0.8, 0.8, 0.8); upper=c(1, 1, 1)`
- **Green**: `lower=c(0, 0.55, 0); upper=c(0.24, 1, 0.24)`
- **Blue**: `lower=c(0, 0, 0.55); upper=c(0.24, 0.24, 1)`

If no background filtering is needed, set bounds to some non-numeric value (NULL, FALSE, "off", etc); any non-numeric value is interpreted as NULL.

iter.max
Inherited from `kmeans`. The maximum number of iterations allowed.

nstart
Inherited from `kmeans`. How many random sets should be chosen?

return.clust
Logical. Should clusters be returned? If `FALSE`, results are plotted but not returned.

color.space
The color space ("rgb", "hsv", or "lab") in which to cluster pixels.

from
Display color space of image if clustering in CIE Lab space, probably either "sRGB" or "Apple RGB", depending on your computer.

ref.white
The reference white passed to `convertColorSpace`; must be specified if using CIE Lab space. See `convertColorSpace`.

Value

A `kmeans` fit object.

Examples

```r
colordistance::getKMeanColors(system.file("extdata", 
"Heliconius/Heliconius_B/Heliconius_07.jpeg", package="colordistance"), n=3, 
return.clust=FALSE, lower=rep(0.8, 3), upper=rep(1, 3))
```
**getKMeansList**  

*Get KMeans clusters for every image in a set*

**Description**

Performs `getKMeanColors` on every image in a set of images and returns a list of kmeans fit objects, where each dataframe contains the RGB coordinates of the clusters and the percentage of pixels in the image assigned to that cluster.

**Usage**

```r
getKMeansList(
  images,
  bins = 10,
  sample.size = 20000,
  plotting = FALSE,
  lower = c(0, 0.55, 0),
  upper = c(0.24, 1, 0.24),
  iter.max = 50,
  nstart = 5,
  img.type = FALSE,
  color.space = "rgb",
  from = "sRGB",
  ref.white
)
```

**Arguments**

- `images`: A character vector of directories, image paths, or a combination of both. Takes either absolute or relative filepaths.
- `bins`: Number of KMeans clusters to fit. Unlike `getImageHist`, this represents the actual final number of bins, rather than the number of breaks in each channel.
- `sample.size`: Number of pixels to be randomly sampled from filtered pixel array for performing fit. If set to TRUE, all pixels are fit, but this can be time-consuming, especially for large images.
- `plotting`: Logical. Should the results of the KMeans fit (original image + histogram of colors and bin sizes) be plotted for each image?
- `lower`: RGB triplet specifying the lower bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]).
- `upper`: RGB triplet specifying the upper bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]). Determining these bounds may take some trial and error, but the following bounds may work for certain common background colors:
  - Black: lower=c(0, 0, 0); upper=c(0.1, 0.1, 0.1)
getLabHist

Generate a 3D histogram based on CIE Lab color coordinates in an image

Description

Computes a histogram in CIE Lab color space by sorting pixels into specified bins.

Usage

getLabHist(
  image,
  bins = 3,
  sample.size = 10000,
  ref.white,
  from = "sRGB",
  lower = c(0.8, 0.8, 0.8),
  upper = c(1, 1, 1)
)

If no background filtering is needed, set bounds to some non-numeric value (NULL, FALSE, "off", etc); any non-numeric value is interpreted as NULL.

Value

A list of kmeans fit objects, where the list element names are the original image names.

Examples

```r
## Not run:
# Takes a few seconds to run
kmeans_list <- colordistance::getKMeansList(dir(system.file("extdata",
  "Heliconius/", package="colordistance"), full.names=TRUE), bins=3,
  lower=rep(0.8, 3), upper=rep(1, 3), plotting=TRUE)
## End(Not run)
```

### Table: getLabHist Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter.max</td>
<td>Inherited from kmeans. The maximum number of iterations allowed.</td>
</tr>
<tr>
<td>nstart</td>
<td>Inherited from kmeans. How many random sets should be chosen?</td>
</tr>
<tr>
<td>img.type</td>
<td>Logical. Should the image extension (.PNG or .JPG) be retained in the list names?</td>
</tr>
<tr>
<td>color.space</td>
<td>The color space (&quot;rgb&quot;, &quot;hsv&quot;, or &quot;lab&quot;) in which to cluster pixels.</td>
</tr>
<tr>
<td>from</td>
<td>Original color space of images if clustering in CIE Lab space, probably either &quot;sRGB&quot; or &quot;Apple RGB&quot;, depending on your computer.</td>
</tr>
<tr>
<td>ref.white</td>
<td>The reference white passed to convertColorSpace; must be specified if using CIE Lab space. See convertColorSpace.</td>
</tr>
</tbody>
</table>
getLabHist

bin.avg = TRUE,
alpha.channel = TRUE,
as.vec = FALSE,
plotting = TRUE,
lower = c(0, 0.55, 0),
upper = c(0.24, 1, 0.24),
title = "path",
a.bounds = c(-128, 127),
b.bounds = c(-128, 127),
...
)

Arguments

image Path to a valid image (PNG or JPG) or a loadImage object.
bins Number of bins for each channel OR a vector of length 3 with bins for each channel. Bins = 3 will result in 3^3 = 27 bins; bins = c(2, 2, 3) will result in 2 * 2 * 3 = 12 bins (2 L, 2 a, 3 b), etc.
sample.size Numeric. How many pixels should be randomly sampled from the non-background part of the image and converted into CIE Lab coordinates? If non-numeric, all pixels will be converted, but this can be very slow (see details).
ref.white Reference white passed to convertColorSpace. Unlike convertColor, no default is provided. See details for explanation of different reference whites.
from Original color space of image, probably either "sRGB" or "Apple RGB", depending on your computer.
bin.avg Logical. Should the returned color clusters be the average of the pixels in that bin (bin.avg=TRUE) or the center of the bin (FALSE)? If a bin is empty, the center of the bin is returned as the cluster color regardless.
alpha.channel Logical. If available, should alpha channel transparency be used to mask background? See removeBackground for more details.
as.vec Logical. Should the bin sizes just be returned as a vector? Much faster if only using chisqDistance for comparison metric.
plotting Logical. Should a histogram of the bin colors and sizes be plotted?
lower, upper RGB or HSV triplets specifying the lower and upper bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]). Determining these bounds may take some trial and error, but the following bounds may work for certain common background colors:
- Black: lower=c(0, 0, 0); upper=c(0.1, 0.1, 0.1)
- White: lower=c(0.8, 0.8, 0.8); upper=c(1, 1, 1)
- Green: lower=c(0.55, 0); upper=c(0.24, 1, 0.24)
- Blue: lower=c(0, 0, 0.55); upper=c(0.24, 0.24, 1)

If no background filtering is needed, set bounds to some non-numeric value (NULL, FALSE, "off", etc); any non-numeric value is interpreted as NULL.
title String for what the title the plot if plotting is on; defaults to the image name.
getLabHist

a.bounds, b.bounds

Numeric ranges for the a (green-red) and b (blue-yellow) channels of Lab color space. Technically, a and b have infinite range, but in practice nearly all values fall between -128 and 127 (the default). Many images will have an even narrower range than this, depending on the lighting conditions and conversion; setting narrower ranges will result in finer-scale binning, without generating empty bins at the edges of the channels.

... Additional arguments passed to barplot.

Details

getLabHist uses convertColorSpace to convert pixels into CIE Lab coordinates, which requires a references white. There are seven CIE standardized illuminants available in colordistance (A, B, C, E, and D50, D55, and D65), but the most common are:

- "A": Standard incandescent lightbulb
- "D65": Average daylight
- "D50": Direct sunlight

Color conversions will be highly dependent on the reference white used, which is why no default is provided. Users should look into standard illuminants to choose an appropriate reference for a dataset.

The conversion from RGB to a standardized color space (XYZ, Lab, or Luv) is approximate, non-linear, and relatively time-consuming. Converting a large number of pixels can be computationally expensive, so convertColorSpace will randomly sample a specified number of rows to reduce the time. The default sample size, 10,000 rows, takes about 1 second to convert from sRGB to Lab space on an early 2015 Macbook with 8 GB of RAM. Time scales about linearly with number of rows converted.

Unlike RGB or HSV color spaces, the three channels of CIE Lab color space do not all range between 0 and 1; instead, L (luminance) is always between 0 and 100, and the a (green-red) and b (blue-yellow) channels generally vary between -128 and 127, but usually occupy a narrower range depending on the reference white. To achieve the best results, ranges for a and b should be restricted to avoid generating empty bins.

Value

A vector or dataframe (depending on whether as.vec = TRUE) of bin sizes and color coordinates.

Examples

```r
path <- system.file("extdata", "Heliconius/Heliconius_B/Heliconius_07.jpeg", package="colordistance")
getLabHist(path, ref.white = "D65", bins = c(2, 3, 3), lower = rep(0.8, 3), upper = rep(1, 3), sample.size = 1000, ylim = c(0, 1))
```
**getLabHistList**

Generate a list of cluster sets in CIE Lab color space

**Description**

Applies `getLabHist` to every image in a provided set of image paths and/or directories containing images.

**Usage**

```r
getLabHistList(
    images,
    bins = 3,
    sample.size = 10000,
    ref.white,
    from = "sRGB",
    bin.avg = TRUE,
    as.vec = FALSE,
    plotting = FALSE,
    pausing = TRUE,
    lower = c(0, 0.55, 0),
    upper = c(0.24, 1, 0.24),
    alpha.channel = TRUE,
    title = "path",
    a.bounds = c(-128, 127),
    b.bounds = c(-128, 127),
    ...
)
```

**Arguments**

- **images** Character vector of directories, image paths, or both.
- **bins** Number of bins for each channel OR a vector of length 3 with bins for each channel. Bins = 3 will result in \(3^3 = 27\) bins; bins = \(c(2, 2, 3)\) will result in \(2 \times 2 \times 3 = 12\) bins (2 L, 2 a, 3 b), etc.
- **sample.size** Numeric. How many pixels should be randomly sampled from the non-background part of the image and converted into CIE Lab coordinates? If non-numeric, all pixels will be converted, but this can be very slow (see details).
- **ref.white** Reference white passed to `convertColorSpace`. Unlike `convertColor`, no default is provided. See details for explanation of different reference whites.
- **from** Original color space of image, probably either "sRGB" or "Apple RGB", depending on your computer.
- **bin.avg** Logical. Should the returned color clusters be the average of the pixels in that bin (bin.avg=TRUE) or the center of the bin (FALSE)? If a bin is empty, the center of the bin is returned as the cluster color regardless.
getLabHistList

as.vec Logical. Should the bin sizes just be returned as a vector? Much faster if only using \texttt{chisqDistance} for comparison metric.

plotting Logical. Should a histogram of the bin colors and sizes be plotted?

pausing Logical. If \texttt{plotting=T}, should the function pause between graphing and wait for user to hit [enter] before continuing? Useful for data/histogram inspection.

lower, upper RGB or HSV triplets specifying the lower and upper bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]). Determining these bounds may take some trial and error, but the following bounds may work for certain common background colors:

- Black: lower=c(0, 0, 0); upper=c(0.1, 0.1, 0.1)
- White: lower=c(0.8, 0.8, 0.8); upper=c(1, 1, 1)
- Green: lower=c(0, 0.55, 0); upper=c(0.24, 1, 0.24)
- Blue: lower=c(0, 0, 0.55); upper=c(0.24, 0.24, 1)

If no background filtering is needed, set bounds to some non-numeric value (\texttt{NULL}, FALSE, "off", etc); any non-numeric value is interpreted as \texttt{NULL}.

alpha.channel Logical. If available, should alpha channel transparency be used to mask background? See \texttt{removeBackground} for more details.

title String for what the title the plot if plotting is on; defaults to the image name.

a.bounds, b.bounds Numeric ranges for the a (green-red) and b (blue-yellow) channels of Lab color space. Technically, a and b have infinite range, but in practice nearly all values fall between -128 and 127 (the default). Many images will have an even narrower range than this, depending on the lighting conditions and conversion; setting narrower ranges will result in finer-scale binning, without generating empty bins at the edges of the channels.

... Additional arguments passed to \texttt{barplot}.

Details

getLabHist uses \texttt{convertColorSpace} to convert pixels into CIE Lab coordinates, which requires a references white. There are seven CIE standardized illuminants available in \texttt{colordistance} (A, B, C, E, and D50, D55, and D65), but the most common are:

- "A": Standard incandescent lightbulb
- "D65": Average daylight
- "D50": Direct sunlight

Color conversions will be highly dependent on the reference white used, which is why no default is provided. Users should look into \texttt{standard illuminants} to choose an appropriate reference for a dataset.

Unlike RGB or HSV color spaces, the three channels of CIE Lab color space do not all range between 0 and 1; instead, L (luminance) is always between 0 and 100, and the a (green-red) and b (blue-yellow) channels generally vary between -128 and 127, but usually occupy a narrower range depending on the reference white. The exception is reference white A (standard incandescent lighting), which tends to have lower values when converting with \texttt{convertColor}.
heatmapColorDistance

Value

A list of `getLabHist` dataframes, 1 per image, named by image name.

Examples

```r
images <- system.file("extdata", "Heliconius/Heliconius_B", package="colordistance")

colordistance::getLabHistList(images, bins = 2, sample.size = 1000, ref.white = "D65", plotting = TRUE, pausing = FALSE, lower = rep(0.8, 3), upper = rep(1, 3), a.bounds = c(-100, 100), b.bounds = c(-127, 100), ylim = c(0, 1))
```

Description

Plots a heatmap of a symmetrical distance matrix in order to visualize similarity/dissimilarity in scores. Values are clustered by similarity using `hclust`.

Usage

```r
heatmapColorDistance(
  clusterList_or_matrixObject,
  main = NULL,
  col = "default",
  margins = c(6, 8),
  ...
)
```

Arguments

- `clusterList_or_matrixObject`: Either a list of identically sized dataframes with 4 columns each (3 color channels + Pct) as output by `extractClusters` or `getHistList`, or a symmetrical distance matrix as output by `getColorDistanceMatrix`.
- `main`: Title for heatmap plot.
- `col`: Color scale for heatmap from low to high. Default is `colorRampPalette(c("royalblue4","ghostwhite","violetred2"))(299)`, where pink is more dissimilar and blue is more similar.
- `margins`: Margins for column and row labels.
- `...`: Additional arguments passed on to `heatmap.2`.

Value

Heatmap representation of distance matrix.
imageClusterPipeline

Generate and plot a color distance matrix from a set of images

Description

Takes images, computes color clusters for each image, and calculates distance matrix/dendrogram from those clusters.

Usage

imageClusterPipeline(
  images,
  cluster.method = "hist",
  distance.method = "emd",
  lower = c(0, 140/255, 0),
  upper = c(60/255, 1, 60/255),
  hist.bins = 3,
  kmeans.bins = 27,
  bin.avg = TRUE,
  norm.pix = FALSE,
  plot.bins = FALSE,
  pausing = TRUE,
  color.space = "rgb",
  ref.white,
  from = "sRGB",
  bounds = c(0, 1),
  sample.size = 20000,
  iter.max = 50,
  nstart = 5,
  img.type = FALSE,
  ordering = "default",
  size.weight = 0.5,
color.weight = 0.5,
plot.heatmap = TRUE,
return.distance.matrix = TRUE,
save.tree = FALSE,
save.distance.matrix = FALSE,
a.bounds = c(-127, 128),
b.bounds = c(-127, 128)
)

Arguments

images Character vector of directories, image paths, or both.
cluster.method Which method for getting color clusters from each image should be used? Must be either "hist" (predetermined bins generated by dividing each channel with equidistant bounds; calls getHistList) or "kmeans" (determine clusters using kmeans fitting on pixels; calls getKMeansList).
distance.method One of four possible comparison methods for calculating the color distances: "emd" (uses EMDistance, recommended), "chisq" (uses chisqDistance), "color.dist" (uses colorDistance; not appropriate if bin.avg=F), or "weighted.pairs" (weightedPairsDistance).
lower RGB or HSV triplet specifying the lower bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]).
upper RGB or HSV triplet specifying the upper bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]). Determining these bounds may take some trial and error, but the following bounds may work for certain common background colors:
  • Black: lower=c(0, 0, 0); upper=c(0.1, 0.1, 0.1)
  • White: lower=c(0.8, 0.8, 0.8); upper=c(1, 1, 1)
  • Green: lower=c(0, 0.55, 0); upper=c(0.24, 1, 0.24)
  • Blue: lower=c(0, 0, 0.55); upper=c(0.24, 0.24, 1)
If no background filtering is needed, set bounds to some non-numeric value (NULL, FALSE, "off", etc); any non-numeric value is interpreted as NULL.
hist.bins Only applicable if cluster.method="hist". Number of bins for each channel OR a vector of length 3 with bins for each channel. Bins=3 will result in 3^3 = 27 bins; bins=c(2, 2, 3) will result in 2*2*3=12 bins (2 red, 2 green, 3 blue), etc. Passed to getHistList.
kmeans.bins Only applicable if cluster.method="kmeans". Number of KMeans clusters to fit. Unlike getImageHist, this represents the actual final number of bins, rather than the number of breaks in each channel.
bin.avg Logical. Should the color clusters used for the distance matrix be the average of the pixels in that bin (bin.avg=TRUE) or the center of the bin (FALSE)? If a bin is empty, the center of the bin is returned as the cluster color regardless. Only applicable if cluster.method="hist", since kmeans clusters are at the center of their assigned pixel clouds by definition.
norm.pix Logical. Should RGB or HSV cluster values be normalized using normalizeRGB?
plot.bins Logical. Should the bins for each image be plotted as they are calculated?
pausing Logical. If plot.bins=TRUE, pause and wait for user keystroke before plotting bins for next image?
color.space The color space ("rgb", "hsv", or "lab") in which to plot pixels.
ref.white The reference white passed to convertColorSpace; must be specified if using color.space = "lab".
from Display color space of image if clustering in CIE Lab space, probably either "sRGB" or "Apple RGB", depending on your computer.
bounds Upper and lower limits for the channels; R reads in images with intensities on a 0-1 scale, but 0-255 is common.
sample.size Only applicable if cluster.method="kmeans". Number of pixels to be randomly sampled from filtered pixel array for performing fit. If set to FALSE, all pixels are fit, but this can be time-consuming, especially for large images. Passed to getKMeansList.
iter.max Only applicable if cluster.method="kmeans". Inherited from kmeans. The maximum number of iterations allowed during kmeans fitting. Passed to getKMeansList.
nstart Only applicable if cluster.method="kmeans". Inherited from kmeans. How many random sets should be chosen? Passed to getKMeansList.
img.type Logical. Should file extensions be retained with labels?
ordering Logical if not left as "default". Should the color clusters in the list be reordered to minimize the distances between the pairs? If left as default, ordering depends on distance method: "emd" and "chisq" do not order clusters ("emd" orders on a case-by-case in the EMDistance function itself and reordering by size similarity would make chi-squared meaningless); "color.dist" and "weighted.pairs" use ordering. To override defaults, set to either T (for ordering) or F (for no ordering).
size.weight Weight of size similarity in determining overall score and ordering (if ordering=T).
color.weight Weight of color similarity in determining overall score and ordering (if ordering=T). Color and size weights do not necessarily have to sum to 1.
plot.heatmap Logical. Should a heatmap of the distance matrix be plotted?
return.distance.matrix Logical. Should the distance matrix be returned to the R environment or just plotted?
save.tree Either logical or a filepath for saving the tree; default if set to TRUE is to save in current working directory as "ColorTree.newick".
save.distance.matrix Either logical or filepath for saving distance matrix; default if set to TRUE is to save in current working directory as "ColorDistanceMatrix.csv".
a.bounds, b.bounds Passed to getLabHistList. Numeric ranges for the a (green-red) and b (blue-yellow) channels of Lab color space. Technically, a and b have infinite range, but in practice nearly all values fall between -128 and 127 (the default). Many images will have an even narrower range than this, depending on the lighting conditions and conversion; setting narrower ranges will result in finer-scale binning, without generating empty bins at the edges of the channels.
loadImage

Import image and generate filtered 2D pixel array(s)

Description

Imports a single image and returns a list with the original image as a 3D array, a 2D matrix with background pixels removed, and the absolute path to the original image.

Usage

loadImage(
  path,
  lower = c(0, 0.55, 0),
  upper = c(0.24, 1, 0.24),
  hsv = TRUE,
  CIELab = FALSE,
  sample.size = 1e+05,
  ref.white = NULL,
  alpha.channel = TRUE,
  alpha.message = FALSE
)

Value

Color distance matrix, heatmap, and saved distance matrix and tree files if saving is TRUE.

Note

This is the fastest way to get a distance matrix for color similarity starting from a folder of images. Essentially, it just calls in a series of other package functions in order: input images -> getImagePaths -> getHistList or getKMeansList followed by extractClusters -> getDistanceMatrix -> plotting -> return/save distance matrix. Sort of railroads you, but good for testing different combinations of clustering methods and distance metrics.

Examples

## Not run:
colordistance::imageClusterPipeline(dir(system.file("extdata", "Heliconius/", package="colordistance"), full.names=TRUE), color.space="hsv", lower=rep(0.8, 3), upper=rep(1, 3), cluster.method="hist", distance.method="emd", hist.bins=3, plot.bins=TRUE, save.tree="example_tree.newick", save.distance.matrix="example_DM.csv")

## End(Not run)
Arguments

**path**  
Path to image (a string).

**lower**  
RGB or HSV triplet specifying the lower bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]).

**upper**  
RGB or HSV triplet specifying the upper bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]). Determining these bounds may take some trial and error, but the following bounds may work for certain common background colors:

- Black: lower=c(0, 0, 0); upper=c(0.1, 0.1, 0.1)
- White: lower=c(0.8, 0.8, 0.8); upper=c(1, 1, 1)
- Green: lower=c(0, 0.55, 0); upper=c(0.24, 1, 0.24)
- Blue: lower=c(0, 0, 0.55); upper=c(0.24, 0.24, 1)

If no background filtering is needed, set bounds to some non-numeric value (NULL, FALSE, "off", etc); any non-numeric value is interpreted as NULL.

**hsv**  
Logical. Should HSV pixel array also be calculated? Setting to FALSE will shave some time off the analysis, but not much (a few microseconds per image).

**CIELab**  
Logical. Should CIEL*a*b color space pixels be calculated from RGB? Requires specification of a reference white (see details).

**sample.size**  
Number of pixels to be randomly sampled from filtered pixel array for conversion. If not numeric, all pixels are converted.

**ref.white**  
String; white reference for converting from RGB to CIEL*a*b color space. Accepts any of the standard white references for convertColor (see details).

**alpha.channel**  
Logical. If available, should alpha channel transparency be used to mask background? See removeBackground for more details.

**alpha.message**  
Logical. Output a message if using alpha channel transparency to mask background? Helpful for troubleshooting with PNGs.

Details

The upper and lower limits for background pixel elimination set the inclusive bounds for which pixels should be ignored for the 2D arrays; while all background pixels are ideally a single color, images photographed against "uniform" backgrounds often contain some variation, and even segmentation done with photo editing software will produce some variance as a result of image compression.

The upper and lower bounds represent cutoffs: any pixel for which the first channel falls between the first upper and lower bounds, the second channel falls between the second upper and lower bounds, and the third channel falls between the third upper and lower bounds, will be ignored. For example, if you have a green pixel with RGB channel values [0.1, 0.9, 0.2], and your upper and lower bounds were (0.2, 1, 0.2) and (0, 0.6, 0) respectively, the pixel would be ignored because 0 <= 0.1 <= 0.2, 0.6 <= 0.9 <= 1, and 0 <= 0.2 <= 0.2. But a pixel with the RGB channel values [0.3, 0.9, 0.2] would not be considered background because 0.3 >= 0.2.

CIEL*a*b color space requires a reference ‘white light’ color (dimly and brightly lit photographs of the same object will have very different RGB palettes, but similar Lab palettes if appropriate
white references are used). The idea here is that the apparent colors in an image depend not just on the "absolute" color of an object (whatever that means), but also on the available light in the scene. There are seven CIE standardized illuminants available in colordistance (A, B, C, E, and D50, D55, and D60), but the most common are:

- "A": Standard incandescent lightbulb
- "D65": Average daylight
- "D50": Direct sunlight

Color conversions will be highly dependent on the reference white used, which is why no default is provided. Users should look into standard illuminants to choose an appropriate reference for a dataset.

Value

A list with original image ($original.rgb, 3D array), 2D matrix with background pixels removed ($filtered.rgb.2d and $filtered.hsv.2d), and path to the original image ($path).

Note

The 3D array is useful for displaying the original image, while the 2D arrays (RGB and HSV) are treated as rows of data for clustering in the rest of the package.

Examples

```r
loadedImg <- colordistance::loadImage(system.file("extdata", "Heliconius/Heliconius_A/Heliconius_01.jpeg", package="colordistance"),
  upper=rep(1, 3), lower=rep(0.8, 3))

loadedImgNoHSV <- colordistance::loadImage(system.file("extdata", "Heliconius/Heliconius_A/Heliconius_01.jpeg", package="colordistance"),
  upper=rep(1, 3), lower=rep(0.8, 3), hsv=FALSE)
```

---

**normalizeRGB**

*Normalize pixel RGB ratios*

**Description**

Converts clusters from raw channel intensity to their fraction of the intensity for that cluster

**Usage**

```r
normalizeRGB(extractClustersObject)
```
orderClusters

Arguments

extractClustersObject
A list of color clusters such as those returned by `extractClusters` or `getHistList`. List must contain identically sized dataframes with color coordinates (R, G, B or H, S, V) as the first three columns.

Value
A list of the same size and structure as the input list, but with the cluster normalized as described.

Note
This is a useful option if your images have a lot of variation in lighting, but obviously comes at the cost of reducing variation (if darker and lighter colors are meaningful sources of variation in the dataset).

For example, a bright yellow (R=1, G=1, B=0) and a darker yellow (R=0.8, G=0.8, B=0) both have 50% red, 50% green, and 0% blue, so their normalized values would be equivalent.

A similar but less harsh alternative would be to use HSV rather than RGB for pixel binning and color similarity clustering by setting `hsv=T` in clustering functions and specifying a low number of 'value' bins (e.g. `bins=c(8,8,2)`).

Examples

```r
cluster.list <- colordistance::getKMeansList(c(system.file("extdata", "Heliconius/Heliconius_A", package="colordistance"), lower=rep(0.8, 3), upper=rep(1, 3)))
cluster.list <- colordistance::extractClusters(cluster.list)
colordistance:::normalizeRGB(cluster.list)
```

orderClusters

Order color clusters to minimize overall color distance between pairs

Description
Reorders clusters to minimize color distance using the Hungarian algorithm as implemented by `solve_LSAP`.

Usage

`orderClusters(extractClustersObject)`

Arguments

extractClustersObject
A list of color clusters such as those returned by `extractClusters` or `getHistList`. List must contain identically sized dataframes with color coordinates (R, G, B or H, S, V) as the first three columns.
Details

Briefly: Euclidean distances between every possible pair of clusters across two dataframes are calculated, and pairs of clusters are chosen in order to minimize the total sum of color distances between the cluster pairs (i.e. A1-B1, A2-B2, etc).

For example, if dataframe A has a black cluster, a white cluster, and a blue cluster, in that order, and dataframe B has a white cluster, a blue cluster, and a grey cluster, in that order, the final pairs might be A1-B3 (black and grey), A2-B2 (blue and blue), and A3-B1 (white and white).

Rows are reordered so that paired rows are in the same row index (in the example, dataframe B would be reshuffled to go grey, blue, white instead of white, grey, blue).

Value

A list with identical data to the input list, but with rows in each dataframe reordered to minimize color distances per cluster pair.

Examples

```r
cluster.list <- colordistance::getKMeansList(c(system.file("extdata", "Heliconius/Heliconius_A", package="colordistance"), lower=rep(0.8, 3), upper=rep(1, 3)))
cluster.list <- colordistance::extractClusters(cluster.list)
colordistance:::orderClusters(cluster.list)
```

Description

Tiny little function wrapper, mostly used for looping or when several plots are output by a single function. Waits for user keystroke to move on to next image or exit.

Usage

```r
pause()
```

Examples

```r
for (i in c(1:5)) {
  print(i)
  if (i < 5) {
    colordistance::pause()
  }
}
```
plotClusters  

Plot clusters in 3D color space

Description

Interactive, 3D `plot_ly` plots of cluster sizes and colors for each image in a list of cluster dataframes in order to visualize cluster output.

Usage

```
plotClusters(
  cluster.list,
  color.space = "rgb",
  p = "all",
  pausing = TRUE,
  ref.white,
  to = "sRGB"
)
```

Arguments

- `cluster.list` A list of identically sized dataframes with 4 columns each (R, G, B, Pct or H, S, V, Pct) as output by `extractClusters` or `getHistList`.
- `color.space` The color space ("rgb", "hsv", or "lab") in which to plot pixels.
- `p` Numeric vector of indices for which elements to plot; otherwise each set of clusters is plotted in succession.
- `pausing` Logical. Should the function pause and wait for user keystroke before plotting the next plot?
- `ref.white` The reference white passed to `convertColorSpace`; must be specified if using `color.space = "lab"`. 
- `to` Display color space of image if clustering in CIE Lab space, probably either "sRGB" or "Apple RGB", depending on your computer.

Value

A 3D `plot_ly` plot of cluster sizes in the specified colorspace for each cluster dataframe provided.

Examples

```
## Not run:
# Takes >10 seconds
cluster.list <- colordistance::getHistList(dir(system.file("extdata", "Heliconius/", package="colordistance"), full.names=TRUE), plotting=FALSE, lower=rep(0.8, 3), upper=rep(1, 3))

colordistance::plotClusters(cluster.list, p=c(1:3, 7:8), pausing=FALSE)
```
plotClustersMulti

Plots cluster sets from several different dataframes on a single plot for easy comparison.

Usage

plotClustersMulti(
  cluster.list,
  color.space = "rgb",
  p = "all",
  title = "",
  ref.white,
  to = "sRGB"
)

Arguments

cluster.list A list of identically sized dataframes with 4 columns each as output by extractClusters, getLabHistList, or getHistList.

color.space The color space ("rgb", "hsv", or "lab") in which to plot pixels.

p Numeric vector of indices for which elements to plot; otherwise all of the cluster sets provided will be plotted together.

title Optional title for the plot.

ref.white The reference white passed to convertColorSpace; must be specified if using color.space = "lab".

to Display color space of image if clustering in CIE Lab space, probably either "sRGB" or "Apple RGB", depending on your computer.

Value

A single plot_ly plot of every cluster in a list of cluster sets. Each cluster is colored by cluster color, proportional to cluster size, and labeled according to the image from which it originated.
Note

Each cluster plotted is colored according to its actual color, and labeled according to the image from which it originated.

Examples

```r
## Not run:
# Takes >10 seconds
cluster.list <- colordistance::getHistList(dir(system.file("extdata", "Heliconius/", package="colordistance"), full.names=TRUE), plotting=FALSE, lower=rep(0.8, 3), upper=rep(1, 3))

colordistance::plotClustersMulti(cluster.list, p=c(1:4), title="Orange and black Heliconius")

colordistance::plotClustersMulti(cluster.list, p=c(5:8), title="Black, yellow, and red Heliconius")

clusterListHSV <- colordistance::getHistList(dir(system.file("extdata", "Heliconius/", package="colordistance"), full.names=TRUE), hsv=TRUE, plotting=FALSE, lower=rep(0.8, 3), upper=rep(1, 3))

colordistance::plotClustersMulti(clusterListHSV, p=c(1:3, 7:8), hsv=TRUE)
## End(Not run)
```

plotHist

*Color histogram of binned image*

Description

Plots a color histogram from a dataframe as returned by `getImageHist`, `getHistList`, or `extractClusters`. Bars are colored according to the color of the bin.

Usage

```r
plotHist(
    histogram,
    pausing = TRUE,
    color.space = "rgb",
    ref.white,
    from = "sRGB",
    main = "default",
    ...)
```
Arguments

histogram  A single dataframe or a list of dataframes as returned by `getLabHist`, `getLabHistList`, or `extractClusters`. First three columns must be color coordinates and fourth column must be cluster size.

pausing Logical. Pause and wait for keystroke before plotting the next histogram?

color.space The color space ("rgb", "hsv", or "lab") in which to plot cluster histogram.

ref.white The reference white passed to `convertColorSpace`; must be specified if using CIE Lab space. See `convertColorSpace`.

from Display color space of image if clustering in CIE Lab space, probably either "sRGB" or "Apple RGB", depending on your computer.

main Title for plot. If "default", the name of the cluster histogram is used.

... Optional arguments passed to the `barplot` function.

Examples

color_df <- as.data.frame(matrix(rep(seq(0, 1, length.out=3), 3), nrow=3, ncol=3))

color_df$Pct <- c(0.2, 0.5, 0.3)

colordistance::plotHist(color_df, main="Example plot")

plotImage

Display an image in a plot window

Description

Plots an image as an image.

Usage

plotImage(img)

Arguments

img Either a path to an image or a `loadImage` object.

Details

Redundant, but a nice sanity check. Used in a few other functions in colordistance package. Takes either a path to an image (RGB or PNG) or an image object as read in by `loadImage`.

Value

A plot of the provided image in the current plot window.
Examples

```
colordistance::plotImage(system.file("extdata", "Heliconius/Heliconius_A/Heliconius_01.jpeg", package="colordistance"))
colordistance::plotImage(loadImage(system.file("extdata", "Heliconius/Heliconius_A/Heliconius_01.jpeg", package="colordistance"), lower=rep(0.8, 3), upper=rep(1, 3)))
```

---

**plotPixels**

*Plot pixels in color space*

Description

Plots non-background pixels according to their color coordinates, and colors them according to their RGB or HSV values. Dimensions are either RGB or HSV depending on flags.

Usage

```
plotPixels(
  img,
  n = 10000,
  lower = c(0, 0.55, 0),
  upper = c(0.25, 1, 0.25),
  color.space = "rgb",
  ref.white = NULL,
  pch = 20,
  main = "default",
  from = "sRGB",
  xlim = "default",
  ylim = "default",
  zlim = "default",
  ...
)
```

Arguments

- **img**
  - Either a path to an image or a `loadImage` object.
- **n**
  - Number of randomly selected pixels to plot; recommend <20000 for speed. If n exceeds the number of non-background pixels in the image, all pixels are plotted. If n is not numeric, all pixels are plotted.
- **lower**
  - RGB or HSV triplet specifying the lower bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]).
- **upper**
  - RGB or HSV triplet specifying the upper bounds for background pixels. Default upper and lower bounds are set to values that work well for a bright green background (RGB [0, 1, 0]). Determining these bounds may take some trial and error, but the following bounds may work for certain common background colors:
• Black: lower=c(0, 0, 0); upper=c(0.1, 0.1, 0.1)
• White: lower=c(0.8, 0.8, 0.8); upper=c(1, 1, 1)
• Green: lower=c(0, 0.55, 0); upper=c(0.24, 1, 0.24)
• Blue: lower=c(0, 0, 0.55); upper=c(0.24, 0.24, 1)

If no background filtering is needed, set bounds to some non-numeric value (NULL, FALSE, "off", etc); any non-numeric value is interpreted as NULL.

color.space
The color space ("rgb", "hsv", or "lab") to use for plotting.

ref.white
The reference white passed to convertColor; must be specified if img does not already contain CIE Lab pixels. See convertColorSpace.

pch
Passed to scatterplot3d.

main
Plot title. If left as "default", image name is used.

from
Original color space of image if plotting in CIE Lab space, probably either "sRGB" or "Apple RGB", depending on your computer.

xlim, ylim, zlim
Ranges for the X, Y, and Z axes. If "default", the widest ranges for each axis according to the specified color space (0-1 for RGB and HSV, 0-100 for L of Lab, -128-127 for a and b of Lab) are used.

... Optional parameters passed to scatterplot3d.

Value
3D plot of pixels in either RGB or HSV color space, colored according to their color in the image. Uses scatterplot3d function.

Note
If n is not numeric, then all pixels are plotted, but this is not recommended. Unless the image has a low pixel count, it takes much longer, and plotting this many points in the plot window can obscure important details.

There are seven CIE standardized illuminants available in colordistance (A, B, C, E, and D50, D55, and D65), but the most common are:

• "A": Standard incandescent lightbulb
• "D65": Average daylight
• "D50": Direct sunlight

Examples

colordistance::plotPixels(system.file("extdata", "Heliconius/Heliconius_B/Heliconius_07.jpeg", package="colordistance"), n=20000, upper=rep(1, 3), lower=rep(0.8, 3), color.space = "rgb", angle = -45)
removeBackground  
Remove background pixels in image

Description

Take an image array (from `readPNG` or `jpeg(readJPEG)`) and remove the background pixels based on transparency (if a PNG with transparency) or color boundaries.

Usage

```r
removeBackground(
  img,
  lower = NULL,
  upper = NULL,
  quietly = FALSE,
  alpha.channel = TRUE
)
```

Arguments

- `img`  
  Image array, either output from `readPNG` or `jpeg(readJPEG)`.
- `lower, upper`  
  RGB or HSV triplets specifying the bounds for background pixels. See `loadImage`.
- `quietly`  
  Logical. Display a message if using transparency?
- `alpha.channel`  
  Logical. If available, should alpha channel transparency be used to mask background? See details.

Details

If `alpha.channel = TRUE`, transparency takes precedence over color masking. If you provide a PNG with any pixels with alpha < 1, `removeBackground` ignores any lower and upper color boundaries and assumes transparent pixels are background. If all pixels are opaque (alpha = 1), color masking will apply.

Value

A list with a 3-dimensional RGB array and a 2-dimensional array of non-background pixels with R, G, B columns.

Examples

```r
# remove background by transparency
img_path <- system.file("extdata/chrysochroa_NPL.png", package = "colordistance")

img_array <- jpeg::readJPEG(img_path)
```
**scatter3dclusters**

Plot 3D clusters in a 2D plot

**Description**

Uses `scatterplot3d` to plot clusters in color space.

**Usage**

```r
scatter3dclusters(
  clusters,
  color.space, 
  ref.white = "D65",
  xlim = "default",
  ylim = "default",
  zlim = "default",
  main = "Color clusters",
  scaling = 10,
  opacity = 0.9,
  plus = 0.01,
  ...
)
```

**Arguments**

- **clusters**: A single dataframe or a list of dataframes as returned by `getLabHist`, `getLabHistList`, or `extractClusters`. First three columns must be color coordinates and fourth column must be cluster size.
- **color.space**: The color space ("rgb", "hsv", or "lab") in which to plot. If not specified, the function uses column names to guess the color space.
- **ref.white**: Standard reference white for converting lab coordinates to RGB coordinates for coloring clusters. One of either "A", "B", "C", "E", "D50", "D55", or "D65".

```r
img_filtered <- removeBackground(img_array)

# remove background by color
img_path <- dir(system.file("extdata/Heliconius",
package = "colordistance"),
recursive = TRUE, full.names = TRUE)[1]
img_array <- jpeg::readJPEG(img_path)

img_filtered <- removeBackground(img_array,
lower = rep(0.8, 3), upper = rep(1, 3))
```
xlim, ylim, zlim

X, Y, and Z-axis limits. If not specified, the defaults are 0-1 for all channels in RGB and HSV space, or 0-100 for L and -100-100 for a and b channels of CIE Lab space.

main
Title for the plot.

scaling
Scaling factor for size of clusters.

opacity
Transparency value for plotting; must be between 0 and 1.

plus
Amount to add to percent column for plotting; can help to make very small (or 0) clusters visible.

... Additional parameters passed to scatterplot3d.

See Also

plotClusters, plotClustersMulti

Examples

clusters <- data.frame(R = runif(20, min = 0, max = 1),
                      G = runif(20, min = 0, max = 1),
                      B = runif(20, min = 0, max = 1),
                      Pct = runif(20, min = 0, max = 1))

# plot in RGB space
scatter3dclusters(clusters, scaling = 15, plus = 0.05)

# overrule determined color space and plot in HSV space
scatter3dclusters(clusters, scaling = 15, plus = 0.05, color.space = "hsv")

weightPairsDistance Distance between color clusters with user-specified color/size weights

Description

Distance metric with optional user input for specifying how much the bin size similarity and color similarity should be weighted when pairing clusters from different color cluster sets.

Usage

weightPairsDistance(
    T1, T2,
    ordering = FALSE,
    size.weight = 0.5,
    color.weight = 0.5
)
**weightedPairsDistance**

**Arguments**

T1  
Dataframe (especially a dataframe as returned by `extractClusters` or `getImageHist`, but first three columns must be coordinates).

T2  
Another dataframe like T1.

ordering  
Logical. Should clusters by paired in order to minimize overall distance scores or evaluated in the order given?

size.weight  
Weight of size similarity in determining overall score and ordering (if ordering=T).

color.weight  
Weight of color similarity in determining overall score and ordering (if ordering=T). Color and size weights do not necessarily have to sum to 1.

**Value**

Similarity score based on size and color similarity of each pair of points in provided dataframes.

**Note**

Use with caution, since weights can easily swing distance scores more dramatically than might be expected. For example, if `size.weight = 1` and `color.weight = 0`, two clusters of identical color but different sizes would not be compared.

**Examples**

```r
cluster.list <- colordistance::getKMeansList(system.file("extdata","Heliconius/Heliconius_B", package="colordistance"), lower=rep(0.8, 3), upper=rep(1, 3))
cluster.list <- colordistance::extractClusters(cluster.list, ordering=TRUE)
colordistance:::weightedPairsDistance(cluster.list[[1]], cluster.list[[2]], size.weight=0.8, color.weight=0.2)
```
Index

barplot, 16, 22, 24, 37

chisqDistance, 2, 10, 15, 21, 24, 27

colorDistance, 3, 10, 27

combineClusters, 4

getList, 4

convertColor, 5, 6, 24, 30, 39

convertColorSpace, 5, 18, 20–24, 28, 34, 35, 37, 39

EMDistance, 7, 10, 27, 28

exportTree, 8

extractClusters, 5, 9, 10, 25, 29, 32, 36, 37, 41

generateDistanceMatrix, 8, 10, 25, 29

getHistColors, 12

getHistList, 4, 9, 10, 12, 25, 27, 29, 32, 36

getImageHist, 5–7, 12, 14, 14, 18, 19, 27, 36

getImagePaths, 16, 29

getKMeanColors, 17, 19

getKMeansList, 5, 14, 19, 27–29

getLabHist, 20, 23, 25, 37, 41

getLabHistList, 23, 28, 37, 41

hclust, 8, 25

heatmap, 2, 25

heatmapColorDistance, 10, 25

imageClusterColorDistance, 10, 25

plot.png, 34, 35

plotClusters, 34, 42

plotClustersMulti, 35, 42

plotHist, 36

plotImage, 37

plotPixels, 38

readPNG, 40

removeBackground, 13, 15, 21, 24, 30, 40

scatter3dclusters, 41

scatterplot3d, 39, 41, 42

solve_LSAP, 9, 32

weightedPairsDistance, 10, 27, 42

jpeg, 40

kmeans, 9, 18, 20, 28

loadImage, 15, 21, 29, 37, 38, 40

normalizeRGB, 9, 13, 15, 28, 31

orderClusters, 32

pause, 33