Package 'handwriter'

August 16, 2021

Title Handwriting Analysis in R

Version 1.0.1

Maintainer James Taylor <jamesetay1@gmail.com>

Description
Process handwriting document into letters, words, and lines. Provides measurements at all levels. Webpage provided at: <https://csafe-isu.github.io/handwriter/index.html>.

Depends R (>= 3.1)

LinkingTo Rcpp, RcppArmadillo

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports Rcpp, ggplot2, igraph, png, reshape2, stringr, rjson, magick, shiny, randomForest

NeedsCompilation yes

Author Nick Berry [aut], James Taylor [cre, aut], Felix Baez-Santiago [aut]

Repository CRAN

Date/Publication 2021-08-16 16:20:02 UTC

R topics documented:

`AddLetterImages` *AddLetterImages*

Description

Pulls out letterlist as its own object, and adds the image matrix as well

Usage

```
AddLetterImages(letterList, dims)
```

Arguments

`letterList`	Letter list from processHandwriting function
`dims`	Dimensions of the original document

Value

letterList with a new matrix ‘image‘ value for each sublist.

Examples

```
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

dims = dim(twoSent_document$image)
withLetterImages = AddLetterImages(twoSent_processList$letterList, dims)
```

addToFeatures *addToFeatures*

Description

addToFeatures

Usage

```
addToFeatures(FeatureSet, LetterList, vectorDims)
```

Arguments

FeatureSet	The current list of features that have been calculated
LetterList	List of all letters and their information
vectorDims	Vectors with image Dims

Value

A list consisting of current features calculated in FeatureSet as well as measures of compactness, loop count, and loop dimensions

add_character_features *add_character_features*

Description

Internal method that adds features to characters

Usage

```
add_character_features(img, letterList, letters, dims)
```

Arguments

img	thinned binary image
letterList	list containing letter characters
letters	individual characters from letterList
dims	image graph dimensions

Value

a list of letters with features applied

add_covariance_matrix *add_covariance_matrix*

Description

add_covariance_matrix

Usage

```
add_covariance_matrix(character_lists, character_features, dims)
```

Arguments

character_lists	Output from processHandwriting$letterLists
character_features	Nested lists associating features to respective characters.
dims	Dimensions of binary image

Value

nested lists associating features to respective characters.

add_line_info *add_line_info*

Description

Associates characters to their respective line numbers Needs improvement if runtime becomes a problem

Usage

```
add_line_info(character_features, dims)
```

Arguments

character_features	All extracted features
dims	Dimensions of binary image

Value

Appends line information to character features

add_word_info *add_word_info*

Description

Associates characters to their respective word numbers by ML on labeled data

Usage

```
add_word_info(letterList, dims)
```

Arguments

letterList	List containing characters
dims	Dimensions of binary image

Value

Appends line information to character features

AllUniquePaths *AllUniquePaths*

Description

Internal function for getting a list of all non loop paths in a writing sample.

Usage

```
AllUniquePaths(adj, graph, graph0)
```

Arguments

adj	adjacent matrix
graph	first skeletonized graph
graph0	second skeletonized graph

Value

a list of all non loop paths

`all_centroids` *all_centroids*

Description

Iterates through extracted character features, extracting all centroids found for later use in line numbering.

Usage

```
all_centroids(character_features)
```

Arguments

`character_features`
: Features extracted from any given document

Value

All centroids concatenated with one another (unlisted)

`all_down_dists` *all_down_dists*

Description

Iterates through extracted character features, extracting all downward distances found for later use in line separating.

Usage

```
all_down_dists(character_features)
```

Arguments

`character_features`
: Features extracted from any given document

Value

All downdistance concatenated with one another (unlisted)

`char_to_feature` *char_to_feature*

Description

Secondary driver of feature extraction Extracts features from a single character

Usage

```
char_to_feature(character, dims, uniqueid)
```

Arguments

`character`	character to extract information from
`dims`	Dimensions of binary image
`uniqueid`	Unique numerical reference to character

Value

List containing features of character

`checkBreakPoints` *checkBreakPoints*

Description

Internal function called by processHandwriting that eliminates breakpoints based on rules to try to coherently separate letters.

Usage

```
checkBreakPoints(candidateNodes, allPaths, nodeGraph, terminalNodes, dims)
```

Arguments

`candidateNodes`	possible breakpoints
`allPaths`	list of paths
`nodeGraph`	graph of nodes; call the getNodeGraph function
`terminalNodes`	nodes at the endpoints of the graph
`dims`	graph dimensions

Value

a graph without breakpoints and separated letters

checkSimplicityBreaks *checkSimplicityBreaks*

Description

Internal function for removing breakpoints that separate graphs that are too simple to be split. Remove break if graph on left and right of the break have 4 or fewer nodes and no loops or double paths. Never remove break on a trough.

Usage

```
checkSimplicityBreaks(
  candidateBreaks,
  pathList,
  loopList,
  letters,
  nodeGraph0,
  nodeList,
  terminalNodes,
  hasTrough,
  dims
)
```

Arguments

candidateBreaks	possible breakpoints
pathList	list of paths
loopList	list of loops
letters	list of individual letter characters
nodeGraph0	skeletonized graph
nodeList	list of nodes
terminalNodes	nodes at the ends of letters
hasTrough	wether or not break has a trough
dims	graph dimensions

Value

removes breakpoints on simple graphs

checkStacking *checkStacking*

Description

Internal function for removing breakpoints that follow all of the rules, but separate two letters that are stacked on top of each other.

Usage

```
checkStacking(candidateBreaks, allPaths, letters, nodeGraph0, dims)
```

Arguments

candidateBreaks	possible breaks for letterpath
allPaths	list of paths
letters	list of individual letter characters
nodeGraph0	skeletonized graph
dims	graph dimensions

Value

stackPtFlag

cleanBinaryImage *cleanBinaryImage*

Description

Removes alpha channel from png image.

Usage

```
cleanBinaryImage(img)
```

Arguments

img	A matrix of 1s and 0s.

Value

png image with the alpha channel removed

countChanges *countChanges*

Description

Internal function for counting 4-connected components around a pixel.

Usage

```
countChanges(coords, img)
```

Arguments

coords	coordinates to consider
img	The non-thinned image as binary bit map

Value

The sum of the 4-connected components around a pixel.

countNodes *countNodes*

Description

Function for counting nodes in a list of letters.

Usage

```
countNodes(letterList, nodes)
```

Arguments

letterList	list containing letter characters
nodes	list of nodes

Value

number of nodes in letterList

create_words *create_words*

Description

creates word objects based on splits found in processHandwriting

Usage

```
create_words(processList)
```

Arguments

processList	Output from processHandwriting - contains all glyph information

Value

list of word objects

Examples

```
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

dims = dim(twoSent_document$image)
words = create_words(twoSent_processList)
words_after_processing = process_words(words, dim(twoSent_document$image), TRUE)
```

crop *crop*

Description

This function crops an image down so that there is 1 pixel of padding on each side of the outermost 0 points.

Usage

```
crop(img)
```

Arguments

img	Full image matrix to be cropped

Value

Cropped image matrix.

csafe *Cursive written word: csafe*

Description

Cursive written word: csafe

Usage

```
csafe
```

Format

Binary image matrix. 111 rows and 410 columns.

Examples

```
csafe_document = list()
csafe_document$image = csafe
plotImage(csafe_document$image)
csafe_document$thin = thinImage(csafe_document$image)
plotImageThinned(csafe_document$image, csafe_document$thin)
csafe_processList = processHandwriting(csafe_document$thin, dim(csafe_document$image))
```

extract_character_features *extract_character_features*

Description

Primary driver of feature extraction. Parses all characters from a processed image.

Usage

```
extract_character_features(img, character_lists, dims)
```

Arguments

img	The thinned image bitmap
character_lists	Output from processHandwriting$letterLists
dims	Dimensions of binary image

Value

nested lists associating features to respective characters.

findMergeNodes *findMergeNodes*

Description

Internal function to merge nodes that are very close together.

Usage

```
findMergeNodes(skel_graph, mergeMat)
```

Arguments

skel_graph	the skeltonized graph
mergeMat	sets of the nodes to merge into a single nodes

Value

The merged node

find_colorpoints *find_colorpoints*

Description

Finds and assigns points for Kneser Triangulation

Usage

```
find_colorpoints(words, dims)
```

Arguments

words	List of words and some glyph level information
dims	The dimensions of the image (important for r/c features)

Value

A new list with word level information for each word.

getLoops *getLoops*

Description

Internal function for getting looped paths.

Usage

```
getLoops(nodeList, graph, graph0, pathList, dims)
```

Arguments

nodeList	A list of all found nodes
graph	first skeletonized graph
graph0	second skeletonized graph
pathList	The current path list to check for loops
dims	dimensions of the image

Value

A list of all loops found

getNodeGraph *getNodeGraph*

Description

Internal function for creating a graph from a path list and node list.

Usage

```
getNodeGraph(allPaths, nodeList)
```

Arguments

allPaths	list of paths
nodeList	list of nodes

Value

a graph of nodes

getNodeOrder *getNodeOrder*

Description

Internal function for ordering nodes in a letter.

Usage

```
getNodeOrder(letter, nodesInGraph, nodeConnectivity, dims)
```

Arguments

letter	letter graph containing nodes to be ordered
nodesInGraph	how many nodes are in the letter
nodeConnectivity	how nodes are connected to each other
dims	graph dimensions

Value

order of the nodes

getNodes *getNodes*

Description

Detect intersection points of an image thinned with thinImage.

Usage

```
getNodes(indices, dims)
```

Arguments

indices	Where to check for intersection at
dims	dimensions of the image

Value

Returns image matrix. 1 is blank, 0 is a node.

get_aspect_info *get_aspect_info*

Description

Extracts aspect ratio & supporting information from a character Relevant Features: Aspect Ratio: Row (Height) over (Column Width) Height, Width (Each measure of pixels) The rest are supporting features that are minor independently.

Usage

```
get_aspect_info(character, dims)
```

Arguments

character	character to extract information from
dims	Dimensions of binary image

Value

List containing aspect_ratio,

get_centroid_info *get_centroid_info*

Description

Extracts centroid & supporting information from a character Relevant Features: Centroid Index: R Index representation of centroid location Centroid x,y: X,Y representations of the centroid, see ?i_to_rci Centroid Horiz Location: How far along horizontally (Represented as a number between 0 and 1) the centroid is in its respective character. Centroid Vertical Location: How far along vertically (Represented as a number between 0 and 1) the centroid is in its respective character. Slope: 'Letter Lean', slope found between the centroids of each disjoint half in a single character. The letter is split in half, each halve's centroid is calculated independently, the slope is taken between the two. Box Density: (Dimensions of box around letter width height) / (how much of the document it covers) //Might be a more document as opposed to letter based feature Pixel Density: Ratio of black to white pixels found in box drawn around the letter.

Usage

```
get_centroid_info(character, dims)
```

Arguments

character	character to extract information from
dims	Dimensions of binary image

Value

List containing centroid, pixel density,letter 'lean', and all supporting information

---

`get_loop_info` *get_loop_info*

Description

Associator of loop to character association Relevant Features: Loop Count, how many loops are found in the letter Loop Major, length of farthest line that can be drawn inside of a loop Loop Minor, length of the perpendicular bisector of the loop major.

Usage

```
get_loop_info(character, dims)
```

Arguments

`character`	Target for loop association
`dims`	Dimensions of binary image

Value

Loop information to respective character

`handwriter` *handwriter*

Description

This package provides a pipeline for the processing of handwritten documents to be used in

Author(s)

Nick Berry

i_to_rc *i_to_rc*

Description

Function for converting indices to respective row, col.

Usage

```
i_to_rc(nodes, dims)
```

Arguments

nodes	nodes to be converted.
dims	dimensions of binary image

Value

returns matrix mapping nodes to respective row,

i_to_rci *i_to_rci*

Description

Function for converting indices to respective row, col and associates the original index.

Usage

```
i_to_rci(nodes, dims, fixed = FALSE)
```

Arguments

nodes	nodes to be converted.
dims	dimensions of binary image
fixed	instead of normal computation of rows, put it in a fixed location.

Value

returns matrix mapping nodes' indices to respective row, col

letterPaths *letterPaths*

Description

Internal function that uses existing breakPoint list to assign letters to the nodes in nodeGraph0.

Usage

```
letterPaths(allPaths, nodeGraph0, breakPoints)
```

Arguments

allPaths	list of every path
nodeGraph0	graph of all nodes
breakPoints	breakpoint list

Value

assigned letters to nodes in graph

line_number_extract *line_number_extract*

Description

Primary logic unit for line number to character association.

Usage

```
line_number_extract(down_dists, all_centroids, dims)
```

Arguments

down_dists	how far down to the next character from each character
all_centroids	List of centroids extracted from cumulative character_features
dims	Dimensions of binary image

Value

List associating line numbers to characters

london *Cursive written word: London*

Description

Cursive written word: London

Usage

```
london
```

Format

Binary image matrix. 148 rows and 481 columns.

Examples

```
london_document = list()
london_document$image = london
plotImage(london_document$image)
london_document$thin = thinImage(london_document$image)
plotImageThinned(london_document$image, london_document$thin)
london_processList = processHandwriting(london_document$thin, dim(london_document$image))
```

loop_extract *loop_extract*

Description

Iterates through all available paths from processHandwriting() Picks out loops for later character association.

Usage

```
loop_extract(allPaths)
```

Arguments

allPaths	All character (formerly letter) paths from processHandwriting()

Value

List of all loops

makeModel *makeModel*

Description

Creates a randomForest word model

Usage

```
makeModel(TaggedJson)
```

Arguments

TaggedJson	Json File with tagged letter data

Value

randomForest model

message *Full page image of the handwritten London letter.*

Description

Full page image of the handwritten London letter.

Usage

```
message
```

Format

Binary image matrix. 1262 rows and 1162 columns.

Examples

```
## Not run:
message_document = list()
message_document$image = message
plotImage(message_document$image)
message_document$thin = thinImage(message_document$image)
plotImageThinned(message_document$image, message_document$thin)
message_processList = processHandwriting(message_document$thin, dim(message_document$image))

## End(Not run)
```

nature1 *Full page image of the 4th sample (nature) of handwriting from the first writer.*

Description

Full page image of the 4th sample (nature) of handwriting from the first writer.

Usage

```
nature1
```

Format

Binary image matrix. 811 rows and 1590 columns.

Examples

```
## Not run:
nature1_document = list()
nature1_document$image = nature1
plotImage(nature1_document$image)
nature1_document$thin = thinImage(nature1_document$image)
plotImageThinned(nature1_document$image, nature1_document$thin)
nature1_processList = processHandwriting(nature1_document$thin, dim(nature1_document$image))

## End(Not run)
```

otsuBinarization *otsuBinarization*

Description

Uses Otsu's Method to binarize given image, performing automatic image thresholding.

Usage

```
otsuBinarization(img, breaks = 512)
```

Arguments

img	image object to be processed
breaks	a single number giving the number of cells for the histogram

Value

separated image into foreground and background

pathLetterAssociate *pathLetterAssociate*

Description

Function associating entries in allPaths to each letter

Usage

```
pathLetterAssociate(allPaths, letter)
```

Arguments

allPaths	list of paths
letter	individual character

Value

associated path to each letter

plotColorNodes *plotColorNodes*

Description

This function returns a plot of a single Word extracted from a document. It plots the color as well.

Usage

```
plotColorNodes(letterList, whichWord, dims, wordInfo)
```

Arguments

letterList	Letter list from processHandwriting function
whichWord	Single word value denoting which line to plot - checked if too big inside function.
dims	Dimensions of the original document
wordInfo	Word information list

Value

Plot of single word.

Examples

```
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

dims = dim(twoSent_document$image)
words = create_words(twoSent_processList)
words_after_processing = process_words(words, dim(twoSent_document$image), TRUE)
plotColorNodes(twoSent_processList$letterList, 3, dims, words_after_processing)
```

---

plotImage *plotImage*

Description

This function plots a basic binary image.

Usage

```
plotImage(x)
```

Arguments

x	Binary matrix, usually from readPNGBinary

Value

Returns plot of x.

Examples

```
csafe_document = list()
csafe_document$image = csafe
plotImage(csafe_document$image)
```

plotImageThinned *plotImageThinned*

Description

This function returns a plot with the full image plotted in light gray and the skeleton printed in black on top.

Usage

```
plotImageThinned(img, thinned)
```

Arguments

img	Full image matrix
thinned	Thinned image matrix

Value

Plot of full and thinned image.

Examples

```
## Not run:
csafe_document = list()
csafe_document$image = nature1
csafe_document$thin = thinImage(csafe_document$image)
plotImageThinned(csafe_document$image, csafe_document$thin)

## End(Not run)
```

plotLetter *plotLetter*

Description

This function returns a plot of a single letter extracted from a document. It uses the letterList parameter from the processHandwriting function and accepts a single value as whichLetter. Dims requires the dimensions of the entire document, since this isn't contained in processHandwriting.

Usage

```
plotLetter(
  letterList,
  whichLetter,
  dims,
  showPaths = TRUE,
  showCentroid = TRUE,
  showSlope = TRUE
)
```

Arguments

letterList	Letter list from processHandwriting function
whichLetter	Single value in 1:length(letterList) denoting which letter to plot.
dims	Dimensions of the original document
showPaths	Whether the calculated paths on the letter should be shown with numbers.
showCentroid	Whether the centroid should be shown
showSlope	whether the slope should be shown

Value

Plot of single letter.

Examples

```
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

dims = dim(twoSent_document$image)
plotLetter(twoSent_processList$letterList, 1, dims)
plotLetter(twoSent_processList$letterList, 4, dims)
```

---

plotLine *plotLine*

Description

This function returns a plot of a single line extracted from a document. It uses the letterList parameter from the processHandwriting function and accepts a single value as whichLetter. Dims requires the dimensions of the entire document, since this isn't contained in processHandwriting.

Usage

```
plotLine(letterList, whichLine, dims)
```

Arguments

letterList	Letter list from processHandwriting function
whichLine	Single value denoting which line to plot - checked if too big inside function.
dims	Dimensions of the original document

Value

Plot of single line.

Examples

```
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

dims = dim(twoSent_document$image)
plotLine(twoSent_processList$letterList, 1, dims)
```

---

plotNodes *plotNodes*

Description

This function returns a plot with the full image plotted in light gray and the skeleton printed in black, with red triangles over the vertices. Also called from plotPath, which is a more useful function, in general.

Usage

```
plotNodes(img, thinned, nodeList, nodeSize = 3, nodeColor = "red")
```

Arguments

img	Full image matrix, unthinned.
thinned	Thinned image matrix
nodeList	Nodelist returned from getNodes.
nodeSize	Size of triangles printed. 3 by default. Move down to 2 or 1 for small text images.
nodeColor	Which color the nodes should be

Value

Plot of full and thinned image with vertices overlaid.

Examples

```
## Not run:
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

twoSent_document$nodes = twoSent_processList$nodes
twoSent_document$breaks = twoSent_processList$breakPoints
plotNodes(twoSent_document$image, twoSent_document$thin, twoSent_document$nodes)
plotNodes(twoSent_document$image, twoSent_document$thin, twoSent_document$breaks)

## End(Not run)
```

`plotNodesLine` *plotNodesLine*

Description

Internal function for drawing a line from two given nodes.

Usage

```
plotNodesLine(img, thinned, nodeList, nodeSize = 3, nodeColor = "red")
```

Arguments

`img`	full image matrix; used to call plotImageThinned()
`thinned`	thinned image matrix; used to call plotImageThinned()
`nodeList`	list of nodes
`nodeSize`	size of node; default set to 3
`nodeColor`	color of node; default set to red

Value

a line in between the two nodes

plotWord *plotWord*

Description

This function returns a plot of a single Word extracted from a document. It uses the letterList parameter from the processHandwriting function and accepts a single value as whichLetter. Dims requires the dimensions of the entire document, since this isn't contained in processHandwriting.

Usage

```
plotWord(letterList, whichWord, dims)
```

Arguments

letterList	Letter list from processHandwriting function
whichWord	Single word value denoting which line to plot - checked if too big inside function.
dims	Dimensions of the original document

Value

Plot of single word.

Examples

```
## Not run:
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

dims = dim(twoSent_document$image)
words = create_words(twoSent_processList)
words_after_processing = process_words(words, dim(twoSent_document$image), TRUE)
plotWord(twoSent_processList$letterList, 1, dims)

## End(Not run)
```

processHandwriting *processHandwriting*

Description

Main driver of handwriting processing. Takes in thin image form and the breakpoints suggested by getNodes and parses the writing into letters. Returns final letter separation points, a list of the paths in the image, and a list of the letter paths in the image.

Usage

```
processHandwriting(img, dims)
```

Arguments

img	Thinned binary image.
dims	Dimensions of thinned binary image.

Value

Returns a list of length 3. Object [[1]] (breakPoints) is the set of final letter separation points. Object [[2]] (pathList) is a list of the paths between the input specified nodes. Object [[3]] (letters) is a list of the pixels in the different letters in the handwriting sample.

Examples

```
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))
```

process_words *process_words*

Description

Gets information on a word level

Usage

```
process_words(words, dims, triangulate = FALSE)
```

Arguments

words	List of words and some glyph level information
dims	The dimensions of the image (important for r/c features)
triangulate	Logical value that begins the triangulation process when set to TRUE.

Value

A new list with word level information for each word.

Examples

```
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

dims = dim(twoSent_document$image)
words = create_words(twoSent_processList)
words_after_processing = process_words(words, dim(twoSent_document$image), TRUE)
```

---

rc_to_i *rc_to_i*

Description

Convert rows and columns to their respective indices. This is index sensitive, so row_y[[1]] should correspond to col_x[[1]]

Usage

```
rc_to_i(row_y, col_x, dims, fixed = FALSE)
```

Arguments

row_y	Row(s) to be converted to an index
col_x	Columns(s) to be converted to an index
dims	Dimensions of binary image
fixed	Logical value asking if row_y is fixed to a point.

Value

Returns index(icies) of all row_y's and col_x's

readPNGBinary *readPNGBinary*

Description

This function reads in and binarizes PNG images from the specified file path.

Usage

```
readPNGBinary(
  path,
  cutoffAdjust = 0,
  clean = TRUE,
  crop = TRUE,
  inversion = FALSE
)
```

Arguments

path	File path for image.
cutoffAdjust	Multiplicative adjustment to the K-means estimated binarization cutoff.
clean	Whether to fill in white pixels with 7 or 8 neighbors. This will help a lot when thinning – keeps from getting little white bubbles in text.
crop	Logical value dictating whether or not to crop the white out around the image. TRUE by default.
inversion	Logical value dictating whether or not to flip each pixel of binarized image. Flipping happens after binarization. FALSE by default.

Value

Returns image from path. 0 represents black, and 1 represents white by default.

Examples

```
## Not run:
csafe_document = list()
csafe_document$image = readPNGBinary("examples/Writing_csafe_single.png")
csafe_document$thin = thinImage(csafe_document$image)
csafe_processList = processHandwriting(csafe_document$thin, dim(csafe_document$image))

## End(Not run)
```

`rgb2grayscale` *rgba2grayscale*

Description

Changes RGB image to grayscale

Usage

```
rgb2grayscale(img)
```

Arguments

`img`	A 3D array with slices R, G, and B

Value

img as a 3D array as grayscale

`rgba2rgb` *rgba2rgb*

Description

Removes alpha channel from png image.

Usage

```
rgba2rgb(img)
```

Arguments

`img`	A 3-d array with slices R, G, B, and alpha.

Value

img as a 3D array with alpha channel removed

runHandwritingViewer *runHandwritingViewer*

Description

This function opens and runs a shiny app that allows for viewing of an object that comes from the 'processHandwriting' function.' Requires **shiny**.

Usage

```
runHandwritingViewer()
```

Value

None

See Also

lattice

Examples

```
## Not run:
runHandWrtingViewer()

## End(Not run)
```

SaveAllLetterPlots *SaveAllLetterPlots*

Description

This function returns a plot of a single letter extracted from a document. It uses the letterList parameter from the processHandwriting function and accepts a single value as whichLetter. Dims requires the dimensions of the entire document, since this isn't contained in processHandwriting. Requires the **magick** package.

Usage

```
SaveAllLetterPlots(letterList, filePaths, dims, bgTransparent = TRUE)
```

Arguments

letterList	Letter list from processHandwriting function
filePaths	Folder path to save images to
dims	Dimensions of original document
bgTransparent	Logical determines if the image is transparent

Value

No return value.

See Also

image_transparent

image_write

image_read

Examples

```
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

dims = dim(twoSent_document$image)
## Not run:
withLetterImages = AddLetterImages(twoSent_processList$letterList, "path/to/save", dims)

## End(Not run)
```

---

thinImage *thinImage*

Description

This function returns a vector of locations for black pixels in the thinned image. Thinning done using Zhang - Suen algorithm.

Usage

```
thinImage(img)
```

Arguments

img	A binary matrix of the text that is to be thinned.

Value

A thinned, one pixel wide, image.

twoSent *Two sentence printed example handwriting*

Description

Two sentence printed example handwriting

Usage

```
twoSent
```

Format

Binary image matrix. 396 rows and 1947 columns

Examples

```
## Not run:
twoSent_document = list()
twoSent_document$image = twoSent
plotImage(twoSent_document$image)
twoSent_document$thin = thinImage(twoSent_document$image)
plotImageThinned(twoSent_document$image, twoSent_document$thin)
twoSent_processList = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))

## End(Not run)
```

whichNeighbors *whichNeighbors*

Description

Internal function for identifying which neighbors are black.

Usage

```
whichNeighbors(coords, img)
```

Arguments

coords	coordinates to consider
img	The image as a bitmap

Value

Return a list of which neighbors are a black pixel

`whichNeighbors0` *whichNeighbors0*

Description

Internal function for identifying which neighbors are black excluding diagonals to the middle point when a non-diagonal between those two vertices exists.

Usage

```
whichNeighbors0(coords, img)
```

Arguments

`coords`	coordinates to consider
`img`	The image as a bitmap

Value

Return a list of which neighbors are a black pixel excluding diagonals to the middle point when a non-diagonal between those two vertices exists.

`whichToFill` *whichToFill*

Description

Finds pixels in the plot that shouldn't be white and makes them black. Quick and helpful cleaning for before the thinning algorithm runs.

Usage

```
whichToFill(img)
```

Arguments

`img`	A binary matrix.

Value

A cleaned up image.

wordModel	*wordModel is the RandomForest model to decide if a word separation has happened*

Description

wordModel is the RandomForest model to decide if a word separation has happened

Index