

Package ‘handwriter’

October 9, 2024

Title Handwriting Analysis in R

Version 3.2.0

Maintainer Stephanie Reinders <srein@iastate.edu>

Description Perform statistical writership analysis of scanned handwritten documents.

Webpage provided at: <<https://github.com/CSAFE-ISU/handwriter>>.

Depends R (>= 3.5.0)

LinkingTo Rcpp, RcppArmadillo

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports doParallel, dplyr, foreach, ggplot2, igraph, lpSolve, magick,
mc2d, png, purrr, Rcpp, reshape2, Rfast, rjags, stringr, tidyr,

Suggests knitr, rmarkdown, testthat (>= 3.0.0), coda, withr

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://github.com/CSAFE-ISU/handwriter>

BugReports <https://github.com/CSAFE-ISU/handwriter/issues>

NeedsCompilation yes

Author Iowa State University of Science and Technology on behalf of its Center
for Statistics and Applications in Forensic Evidence [aut, cph,
fnd],

Nick Berry [aut],

Stephanie Reinders [aut, cre],

James Taylor [aut],

Felix Baez-Santiago [ctb],

Jon González [ctb]

Repository CRAN

Date/Publication 2024-10-09 19:30:02 UTC

Contents

about_variable	3
addToFeatures	3
analyze_questioned_documents	4
calculate_accuracy	5
cleanBinaryImage	6
csafe	6
drop_burnin	7
example_analysis	7
example_cluster_template	8
example_model	9
extractGraphs	10
fit_model	11
format_template_data	13
get_clusters_batch	13
get_cluster_fill_counts	14
get_credible_intervals	15
get_posterior_probabilities	16
graphToPrototype	17
london	17
make_clustering_template	18
message	19
nature1	20
plotImage	20
plotImageThinned	21
plotLetter	22
plotLine	23
plotNodes	23
plot_cluster_centers	24
plot_cluster_fill_counts	25
plot_cluster_fill_rates	26
plot_credible_intervals	26
plot_posterior_probabilities	27
plot_trace	28
processDocument	28
processHandwriting	29
process_batch_dir	30
process_batch_list	30
readPNGBinary	31
read_and_process	32
rgb2grayscale	33
rgba2rgb	33
thinImage	34
twoSent	34
whichToFill	35

about_variable	<i>About Variable</i>
----------------	-----------------------

Description

about_variable() returns information about the model variable.

Usage

```
about_variable(variable, model)
```

Arguments

variable	A variable in the fitted model output by <code>fit_model()</code>
model	A fitted model created by <code>fit_model()</code>

Value

Text that explains the variable

Examples

```
about_variable(  
  variable = "mu[1,2]",  
  model = example_model  
)
```

addToFeatures	<i>addToFeatures</i>
---------------	----------------------

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

analyze_questioned_documents

Analyze Questioned Documents

Description

analyze_questioned_documents() estimates the posterior probability of writership for the questioned documents using Markov Chain Monte Carlo (MCMC) draws from a hierarchical model created with [fit_model\(\)](#).

Usage

```
analyze_questioned_documents(  
  main_dir,  
  questioned_docs,  
  model,  
  num_cores,  
  writer_indices,  
  doc_indices  
)
```

Arguments

main_dir	A directory that contains a cluster template created by make_clustering_template()
questioned_docs	A directory containing questioned documents
model	A fitted model created by fit_model()
num_cores	An integer number of cores to use for parallel processing with the doParallel package.
writer_indices	A vector of start and stop characters for writer IDs in file names
doc_indices	A vector of start and stop characters for document names in file names

Value

A list of likelihoods, votes, and posterior probabilities of writership for each questioned document.

Examples

```
## Not run:
main_dir <- "/path/to/main_dir"
questioned_docs <- "/path/to/questioned_images"
analysis <- analyze_questioned_documents(
  main_dir = main_dir,
  questioned_docs = questioned_docs,
  model = model,
  num_cores = 2,
  writer_indices = c(2, 5),
  doc_indices = c(7, 18)
)
analysis$posterior_probabilities

## End(Not run)
```

calculate_accuracy	<i>Calculate Accuracy</i>
--------------------	---------------------------

Description

Fit a model with `fit_model()` and calculate posterior probabilities of writership with `analyze_questioned_documents()` of a set of test documents where the ground truth is known. Then use `calculate_accuracy()` to measure the accuracy of the fitted model on the test documents. Accuracy is calculated as the average posterior probability assigned to the true writer.

Usage

```
calculate_accuracy(analysis)
```

Arguments

analysis Writership analysis output by `analyze_questioned_documents`

Value

The model's accuracy on the test set as a number

Examples

```
# calculate the accuracy for example analysis performed on test documents and a model with 1 chain
calculate_accuracy(example_analysis)

## Not run:
main_dir <- "/path/to/main_dir"
test_images_dir <- "/path/to/test_images"
analysis <- analyze_questioned_documents(
  main_dir = main_dir,
```

```

questioned_docs = test_images_dir,
model = model,
num_cores = 2,
writer_indices = c(2, 5),
doc_indices = c(7, 18)
)
calculate_accuracy(analysis)

## End(Not run)

```

cleanBinaryImage	<i>cleanBinaryImage</i>
------------------	-------------------------

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

csafe	<i>Cursive written word: csafe</i>
-------	------------------------------------

Description

Cursive written word: csafe

Usage

```
csafe
```

Format

Binary image matrix. 111 rows and 410 columns.

Examples

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

drop_burnin

Drop Burn-In

Description

drop_burnin() removes the burn-in from the Markov Chain Monte Carlo (MCMC) draws.

Usage

```
drop_burnin(model, burn_in)
```

Arguments

model A list of MCMC draws from a model fit with `fit_model()`.
burn_in An integer number of starting iterations to drop from each MCMC chain.

Value

A list of data frames of MCMC draws with burn-in dropped.

Examples

```
model <- drop_burnin(model = example_model, burn_in = 25)
plot_trace(variable = "mu[1,2]", model = example_model)
```

example_analysis

Example of writership analysis

Description

Example of writership analysis

Usage

```
example_analysis
```

Format

The results of `analyze_questioned_documents()` stored in a named list with 5 items:

graph_measurements A data frame of that shows the writer, document name, cluster assignment, slope, principle component rotation angle, and wrapped principle component rotation angle for each training graph in each questioned documents.

cluster_fill_counts A data frame of the cluster fill counts for each questioned document.

likelihoods A list of data frames where each data frame contains the likelihoods for a questioned document for each MCMC iteration.

votes A list of vote tallies for each questioned document.

posterior_probabilites A list of posterior probabilities of writership for each questioned document and each known writer in the closed set used to train the hierarchical model.

Examples

```
plot_cluster_fill_counts(formatted_data = example_analysis)
plot_posterior_probabilities(analysis = example_analysis)
```

example_cluster_template

Example cluster template

Description

An example cluster template created with `make_clustering_template()`. The cluster template was created from handwriting samples "w0016_s01_pLND_r01.png", "w0080_s01_pLND_r01.png", "w0124_s01_pLND_r01.png", "w0138_s01_pLND_r01.png", and "w0299_s01_pLND_r01.png" from the CSAFE Handwriting Database. The template has K=5 clusters.

Usage

```
example_cluster_template
```

Format

A list containing a single cluster template created by `make_clustering_template()`. The cluster template was created by sorting a random sample of 1000 graphs from 10 training documents into 10 clusters with a K-means algorithm. The cluster template is a named list with 16 items:

centers_seed An integer for the random number generator.

cluster A vector of cluster assignments for each graph used to create the cluster template.

centers The final cluster centers produced by the K-Means algorithm.

K The number of clusters to build (10) with the K-means algorithm.

n The number of training graphs to use (1000) in the K-means algorithm.

- docnames** A vector that lists the training document from which each graph originated.
- writers** A vector that lists the writer of each graph.
- iters** The maximum number of iterations for the K-means algorithm (3).
- changes** A vector of the number of graphs that changed clusters on each iteration of the K-means algorithm.
- outlierCutoff** A vector of the outlier cutoff values calculated on each iteration of the K-means algorithm.
- stop_reason** The reason the K-means algorithm terminated.
- wcd** A matrix of the within cluster distances on each iteration of the K-means algorithm. More specifically, the distance between each graph and the center of the cluster to which it was assigned on each iteration.
- wcss** A vector of the within-cluster sum of squares on each iteration of the K-means algorithm.

Examples

```
# view cluster fill counts for template training documents
template_data <- format_template_data(example_cluster_template)
plot_cluster_fill_counts(template_data, facet = TRUE)
```

example_model

Example of a hierarchical model

Description

Example of a hierarchical model

Usage

```
example_model
```

Format

A hierarchical model created by `fit_model` with a single chain of 100 MCMC iterations. It is a named list of 4 objects:

- graph_measurements** A data frame of model training data that shows the writer, document name, cluster assignment, slope, principle component rotation angle, and wrapped principle component rotation angle for each training graph.
- cluster_fill_counts** A data frame of the cluster fill counts for each model training document.
- rjags_data** The model training information from `graph_measurements` and `cluster_fill_counts` formatted for RJAGS.
- fitted_model** A model fit using the `rjags_data` and the RJAGS and coda packages. It is an MCMC list that contains a single MCMC object.

Examples

```
# convert to a data frame and view all variable names
df <- as.data.frame(coda::as.mcmc(example_model$fitted_model))
colnames(df)

# view a trace plot
plot_trace(variable = "mu[1,1]", model = example_model)

# drop the first 25 MCMC iterations for burn-in
model <- drop_burnin(model = example_model, burn_in = 25)

## Not run:
# analyze questioned documents
main_dir <- /path/to/main_dir
questioned_docs <- /path/to/questioned_documents_directory
analysis <- analyze_questioned_documents(
  main_dir = main_dir,
  questioned_docs = questioned_docs
  model = example_model
  num_cores = 2
)
analysis$posterior_probabilities

## End(Not run)
```

extractGraphs

Extract Graphs

Description

‘r lifecycle::badge("superseded")‘

Usage

```
extractGraphs(source_folder = getwd(), save_folder = getwd())
```

Arguments

source_folder path to folder containing .png images
save_folder path to folder where graphs are saved to

Details

Development on ‘extractGraphs()‘ is complete. We recommend using ‘process_batch_dir()‘ instead.

Extracts graphs from .png images and saves each by their respective writer.

Value

saves graphs in an rds file

Examples

```
## Not run:
sof = "path to folder containing .png images"
saf = "path to folder where graphs will be saved to"
extractGraphs(sof, saf)

## End(Not run)
```

fit_model

Fit Model

Description

fit_model() fits a Bayesian hierarchical model to the model training data in model_docs and draws samples from the model as Markov Chain Monte Carlo (MCMC) estimates.

Usage

```
fit_model(
  main_dir,
  model_docs,
  num_iters,
  num_chains = 1,
  num_cores,
  writer_indices,
  doc_indices,
  a = 2,
  b = 0.25,
  c = 2,
  d = 2,
  e = 0.5
)
```

Arguments

main_dir	A directory that contains a cluster template created by make_clustering_template()
model_docs	A directory containing model training documents
num_iters	An integer number of iterations of MCMC.
num_chains	An integer number of chains to use.
num_cores	An integer number of cores to use for parallel processing clustering assignments. The model fitting is not done in parallel.

writer_indices	A vector of the start and stop character of the writer ID in the model training file names. E.g., if the file names are writer0195_doc1, writer0210_doc1, writer0033_doc1 then writer_indices is 'c(7,10)'.
doc_indices	A vector of the start and stop character of the "document name" in the model training file names. This is used to distinguish between two documents written by the same writer. E.g., if the file names are writer0195_doc1, writer0195_doc2, writer0033_doc1, writer0033_doc2 then doc_indices are 'c(12,15)'.
a	The shape parameter for the Gamma distribution in the hierarchical model
b	The rate parameter for the Gamma distribution in the hierarchical model
c	The first shape parameter for the Beta distribution in the hierarchical model
d	The second shape parameter for the Beta distribution in the hierarchical model
e	The scale parameter for the hyper prior for mu in the hierarchical model

Value

A list of training data used to fit the model and the fitted model

Examples

```
## Not run:
main_dir <- "/path/to/main_dir"
model_docs <- "path/to/model_training_docs"
questioned_docs <- "path/to/questioned_docs"

model <- fit_model(
  main_dir = main_dir,
  model_docs = model_docs,
  num_iters = 100,
  num_chains = 1,
  num_cores = 2,
  writer_indices = c(2, 5),
  doc_indices = c(7, 18)
)

model <- drop_burnin(model = model, burn_in = 25)

analysis <- analyze_questioned_documents(
  main_dir = main_dir,
  questioned_docs = questioned_docs,
  model = model,
  num_cores = 2
)
analysis$posterior_probabilities

## End(Not run)
```

format_template_data *Format Template Data*

Description

format_template_data() formats the template data for use with [plot_cluster_fill_counts\(\)](#). The output is a list that contains a data frame called cluster_fill_counts.

Usage

```
format_template_data(template)
```

Arguments

template A single cluster template created by [make_clustering_template\(\)](#)

Value

List that contains the cluster fill counts

Examples

```
template_data <- format_template_data(template = example_cluster_template)
plot_cluster_fill_counts(formatted_data = template_data, facet = TRUE)
```

get_clusters_batch *get_clusters_batch*

Description

get_clusters_batch

Usage

```
get_clusters_batch(
  template,
  input_dir,
  output_dir,
  writer_indices,
  doc_indices,
  num_cores = 1,
  save_master_file = FALSE
)
```

Arguments

<code>template</code>	A cluster template created with <code>make_clustering_template</code>
<code>input_dir</code>	A directory containing graphs created with <code>process_batch_dir</code>
<code>output_dir</code>	Output directory for cluster assignments
<code>writer_indices</code>	Vector of start and end indices for the writer id in the graph file names
<code>doc_indices</code>	Vector of start and end indices for the document id in the graph file names
<code>num_cores</code>	Integer number of cores to use for parallel processing
<code>save_master_file</code>	TRUE or FALSE. If TRUE, a master file named 'all_clusters.rds' containing the cluster assignments for all documents in the input directory will be saved to the output directory. If FALSE, a master file will not be saved, but the individual files for each document in the input directory will still be saved to the output directory.

Value

A list of cluster assignments

Examples

```
## Not run:
template <- readRDS('path/to/template.rds')
get_clusters_batch(template=template, input_dir='path/to/dir', output_dir='path/to/dir',
writer_indices=c(2,5), doc_indices=c(7,18), num_cores=1)

get_clusters_batch(template=template, input_dir='path/to/dir', output_dir='path/to/dir',
writer_indices=c(1,4), doc_indices=c(5,10), num_cores=5)

## End(Not run)
```

```
get_cluster_fill_counts
      get_cluster_fill_counts
```

Description

`get_cluster_fill_counts()` creates a data frame that shows the number of graphs in each cluster for each input document.

Usage

```
get_cluster_fill_counts(df)
```

Arguments

`df` A data frame with columns `writer`, `doc`, and `cluster`. Each row corresponding to a graph and lists the writer of that graph, the document from which the graph was obtained, and the cluster to which that graph is assigned.

Value

A dataframe of cluster fill counts for each document in the input data frame.

Examples

```
writer <- c(rep(1, 20), rep(2, 20), rep(3, 20))
docname <- c(rep('doc1', 20), rep('doc2', 20), rep('doc3', 20))
doc <- c(rep(1, 20), rep(2, 20), rep(3, 20))
cluster <- sample(3, 60, replace=TRUE)
df <- data.frame(docname, writer, doc, cluster)
get_cluster_fill_counts(df)
```

`get_credible_intervals`*Get Credible Intervals*

Description

In a model created with `fit_model()` the pi parameters are the estimate of the true cluster fill count for a particular writer and cluster. The function `get_credible_intervals()` calculates the credible intervals of the pi parameters for each writer in the model.

Usage

```
get_credible_intervals(model, interval_min = 0.05, interval_max = 0.95)
```

Arguments

`model` A model output by `fit_model()`

`interval_min` The lower bound for the credible interval. The number must be between 0 and 1.

`interval_max` The upper bound for the credible interval. The number must be greater than `interval_min` and must be less than 1.

Value

A list of data frames. Each data frame lists the credible intervals for a single writer.

Examples

```
get_credible_intervals(model=example_model)
get_credible_intervals(model=example_model, interval_min=0.05, interval_max=0.95)
```

get_posterior_probabilities

Get Posterior Probabilities

Description

Get the posterior probabilities for questioned document analyzed with [analyze_questioned_documents\(\)](#).

Usage

```
get_posterior_probabilities(analysis, questioned_doc)
```

Arguments

analysis The output of [analyze_questioned_documents\(\)](#). If more than one questioned document was analyzed with this function, then the data frame `analysis$posterior_probabilities` lists the posterior probabilities for all questioned documents. [get_posterior_probabilities\(\)](#) creates a data frame of the posterior probabilities for a single questioned document and sorts the known writers from the most likely to least likely to have written the questioned document.

questioned_doc The filename of the questioned document

Value

A data frame of posterior probabilities for the questioned document

Examples

```
get_posterior_probabilities(
  analysis = example_analysis,
  questioned_doc = "w0030_s03_pWOZ_r01"
)
```

graphToPrototype	<i>Convert graph to a prototype</i>
------------------	-------------------------------------

Description

A graph prototype consists of the starting and ending points of each path in the graph, as well as and evenly spaced points along each path. The prototype also stores the center point of the graph. All points are represented as xy-coordinates and the center point is at (0,0).

Usage

```
graphToPrototype(graph, numPathCuts = 8)
```

Arguments

graph	A graph from a handwriting sample
numPathCuts	Number of segments to cut the path(s) into

Value

List of pathEnds, pathQuarters, and pathCenters given as (x,y) coordinates with the graph centroid at (0,0). The returned list also contains path lengths. pathQuarters gives the (x,y) coordinates of the path at the cut points and despite the name, the path might not be cut into quarters.

london	<i>Cursive written word: London</i>
--------	-------------------------------------

Description

Cursive written word: London

Usage

```
london
```

Format

Binary image matrix. 148 rows and 481 columns.

Examples

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

```
make_clustering_template
```

Make Clustering Template

Description

`make_clustering_template()` applies a K-means clustering algorithm to the input handwriting samples pre-processed with `process_batch_dir()` and saved in the input folder `main_dir > data > template_graphs`. The K-means algorithm sorts the graphs in the input handwriting samples into groups, or *clusters*, of similar graphs.

Usage

```
make_clustering_template(  
  main_dir,  
  template_docs,  
  writer_indices,  
  centers_seed,  
  K = 40,  
  num_dist_cores = 1,  
  max_iters = 25  
)
```

Arguments

<code>main_dir</code>	Main directory that will store template files
<code>template_docs</code>	A directory containing template training images
<code>writer_indices</code>	A vector of the starting and ending location of the writer ID in the file name.
<code>centers_seed</code>	Integer seed for the random number generator when selecting starting cluster centers.
<code>K</code>	Integer number of clusters
<code>num_dist_cores</code>	Integer number of cores to use for the distance calculations in the K-means algorithm. Each iteration of the K-means algorithm calculates the distance between each input graph and each cluster center.
<code>max_iters</code>	Maximum number of iterations to allow the K-means algorithm to run

Value

List containing the cluster template

Examples

```
## Not run:  
main_dir <- "path/to/folder"  
template_docs <- "path/to/template_training_docs"  
template_list <- make_clustering_template(  
  main_dir,  
  template_docs,  
  writer_indices,  
  centers_seed,  
  K = 40,  
  num_dist_cores = 1,  
  max_iters = 25  
)
```

```
main_dir = main_dir,  
template_docs = template_docs,  
writer_indices = c(2, 5),  
K = 10,  
num_dist_cores = 2,  
max_iters = 25,  
centers_seed = 100,  
)  
  
## End(Not run)
```

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

```
message_document <- list()  
message_document$image <- message  
plotImage(message_document)  
  
## Not run:  
message_document <- list()  
message_document$image <- message  
plotImage(message_document)  
message_document$thin <- thinImage(message_document$image)  
plotImageThinned(message_document)  
message_processList <- processHandwriting(message_document$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

```
nature1_document <- list()
nature1_document$image <- nature1
plotImage(nature1_document)

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

## End(Not run)
```

plotImage *Plot Image*

Description

This function plots a basic black and white image.

Usage

```
plotImage(doc)
```

Arguments

doc A document processed with `processDocument()` or a binary matrix (all entries are 0 or 1)

Value

ggplot plot

Examples

```
csafe_document <- list()
csafe_document$image <- csafe
plotImage(csafe_document)

## Not run:
document <- processDocument('path/to/image.png')
plotImage(document)

## End(Not run)
```

`plotImageThinned` *Plot Thinned Image*

Description

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

Usage

```
plotImageThinned(doc)
```

Arguments

`doc` A document processed with [processHandwriting\(\)](#)

Value

ggplot plot of thinned image

Examples

```
csafe_document <- list()
csafe_document$image <- csafe
csafe_document$thin <- thinImage(csafe_document$image)
plotImageThinned(csafe_document)
```

plotLetter

Plot Letter

Description

This function returns a plot of a single graph extracted from a document. It uses the letterList parameter from the [processHandwriting\(\)](#) or [processDocument\(\)](#) function and accepts a single value as whichLetter. Dims requires the dimensions of the entire document, since this isn't contained in [processHandwriting\(\)](#) or [processDocument\(\)](#).

Usage

```
plotLetter(
  doc,
  whichLetter,
  showPaths = TRUE,
  showCentroid = TRUE,
  showSlope = TRUE,
  showNodes = TRUE
)
```

Arguments

doc	A document processed with processHandwriting() or processDocument()
whichLetter	Single value in 1:length(letterList) denoting which letter to plot.
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
showNodes	Whether the nodes should be shown

Value

Plot of single letter.

Examples

```
twoSent_document = list()
twoSent_document$image = twoSent
twoSent_document$thin = thinImage(twoSent_document$image)
twoSent_document$process = processHandwriting(twoSent_document$thin, dim(twoSent_document$image))
plotLetter(twoSent_document, 1)
plotLetter(twoSent_document, 4, showPaths = FALSE)
```

plotLine	<i>Plot Line</i>
----------	------------------

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

ggplot 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	<i>Plot Nodes</i>
-----------	-------------------

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(doc, plot_break_pts = FALSE, nodeSize = 3, nodeColor = "red")
```

Arguments

doc	A document processed with <code>processHandwriting()</code>
plot_break_pts	Logical value as to whether to plot nodes or break points. <code>plot_break_pts=FALSE</code> plots nodes and <code>plot_break_pts=TRUE</code> plots break point.
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

```

csafe_document <- list()
csafe_document$image <- csafe
csafe_document$thin <- thinImage(csafe_document$image)
csafe_document$process <- processHandwriting(csafe_document$thin, dim(csafe_document$image))
plotNodes(csafe_document)
plotNodes(csafe_document, nodeSize=6, nodeColor="black")

```

plot_cluster_centers *Plot Template Cluster Centers*

Description

Plot the cluster centers of a cluster template created with `make_clustering_template`. This function uses a K-Means type algorithm to sort graphs from training documents into clusters. On each iteration of the algorithm, it calculates the mean graph of each cluster and finds the graph in each cluster that is closest to the mean graph. The graphs closest to the mean graphs are used as the cluster centers for the next iteration. Handwriter stores the cluster centers of a cluster template as graph prototypes. A graph prototype consists of the starting and ending points of each path in the graph, as well as and evenly spaced points along each path. The prototype also stores the center point of the graph. All points are represented as xy-coordinates and the center point is at (0,0).

Usage

```
plot_cluster_centers(template, plot_graphs = FALSE, size = 100)
```

Arguments

template	A cluster template created with <code>make_clustering_template</code>
plot_graphs	TRUE plots all graphs in each cluster in addition to the cluster centers. FALSE only plots the cluster centers.
size	The size of the output plot

Value

A plot

Examples

```
# plot cluster centers from example template
plot_cluster_centers(example_cluster_template)
plot_cluster_centers(example_cluster_template, plot_graphs = TRUE)
```

plot_cluster_fill_counts

Plot Cluster Fill Counts

Description

Plot the cluster fill counts for each document in formatted_data.

Usage

```
plot_cluster_fill_counts(formatted_data, facet = TRUE)
```

Arguments

formatted_data Data created by [format_template_data\(\)](#), [fit_model\(\)](#), or [analyze_questioned_documents\(\)](#)

facet TRUE uses facet_wrap to create a subplot for each writer. FALSE plots the data on a single plot.

Value

ggplot plot of cluster fill counts

Examples

```
# Plot cluster fill counts for template training documents
template_data <- format_template_data(example_cluster_template)
plot_cluster_fill_counts(formatted_data = template_data, facet = TRUE)

# Plot cluster fill counts for model training documents
plot_cluster_fill_counts(formatted_data = example_model, facet = TRUE)

# Plot cluster fill counts for questioned documents
plot_cluster_fill_counts(formatted_data = example_analysis, facet = FALSE)
```

`plot_cluster_fill_rates`*Plot Cluster Fill Rates*

Description

Plot the cluster fill rates for each document in `formatted_data`.

Usage

```
plot_cluster_fill_rates(formatted_data, facet = FALSE)
```

Arguments

`formatted_data` Data created by `format_template_data()`, `fit_model()`, or `analyze_questioned_documents()`
`facet` TRUE uses `facet_wrap` to create a subplot for each writer. FALSE plots the data on a single plot.

Value

ggplot plot of cluster fill rates

Examples

```
# Plot cluster fill rates for template training documents
template_data <- format_template_data(example_cluster_template)
plot_cluster_fill_rates(formatted_data = template_data, facet = TRUE)

# Plot cluster fill rates for model training documents
plot_cluster_fill_rates(formatted_data = example_model, facet = TRUE)

# Plot cluster fill rates for questioned documents
plot_cluster_fill_rates(formatted_data = example_analysis, facet = FALSE)
```

`plot_credible_intervals`*Plot Credible Intervals*

Description

Plot credible intervals for the model's π parameters that estimate the true writer cluster fill counts.

Usage

```
plot_credible_intervals(  
  model,  
  interval_min = 0.025,  
  interval_max = 0.975,  
  facet = FALSE  
)
```

Arguments

model	A model created by fit_model()
interval_min	The lower bound of the credible interval. It must be greater than zero and less than 1.
interval_max	The upper bound of the credible interval. It must be greater than the interval minimum and less than 1.
facet	TRUE uses <code>facet_wrap</code> to create a subplot for each writer. FALSE plots the data on a single plot.

Value

ggplot plot credible intervals

Examples

```
plot_credible_intervals(model = example_model)  
plot_credible_intervals(model = example_model, facet = TRUE)
```

plot_posterior_probabilities
Plot Posterior Probabilities

Description

Creates a tile plot of posterior probabilities of writership for each questioned document and each known writer analyzed with [analyze_questioned_documents\(\)](#).

Usage

```
plot_posterior_probabilities(analysis)
```

Arguments

analysis	A named list of analysis results from analyze_questioned_documents() .
----------	--

Value

A tile plot of posterior probabilities of writership.

Examples

```
plot_posterior_probabilities(analysis = example_analysis)
```

plot_trace

Plot Trace

Description

Create a trace plot for all chains for a single variable of a fitted model created by `fit_model()`. If the model contains more than one chain, the chains will be combined by pasting them together.

Usage

```
plot_trace(variable, model)
```

Arguments

variable	The name of a variable in the model
model	A model created by <code>fit_model()</code>

Value

A trace plot

Examples

```
plot_trace(model = example_model, variable = "pi[1,1]")
plot_trace(model = example_model, variable = "mu[2,3]")
```

processDocument

Process Document

Description

Load a handwriting sample from a PNG image. Then binarize, thin, and split the handwriting into graphs.

Usage

```
processDocument(path)
```

Arguments

path	File path for handwriting document. The document must be in PNG file format.
------	--

Value

The processed document as a list

Examples

```
image_path <- system.file("extdata", "phrase_example.png", package = "handwriter")
doc <- processDocument(image_path)
plotImage(doc)
plotImageThinned(doc)
plotNodes(doc)
```

processHandwriting *Process Handwriting by Component*

Description

The main driver of handwriting processing. Takes in an image of thinned handwriting created with `thinImage()` and splits the the handwriting into shapes called *graphs*. Instead of processing the entire document at once, the thinned writing is separated into connected components and each component is split into graphs.

Usage

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

Arguments

`img` Thinned binary image created with `thinImage()`.
`dims` Dimensions of thinned binary image.

Value

A list of the processed image

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_batch_dir *Process Batch Directory*

Description

Process a list of handwriting samples saved as PNG images in a directory: (1) Load the image and convert it to black and white with `readPNGBinary()` (2) Thin the handwriting to one pixel in width with `thinImage()` (3) Run `processHandwriting()` to split the handwriting into parts called *edges* and place *nodes* at the ends of edges. Then combine edges into component shapes called *graphs*. (4) Save the processed document in an RDS file. (5) Optional. Return a list of the processed documents.

Usage

```
process_batch_dir(input_dir, output_dir = ".", skip_docs_on_retry = TRUE)
```

Arguments

input_dir	Input directory that contains images
output_dir	A directory to save the processed images
skip_docs_on_retry	Logical whether to skip documents in input_dir that caused errors on a previous run. The errors and document names are stored in output_dir > problems.txt. If this is the first run, process_batch_list will attempt to process all documents in input_dir.

Value

No return value, called for side effects

Examples

```
## Not run:
process_batch_dir("path/to/input_dir", "path/to/output_dir")

## End(Not run)
```

process_batch_list *Process Batch List*

Description

Process a list of handwriting samples saved as PNG images: (1) Load the image and convert it to black and white with `readPNGBinary()` (2) Thin the handwriting to one pixel in width with `thinImage()` (3) Run `processHandwriting()` to split the handwriting into parts called *edges* and place *nodes* at the ends of edges. Then combine edges into component shapes called *graphs*. (4) Save the processed document in an RDS file. (5) Optional. Return a list of the processed documents.

Usage

```
process_batch_list(images, output_dir, skip_docs_on_retry = TRUE)
```

Arguments

`images` A vector of image file paths

`output_dir` A directory to save the processed images

`skip_docs_on_retry`
Logical whether to skip documents in the `images` argument that caused errors on a previous run. The errors and document names are stored in `output_dir > problems.txt`. If this is the first run, `process_batch_list` will attempt to process all documents in the `images` argument.

Value

No return value, called for side effects

Examples

```
## Not run:  
images <- c('path/to/image1.png', 'path/to/image2.png', 'path/to/image3.png')  
process_batch_list(images, "path/to/output_dir", FALSE)  
process_batch_list(images, "path/to/output_dir", TRUE)  
  
## End(Not run)
```

readPNGBinary

Read PNG Binary

Description

This function reads in and binarizes a PNG image.

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

```
image_path <- system.file("extdata", "phrase_example.png", package = "handwriter")
csafe_document <- list()
csafe_document$image = readPNGBinary(image_path)
plotImage(csafe_document)
```

read_and_process *Read and Process*

Description**[Superseded]**

Development on `read_and_process()` is complete. We recommend using `processDocument()`. `read_and_process(image_name, "document")` is equivalent to `processDocument(image_name)`.

Usage

```
read_and_process(image_name, transform_output)
```

Arguments

image_name	The file path to an image
transform_output	The type of transformation to perform on the output

Value

A list of the processed image components

Examples

```
# use handwriting example from handwriter package
image_path <- system.file("extdata", "phrase_example.png", package = "handwriter")
doc <- read_and_process(image_path, "document")
```

rgb2grayscale	<i>rgba2grayscale</i>
---------------	-----------------------

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	<i>rgba2rgb</i>
----------	-----------------

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

thinImage	<i>thinImage</i>
-----------	------------------

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	<i>Two sentence printed example handwriting</i>
---------	---

Description

Two sentence printed example handwriting

Usage

```
twoSent
```

Format

Binary image matrix. 396 rows and 1947 columns

Examples

```
twoSent_document <- list()
twoSent_document$image <- twoSent
plotImage(twoSent_document)

## Not run:
twoSent_document <- list()
twoSent_document$image <- twoSent
plotImage(twoSent_document)
twoSent_document$thin <- thinImage(twoSent_document$image)
plotImageThinned(twoSent_document)
```

```
twoSent_processList <- processHandwriting(twoSent_document$thin, dim(twoSent_document$image))  
## End(Not run)
```

*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.

Index

- * **datasets**
 - csafe, 6
 - example_analysis, 7
 - example_cluster_template, 8
 - example_model, 9
 - london, 17
 - message, 19
 - nature1, 20
 - twoSent, 34

- about_variable, 3
- addToFeatures, 3
- analyze_questioned_documents, 4, 5
- analyze_questioned_documents(), 5, 8, 16, 25–27

- calculate_accuracy, 5
- cleanBinaryImage, 6
- csafe, 6

- drop_burnin, 7

- example_analysis, 7
- example_cluster_template, 8
- example_model, 9
- extractGraphs, 10

- fit_model, 9, 11
- fit_model(), 3–5, 7, 15, 25–28
- format_template_data, 13
- format_template_data(), 25, 26

- get_cluster_fill_counts, 14
- get_clusters_batch, 13
- get_credible_intervals, 15
- get_posterior_probabilities, 16
- get_posterior_probabilities(), 16
- graphToPrototype, 17

- london, 17

- make_clustering_template, 14, 18, 24
- make_clustering_template(), 4, 8, 11, 13
- message, 19

- nature1, 20

- plot_cluster_centers, 24
- plot_cluster_fill_counts, 25
- plot_cluster_fill_counts(), 13
- plot_cluster_fill_rates, 26
- plot_credible_intervals, 26
- plot_posterior_probabilities, 27
- plot_trace, 28
- plotImage, 20
- plotImageThinned, 21
- plotLetter, 22
- plotLine, 23
- plotNodes, 23
- process_batch_dir, 14, 30
- process_batch_dir(), 18
- process_batch_list, 30
- processDocument, 28
- processDocument(), 20, 22, 32
- processHandwriting, 29
- processHandwriting(), 21, 22, 24, 30

- read_and_process, 32
- readPNGBinary, 31
- readPNGBinary(), 30
- rgb2grayscale, 33
- rgba2rgb, 33

- thinImage, 34
- thinImage(), 29, 30
- twoSent, 34

- whichToFill, 35