Package 'sentimentIt'

August 31, 2017

Title Crowdsourced Pairwise Sentiment Text Analysis

Version 0.1

Description This package interacts with the SentimentIt API for sentiment analysis of text through crowdsourcing. By having workers complete pairwise comparisons of documents, a researcher-specified latent trait of interest is estimated, with full posteriors, for each document in the set. If the fullest functionality of the package is exploited, the documents will be sent to the server, workers will be monitored for performance and removed from future participation if providing poor quality data, all of the data will be saved and returned, and a Stan model will be run, saved, and returned.

Depends R (>= 3.1.0)

Imports plyr, jsonlite, httr, RCurl, rstan

Suggests testthat

LazyData true

License GPL (>= 2)

RoxygenNote 5.0.1

NeedsCompilation no

Author David Carlson [aut, cre],
Jacob M. Montgomery [aut]

Maintainer David Carlson <carlson.david@wustl.edu>

R topics documented:

`authenticate` *Authenticate researcher and retrieve token*

Description

Authenticates researcher and returns authorization token. Primarily used internally but can be used if the researcher wishes to use the API directly.

Usage

```
authenticate(email, password)
```

Arguments

`email`	The email used by the researcher to register with SentimentIt.
`password`	The password associated with the account.

Value

auth_token Character for the authorization token, which expires after the researcher signs out.

Author(s)

David Carlson

See Also

`sentimentIt` `batchStatus` `checkCert` `checkWorkers` `createBatches` `createCert` `createPairwise` `createTasks` `fitStan` `fitStanHier` `makeCompsSep` `readInData` `readText` `repostExpired` `reviews` `revokeCert` `signout`

batchStatus *Check the status of batch(es)*

Description

This function checks the status of the batch_id number(s). This returns a data frame with the batch ID and the number of comparisons submitted and the number completed.

Usage

```
batchStatus(email, password, batch_id)
```

Arguments

email	The researcher's email used for SentimentIt registration
password	The researcher's password used for SentimentIt
batch_id	Vector of batch IDs to check

Value

output Dataframe with the variables:

- id
- total_count
- submitted count
- completed_count
- expired_count

Author(s)

David Carlson

See Also

sentimentIt authenticate checkCert checkWorkers createBatches createCert createPairwise createTasks fitStan fitStanHier makeCompsSep readInData readText repostExpired reviews revokeCert signout

Examples

```
## Not run:
batchStatus(email = 'researcher@school.edu', password = 'uniquePassword', batch_id = batch_ids[1])

## End(Not run)
```

checkCert *Checks if a worker has a certification*

Description

Checks if a worker has a certification

Usage

```
checkCert(email, password, cert, worker)
```

Arguments

email	The researcher's email used for SentimentIt registration
password	The researcher's password used for SentimentIt
cert	The name of the certification given to the workers.
worker	The MTurk worker ID you want to check for certification.

Value

TRUE or FALSE indicating if the worker has the certification.

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkWorkers createBatches createCert createPairwise createTasks fitStan fitStanHier makeCompsSep readInData readText repostExpired reviews revokeCert signout

Examples

```
## Not run:
x <- "ab1"
y <- c("a204")
check <- checkCert(email, password, x, y)

## End(Not run)
```

checkWorkers *Detect outlying workers*

Description

Detect outlying workers

Usage

```
checkWorkers(stan_fit, data, cut_point = 1, cut_proportion = 0.9,
  n.questions = 50, plot_hist = FALSE, hist_path = NULL)
```

Arguments

stan_fit	A stan fit.
data	The data used to fit stan.
cut_point	A cutoff point to classify posterior coefficients. The proportion of posterior coefficients below cut_point is used to determine outliers. (Default is 1)
cut_proportion	A cutoff proportion of posterior coefficients below cut_point. If the proportion of posterior coefficients below cut_points is higher than cut_proportion, a worker will be considered as an outlier provided that she answers more than the number of questions in n.questions. (Default is 0.9)
n.questions	The number of questions to consider in order to determine suggested banned workers. (Default is 50)
plot_hist	If TRUE, plot the histogram of workers with a rug plot. Default is FALSE
hist_path	Save the histogram to path and file name specified. Default is NULL and no plot is saved.

Value

ban_workers A vector of outlying workers' IDs whose proportion of posterior coefficients below cut_point is greater than cut_proportion and who answered more than the number of questions in n.questions

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert createBatches createCert createPairwise createTasks fitStan fitStanHier makeCompsSep readInData readText repostExpired reviews revokeCert signout

Examples

```
## Not run:
ban_workers <- checkWorkers(stan_fit = fit, data = output)

## End(Not run)
```

`createBatches` *Create new batches*

Description

This function creates new batches with the desired Task setting and the desired number of batches.

Usage

```
createBatches(email, password, task_setting_id, num_batches = 1)
```

Arguments

`email`	The researcher's email used for SentimentIt registration
`password`	The researcher's password used for SentimentIt
`task_setting_id`	ID of Task setting to use
`num_batches`	Number of separate batches to create. Default is 1.

Value

batch_ids Vector of batch IDs created

Author(s)

David Carlson

See Also

`sentimentIt` `authenticate` `batchStatus` `checkCert` `checkWorkers` `createCert` `createPairwise` `createTasks` `fitStan` `fitStanHier` `makeCompsSep` `readInData` `readText` `repostExpired` `reviews` `revokeCert` `signout`

Examples

```
## Not run:
batch_ids <- createBatches(email = 'researcher@school.edu',
    password = 'uniquePassword',
    task_setting_id = 8, num_batches = 3)

## End(Not run)
```

createCert *Grants certifications to workers*

Description

This function will grant a certification to a worker, and if the certification does not exist it will be created.

Usage

```
createCert(email, password, cert, workers)
```

Arguments

email	The researcher's email used for SentimentIt registration
password	The researcher's password used for SentimentIt
cert	The name of the certification given to the workers.
workers	The workers you want to grant certification.

Value

Character indicating number of workers certified.

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createPairwise createTasks fitStan fitStanHier makeCompsSep readInData readText repostExpired reviews revokeCert signout

Examples

```
## Not run:
createCert(email = 'researcher@school.edu',
    password = 'uniquePassword',
    cert= 'bannedmovie_reviews',
    workers = ban_workers)

## End(Not run)
```

createPairwise *Creates a matrix of random pairs for comparison*

Description

This function randomly samples from all possible pairings of IDs. In other words, no comparison will be duplicated.

Usage

```
createPairwise(ids, number_per, pairwise_path = "pairwise.Rdata")
```

Arguments

ids	The id numbers of the texts you want to use.
number_per	How many documents per batch to be compared.
pairwise_path	Path to save the created pairwise matrix. Default is pairwise.Rdata

Value

A matrix of random pairwise comparisons without repitition.

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createTasks fitStan fitStanHier makeCompsSep readInData readText repostExpired reviews revokeCert signout

createTasks *Create new tasks for comparisons*

Description

This function posts a batch or comparisons to Mechanical Turk as tasks

Usage

```
createTasks(email, password, comp_ids = NULL, task_setting_id = NULL,
  batch_id = NULL)
```

Arguments

email	The researcher's email used for SentimentIt registration
password	The researcher's password used for SentimentIt
comp_ids	Vector of comparison ids for which to create tasks, if comparisons were not created under a batch. Default is NULL. Leave as NULL if batch_id is provided.
task_setting_id	Task setting to use for the created tasks, if comparisons were not created under a batch. Default is NULL. Leave as NULL if batch_id is provided.
batch_id	Batch ID to be used if comparisons created under a batch. Default is NULL.

Value

out ID for batch of comparisons

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createPairwise fitStan fitStanHier makeCompsSep readInData readText repostExpired reviews revokeCert signout

Examples

```
## Not run:
createTasks(email = 'researcher@school.edu', password = 'uniquePassword', batch_id = batch_ids[1])

## End(Not run)
```

---

fitStan *Fit a Stan model with results from SentimentIt*

Description

Fit a random utility model using Hamiltonian MCMC in Stan with data retrieved from the SentimentIt platform.

Usage

```
fitStan(email = NULL, password = NULL, data, chains = 3, iter = 2500,
  seed = 1234, n.cores = 3)
```

Arguments

email	The researcher's email used for SentimentIt registration. Default is NULL and only needs to be provided if batch numbers are used instead of data.
password	The researcher's password used for SentimentIt. Default is NULL and only needs to be provided if batch numbers are used instead of data.
data	A csv file or a vector of batch numbers.
chains	The number of chains. (Default is 3)
iter	The number of iteration. (Default is 2500)
seed	Set seed. (Defalt is 1234)
n.cores	Number of cores to be used in stan fit. (Default is 3)

Value

fit Stan fit object

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createPairwise createTasks fitStanHier makeCompsSep readInData readText repostExpired reviews revokeCert signout

Examples

```
## Not run:
fit <- fitStan(data = batch_ids)
fit <- fitStan(data = output)

## End(Not run)
```

---

fitStanHier *Fit a Stan hierarchical model with results from SentimentIt*

Description

Fit a multilevel random utility model using Hamiltonian MCMC in Stan with data retrieved from the SentimentIt platform.

Usage

```
fitStanHier(email = NULL, password = NULL, data, hierarchy_data,
  hierarchy_var, chains = 3, iter = 2500, seed = 1234, n.cores = 3)
```

Arguments

email	The researcher's email used for SentimentIt registration. Default is NULL and only needs to be provided if batch numbers are used instead of data.
password	The researcher's password used for SentimentIt. Default is NULL and only needs to be provided if batch numbers are used instead of data.
data	A csv file or a vector of batch numbers.
hierarchy_data	A file that contains the variable that is used as a hierarchy.
hierarchy_var	A name of the variable in hierarchy_data that is used as a hierarchy.
chains	The number of chains. (Default is 3)
iter	The number of iteration. (Default is 2500)
seed	Set seed. (Default is 1234)
n.cores	Number of cores to be used in stan fit. (Default is 3)

Value

fit_heir The heirarchical Stan fit object

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createPairwise createTasks fitStan makeCompsSep readInData readText repostExpired reviews revokeCert signout

---

makeCompsSep *Create comparisons and post to SentimentIt*

Description

Creates random comparisons of document IDs using createPairwise. These comparisons are posted on the SentimentIt server and receive unique IDs from the system.

Usage

```
makeCompsSep(email, password, ids, number_per, batch_id, question,
  per_batch = 1000, pairwise_path = "pairwise.Rdata",
  ids_as_comps = FALSE)
```

Arguments

email	The researcher's email used for SentimentIt registration
password	The researcher's password used for SentimentIt registration
ids	Numerical vector of IDs for the documents
number_per	Number of comparisons desired
batch_id	Batch IDs to be used for the tasks
question	A character of the question the worker will see once the worker selects the task
per_batch	Number of comparisons per batch desired. Defaulted to 1000.
pairwise_path	File path to store matrix of document IDs used for comparisons. Default is pairwise.Rdata
ids_as_comps	an indicator as to whether or not the IDs provided refer to comparison IDs rather than document IDs. Default is FALSE.

Value

out A table with the text and correspondings ID's that have been sent.

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createPairwise createTasks fitStan fitStanHier readInData readText repostExpired reviews revokeCert signout

Examples

```
## Not run:
docInfo <- read.table(email, password, "ReviewsWithIds",header=TRUE)
makeCompsSep(email = 'researcher@school.edu',
    password = 'uniquePassword',
    ids = docInfo[,'ids'], number_per = 10,
    batch_id = batch_ids,
    question = 'Below is text taken from
        two movie reviews. Please choose
        the text that you think comes from
        the most positive review',
    pairwise_path = 'Comparisons/first10.Rdata')

## End(Not run)
```

readInData *Read in data provided by the workers*

Description

Reads in the complete data for specified batch numbers from the server. This data contains the selection made for every completed comparison in the batch.

Usage

```
readInData(email, password, batch_id)
```

Arguments

email	The researcher's email used for SentimentIt registration
password	The researcher's password used for SentimentIt
batch_id	A vector of batch numbers to download data from.

Value

output a dataframe with the data from the specified batches with columns

batch_id the batch number
comparison_id id number of the comparison being made
document_id id number of the document
result result of document comparison
task_id id of the task
worker_id id of the worker who did the comparison
completed_at time comaparison was completed

Author(s)

Jacob M. Montgomery

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createPairwise createTasks fitStan fitStanHier makeCompsSep readText repostExpired reviews revokeCert signout

Examples

```
## Not run:
output <- readInData(email = 'researcher@school.edu',
    password = 'uniquePassword', batch_id = batch_ids[1])

## End(Not run)
```

readText *Find/create documents and retrieve IDs*

Description

Reads in text, posts the text to the server, and returns the document IDs after saving the result to a new table. If text already exists on the server, the associated ID is returned (there will never be duplicated text).

Usage

```
readText(email, password, read_documents_from, write_documents_to = NULL,
  what = "character", sep = "\n", quiet = TRUE, index = NULL,
  which_source = "apiR", ...)
```

Arguments

email	The researcher's email associated with the SentimentIt account.
password	The researcher's password associated with the SentimentIt account.
read_documents_from	The file path the data will be drawn form, or a vector of text.
write_documents_to	The file path to write the original data, merged with the document IDs. Default is NULL and the results will not be saved, but only returned. For best functionality specify a csv.
what	Argument passed to scan() function. Default is character. Only needed when index is NULL.
sep	Argument passed to read.table() function. Default is line break. Only needed when index is not NULL.
quiet	Argument passed to scan(). Default is TRUE. Only needed when index is NULL.
index	The index number of the table to extract the text from, or the name of the column. Default is NULL, indicating the text was not sent in a table.
which_source	Source used within SentimentIt server assoicated with document uploads. Only used for later reference. Default is apiR.
...	Additional arguments passed to either scan() or read.table() depending on type of data used

Value

A data frame with the provided data and the IDs appended

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createPairwise createTasks fitStan fitStanHier makeCompsSep readInData repostExpired reviews revokeCert signout

Examples

```
## Not run:
data(reviews)

docInfo <- readText(email = 'researcher@school.edu',
    password = 'uniquePassword', read_documents_from = reviews,
    write_documents_to = "ReviewsWithIds.csv", index = 'Review')

## End(Not run)
```

---

repostExpired *Repost expired tasks from a batch*

Description

This will repost all of the expired tasks from the vector of batch IDs. It is common for a few tasks in a batch to be overlooked and remain incomplete.

Usage

```
repostExpired(email, password, batch_id)
```

Arguments

email	The researcher's email used for SentimentIt registration
password	The researcher's password used for SentimentIt
batch_id	ID of batch to check

Value

out reposted count

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createPairwise createTasks fitStan fitStanHier makeCompsSep readInData readText reviews revokeCert signout

Examples

```
## Not run:
repostExpired(email = 'researcher@school.edu', password = 'uniquePassword', batch_id = batch_ids)

## End(Not run)
```

`reviews` *Movie reviews example data set*

Description

An example dataset of Rotten Tomato movie reviews with the number of stars given by the user. This is the first application in the associated paper introducing SentimentIt.

Usage

```
reviews
```

Format

A data frame with 500 rows and 2 columns:

Stars rating on scale from 1-5 by Mechanical Turk worker

Review The movie review looked over by worker

Source

https://www.rottentomatoes.com

See Also

`sentimentIt` `authenticate` `batchStatus` `checkCert` `checkWorkers` `createBatches` `createCert` `createPairwise` `createTasks` `fitStan` `fitStanHier` `makeCompsSep` `readInData` `readText` `repostExpired` `revokeCert` `signout`

revokeCert *Revoke a certification for workers*

Description

This function simply removes workers from the list of approved workers and they will no longer be able to perform the tasks requiring the certification.

Usage

```
revokeCert(email, password, cert, workers)
```

Arguments

email	The researcher's email used for SentimentIt registration
password	The researcher's password used for SentimentIt
cert	The name of the certification given to the workers.
workers	The workers you want to grant certification.

Value

The number of workers revoked.

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createPairwise createTasks fitStan fitStanHier makeCompsSep readInData readText repostExpired reviews signout

Examples

```
## Not run:
revokeCert(email = 'researcher@school.edu',
    password = 'uniquePassword',
    cert = 'snippets', workers = ban_workers)

## End(Not run)
```

sentimentIt *sentimentIt*

Description

This package interacts with the SentimentIt API to analyze text and capture latent traits of interest.

`sentimentIt()` is a wrapper function of all base functionality.

Usage

```
sentimentIt(email, password, read_documents_from, write_documents_to = NULL,
  what = "character", sep = "\n", quiet = TRUE, index = NULL,
  which_source = "apiR", number_per = 20, task_setting_id, question,
  per_batch = 1000, pairwise_path = "pairwise.Rdata", certone = NULL,
  certtwo = NULL, timed = TRUE, time_per = 1, mintime = 8,
  maxtime = 22, rate = 1/3, threshold = 5, check_workers_at = NULL,
  cut_point = 1, cut_proportion = 0.9, n.questions = 50,
  plot_hist = FALSE, hist_path = NULL, rest_time = 60,
  hierarchy_data = NULL, hierarchy_var = NULL, return_stan = FALSE,
  stan_file = NULL, return_data = TRUE, data_file = NULL, chains = 3,
  iter = 2500, seed = 1234, n.cores = 3, wait_to_repost = 2, ...)
```

Arguments

`email`	The researcher's email used for SentimentIt registration
`password`	The researcher's password used for SentimentIt
`read_documents_from`	File path of the text to be analyzed
`write_documents_to`	Where to write the text with document IDs appended. Defaulted to NULL. For best functionality specify a csv.
`what`	Argument passed to scan() function when reading in text data. Default is character. Only needed when index is NULL.
`sep`	Argument passed to read.table() function when reading in data from a data frame. Default is line break. Only needed when index is not NULL.
`quiet`	Argument passed to scan() function when reading in text data. Default is TRUE. Only needed when index is NULL.
`index`	The index number of the table to extract the text from, or the name of the column. Default is NULL, indicating the text was not sent in a table.
`which_source`	Source used within SentimentIt server assoicated with document uploads. Only used for later reference. Default is apiR.
`number_per`	How many documents per batch to be compared. Default is 20.
`task_setting_id`	ID of task setting to use for comparisons created on SentimentIt platform.

question	The question the worker will be asked when comparing documents.
per_batch	Number of comparisons per batch desired. Defaulted to 1000.
pairwise_path	Path to save the created pairwise matrix of document IDs used for comparison. Default is pairwise.Rdata
certone	The certification needed to complete tasks. Default is NULL. If a certification is needed in the SentimentIt platform, this is only needed if the researcher wishes to check and ban workers.
certtwo	The certification granted if the worker is banned. Default is NULL
timed	Logical indicating whether or not the batches should be sent to MTurk based on time rather than batch completion status. Default is TRUE, the recommended setting. Expired tasks will be reposted regardless.
time_per	Time in hours to wait to post new batches. Default is 1.
mintime	This is the earliest time in the day, in 24 hour time, to post batches. Default is 8. This is based on system time and therefore timezones ought be considered.
maxtime	This is the latest time in the day, in 24 hour time, to post batches. Default is 22
rate	Time, in hours, to wait to check if a batch is (near) completed. Only needed if not posting based on time. Default is 1/3.
threshold	When posting batches based on completion status, this is the number of comparisons remaining at which point the batch can be considered (near) complete. Default is 5. This threshold is ignored after 4 hours, and the next batch is posted.
check_workers_at	This is a vector of how often to check workers, e.g. `c(1,3)` would be after the first and third batch. Default is NULL and workers are not checked.
cut_point	A cutoff point to classify posterior worker estimates. The proportion of posterior draws below `cut_point` is used to determine outliers. (Default is 1)
cut_proportion	A cutoff proportion of posterior draws of worker estimates below `cut_point`. If the proportion of posterior coefficients below `cut_points` is higher than `cut_proportion`, a worker will be considered an outlier if answering more than the number of questions in `n.questions`. (Default is 0.9)
n.questions	The number of questions an outlying worker has to answer in order to be banned. (Default is 50).
plot_hist	If TRUE, plot the histogram of workers with a rug plot. Default is FALSE
hist_path	Save the histogram to path and file name specified. Default is NULL and no plot is saved. If mutliple checks are performed the file will be overwritten unless a vector of paths is provided equal to the number of checks performed. If fewer are provided, the last one will be overwritten until completion.
rest_time	The amount of time in seconds to wait to post a batch after the comparisons are created. This ensures the comparisons are ready before sending the request. Default is 60.
hierarchy_data	A file that contains the variable that is used as a hierarchy group. Default is NULL and is only needed if heirarchical analysis is desired.
hierarchy_var	A column name or number of the variable in `hierarchy_data` that is used as a hierarchy. Default is NULL and is only needed if heirarchical analysis is desired.

return_stan	Return the Stan fit object from the final data analysis if TRUE as part of the return. Default is FALSE
stan_file	The path and name of the file to save the Stan fitted model. Default is NULL and the fit is not saved.
return_data	A logical indicating if the data from the tasks should be returned. Default is TRUE
data_file	The directory and name of the file to save the HIT data. Default is NULL and the data is not saved
chains	The number of chains to use for all Stan runs. Default is 3
iter	The number of iterations for all Stan runs. Default is 2500
seed	The seed to use for Stan runs. Default is 1234
n.cores	Number of cores to be used in Stan runs. Default is 3. Note that, because this wrapper will generally run for extended periods of time, the researcher may want to leave some cores free for other processes
wait_to_repost	Amount of time in hours to wait to repost expired tasks after all batches have been posted. Default is 2 hours. This this should be at least as long as the duration of the tasks set in the GUI, as only those tasks that have not been completed by this time are considered expired. This is also the time that the system will wait after reposting to download the complete data if any are expired.
...	Additional arguments passed to either scan() or read.table() depending on type of data used

Details

The SentimentIt system is designed to estimate a researcher-specified latent trait of interest for text corpora. The system is introduced in Carlson and Montgomery "A pairwise comparison framework for fast, flexible, and reliable human coding of political texts." The system crowd-sources pairwise comparisons of hte documents, currently through Amazon Mechanical Turk. This is a cognitively simple task and much easier for workers to achieve reliably than thermometer-type coding or similar approaches. The workers are simply asked which of the two documents is further along some dimension of interest, e.g. positivity. Once a sufficient number of comparisons are done per document (when randomly selecting comparisons we find 20 is a reasonable number), the traits are estimated using a random utility model via Hamiltonian MCMC in Stan. This process generates both point estimates and full posterior distributions. This is a reliable and efficient way to estimate the underlying sentiment in text, a task often too difficult for machines and too time-consuming, costly, and unreliable for expert coders.

This package is designed to allow nearly full implementation of the system. Some start-up is necessary outside of the R environment. To begin using SentimentIt, navigate to https://github.com/carlson9/SentimentIt-Public. Here you will find detailed information on setting up a SentimentIt account, setting up an MTurk account, linking the two, setting up certifications and training for workers, and creating HIT settings. Once these steps have been completed, all that is needed is the HIT Setting ID number you wish to use for your task. You can then proceed to use the package. The documentation also explains the purpose of these settings and guidance for best practices.

First, the base-level functionality of the package is explained to give an intuition behind the process. We then show how the entire process can be done using the wrapper function sentimentIt. All

argument names are the same in the wrapper function as they are in base-level functions. Data on Rotten Tomato movie reviews comes with the package (reviews) and we will use this data to demonstrate how to uncover the positivity of a movie review which can then be benchmarked to the stars the reviewer provided.

We begin by reading in the data using the readText function. If we want to use the movie review data in the package, we could run the following code:

```
data(reviews)

docInfo <- readText(email = 'researcher@school.edu',  password = 'uniquePassword',  read_documents_f
```

This would load the reviews data, read the text from the column specified by index (which can alternatively be a numeric for the column), put the text on the SentimentIt server, which will return unique ID numbers for the documents, and the function readText will both return a dataframe with the inputted data with the IDs appended, and write this data frame to "ReviewsWithIds.csv". View the function documentation for all possible arguments, which, again, have the same names as the higher level wrapper function.

Now that the data is on the server and we have the document IDs, we can create the batches of pairwise comparisons. We start with 10 comparisons per document, leading to 2,500 comparisons. We wish to send them out in batches of 1,000, so we need three batches. We run the following function:

```
batch_ids <- createBatches(email = 'researcher@school.edu',  password = 'uniquePassword',  task_sett
```

This assigns batch_ids to the batch identification numbers returned from the server. The first argument following authentication, task_setting_id refers to the HIT setting we wish to use. These can be set on our server using a simple form to indicate what certification (if any) is required, how long workers have to complete the task once they have started, how much money they should be paid, and how long HITs should be posted for completion before expiration. In this case, the HIT setting requires a training certification, pays the workers \$0.04 per HIT, allows the worker to take up to an hour answering the question, and leaves the HIT active on MTurk for 2 hours. This also dictates what the workers will see before selecting the HIT, which should include the URL to the certification if a certification is used. If the workers follow the link and successfully complete the training, their worker ID will be automatically added to the list of approved workers and complete the associated HITs. If they fail the certification, they are banned from retaking the training or answering HITs associated with this setting.

Now that the batch settings are specified, we can create 10 random pairwise comparisons. We first need to retrieve the document IDs created earlier, written to the file specified, then create the comparisons using these IDs. The following code will set up the comparisons:

```
makeCompsSep(email = 'researcher@school.edu',  password = 'uniquePassword',  ids = docInfo[,'ids'],
Please choose the text that you think comes          from the most positive review',  pairwise_path =
```

The first argument, ids, indicates the numerical IDs for the documents. The argument number_per is the number of comparisons desired (in this case 10). The batch_id argument indicates the batch IDs to be used for the HITs. The question argument specifies the question the worker will see once the worker selects the HIT. There is an argument per_batch indicating the number of comparisons per batch desired, defaulted to 1,000. If the number of comparisons is not a multiple of this number, the final batch will have fewer comparisons. We have 2,500 comparisons, so the final batch only has 500 comparisons. The number of comparisons per batch could have been automatically determined, but forcing this number to be provided ensures no mistakes are made, such as providing too few batches. The pairwise_path argument is used to save the comparisons that were created to a

specified path and file name where the comparisons should be stored. The function returns the batch IDs as returned by the server (which we already have stored). This serves as an assurance that the function has been correctly called. Full argument documentation is available under makeCompsSep, and all arguments have the same name as the wrapper.

Now that the comparisons are set up and on the server, we can post them as HITs to AMT. If we wish to send our first batch, we run:

```
createTasks(email = 'researcher@school.edu',  password = 'uniquePassword',  batch_id = batch_ids[1])
```

The argument batch_id is the identification number for the batch that we want to send, which we retrieved from the call to createBatches. Full documentation for the arguments is available under createTasks.

At this point, we want to occasionally check the status of a batch. That is, how many of the comparisons have been completed. To do this we run the function:

```
batchStatus(email = 'researcher@school.edu',  password = 'uniquePassword',  batch_id = batch_ids[1])
```

The only argument to this function, other than authentication, is batch_id, the ID of the batch you wish to check. This could be a vector of batch IDs. This returns a dataframe with the batch ID and the number of comparisons total, submitted, completed, and expired.

Once the batch is completed (or near completed), we can check the workers to find any that are deviant. First, we need to read in the data from the server. We accomplish this by running:

```
output <- readInData(email = 'researcher@school.edu',  password = 'uniquePassword',  batch_id = batc
```

The argument to this function is batch_id, which could be a scalar or a vector of batch ID numbers. The returned output is a data frame with the following columns: batch_id, comparison_id, document_id, result, hit_id, worker_id, and completed_at. The data is organized by document-comparison, and the result is an indicator if the document was chosen over the other document in the given comparison. (There are, therefore, two rows for every comparison conveniently grouped by comparison so every odd row is immediately followed by an entry for the other document in the same comparison.) The worker_id is the AMT identification number of the worker, important for keeping track of the performance of workers to determine deviant (or highly reliable) workers. Finally, completed_at is a time stamp for the HIT completion time. This can be used to determine how quickly workers are completing tasks. If a worker is finishing HITs in a very fast amount of time this may suggest the worker is simply clicking through as fast as possible. In our experience only a very small proportion of workers do this, and it is quite obvious if workers are simply providing insincere evaluations.

We now need to fit the Stan model to estimate worker reliability. For the non-hierarchical model, which we used in this example, we run:

```
fit <- fitStan(data = output)
```

The data can alternatively be a (vector of) batch number(s) rather than actual data, allowing the researcher to skip the earlier step. See the documentation here or under fitStan for a full list of arguments. The latter arguments are all used in the call to Stan through rstan.

There is a related function that fits the hierarchical Stan model, fitStanHier. The arguments are the same as fitStan, but with two additional arguments, hierarchy_data and hierarchy_var. In order to fit the hierarchical model, the data used to set up the documents and comparisons needs to be provided in order to map the document IDs to their respective higher-level grouping. The higher-level group could be long documents while the lower level is paragraphs within the documents.

Once the model is fitted, we can check for outlier workers. To do so, we run:

```
ban_workers <- checkWorkers(stan_fit = fit, data = output)
```

The first argument is the `rstan` object obtained from the previous step, and the data is the output from the batch(es). The function has optional arguments with defaults. Read the documentation here or under `checkWorkers`.

We want to revoke the certification for these workers and add them to the list of banned workers for this task. We keep track of banned workers by granting them a different certification that indicates their certifications have been revoked. This is also how we keep track of workers that fail the qualification. First, to revoke the certification, we run:

```
revokeCert(email = 'researcher@school.edu',  password = 'uniquePassword',  cert = 'snippets', worker
```

The `cert` argument is the name of the certification as used on the server, which in this case is titled snippets for movie reviews. The next argument is a vector of worker IDs obtained from the previous step. We now add them to banned list:

```
createCert(email = 'researcher@school.edu',  password = 'uniquePassword',  cert= 'bannedmovie_revie
```

This will grant the workers the certification bannedmovie_reviews which indicates they can no longer participate in HITs requiring the snippets qualification.

We can now proceed with posting the other batches, checking workers whenever we choose. Once all the batches are completed, we find it common that a few HITs remain incomplete with a few HITs per batch overlooked. We can then run:

```
repostExpired(email = 'researcher@school.edu',  password = 'uniquePassword',  batch_id = batch_ids)
```

This will repost all of the expired HITs from the vector of batch IDs. Finally, we can run `readInData` and `fitStan` to retrieve final estimates of all the data.

Finally, after all the data has been retrieved, run `signout(email, password)` to sign out of the session. The function `authenticate` signs the researcher in, but this function is called internally and there should be no reason the researcher would need this functionality, unless using the API directly. Note that if the qualification survey remains active, the researcher will be signed back in if the survey is taken. This should not cause issues but taking a survey offline will prevent this from happening.

The functionality discussed to this point is the lowest level of functionality, where the researcher has the most control. All of this functionality can be automated through the wrapper function, allowing new batches to be posted either as a function of time or as batches are (nearly) completed. The example provided in the Examples section would perform the following:

- All documents in the `reviews` dataframe will be read in and passed to our servers.
- HIT settings for the task will be assigned (question wording, compensation, duration, etc.) and the snippet certification will be required.
- Ten random pairwise comparisons per document will be created.
- All unique identifiers for documents will be stored at ReviewsWithIds.csv
- Comparisons will be posted to AMT in batches of 1,000.
- New batches will be posted once the current batch is completed up until the `threshold` of five incomplete comparisons. Completion status will be checked every `rate=1/3` of an hour.
- A Stan model with three chains and 2,500 iterations will be fit when the workers are checked and when the final data is analyzed.

- Workers will be evaluated after the first 1,000 and second 1,000 comparisons are complete, and workers with `0.9` of the posterior draws falling below the default cut point of `b_k=1`, will be banned from completing future tasks.
- After posting comparisons, the function will wait `rest_time = 60` seconds before posting HITs to Mechanical Turk to ensure all of the comparisons are ready to be posted.
- All incomplete tasks will be re-posted.
- After all tasks are completed, the data and Stan estimates of all model parameters will be returned and the researcher will be signed out.

There are several advantages to this higher-level approach. First and foremost, this functionality makes the process of interacting with online workers simple and efficient from the perspective of a researcher. Once the qualifications and HIT settings have been created, a single command can supervise the collection of worker evaluations and the creation of a meaningful measure, even if this process requires several days. However, a further advantage of this approach is that it makes the process of turning text into data highly replicable. Researchers wishing to evaluate the reliability of any measure can simply re-run the task using the original call to create a replication of the original measure. Thus, the SentimentIt platform has the potential to bring about a higher degree of transparency to the task of turning natural language into meaningful data.

Some important things to note, however:

- The wrapper function will take over your R session, so you'll to start a new session to continue using R or run this from a terminal. If running a Linux/Unix environment the software `screen` can allow running commands from the terminal and will continue running even if the terminal window closes.
- It is recommended that all saving options are utilized. If a power outage or system crash interrupts the process having the information saved will greatly ease starting it back up.
- Because Stan will intermittently run on the specified number of cores, your machine may lose computing power throughout the process. Consider this when specifying the number of cores.

Value

sentimentItOut A list with the requested information returned. The data with document IDs will always be returned, regardless if saved elsewhere.

Author(s)

David Carlson <carlson.david@wustl.edu> and Jacob M. Montgomery <jacob.montgomery@wustl.edu>

See Also

`authenticate` `batchStatus` `checkCert` `checkWorkers` `createBatches` `createCert` `createPairwise` `createTasks` `fitStan` `fitStanHier` `makeCompsSep` `readInData` `readText` `repostExpired` `reviews` `revokeCert` `signout`

Examples

```
## Not run:
```

```
data(reviews)
movies <- sentimentIt(email = 'researcher@school.edu',
   password = 'uniquePassword',
   read_documents_from = reviews,
   write_documents_to = 'ReviewsWithIds.csv',
   index = 'Review', task_setting_id = 8,
   number_per = 10,
   question = 'Below is text taken from
       two movie reviews. Please
       choose the text that you
       think comes from the most
       positive review',
   pairwise_path = 'Comparisons.Rdata',
   certone = 'snippets', certtwo = 'bannedmovie_reviews',
   timed = FALSE, check_workers_at = c(1,2),
   rest_time = 60, rate = 1/3, threshold = 5,
   return_stan = TRUE, return_data = TRUE)

## End(Not run)
```

`signout` *Signs the researcher out of the session*

Description

This function will revoke the authorization token and sign the researcher out of the session. However, if the certification surveys remain active, if someone takes the survey the researcher will be signed back in.

Usage

```
signout(email, password)
```

Arguments

`email`	The email used by the researcher to register with SentimentIt.
`password`	The password associated with the account.

Value

success TRUE if sign out succeeded. Function will be called recursively until success.

Author(s)

David Carlson

See Also

sentimentIt authenticate batchStatus checkCert checkWorkers createBatches createCert createPairwise createTasks fitStan fitStanHier makeCompsSep readInData readText repostExpired reviews revokeCert

Index