Package 'Twitmo'

October 12, 2022

Type Package

Title Twitter Topic Modeling and Visualization for R

Version 0.1.2

Description Tailored for topic modeling with tweets and fit for visualization tasks in R. Collect, pre-process and analyze the contents of tweets using LDA and structural topic models (STM). Comes with visualizing capabilities like tweet and hashtag maps and built-in support for 'LDAvis'.

License MIT + file LICENSE

URL https://github.com/abuchmueller/Twitmo

BugReports https://github.com/abuchmueller/Twitmo/issues

Encoding UTF-8

Depends R (>= 3.5.0)

Imports jsonlite, stats, plyr, stopwords, stringr, dplyr, readr, magrittr, rtweet, quanteda, quanteda.textstats, topicmodels, stm, tidyr, rlang, maps, LDAvis, leaflet, ldatuning, stringi, tm

RoxygenNote 7.1.2

Suggests rmarkdown, knitr, tidytext, modeltools, servr

NeedsCompilation no

Author Andreas Buchmueller [aut, cre] (github.com/abuchmueller), Gillian Kant [aut, ths] (<https://orcid.org/0000-0003-2346-2841>), Christoph Weisser [aut, ths] (<https://orcid.org/0000-0003-0616-1027>), Benjamin Saefken [aut, ths] (<https://orcid.org/0000-0003-4702-3333>), Thomas Kneib [rev, ths, dgs] (<https://orcid.org/0000-0003-3390-0972>), Krisztina Kis-Katos [rev] (<https://orcid.org/0000-0003-2459-1274>)

Maintainer Andreas Buchmueller <a.buchmueller@stud.uni-goettingen.de>

Repository CRAN

Date/Publication 2021-12-06 15:30:02 UTC

R topics documented:

`cluster_tweets` *Cluster tweets on an interactive map*

Description

Plot into clusters on an interactive map

Usage

```
cluster_tweets(data, ...)
```

Arguments

`data`	A data frame of tweets parsed by load_tweets or returned by pool_tweets.
`...`	Extra arguments passed to markerClusterOptions

Details

This function can be used to create interactive maps on OpenStreetView.

Value

Interactive leaflet map

See Also

tileOptions

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

pool <- pool_tweets(mytweets)
cluster_tweets(mytweets)

# OR
cluster_tweets(pool$data)

## End(Not run)
```

`filter_tweets` *Filter tweets*

Description

Filter tweets by keywords.

Usage

```
filter_tweets(data, keywords, include = TRUE)
```

Arguments

`data`	Data frame of parsed tweets. Obtained either by using `load_tweets` or `stream_in` in conjunction with `tweets_with_users`.
`keywords`	Character string of keywords for black- or whitelisting provided via a comma separated character string.
`include`	Logical. Indicate where to perform exclusive or inclusive filtering. Inclusive filtering is akin to whitelisting keywords. Exclusive filtering is blacklisting certain keywords.

Details

Use this function if you want your Tweets to contain certain keywords. This can be used for iterative filtering to create more coherent topic models. Keyword filtering is always case insensitive (lowercase).

Value

Data frame of Tweets containing specified keywords

Examples

```
## Not run:

library(Twitmo)


# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Exclude Tweets that mention "football" and/or "mood"
keyword_dict <- "football,mood"
mytweets_reduced <- filter_tweets(mytweets, keywords = keyword_dict, include = FALSE)

## End(Not run)
```

`find_lda` *Find best LDA model*

Description

Find the optimal hyperparameter k for your LDA model

Usage

```
find_lda(pooled_dfm, search_space = seq(1, 10, 2), method = "Gibbs", ...)
```

Arguments

`pooled_dfm`	object of class dfm (see dfm) containing (pooled) tweets
`search_space`	Vector with number of topics to compare different models.
`method`	The method to be used for fitting. Currently method = "VEM" or method = "Gibbs" are supported.
`...`	Additional arguments passed to FindTopicsNumber.

Value

Plot with different metrics compared.

See Also

FindTopicsNumber

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Pool tweets into longer pseudo-documents
pool <- pool_tweets(data = mytweets)
pooled_dfm <- pool$document_term_matrix

# use the ldatuner to compare different K
find_lda(pooled_dfm, search_space = seq(1, 10, 1),  method = "Gibbs")

## End(Not run)
```

`find_stm` *Find best STM/CTM*

Description

Gridsearch for optimal K for your STM/CTM

Usage

```
find_stm(data, search_space = seq(4, 20, by = 2), ...)
```

Arguments

`data`	Either a pooled dfm object returned by pool_tweets or a named list of pre-processed tweets for stm modeling returned by `fit_stm`.
`search_space`	Vector with number of topics to compare different models.
`...`	Additional parameters passed to searchK

Details

Wrapper function around `searchK` for pooled dfm objects returned by pool_tweets and prepped stm documents returned by `fit_stm`.

Value

Plot with different metrics compared.

See Also

searchK

searchK

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Pool tweets into longer pseudo-documents
pool <- pool_tweets(data = mytweets)
pooled_dfm <- pool$document_term_matrix

# compare different K for CTM
find_stm(pooled_dfm, search_space = seq(1, 10, 1))

# OR

# compare different K for STM
prepped_stm <- stm_model$prep
find_stm(prepped_stm, search_space = seq(4, 16, by = 2))

## End(Not run)
```

`fit_ctm` *Fit CTM (Correlated topic model)*

Description

Estimate a CTM topic model.

Usage

```
fit_ctm(pooled_dfm, n_topics = 2L, ...)
```

Arguments

`pooled_dfm`	Object of class dfm (see dfm) containing (pooled) Tweets.
`n_topics`	Integer with number of topics
`...`	Additional arguments passed to stm.

Value

Object of class stm

See Also

stm

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Pool tweets into longer pseudo-documents
pool <- pool_tweets(data = mytweets)
pooled_dfm <- pool$document_term_matrix

# fit your CTM with 7 topics
ctm_model <- fit_ctm(pooled_dfm, n_topics =7)

## End(Not run)
```

`fit_lda` *Fit LDA Topic Model*

Description

Estimate a LDA topic model using VEM or Gibbs Sampling.

Usage

```
fit_lda(pooled_dfm, n_topics, ...)
```

Arguments

`pooled_dfm`	Object of class dfm (see dfm) containing (pooled) tweets.
`n_topics`	Integer with number of topics.
`...`	Additional arguments passed to LDA.

Value

Object of class LDA.

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Pool tweets into longer pseudo-documents
```

```
pool <- pool_tweets(data = mytweets)
pooled_dfm <- pool$document_term_matrix

# fit your LDA model with 7 topics
model <- fit_lda(pooled_dfm, n_topics = 7, method = "Gibbs")

## End(Not run)
```

`fit_stm` *Fit STM (Structural topic model)*

Description

Estimate a structural topic model

Usage

```
fit_stm(
  data,
  n_topics = 2L,
  xcov,
  remove_punct = TRUE,
  stem = TRUE,
  remove_url = TRUE,
  remove_emojis = TRUE,
  stopwords = "en",
  ...
)
```

Arguments

`data`	Data frame of parsed tweets. Obtained either by using `load_tweets` or `stream_in` in conjunction with `tweets_with_users`.
`n_topics`	Integer with number of topics.
`xcov`	Either a \[stats]formula with an empty left-hand side specifying external covariates (meta data) to use.e.g. `~favourites_count + retweet_count` or a character vector (`c("favourites_count", "retweet_count")`) or comma seperated character string (`"favourites_count,retweet_count"`) with column names implying which metadata to use as external covariates.
`remove_punct`	Logical. Indicates whether punctuation (includes Twitter hashtags and usernames) should be removed. Defaults to TRUE.
`stem`	Logical. If `TRUE` turn on word stemming for terms.
`remove_url`	Logical. If `TRUE` find and eliminate URLs beginning with http(s).
`remove_emojis`	Logical. If `TRUE` all emojis will be removed from tweets.
`stopwords`	a character vector, list of character vectors, dictionary or collocations object. See pattern for details. Defaults to stopwords("english").
`...`	Additional arguments passed to stm.

Details

Use this to function estimate a STM from a data frame of parsed Tweets. Works with unpooled Tweets only. Pre-processing and fitting is done in one run.

Value

Object of class stm. Additionally, pre-processed documents are appended into a named list called "prep".

See Also

stm

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# fit STM with tweets
stm_model <- fit_stm(mytweets, n_topics = 7,
                     xcov = ~ retweet_count + followers_count + reply_count +
                     quote_count + favorite_count,
                     remove_punct = TRUE,
                     remove_url = TRUE,
                     remove_emojis = TRUE,
                     stem = TRUE,
                     stopwords = "en")

## End(Not run)
```

---

get_tweets *Sample tweets by streaming or searching*

Description

Collect Tweets via streaming or searching.

Usage

```
get_tweets(
  method = "stream",
  location = c(-180, -90, 180, 90),
  timeout = Inf,
  keywords = "",
```

```
  n_max = 100L,
  file_name = NULL,
  ...
)
```

Arguments

method	Character string. Supported methods are streaming and searching. The default method is streaming `method = 'stream'`. This is the recommended method as it allows to collect larger volumes of data over time. Use `method = 'search'` if you want to collect Tweets from the past 9 days.
location	Character string of location to sample from. Can be a three letter country code i.e. "USA" or a city name like "berlin". Use `Twitmo:::bbox_country` for all supported country locations or `rtweet:::citycoords` for a list of supported cities. Alternatively, use a vector of doubles with four latitude/longitude bounding box points provided via a vector of length 4, in the following format c(sw.long, sw.lat, ne.long, ne.lat) e.g., c(-125, 26, -65, 49).
timeout	Integer. Limit streaming time in seconds. By default will stream indefinitely until user interrupts by pressing [ctrl + c].
keywords	Character string of keywords provided via a comma separated character string. Only for searching Tweets.If you want to stream Tweets for a certain location AND filter by keywords use the location parameter and after sampling use the filter_tweets function. If you are using the search method instead of streaming keywords WILL work together with a location but will yield only a very limited number of Tweets.
n_max	Integer value. Only applies to the `search` method. Limit how many Tweets are collected.
file_name	Character string of desired file path and file name where Tweets will be saved. If not specified, will write to stream_tweets.json in the current working directory.
...	Additional arguments passed to stream_tweets or search_tweets.

Details

A function that calls on stream_tweets and search_tweets (depending on the specified method) and is specifically tailored for sampling geo-tagged data. This function provides supports additional arguments like location for convenient sampling of geo-tagged Tweets. Tweets can be searched up to 9 days into the past.

Value

Either a json file in the specified directory.

References

https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets https://developer.twitter.com/en/docs/twitter-api/v1/tweets/sample-realtime/api-reference/get-statuses-sample

See Also

stream_tweets, search_tweets

Examples

```
## Not run:

# live stream tweets from Germany for 60 seconds and save to current working directory
get_tweets(method = "stream",
           location = "DEU",
           timeout = 60,
           file_name = "german_tweets.json")

# OR
# live stream tweets from berlin for an hour
get_tweets(method = "stream",
           location = "berlin",
           timeout = 3600,
           file_name = "berlin_tweets.json")

# OR
# use your own bounding box coordinates to strean tweets indefinitely (interrupt to stop)
get_tweets(method = 'stream',
           location = c(-125, 26, -65, 49),
           timeout = Inf)


## End(Not run)
```

---

lda_distribution *View distribution of fitted LDA Models*

Description

View the distribution of your fitted LDA model.

Usage

```
lda_distribution(lda_model, param = "gamma", tidy = FALSE)
```

Arguments

lda_model	Object of class LDA).
param	String. Specify either "beta" to return the term distribution over topics (term per document) or "gamma" for the document distribution over. topics (i.e. hashtag pool per topic probability).
tidy	Logical. Specify TRUE for return distribution in tidy format (tbl).

Value

Data frame or tbl of Term (beta) or document (gamma) distribution over topics.

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Pool tweets into longer pseudo-documents
pool <- pool_tweets(data = mytweets)
pooled_dfm <- pool$document_term_matrix

# fit your LDA model with 7 topics
model <- fit_lda(pooled_dfm, n_topics = 7, method = "Gibbs")

# Choose either "beta" to return the term distribution
# over topics (term per document) or "gamma" for the document distribution over
# topics (hashtag pool per topic probability)
lda_distribution(model, param = "gamma")

## End(Not run)
```

---

lda_hashtags *View Documents (hashtags) heavily associated with topics*

Description

Convenience Function to extract the most likely topics for each hashtag.

Usage

```
lda_hashtags(lda_model)
```

Arguments

lda_model	Fitted LDA Model. Object of class LDA).

Value

Data frame with most likely topic for each hashtag.

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Pool tweets into longer pseudo-documents
pool <- pool_tweets(data = mytweets)
pooled_dfm <- pool$document_term_matrix

# fit your LDA model with 7 topics
model <- fit_lda(pooled_dfm, n_topics = 7, method = "Gibbs")

lda_hashtags(model)

## End(Not run)
```

`lda_terms` *View Terms heavily associated with each topic*

Description

Convenience Function to extract the most likely terms for each topic.

Usage

```
lda_terms(lda_model, n_terms = 10)
```

Arguments

`lda_model`	Fitted LDA Model. Object of class LDA).
`n_terms`	Integer number of terms to return.

Value

Data frame with top n terms for each topic.

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Pool tweets into longer pseudo-documents
```

```
pool <- pool_tweets(data = mytweets)
pooled_dfm <- pool$document_term_matrix

# fit your LDA model with 7 topics
model <- fit_lda(pooled_dfm, n_topics = 7, method = "Gibbs")


## End(Not run)
```

`load_tweets` *Converts Twitter stream data (JSON file) into parsed data frame*

Description

Parse JSON files of collected Tweets

Usage

```
load_tweets(file_name)
```

Arguments

`file_name`	Character string. Name of JSON file with data collected by stream_tweets or `get_tweets()`.

Details

This function replaces parse_stream which has been deprecated in rtweet 0.7 but is included here to ensure backwards compatibility for data streamed with older versions of `rtweet`. Alternatively stream_in in conjunction with tweets_with_users and lat_lng can be used if data has been collected with rtweet 0.7 or newer.

Value

A data frame of tweets data with additional meta data

See Also

parse_stream, stream_in, tweets_with_users

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
raw_path <- system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo")
```

```
mytweets <- load_tweets(raw_path)

## End(Not run)
```

`plot_hashtag` *Plot tweets containing certain hashtag*

Description

Plot the locations of certain hashtag on a static map with base plot.

Usage

```
plot_hashtag(
  data,
  region = ".",
  alpha = 0.01,
  hashtag = "",
  ignore_case = TRUE,
  ...
)
```

Arguments

`data`	A data frame of tweets parsed by load_tweets or returned by pool_tweets.
`region`	Character vector specifying region. Returns a world map by default. For higher resolutions specify a region.
`alpha`	A double between 0 and 1 specifying the opacity of plotted points. See iso3166 for country codes.
`hashtag`	Character vector of the hashtag you want to plot.
`ignore_case`	Logical, if TRUE will ignore case of hashtag.
`...`	Extra arguments passed to polygon or lines.

Details

This function can be used to generate high resolution spatial plots of hashtags Works with data frames of tweets returned by pool_tweets as well as data frames read in by load_tweets and then augmented by lat/lng coordinates with lat_lng. For larger view resize the plot window then call `plot_tweets` again.

Value

Maps where each dot represents a tweet.

See Also

map, iso3166

Examples

```
## Not run:
library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Plot tweets on mainland USA region
plot_hashtag(mytweets,
             region = "USA(?!:Alaska|:Hawaii)",
             hashtag = "breakfast",
             ignore_case=TRUE,
             alpha=1)

# Add title
title("My hashtags on a map")

## End(Not run)
```

---

plot_tweets *Plot tweets on a static map*

Description

Plot tweets on a static map with base plot.

Usage

```
plot_tweets(data, region = ".", alpha = 0.01, ...)
```

Arguments

`data`	A data frame of tweets parsed by load_tweets or returned by pool_tweets.
`region`	Character vector specifying region. Returns a world map by default. For higher resolutions specify a region.
`alpha`	A double between 0 and 1 specifying the opacity of plotted points. See iso3166 for country codes.
`...`	Extra arguments passed to polygon or lines.

Details

This function can be used to generate high resolution spatial plots of tweets. Works with data frames of tweets returned by pool_tweets as well as data frames read in by load_tweets and then augmented by lat/lng coordinates with lat_lng. For larger view resize the plot window then call `plot_tweets` again.

Value

Maps where each dot represents a tweet.

See Also

map, iso3166

Examples

```
## Not run:

library(Twitmo)

# Plot tweets on mainland USA
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

plot_tweets(mytweets, region = "USA(?!:Alaska|:Hawaii)", alpha=1)
# Add title
title("My tweets on a map")

## End(Not run)
```

---

pool_tweets *Prepare Tweets for topic modeling by pooling*

Description

This function pools a data frame of parsed tweets into document pools.

Usage

```
pool_tweets(
  data,
  remove_numbers = TRUE,
  remove_punct = TRUE,
  remove_symbols = TRUE,
  remove_url = TRUE,
  remove_emojis = TRUE,
  remove_users = TRUE,
  remove_hashtags = TRUE,
  cosine_threshold = 0.9,
  stopwords = "en",
  n_grams = 1L
)
```

Arguments

data	Data frame of parsed tweets. Obtained either by using load_tweets or stream_in in conjunction with tweets_with_users.
remove_numbers	Logical. If TRUE remove tokens that consist only of numbers, but not words that start with digits, e.g. 2day. See tokens.
remove_punct	Logical. If TRUE remove all characters in the Unicode "Punctuation" [P] class, with exceptions for those used as prefixes for valid social media tags if preserve_tags = TRUE. See tokens
remove_symbols	Logical. If TRUE remove all characters in the Unicode "Symbol" [S] class.
remove_url	Logical. If TRUE find and eliminate URLs beginning with http(s).
remove_emojis	Logical. If TRUE all emojis will be removed from tweets.
remove_users	Logical. If TRUE will remove all mentions of user names from documents.
remove_hashtags	Logical. If TRUE will remove hashtags (not only the symbol but the hashtagged word itself) from documents.
cosine_threshold	Double. Value between 0 and 1 specifying the cosine similarity threshold to be used for document pooling. Tweets without a hashtag will be assigned to document (hashtag) pools based upon this metric. Low thresholds will reduce topic coherence by including a large number of tweets without a hashtag into the document pools. Higher thresholds will lead to more coherent topics but will reduce document sizes.
stopwords	a character vector, list of character vectors, dictionary or collocations object. See pattern for details. Defaults to stopwords("english").
n_grams	Integer vector specifying the number of elements to be concatenated in each n-gram. Each element of this vector will define a n in the n-gram(s) that are produced. See tokens_ngrams

Details

Pools tweets by hashtags using cosine similarity to create longer pseudo-documents for better LDA estimation and creates n-gram tokens. The method applies an implementation of the pooling algorithm from Mehrotra et al. 2013.

Value

List with corpus object and dfm object of pooled tweets.

References

Mehrotra, Rishabh & Sanner, Scott & Buntine, Wray & Xie, Lexing. (2013). Improving LDA Topic Models for Microblogs via Tweet Pooling and Automatic Labeling. 889-892. 10.1145/2484028.2484166.

See Also

tokens, dfm

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

pool <- pool_tweets(data = mytweets,
                    remove_numbers = TRUE,
                    remove_punct = TRUE,
                    remove_symbols = TRUE,
                    remove_url = TRUE,
                    remove_users = TRUE,
                    remove_hashtags = TRUE,
                    remove_emojis = TRUE,
                    cosine_threshold = 0.9,
                    stopwords = "en",
                    n_grams = 1)

## End(Not run)
```

`predict_lda` *Predict topics of tweets using fitted LDA model*

Description

Predict topics of tweets using fitted LDA model.

Usage

```
predict_lda(
  data,
  lda_model,
  response = "max",
  remove_numbers = TRUE,
  remove_punct = TRUE,
  remove_symbols = TRUE,
  remove_url = TRUE
)
```

Arguments

`data`	Data frame of parsed tweets. Obtained either by using `load_tweets` or `stream_in` in conjunction with `tweets_with_users`.
`lda_model`	Fitted LDA Model. Object of class LDA.
`response`	Type of response. Either "prob" for probabilities or "max" one topic (default).

remove_numbers	Logical. If TRUE remove tokens that consist only of numbers, but not words that start with digits, e.g. 2day. See tokens.
remove_punct	Logical. If TRUE remove all characters in the Unicode "Punctuation" [P] class, with exceptions for those used as prefixes for valid social media tags if preserve_tags = TRUE. See tokens
remove_symbols	Logical. If TRUE remove all characters in the Unicode "Symbol" [S] class.
remove_url	Logical. If TRUE find and eliminate URLs beginning with http(s).

Value

Data frame of topic predictions or predicted probabilities per topic (see response).

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Pool tweets into longer pseudo-documents
pool <- pool_tweets(data = mytweets)
pooled_dfm <- pool$document_term_matrix

# fit your LDA model with 7 topics
model <- fit_lda(pooled_dfm, n_topics = 7, method = "Gibbs")

# Predict topics of tweets using your fitted LDA model
predict_lda(mytweets, model, response = "prob")

## End(Not run)
```

---

to_ldavis *Create interactive visualization with LDAvis*

Description

Converts LDA topic model to LDAvis compatible json string and starts server. May require servr Package to run properly. For conversion of STM topic models use toLDAvis.

Usage

```
to_ldavis(fitted, corpus, doc_term)
```

Arguments

fitted	Fitted LDA Model. Object of class LDA)
corpus	Document corpus. Object of class corpus)
doc_term	document term matrix (dtm).

Details

Beware that `to_ldavis` might fail if the corpus contains documents that consist ONLY of numbers, emojis or punctuation e.g. do not contain a single character string. This is due to a limitation in the `topicmodels` package used for model fitting that does not consider such terms as words and omits them causing the posterior to differ in length from the corpus. If you encounter such an error, redo your pre-processing and exclude emojis, punctuation and numbers. When using `pool_tweets` you can remove emojis by specifying `remove_emojis = TRUE`.

Value

Invisible Object (see serVis)).

See Also

toLDAvis

Examples

```
## Not run:

library(Twitmo)

# load tweets (included in package)
mytweets <- load_tweets(system.file("extdata", "tweets_20191027-141233.json", package = "Twitmo"))

# Pool tweets into longer pseudo-documents
pool <- pool_tweets(data = mytweets)
pooled_dfm <- pool$document_term_matrix
pooled_corp <- pool$corpus

# fit your LDA model with 7 topics
model <- fit_lda(pooled_dfm, n_topics = 7, method = "Gibbs")

# Explore your topics with LDAvis
to_ldavis(model, pooled_corp, pooled_dfm)

## End(Not run)
```

Index