Package 'LDAvis'

July 21, 2025

Title Interactive Visualization of Topic Models

Version 0.3.2

Description Tools to create an interactive web-based visualization of a topic model that has been fit to a corpus of text data using Latent Dirichlet Allocation (LDA). Given the estimated parameters of the topic model, it computes various summary statistics as input to an interactive visualization built with D3.js that is accessed via a browser. The goal is to help users interpret the topics in their LDA topic model.

Depends R (>= 2.10)

Imports proxy, RJSONIO, parallel

License MIT + file LICENSE

Suggests mallet, lda, topicmodels, gistr (>= 0.0.8.99), servr, shiny, knitr, rmarkdown, digest, htmltools

LazyData true

VignetteBuilder knitr

URL https://github.com/cpsievert/LDAvis

BugReports https://github.com/cpsievert/LDAvis/issues

NeedsCompilation no

Author Carson Sievert [aut, cre], Kenny Shirley [aut]

Maintainer Carson Sievert <cpsievert1@gmail.com>

Repository CRAN

Date/Publication 2015-10-24 08:21:16

Contents

`createJSON` *Create the JSON object to read into the javascript visualization*

Description

This function creates the JSON object that feeds the visualization template. For a more detailed overview, see `vignette("details", package = "LDAvis")`

Usage

```
createJSON(phi = matrix(), theta = matrix(), doc.length = integer(),
  vocab = character(), term.frequency = integer(), R = 30,
  lambda.step = 0.01, mds.method = jsPCA, cluster, plot.opts = list(xlab =
  "PC1", ylab = "PC2"), ...)
```

Arguments

`phi`	matrix, with each row containing the distribution over terms for a topic, with as many rows as there are topics in the model, and as many columns as there are terms in the vocabulary.
`theta`	matrix, with each row containing the probability distribution over topics for a document, with as many rows as there are documents in the corpus, and as many columns as there are topics in the model.
`doc.length`	integer vector containing the number of tokens in each document of the corpus.
`vocab`	character vector of the terms in the vocabulary (in the same order as the columns of `phi`). Each term must have at least one character.
`term.frequency`	integer vector containing the frequency of each term in the vocabulary.
`R`	integer, the number of terms to display in the barcharts of the interactive viz. Default is 30. Recommended to be roughly between 10 and 50.
`lambda.step`	a value between 0 and 1. Determines the interstep distance in the grid of lambda values over which to iterate when computing relevance. Default is 0.01. Recommended to be between 0.01 and 0.1.
`mds.method`	a function that takes `phi` as an input and outputs a K by 2 data.frame (or matrix). The output approximates the distance between topics. See jsPCA for details on the default method.
`cluster`	a cluster object created from the parallel package. If supplied, computations are performed using parLapply instead of lapply.

plot.opts	a named list used to customize various plot elements. By default, the x and y axes are labeled "PC1" and "PC2" (principal components 1 and 2), since jsPCA is the default scaling method.
...	not currently used.

Details

The function first computes the topic frequencies (across the whole corpus), and then it reorders the topics in decreasing order of frequency. The main computation is to loop through the topics and through the grid of lambda values (determined by `lambda.step`) to compute the `R` most *relevant* terms for each topic and value of lambda.

Value

A string containing JSON content which can be written to a file or feed into serVis for easy viewing/sharing. One element of this string is the new ordering of the topics.

References

Sievert, C. and Shirley, K. (2014) *LDAvis: A Method for Visualizing and Interpreting Topics*, ACL Workshop on Interactive Language Learning, Visualization, and Interfaces. http://nlp.stanford.edu/events/illvi2014/papers/sievert-illvi2014.pdf

See Also

serVis

Examples

```
## Not run:
data(TwentyNewsgroups, package="LDAvis")
# create the json object, start a local file server, open in default browser
json <- with(TwentyNewsgroups,
             createJSON(phi, theta, doc.length, vocab, term.frequency))
serVis(json) # press ESC or Ctrl-C to kill

# createJSON() reorders topics in decreasing order of term frequency
RJSONIO::fromJSON(json)$topic.order

# You may want to just write the JSON and other dependency files
# to a folder named TwentyNewsgroups under the working directory
serVis(json, out.dir = 'TwentyNewsgroups', open.browser = FALSE)
# then you could use a server of your choice; for example,
# open your terminal, type `cd TwentyNewsgroups && python -m SimpleHTTPServer`
# then open http://localhost:8000 in your web browser

# A different data set: the Jeopardy Questions+Answers data:
# Install LDAvisData (the associated data package) if not already installed:
# devtools::install_github("cpsievert/LDAvisData")
library(LDAvisData)
data(Jeopardy, package="LDAvisData")
```

```
json <- with(Jeopardy,
             createJSON(phi, theta, doc.length, vocab, term.frequency))
serVis(json) # Check out Topic 22 (bodies of water!)

# If you have a GitHub account, you can even publish as a gist
# which allows you to easily share with others!
serVis(json, as.gist = TRUE)

# Run createJSON on a cluster of machines to speed it up
system.time(
json <- with(TwentyNewsgroups,
             createJSON(phi, theta, doc.length, vocab, term.frequency))
)
#   user  system elapsed
# 14.415   0.800  15.066
library("parallel")
cl <- makeCluster(detectCores() - 1)
cl # socket cluster with 3 nodes on host 'localhost'
system.time(
 json <- with(TwentyNewsgroups,
   createJSON(phi, theta, doc.length, vocab, term.frequency,
     cluster = cl))
)
#   user  system elapsed
#  2.006   0.361   8.822

# another scaling method (svd + tsne)
library("tsne")
svd_tsne <- function(x) tsne(svd(x)$u)
json <- with(TwentyNewsgroups,
             createJSON(phi, theta, doc.length, vocab, term.frequency,
                        mds.method = svd_tsne,
                        plot.opts = list(xlab="", ylab="")
                        )
             )
serVis(json) # Results in a different topic layout in the left panel


## End(Not run)
```

`jsPCA` *Dimension reduction via Jensen-Shannon Divergence & Principal Components*

Description

Dimension reduction via Jensen-Shannon Divergence & Principal Components

Usage

```
jsPCA(phi)
```

Arguments

phi	matrix, with each row containing the distribution over terms for a topic, with as many rows as there are topics in the model, and as many columns as there are terms in the vocabulary.

---

renderVis *Create an LDAvis output element*

Description

Shiny server output function customized for animint plots (similar to `shiny::plotOutput` and friends).

Usage

```
renderVis(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

expr	An expression that generates a plot.
env	The environment in which to evaluate `expr`.
quoted	Is expr a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

See Also

http://shiny.rstudio.com/articles/building-outputs.html

---

runShiny *Run shiny/D3 visualization*

Description

This function is deprecated as of version 0.2

Usage

```
runShiny(phi, term.frequency, vocab, topic.proportion)
```

Arguments

phi	a matrix with W rows, one for each term in the vocabulary, and K columns, one for each topic, where each column sums to one. Each column is the multinomial distribution over terms for a given topic in an LDA topic model.
term.frequency	an integer vector of length W containing the frequency of each term in the vocabulary.
vocab	a character vector of length W containing the unique terms in the corpus.
topic.proportion	a numeric vector of length K containing the proportion of each topic in the corpus.

---

serVis *View and/or share LDAvis in a browser*

Description

View and/or share LDAvis in a browser.

Usage

```
serVis(json, out.dir = tempfile(), open.browser = interactive(),
  as.gist = FALSE, ...)
```

Arguments

json	character string output from createJSON.
out.dir	directory to store html/js/json files.
open.browser	Should R open a browser? If yes, this function will attempt to create a local file server via the servr package. This is necessary since the javascript needs to access local files and most browsers will not allow this.
as.gist	should the vis be uploaded as a gist? Will prompt for an interactive login if the GITHUB_PAT environment variable is not set. For more details, see https://github.com/ropensci/gistr#authentication.
...	arguments passed onto gistr::gist_create

Details

This function will place the necessary html/js/css files (located in system.file("htmljs", package = "LDAvis")) in a directory specified by out.dir, start a local file server in that directory (if necessary), and (optionally) open the default browser in this directory. If as.gist=TRUE, it will attempt to upload these files as a gist (in this case, please make sure you have the gistr package installed as well as your 'github.username' and 'github.password' set in options.)

Value

An invisible object.

Author(s)

Carson Sievert

See Also

createJSON

Examples

```
## Not run:
# Use of serVis is documented here:
help(createJSON, package = "LDAvis")

## End(Not run)
```

TwentyNewsgroups *Twenty Newsgroups Data*

Description

Twenty Newsgroups Data

Usage

```
TwentyNewsgroups
```

Format

A list elements extracted from a topic model fit to this data

phi phi, a matrix with the topic-term distributions

theta theta, a matrix with the document-topic distributions

doc.length doc.length, a numeric vector with token counts for each document

vocab vocab, a character vector containing the terms

term.frequency term.frequency, a numeric vector of observed term frequencies

Source

http://qwone.com/~jason/20Newsgroups/

`visOutput` *Shiny ui output function*

Description

Shiny ui output function

Usage

```
visOutput(outputId)
```

Arguments

`outputId`	output variable to read the plot from

See Also

http://shiny.rstudio.com/articles/building-outputs.html

Index