ldagibbs: A command for topic modeling in Stata using latent Dirichlet allocation

Carlo Schwarz
University of Warwick
Coventry, UK
c.r.schwarz@warwick.ac.uk

Abstract. In this article, I introduce the `ldagibbs` command, which implements latent Dirichlet allocation in Stata. Latent Dirichlet allocation is the most popular machine-learning topic model. Topic models automatically cluster text documents into a user-chosen number of topics. Latent Dirichlet allocation represents each document as a probability distribution over topics and represents each topic as a probability distribution over words. Therefore, latent Dirichlet allocation provides a way to analyze the content of large unclassified text data and an alternative to predefined document classifications.

Keywords: st0515, ldagibbs, machine learning, latent Dirichlet allocation, Gibbs sampling, topic model, text analysis

1 Introduction

Text data are a potentially rich source of information for researchers. For example, many datasets consist of large collections of unclassified text data. These text collections, which are usually referred to as "corpora", can consist of many thousands of individual documents with millions of words. Hence, a text corpus often cannot be used for statistical analysis without automated text analysis or a machine-learning algorithm. In this article, I therefore introduce the `ldagibbs` command for latent Dirichlet allocation (LDA), which was developed by Blei, Ng, and Jordan (2003). LDA is the most popular topic model and allows for the automatic clustering of any kind of text documents into a user-chosen number of clusters of similar content, usually referred to as "topics".

LDA relies only on text data; therefore, researchers can apply LDA for nearly all tasks of text clustering. Additionally, LDA can cluster corpora independent of the number of documents and language of texts. LDA's topic clustering already has been proven to be reliable for many different text corpora. For example, LDA is capable of grouping together medical studies (Wu et al. 2012), statements from the transcripts of the Federal Reserve's Open Market Committee (Hansen, McMahon, and Prat Forthcoming), and journal articles (Griffiths and Steyvers 2004).

In its clustering, LDA uses a probabilistic model of the text data. This model exploits the fact that when talking about the same topic, authors tend to use similar words. Hence, in texts about the same topic, similar words tend to co-occur. LDA uses the co-occurrences of words to describe each topic as a probability distribution over words and to describe each document as a probability distribution over topics.

The `ldagibbs` command conveniently runs LDA in Stata. Its output enables a comparison of similarities between documents and topics, making it easy to analyze their previously hidden relationships.

`ldagibbs` extends the text analysis capabilities of Stata, making it possible to compare the similarities between entire documents and to compare string similarities based on their Levenshtein edit distances with the `strdist` command (Barker 2012). While `screening` (Belotti and Depalo 2010) already allows one to search texts for individual keywords, and `txttool` (Williams and Williams 2014) allows one to create a bag-of-words representation of text data, `ldagibbs` provides a new option for large-scale document-content analysis. Thereby, `ldagibbs` makes previously unused text data usable for research.

2   LDA

LDA consists of two parts: The first is the probabilistic model, which describes the text data as a likelihood function. The second is an approximate inference algorithm, because simply maximizing this likelihood function is computationally unfeasible.

2.1   Probabilistic model

The probabilistic model of LDA assumes that each document d of the D documents in the corpus can be described as a probabilistic mixture of T topics. These probabilities are contained in a document topic vector $\boldsymbol{\theta}_d$ of length T. The output of LDA is a $D \times T$ matrix $\boldsymbol{\theta}$ containing $P(t_t|d_d)$, which is the probability of document d belonging to topic t:

$$\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_D \end{pmatrix} = \begin{pmatrix} P(t_1|d_1) & \cdots & P(t_T|d_1) \\ \vdots & \ddots & \vdots \\ P(t_1|d_D) & \cdots & P(t_T|d_D) \end{pmatrix}$$

Each topic $t \in T$ is defined by a probabilistic distribution over the vocabulary (the set of unique words in all documents) of size V. A topic, therefore, describes how likely it is to observe a word conditional on a topic. For example, documents regarding physics might have a high probability for words such as "particle", "quantum", and "boson", while the most frequent words in documents regarding medicine could be "clinical", "virus", and "infection". The word-probability vectors of each topic are contained in a matrix $\boldsymbol{\phi}$ of dimensions $V \times T$.

$$\boldsymbol{\phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_T) = \begin{pmatrix} P(w_1|t_1) & \cdots & P(w_1|t_T) \\ \vdots & \ddots & \vdots \\ P(w_V|t_1) & \cdots & P(w_V|t_T) \end{pmatrix}$$

The probabilities $P(w_v|t_t)$ in the $\boldsymbol{\phi}_t$ vectors describe how likely it is to observe word v from the vocabulary, conditional on topic t. In this way, the $\boldsymbol{\phi}_t$ vectors allow one to

judge the content of each topic, because LDA does not provide any additional topic labels.

Given the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, LDA's probabilistic model assumes that the text in the corpus is generated by the following process:

1. Draw a word-probability distribution $\boldsymbol{\phi} \sim \text{Dir}(\beta)$.

2. For each document d in the corpus:

    a. Draw topic proportions $\boldsymbol{\theta}_d \sim \text{Dir}(\alpha)$.

    b. For each of the N_d words w_d in d:

        i. Draw a topic assignment $z_{d,n} \sim \text{Mult}(\boldsymbol{\theta}_d)$.

        ii. Draw a word $w_{d,n}$ from $p(w_{d,n}|z_{d,n}, \boldsymbol{\phi})$.

In this model, α and β are hyperparameters, which are necessary for the Gibbs sampling process. Given the probabilistic model, the overall likelihood of the corpus with respect to the model parameters is

$$\prod_{d=1}^{D} P\left(\boldsymbol{\theta}_d|\alpha\right) \left\{ \prod_{n=1}^{N_d} \sum_{z_{d,n}} P\left(z_{d,n}|\boldsymbol{\theta}_d\right) P\left(w_{d,n}|z_{d,n}, \boldsymbol{\phi}\right) \right\} \tag{1}$$

$P(\boldsymbol{\theta}_d|\alpha)$ in (1) is the likelihood of observing the topic distribution of $\boldsymbol{\theta}_d$ of document d, conditional on α. $P(z_{d,n}|\boldsymbol{\theta}_d)$ is the likelihood of the individual topic assignment $z_{d,n}$ of word n in document d, conditional on the topic distribution of the document. Finally, $P(w_{d,n}|z_{d,n}, \boldsymbol{\phi})$ is the probability to observe a specific word, conditional on the topic assignment of the word and the word probabilities of the given topic, which are contained in $\boldsymbol{\phi}$. By calculating the sum over all possible topic assignments ($\sum_z$), the product over all N_d words in a document ($\prod_{n=1}^{N_d}$), and the product over all documents in the corpus ($\prod_{d=1}^{D}$), we obtain the likelihood of observing the words in the documents.

The LDA classification is based on finding the optimal topic assignment $z_{d,n}$ for each word in each document and the optimal word probabilities $\boldsymbol{\phi}$ for each topic that maximizes this likelihood. Maximizing this likelihood would require summing over all possible topic assignment for all words in all documents, which is computationally unfeasible. Therefore, alternative methods to approximate the likelihood function have been developed. The `ldagibbs` command uses the Gibbs sampling method developed by Griffiths and Steyvers (2004).

2.2 Approximate inference using Gibbs sampling

Gibbs sampling is a Markov chain Monte Carlo algorithm, which is based on repeatedly drawing new samples conditional on all other data. The `WinBUGS` command (Thompson, Palmer, and Moreno 2006) provides a general framework for the implementation of the Gibbs sampler in Stata.

In the specific case of LDA, the Gibbs sampler relies on iteratively updating the topic assignment of words, conditional on the topic assignments of all other words. Because Gibbs sampling is a Bayesian technique, it requires priors for the values of the hyperparameters α and β. These initial parameter values should lie inside the unit interval. The prior for α is usually chosen based on the number of topics T, while the prior for β depends on the size of the vocabulary. The higher the number of topics or the larger the vocabulary, the smaller should be the priors for α and β. As a heuristic for the choice of priors, Griffiths and Steyvers (2004) suggest $\alpha = 50/T$ (only applicable if $T > 50$) and $\beta = 0.1$. In general, the choice of the priors should not influence the outcome of the sampling process.

As the first step, `ldagibbs` splits the documents into individual words—so-called word tokens. These word tokens are then randomly assigned to one of the T topics with equal probabilities. This provides an initial assignment of words and thereby documents to topics for the sampling process. Afterward, `ldagibbs` samples new topic assignments for each of the word tokens. The probability of a word token being assigned to topic t based on the probabilistic model is

$$P(z_{d,n} = t | w_{d,n}, \boldsymbol{\phi}) \propto P(w_{d,n} | z_{d,n} = t, \boldsymbol{\phi}) \times P(z_{d,n} = t)$$

The Gibbs sampler uses the topic assignment of all other tokens to obtain approximate values for $P(w_{d,n} | z_{d,n} = t, \boldsymbol{\phi})$ and $P(z_{d,n} = t)$. $P(w_{d,n} | z_{d,n} = t, \boldsymbol{\phi})$ is given by the number of words identical to $w_{d,n}$ assigned to topic t divided by the total number of words assigned to that topic. For example, if a specific word is assigned to topic t some x times, then $P(w_{d,n} | z_{d,n} = t, \boldsymbol{\phi}) = (x + \beta)/(N^t + V\beta)$, where N^t is the total number of words in the corpus assigned to topic t, and β is the chosen prior for β. Similarly, $P(z_{d,n} = t)$ is given by the fraction of words in a document currently assigned to topic t plus the prior for α.

In each iteration, `ldagibbs` samples new topic assignments for all words in the corpus. By running the Gibbs sampler for a burn-in of several hundred iterations, the Markov chain converges toward the maximum of the likelihood function. In the example of Griffiths and Steyvers (2004), the sampler already reaches convergence after 150 iterations. To analyze the convergence of the Gibbs sampler, `ldagibbs` provides an option called `likelihood`. When the `likelihood` option is specified, the Gibbs sampler reports the log likelihood and the change of the log likelihood of the model every 50 iterations. The log likelihood should initially increase with the number of iterations of the Gibbs sampler until only small changes occur.[1]

After the burn-in period, samples are taken from the Gibbs sampler to obtain an approximation of the parameter values of $\boldsymbol{\theta}_d$ and $\boldsymbol{\phi}$. To guarantee the statistical independence of these samples, additional iterations of the sampling process should be run in between the samples.

---

1. The log likelihood will be negative in nearly all cases. Furthermore, it is possible that the likelihood falls in some iterations of the Gibbs sampler, especially because the log likelihood is calculated using only the current state of the Gibbs sampler.

Like most Gibbs sampling applications, it is possible that the Gibbs sampler converges toward a local instead of the global maximum of the likelihood function. Users can experiment with different random seeds to compare different sampling results. Often, the local maximum result in similar topics and document clusters.

3 Implementation

This section describes how to use the `ldagibbs` command to fit the topic model to text data. For `ldagibbs`, each observation in the dataset should represent one of the documents the user wants to cluster. Long documents can also be split into paragraphs to cluster each paragraph separately. In both cases, the text strings for the clustering should be contained in one variable. It is advisable to clean nonalphanumerical characters from these text strings because they can potentially interfere with the clustering.

In theory, LDA can be applied to any type of text data independently of the length of the individual texts. LDA works well for abstracts of scientific articles or when the individual text documents contain at least 50–100 words. The number of words required for easily interpretable topics varies with the type of document and the size of the vocabulary. When the individual text documents are very short, LDA might generate less meaningful topics. For example, LDA does not perform as well when applied to tweets (Hong and Davison 2010; Zhao et al. 2011), because in these cases, LDA has too few word co-occurrence information. This problem can be overcome either by combining short documents together (Quan et al. 2015) or by using an alternative topic model (for example, see Rosen-Zvi et al. [2004], Yan et al. [2013], or Kim and Shim [2014]).

The `ldagibbs` command requires only the variable containing the text strings as an input. The individual options of the command allow one to modify the behavior of the Gibbs sampler and specify names for the output variables. For convenience, `ldagibbs` also includes some basic text cleaning capabilities to remove stopwords and short words from the data. For more advanced text cleaning options, see the `txttool` command (Williams and Williams 2014).

3.1 Syntax

`ldagibbs` *varname* [, <u>topics</u>(*integer*) <u>burnin_iter</u>(*integer*) <u>alpha</u>(*real*)
   <u>beta</u>(*real*) <u>samples</u>(*integer*) <u>sampling_iter</u>(*integer*) <u>seed</u>(*integer*)
   <u>likeli</u>hood <u>min_char</u>(*integer*) <u>stopwords</u>(*string*) <u>name_new_var</u>(*string*)
   <u>normalize</u> <u>mat_save</u> <u>path</u>(*string*)]

3.2   Options

Gibbs sampler options

topics(*integer*) specifies the number of topics LDA should create. The default is
    topics(10).

burnin_iter(*integer*) specifies how many iterations the Gibbs sampler should run as a
    burn-in period. The default is burnin_iter(500).

alpha(*real*) sets the prior for the topic-probability distribution. The value for alpha()
    should be between 0 and 1. As a heuristic, users can use $\alpha = 50/T$ (only applicable
    if $T > 50$). The default is alpha(0.25).

beta(*real*) sets the prior for the word-probability distribution. The value for beta()
    should be between 0 and 1. The default is beta(0.1).

samples(*integer*) specifies how many samples the algorithm should collect after the
    burn-in period. To obtain robust results, at least 10 samples should be taken. The
    default is samples(10).

sampling_iter(*integer*) specifies how many iterations the Gibbs sampler should ig-
    nore between the individual samples. Running additional iterations of the Gibbs
    sampler guarantees the statistical independence of the samples. The default is
    sampling_iter(50).

seed(*integer*) sets the seed for the random-number generator to guarantee the repro-
    ducibility of the results. The default is seed(0).

likelihood specifies that the Gibbs sampler calculate and report the log likelihood of
    the LDA model every 50 iterations. This option allows one to analyze the convergence
    of the Gibbs sampler but slows down the sampling process.

Text cleaning options

min_char(*integer*) allows the removal of short words from the texts. Words with fewer
    characters than *integer* will be excluded from the sampling algorithm. The default
    is min_char(0).

stopwords(*string*) specifies a list of words to exclude from the Gibbs sampler. Usually,
    highly frequent words, such as "I" or "you", are removed from the text because these
    words do not help classify the documents. Predefined stopword lists for different
    languages are available online.

Output options

name_new_var(*string*) specifies the name of the output variables created by ldagibbs.
    These variables contain the topic assignments for each document. *string* should be
    unique in the dataset. The default is name_new_var("topic_prob"), such that the

names of the new variables will be `topic_prob1`, `topic_prob2`, ..., `topic_prob`T, where T is the total number of topics.

`normalize` specifies that `ldagibbs` normalize the raw topic assignment counts to probabilities. By default, `ldagibbs` returns the raw topic assignment counts. You almost always want to specify this option.

`mat_save` specifies to save the word-probability matrix. This matrix defines the most frequent words in each topic. By default, the matrix will not be saved.

`path(`*string*`)` sets the path where the word-probability matrix is to be saved.

3.3 Output

`ldagibbs` generates T new variables that describe the topics of each document. If `normalize` is specified, each variable contains the probability of a document belonging to topic $t \in T$. Without `normalize`, the variable contains the number of words in the document being assigned to topic t plus $\alpha \times s$, where s is the number of samples drawn (`samples(`s`)`).

The content of each topic can be analyzed using the word-probability vectors. To save these vectors, the `mat_save` option must be specified. `ldagibbs` then saves a Mata matrix file with the name `word_prob` in the folder specified in the `path()` option.[2] The file contains the word probabilities for each of the T topics. To analyze the word-probability vectors, `ldagibbs` also provides the `wprobimport` command, which imports the stored word-probability data into a Stata dataset. Sorting the `word_prob` data in descending order by one of the variables will ensure the most frequent words in each topic are displayed.

4 Examples

Example 1: Clustering laws

The first example dataset contains the title and the summary of 1,000 bills from the U.S. Congress. After opening it and combining the title and the summary for each bill, nonalphanumerical characters are removed from the strings. This provides the raw strings for LDA.

```
. use example_data_laws
. *combine title and summary
. generate text_strings =  title + " " + summary
. *remove nonalphanumerical characters
. replace text_strings = subinstr( text_strings, ".", " ", .)
(613 real changes made)
```

---

2. Storing the data as a Mata matrix is the most efficient solution because the Stata matrix size is limited; thus, storing the results as a Stata dataset could potentially be memory intensive.

```
. replace text_strings = subinstr( text_strings, "!", " ", .)
(0 real changes made)
. replace text_strings = subinstr( text_strings, "?", " ", .)
(0 real changes made)
. replace text_strings = subinstr( text_strings, ":", " ", .)
(142 real changes made)
. replace text_strings = subinstr( text_strings, ";", " ", .)
(142 real changes made)
. replace text_strings = subinstr( text_strings, ",", " ", .)
(361 real changes made)
. replace text_strings = subinstr( text_strings, "(", " ", .)
(189 real changes made)
. replace text_strings = subinstr( text_strings, ")", " ", .)
(189 real changes made)
. replace text_strings = stritrim(text_strings)
(413 real changes made)
. list text_strings if _n<=10
```

	text_strings
1.	Social Security and Medicare Lock-Box Act of 2001 Requires any Feder..
2.	Economic Growth and Tax Relief Act of 2001 Prohibits minimum bracket..
3.	Energy Policy Act of 2002 Sec 206 Grants FERC jurisdiction over an e..
4.	Reserved for the speaker
5.	Marriage Penalty and Family Tax Relief Act of 2001 Repeals provision..
6.	CARE Act of 2002 Sec 103 Sets forth a rule for determining the amoun..
7.	Death Tax Elimination Act of 2001 Title III Unified Credit Replaced ..
8.	Reserved for the speaker
9.	Railroad Retirement and Survivors´ Improvement Act of 2001 Sec 102 R..
10.	Financial Contract Netting Improvement Act of 2001

The LDA Gibbs sampling process is started using the `ldagibbs` command. For the small example dataset, LDA takes a few minutes to execute. Because the time needed for sampling increases proportional to the number of words in all documents, running LDA for large datasets with tens of thousands of observations can take several hours. The sampler reports each time 50 iterations have finished and each time a new sample is collected. This provides an estimate of how long the entire sampling process will take.

```
. global stopwords "a able about across after all almost also am among an and
> any are as at be because been but by can cannot could dear did do does either
> else ever every for from get got had has have he her hers him his how however
> i if in into is it its just least let like likely may me might most must my
> neither no nor not of off often on only or other our own rather said say says
> she should since so some than that the their them then there these they this
> tis to too twas us wants was we were what when where which while who whom why
> will with would yet you your"
```

```
. ldagibbs text_strings, topics(5) alpha(0.25) beta(0.1) seed(3) burnin_iter(1000)
> samples(10) sampling_iter(50) min_char(5) normalize mat_save
> stopwords("$stopwords")
*************************************
********* Parameters for LDA *********
*************************************
Number of Topics: 5
Prior Alpha: .25
Prior Beta: .1
Number of Iterations: 1000
Number of Samples: 10
Iterations between Samples: 50
Seed: 3
Minimal Word Length: 5
*************************************

  *************************************
  ******** Preparing Documents ********
  *************************************

  Processing Document:
  500
  Processing Document:
  1000
  *************************************
  **** Initializing Gibbs Sampler *****
  *************************************


  *************************************
  ******* Running Gibbs Sampler *******
  *************************************

  Iteration:
  50
  Iteration:
  100
```
  (*output omitted*)
```
  Iteration:
  1000
  Sample:
  2
```
  (*output omitted*)
```
  Iteration:
  50
```

After LDA is finished, one can sort the documents by the assigned topics. Even in the small example dataset, LDA accurately classifies relevant bills (for example, bills about educational policy) into one topic.

```
. *list example topics
. forvalues i=1/5 {
  2. gsort -topic_prob`i´
  3. list title topic_prob`i´ if _n<=5, str(45)
  4. }
```

	title	topic_p~1
1.	MD-CARE Act	.97647059
2.	Public Education Reinvestment, Reinvention, a..	.97429429
3.	National Science Education Act	.97394844
4.	Character Counts for the 21st Century Act	.96168582
5.	Excellence and Accountability in Education Act	.95501449

(*output omitted*)

As a final component of analysis, one can load the information from the word-probability matrix by using the `wprobimport` command. After importing the data, the most frequent words in each topic are listed. This allows the user to assign labels to the topics if required. It is worth noting again that the individual ϕ_t vectors for each topic describe how likely it is to see a word conditional on a topic. Hence, the sum of all word probabilities within each individual ϕ_t vector is 1, but the word probabilities across topics will not add up to 1.

```
. wprobimport using "word_prob"
. *list most frequent words in each topic
. forvalues i=1/5 {
  2. gsort -word_prob`i´
  3. list words word_prob`i´ if _n<=5
  4. }
```

	words	word_pr~1
1.	education	.02226475
2.	secretary	.01650924
3.	programs	.01549032
4.	program	.01342494
5.	requires	.01320464

	words	word_pr~2
1.	amend	.02118311
2.	health	.01655658
3.	revenue	.0138505
4.	internal	.01347223
5.	certain	.01341403

	words	word_pr~3
1.	energy	.02905464
2.	national	.01797795
3.	secretary	.01628051
4.	federal	.01409986
5.	program	.0095775

	words	word_pr~4
1.	bankruptcy	.01477839
2.	property	.0123517
3.	debtor	.01002773
4.	court	.00991218
5.	certain	.00885933

	words	word_pr~5
1.	federal	.02166268
2.	states	.0160524
3.	united	.01468338
4.	state	.01199903
5.	protection	.00836174

Example 2: Clustering economic articles

The second example uses titles and abstracts from 10,000 economic articles published from 2010–2015. The data are prepared similarly to the first example. Afterward, the Gibbs sampler is started, this time specifying the likelihood option. The reported likelihood numbers indicate that the Gibbs sampler converges after around 400 iterations, because afterward the likelihood changes only marginally.

```
. ldagibbs text_strings, topics(15) alpha(0.25) beta(0.1) seed(6) burnin_iter(600)
> samples(10) sampling_iter(50) min_char(5) normalize likelihood
> stopwords("$stopwords")
************************************
********* Parameters for LDA *********
************************************

  (output omitted)

  ************************************
  ******** Preparing Documents ********
  ************************************

  Processing Document:
  500
  (output omitted)
  Processing Document:
  10000
```

```
************************************
**** Initializing Gibbs Sampler *****
************************************

************************************
******* Running Gibbs Sampler *******
************************************

Iteration:
50
Log-Likelihood (Change):
-5207989(443079)
Iteration:
100
Log-Likelihood (Change):
-5188673(19316)
Iteration:
150
Log-Likelihood (Change):
-5183687(4986)
Iteration:
200
Log-Likelihood (Change):
-5182548(1139)
Iteration:
250
Log-Likelihood (Change):
-5181958(589)
Iteration:
300
Log-Likelihood (Change):
-5181721(237)
Iteration:
350
Log-Likelihood (Change):
-5180363(1358)
Iteration:
400
Log-Likelihood (Change):
-5180061(302)
Iteration:
450
Log-Likelihood (Change):
-5180449(-389)
Iteration:
500
Log-Likelihood (Change):
-5180211(238)
Iteration:
550
Log-Likelihood (Change):
-5179952(259)
Iteration:
600
Log-Likelihood (Change):
-5179954(-2)
```

   (*output omitted*)

As a next step, one can generate indicator variables for the topic with the largest topic share in each article and plot the average topic shares over time.

```
. egen max_prob = rowmax(topic_prob*)
. replace max_prob = round(max_prob,0.0000001)
(5,742 real changes made)
. forvalues i=1/15 {
  2. gsort -topic_prob`i´
  3. display as error `i´
  4. list title if _n<5
  5. replace topic_prob`i´ = float(round(topic_prob`i´,0.0000001))
  6. generate topic_`i´ = (max_prob==topic_prob`i´)
  7. }
1
```

	title
1.	Estimating panel data models in the presence of endogeneity and se..
2.	Residual based tests for cointegration in dependent panels
3.	Estimation in Partially Linear Single-Index Panel Data Models With..
4.	Semiparametric GMM estimation of spatial autoregressive models

```
(10,000 real changes made)
2
```

	title
1.	Corporate governance when founders are directors
2.	Corporate governance and capital allocations of diversified firms
3.	Shareholder governance, bondholder governance, and managerial risk..
4.	Staggered boards, corporate opacity and firm value

```
(10,000 real changes made)
```
   (output omitted)
```
. preserve
. collapse (mean)  topic_prob*, by(pub_year)
```

```
. twoway line topic_prob1 pub_year || line topic_prob2 pub_year ||
> line topic_prob4 pub_year || line topic_prob8 pub_year ||
> line topic_prob12 pub_year, legend(label(1 "Econometrics")
> label(2 "Corporate governance") label(3 "Game theory")
> label(4 "Monetary economics")  label(5 "Labor economy") )
> xtitle("Year") ytitle("Average topic share") graphregion(color(white))
```
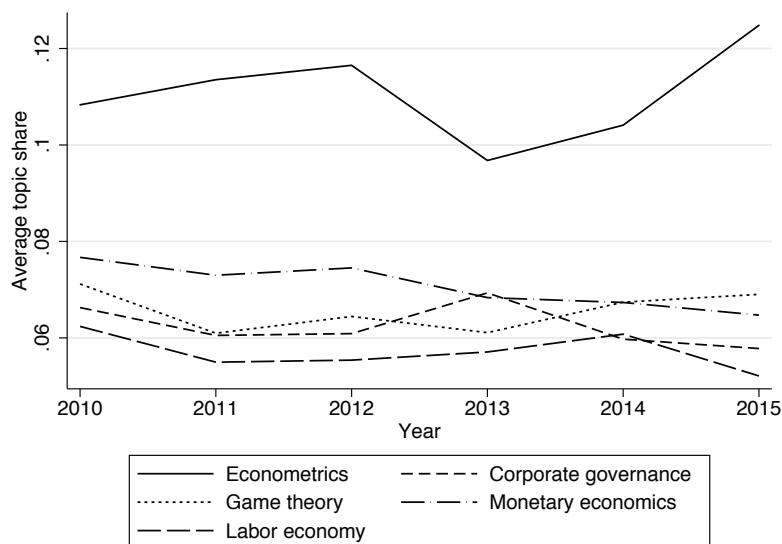

Figure 1. Topic trends over time

As we can see in figure 1, the econometrics topic[3] has by far the largest average share of the five plotted topics. Moreover, after a drop in 2013, the econometrics topic becomes again more prominent in the corpus in 2014 and 2015. One can complement this analysis by fitting a regression model using the number of citations and the previously created indicator variables.

---

3. The topic labels in figure 1 were given based on the titles of the five articles with the largest share in each of the topics.

```
. regress citations topic_1-topic_15 c.topic_1#i.pub_year i.pub_year
```

Source	SS	df	MS		Number of obs	=	10,000
					F(25, 9974)	=	42.90
Model	1011163.27	25	40446.5307		Prob > F	=	0.0000
Residual	9402745.73	9,974	942.72566		R-squared	=	0.0971
					Adj R-squared	=	0.0948
Total	10413909	9,999	1041.49505		Root MSE	=	30.704

citations	Coef.	Std. Err.	t	P>|t|	[95% Conf. Interval]	
topic_1	-7.830208	2.784147	-2.81	0.005	-13.2877	-2.372718
topic_2	-6.149039	1.611277	-3.82	0.000	-9.307467	-2.99061
topic_3	6.123092	1.599216	3.83	0.000	2.988307	9.257878
topic_4	-7.347089	1.611442	-4.56	0.000	-10.50584	-4.188337
topic_5	6.878272	1.444997	4.76	0.000	4.045787	9.710758
topic_6	-.6757591	1.510611	-0.45	0.655	-3.636861	2.285343
topic_7	4.887668	1.644834	2.97	0.003	1.663462	8.111874
topic_8	8.545773	1.513232	5.65	0.000	5.579532	11.51201
topic_9	8.915616	1.487744	5.99	0.000	5.999337	11.8319
topic_10	-2.372908	1.653648	-1.43	0.151	-5.614392	.8685759
topic_11	-.6464624	1.771297	-0.36	0.715	-4.118562	2.825637
topic_12	1.921868	1.686994	1.14	0.255	-1.38498	5.228716
topic_13	-4.580828	1.508604	-3.04	0.002	-7.537996	-1.62366
topic_14	-8.313427	1.651837	-5.03	0.000	-11.55136	-5.075493
topic_15	-1.756457	1.824063	-0.96	0.336	-5.331989	1.819075
pub_year#						
c.topic_1						
2011	1.082498	3.622517	0.30	0.765	-6.018367	8.183363
2012	1.344766	3.654425	0.37	0.713	-5.818646	8.508177
2013	7.94672	3.783296	2.10	0.036	.5306967	15.36274
2014	5.019593	3.673452	1.37	0.172	-2.181115	12.2203
2015	6.619671	3.539678	1.87	0.061	-.3188111	13.55815
pub_year						
2011	-4.859705	1.150878	-4.22	0.000	-7.115658	-2.603751
2012	-9.511767	1.164812	-8.17	0.000	-11.79503	-7.2285
2013	-16.19891	1.137882	-14.24	0.000	-18.42938	-13.96843
2014	-20.13599	1.131391	-17.80	0.000	-22.35375	-17.91824
2015	-24.88938	1.132622	-21.98	0.000	-27.10955	-22.66921
_cons	31.07525	1.215859	25.56	0.000	28.69192	33.45858

The regression results allow for a comparison of the citations of articles in different topics over time. Thereby, LDA provides a way to identify trending topics and analyze the shifting importance of research fields.

5 Acknowledgments

6   References

Barker, M. 2012. strdist: Stata module to calculate the Levenshtein distance, or edit distance, between strings. Statistical Software Components S457547, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s457547.html.

Belotti, F., and D. Depalo. 2010. Translation from narrative text to standard codes variables with Stata. *Stata Journal* 10: 458–481.

Blei, D. M., A. Y. Ng, and M. I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3: 993–1022.

Griffiths, T. L., and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101: 5228–5235.

Hansen, S., M. McMahon, and A. Prat. Forthcoming. Transparency and deliberation within the FOMC: A computational linguistics approach. *Quarterly Journal of Economics*.

Hong, L., and B. D. Davison. 2010. Empirical study of topic modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*, 80–88. New York: ACM.

Kim, Y., and K. Shim. 2014. TWILITE: A recommendation system for Twitter using a probabilistic model based on latent Dirichlet allocation. *Information Systems* 42: 59–77.

Quan, X., C. Kit, Y. Ge, and S. J. Pan. 2015. Short and sparse text topic modeling via self-aggregation. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2270–2276. Buenos Aires, Argentina: AAAI Press.

Rosen-Zvi, M., T. Griffiths, M. Steyvers, and P. Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence*, 487–494. Banff, Canada: AUAI Press.

Thompson, J., T. Palmer, and S. Moreno. 2006. Bayesian analysis in Stata with WinBUGS. *Stata Journal* 6: 530–549.

Williams, U., and S. P. Williams. 2014. txttool: Utilities for text analysis in Stata. *Stata Journal* 14: 817–829.

Wu, Y., M. Liu, W. J. Zheng, Z. Zhao, and H. Xu. 2012. Ranking gene–drug relationships in biomedical literature using latent Dirichlet allocation. *Pacific Symposium on Biocomputing* 422–433.

Yan, X., J. Guo, Y. Lan, and X. Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd International World Wide Web Conference*, 1445–1456. New York: ACM.

Zhao, W. X., J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. 2011. Comparing Twitter and traditional media using topic models. In *Proceedings of the 33rd European Conference on Information Retrieval Research*, 338–349. Berlin: ECIR.

About the author

Carlo Schwarz is a PhD student at the University of Warwick and the Centre for Competitive Advantage in the Global Economy (CAGE) in the UK. His research interests are in the fields of applied microeconomics and political economy, with a focus on text analysis and machine learning.