LDA: Latent Dirichlet Allocation

Xin Lei and Rob McCulloch

1. Introduction

LDA is an unsupervised technique for looking at a corpus (set) of documents.

Each document is treated as a bag of words.

We process all the words in all the documents and come up with a list of *terms* which exhaustively lists all the possible values for each word in each document.

Let V be the number of terms and D be the number of documents.

Then our data consists of the values w_{dl}:

$$w_{dl} \in \{1, 2, \ldots, V\}, \quad d = 1, 2, \ldots, D, \quad l = 1, 2, \ldots, N_d.$$

giving the word id, for the l^{th} work in the d^{th} document. N_d is the number of words in document d.

A set of K *latent topics* is considered.
A topic, is a distribution over the set of possible terms.

Each document is characterized by a distribution over topics.

Then, each word, w_{dl}, in a given document is generated by first drawing the topic from the document's topic distribution and then drawing the term from the drawn topic distribution.

That is a mouthful!!

We will take it apart piece by piece.

2. Topics

A *topic* will be a distribution over the set of possible terms indexed by $\{1, 2, \ldots, V\}$.

For the k^{th} topic and random word w, let

$$\beta_{vk} = P(w = v)$$

for terms $v \in \{1, 2, \ldots, V\}$.

β_{vk} is the probability a word turns out to be term v.

Let

$$\beta_k = (\beta_{1k}, \beta_{2k}, \ldots, \beta_{Vk})'.$$

Then,

$$0 \leq \beta_{vk} \leq 1, \ \sum_v \beta_{vk} = 1.$$

The idea is that a "topic" makes certain terms more likely.

Let β be the matrix whose columns are the topics

$$\beta = [\beta_1, \beta_2, \ldots, \beta_K]$$

So, β is $V \times K$, where V is the number of terms and K is the number of topics.

3. Topic Weights

Our model idea is that each document will favor some of the topics.

To capture this, let z_{dl} be the topic of w_{dl}.

That is, each word gets to come from a specific topic.

$$P(w_{dl} = v | z_{dl} = k, \beta) = \beta_{vk}.$$

Then each document has probabilities for the topics:

$$P(z_{dl} = k) = \theta_{dk}.$$

If we let

$$\theta_d = (\theta_{d1}, \theta_{d2}, \ldots, \theta_{dK})$$

then θ_d gives the topic distributions (weights) for document d.

So, each word in each document gets to have it's own topic, but, all the words in the same document have its topic drawn from the same distribution θ_d.

4. Dirichlet Priors

So each θ_d and each β_k is a probability vector.

We will put *prior distributions* on these probability vectors using the Dirichlet distribution.

In general, let $p = (p_1, p_2, \ldots, p_m)$ be a probability vector:

$$0 \leq p_i \leq 1, \ \sum p_i = 1.$$

Then the Dirichlet distribution has density

$$p(p|\alpha) = \frac{\Gamma(\sum \alpha_i)}{\prod \Gamma(\alpha_i)} \prod p_i^{\alpha_i - 1}$$

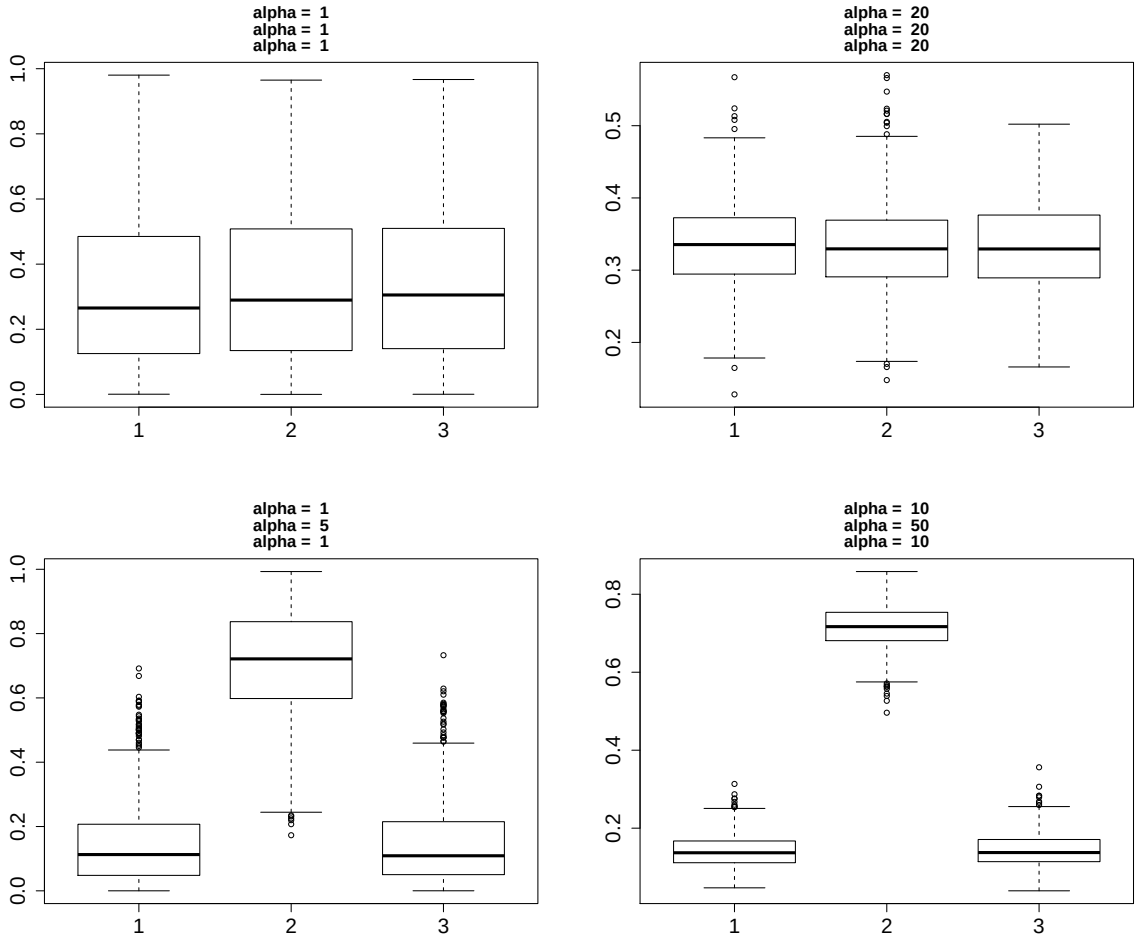
where Γ is the gamma function.

Note:

$$E(p_i) = \frac{\alpha_i}{\sum \alpha_i}.$$

The bigger the α_i are the tighter the distribution.
We say $p \sim \mathit{Dirichlet}(\alpha)$ where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$.

Draws from the Dirichlet for different α.



Let $\vec{1} = (1, 1, \ldots, 1)$.

So, finally, our model is completed by letting

$$\theta_d \sim \textit{Dirichlet}(\alpha \vec{1}_K), \text{ iid.}$$

where here α is a scalar.

$$\beta_k \sim \textit{Dirichlet}(\gamma \vec{1}_V), \text{ iid}$$

where here γ is a scalar.
Typically very small values for α and γ are using so that we start with a prior that weakly suggest the probabilities are spread out.

5. The Full Model

We are using a Bayesian approach where we put a joint distribution on all quantities of interest and then compute the posterior given observed data.

Our full joint is

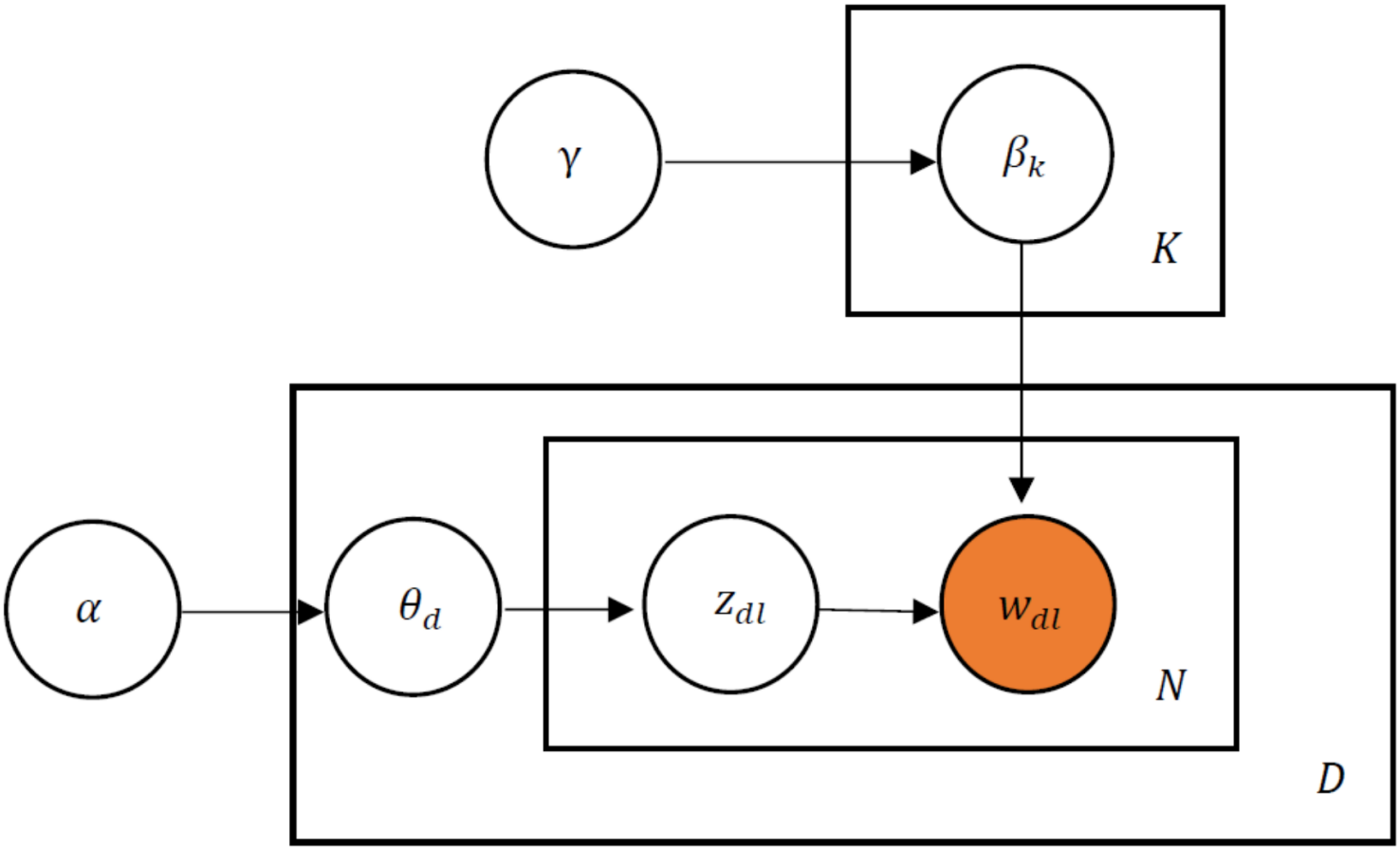
$$p(\beta, \theta, w, z) = \prod p(\beta_k|\gamma) \prod p(\theta_d|\alpha) \prod (p(z_{dl}|\theta_d)p(w_{dl}|z_{dl}, \beta))$$

Then we compute the *posterior*

$$p(\beta, \theta, z|w)$$

We can estimate unknowns with their posterior means.

Here is the picture (DAG) of the model:



6. Example: Newsgroup Postings

The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups.

It was originally collected by Ken Lang, probably for his Newsweeder: Learning to filter netnews paper.

The articles are typical postings and thus have headers including subject lines, signature files, and quoted portions of other articles.

Here is an example document:

```
> print(data$text[1])
[1] "From: cubbie@garnet.berkeley.edu (                                    )
Subject: Re: Cubs behind Marlins? How? Article-I.D.: agate.1pt592$f9a Organization:
University of California, Berkeley Lines: 12 NNTP-Posting-Host: garnet.berkeley.edu
gajarsky@pilot.njin.net writes:  morgan and guzman will have era's 1 run higher than last year,
and  the cubs will be idiots and not pitch harkey as much as hibbard.
castillo won't be good (i think he's a stud pitcher)
This season so far, Morgan and Guzman helped to lead the Cubs        at top in ERA,
even better than THE rotation at Atlanta.        Cubs ERA at 0.056 while Braves at 0.059.
We know it is early        in the season, we Cubs fans have learned how to enjoy the
short triumph while it is still there. "
```

We have 11,314 documents from 20 different newsgroups.

Class	Target	# docs
alt.atheism	0	480
comp.graphics	1	584
comp.os.ms-windows.misc	2	591
comp.sys.ibm.pc.hardware	3	590
comp.sys.mac.hardware	4	578
comp.windows.x	5	593
misc.forsale	6	585
rec.autos	7	594
rec.motorcycles	8	598
rec.sport.baseball	9	597
rec.sport.hockey	10	600
sci.crypt	11	595
sci.electronics	12	591
sci.med	13	594
sci.space	14	593
soc.religion.christian	15	599
talk.politics.guns	16	546
talk.politics.mideast	17	564
talk.politics.misc	18	465
talk.religion.misc	19	377

We then attempt to clean the data:

```
#In computing, stop words are words which are filtered out
#before or after processing of natural language data (text). ...
#Other search engines remove some of the most common words including
#lexical words, such as "want" from a query in order to improve performance.

stopword=read.table("stopwords.txt")
nn=as.character(stopword$V1)
#clean the data
for(i in seq(doc)){ #for each document in doc, drop these symbols
  doc[[i]]<-gsub("/"," ",doc[[i]])
  doc[[i]]<-gsub("@"," ",doc[[i]])
  doc[[i]]<-gsub("-"," ",doc[[i]])
  doc[[i]]<-gsub("[[:punct:]]"," ",doc[[i]])
  doc[[i]]<-gsub("0-9"," ",doc[[i]])
}
# try: print(doc[[1]])
doc <- tm_map(doc, PlainTextDocument)
doc<-tm_map(doc,content_transformer(removeNumbers))
doc<-tm_map(doc,content_transformer(removePunctuation))
doc<-tm_map(doc,content_transformer(tolower))
doc<-tm_map(doc,content_transformer(stemDocument))
doc<-tm_map(doc,content_transformer(removeWords),c(stopwords("english"),nn))
```

After cleaning this is what the first document looks like.

```
> print(doc[[1]][1])
$content
[1] "  cubbie garnet berkeley
subject    cubs  marlins
article  d   agate pt fa organization  university  california  berkeley lines
nntp posting host  garnet berkeley    gajarsky pilot njin net writes   morgan
guzman   era s  run higher   year     cubs   idiots   pitch harkey    hibbard
castillo won t  good     s  stud pitcher
season    morgan  guzman helped
lead  cubs         top  era   better   rotation  atlanta
cubs era     braves         early          season   cubs fans  learned
enjoy         short triumph       "
```

Next we come up with a list "terms", the possible values for each word.

```
dtm_20_1<-DocumentTermMatrix(doc) #package tm (text mining)
print(dim(dtm_20_1))
summary(col_sums(dtm_20_1)) #summarize total time a term is used.
```

```
[1] 11314 93636
> summary(col_sums(dtm_20_1)) #summarize total time a term is used.
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     1.0     2.0    16.6     6.0 12260.0
```

Finally, we get rid of all the terms such that tf-idf is less than the median.

```
summary(term_tfidf)
dtm_20_1 <- dtm_20_1[,term_tfidf >= 0.06586]
summary(col_sums(dtm_20_1))
print(dim(dtm_20_1))
[1] 11314 46817
```

```
> set.seed(27)
> junk = colnames(dtm_20_1) # terms
> ii = sample(1:length(junk),20)
> print(junk[ii])
 [1] "wjbc"         "billingham"   "tempers"      "froumentin"   "demarrais"
 [6] "hsi"          "bdown"        "abou"         "chanting"     "crone"
[11] "ofnwygwbvoa"  "suicidal"     "lifespan"     "reso"         "teskey"
[16] "sentient"     "xls"          "bls"          "neurological" "arrognace"
```

Create a train and test.

```
set.seed(99)
id_20=sample(1:11314,size=9000,replace=FALSE)
train=dtm_20_1[id_20,]
test=dtm_20_1[-id_20,]
save(train,test,file="lda-train-test.rda")
```

```
> dim(train)
[1]  9000 46817
> dim(test)
[1]  2314 46817
```

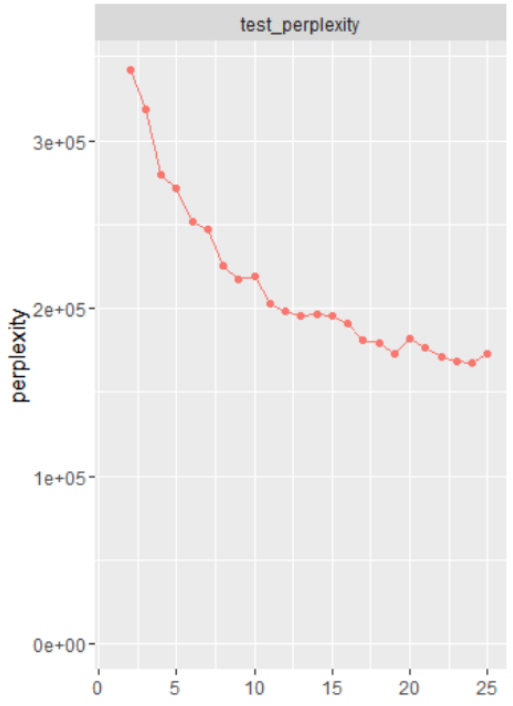
Next we fit a bunch of LDA models with varying number of topics to try to get a feeling for what the right number of topics is.

We look at the *perplexity* on the test data.

$$perplexity(Data_{test}) = exp\{-\frac{\sum_{d=1}^{D} \log p(w_d)}{\sum_{d=1}^{D} N_d}\}$$

Really just our old friend the deviance.

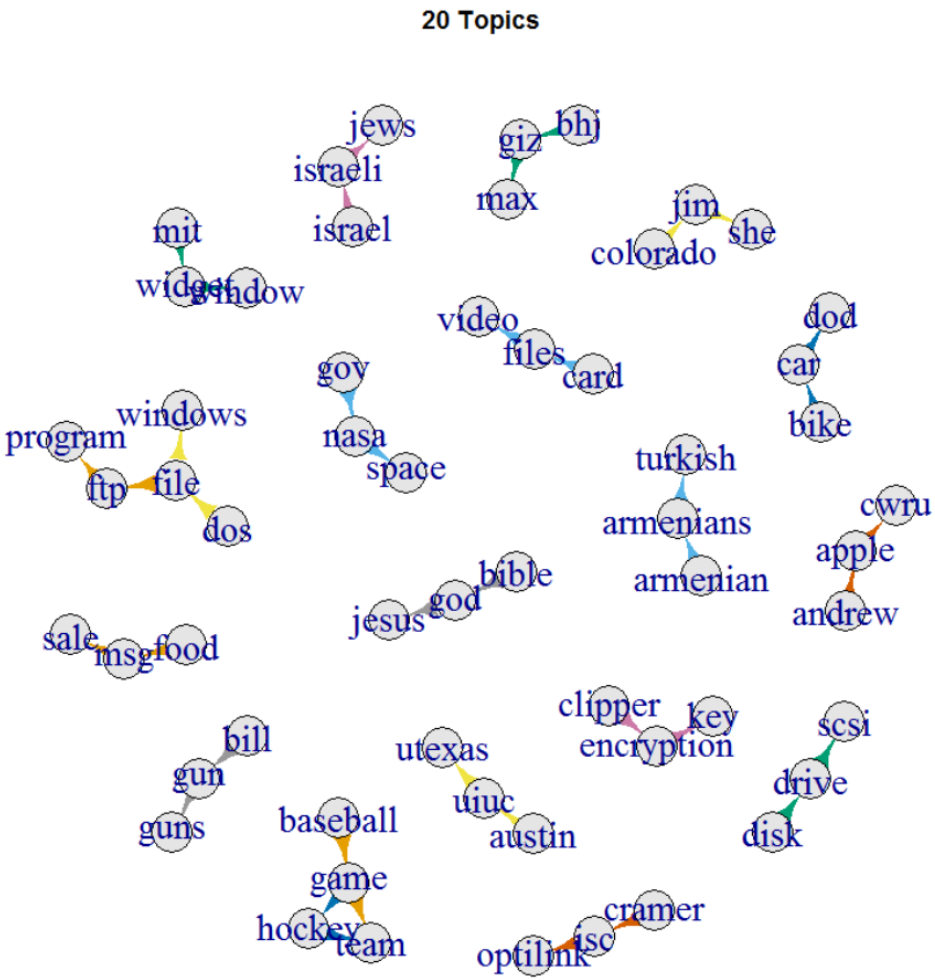
perplexity on the test data.



x axis is the number of topics.

Seems like 15-25 topics will do.

We use 20 topics.
Each topic is labeled the 3 highest probability terms from the estimated β_k.



```
> top_terms<-terms(model_20_1, k=3, threshold=0.002)
> str(top_terms)
 chr [1:3, 1:20] "caltech" "cwru" "keith" "jim" "mit" ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:20] "Topic 1" "Topic 2" "Topic 3" "Topic 4" ...
> top_terms
     Topic 1   Topic 2     Topic 3    Topic 4    Topic 5 Topic 6 Topic 7
[1,] "caltech" "jim"       "colorado" "baseball" "fbi"   "bhj"   "gov"
[2,] "cwru"    "mit"       "cramer"   "game"     "gun"   "giz"   "nasa"
[3,] "keith"   "objective" "virginia" "games"    "guns"  "max"   "space"
     Topic 8 Topic 9      Topic 10 Topic 11     Topic 12    Topic 13
[1,] "bible" "chip"       "mit"    "bike"       "armenian"  "graphics"
[2,] "god"   "encryption" "server" "motorcycle" "armenians" "image"
[3,] "jesus" "key"        "window" "ride"       "car"       "windows"
     Topic 14  Topic 15         Topic 16 Topic 17 Topic 18  Topic 19 Topic 20
[1,] "greek"   "food"           "banks"  "card"   "israel"  "buf"    "game"
[2,] "turkish" "msg"            "gordon" "drive"  "israeli" "file"   "hockey"
[3,] "uiuc"    "stephanopoulos" "pitt"   "scsi"   "jews"    "output" "team"
```

```
> top_terms_5<-terms(model_20_1, k=5, threshold=0.002)
> top_terms_5
     Topic 1     Topic 2     Topic 3    Topic 4    Topic 5    Topic 6 Topic 7
[1,] "caltech"   "berkeley"  "colorado" "baseball" "fbi"      "bhj"   "access"
[2,] "cleveland" "jim"       "cramer"   "game"     "firearms" "bxn"   "gov"
[3,] "cwru"      "mit"       "optilink" "games"    "gun"      "giz"   "launch"
[4,] "keith"     "objective" "sex"      "players"  "guns"     "max"   "nasa"
[5,] "scx"       "sun"       "virginia" "team"     "stratus"  "oil"   "space"
     Topic 8        Topic 9      Topic 10 Topic 11     Topic 12    Topic 13
[1,] "bible"        "chip"       "file"   "bike"       "armenia"   "file"
[2,] "christianity" "clipper"    "mit"    "bmw"        "armenian"  "files"
[3,] "faith"        "encryption" "motif"  "motorcycle" "armenians" "graphics"
[4,] "god"          "key"        "server" "ride"       "car"       "image"
[5,] "jesus"        "privacy"    "window" "riding"     "turkish"   "windows"
     Topic 14  Topic 15         Topic 16 Topic 17 Topic 18  Topic 19  Topic 20
[1,] "cso"     "clinton"        "banks"  "card"   "arab"    "buf"     "game"
[2,] "greek"   "food"           "geb"    "disk"   "israel"  "col"     "hockey"
[3,] "key"     "msg"            "gordon" "drive"  "israeli" "entries" "nhl"
[4,] "turkish" "stephanopoulos" "pitt"   "mac"    "jewish"  "file"    "season"
[5,] "uiuc"    "tax"            "radar"  "scsi"   "jews"    "output"  "team"
```

And here are some θ_d estimates.

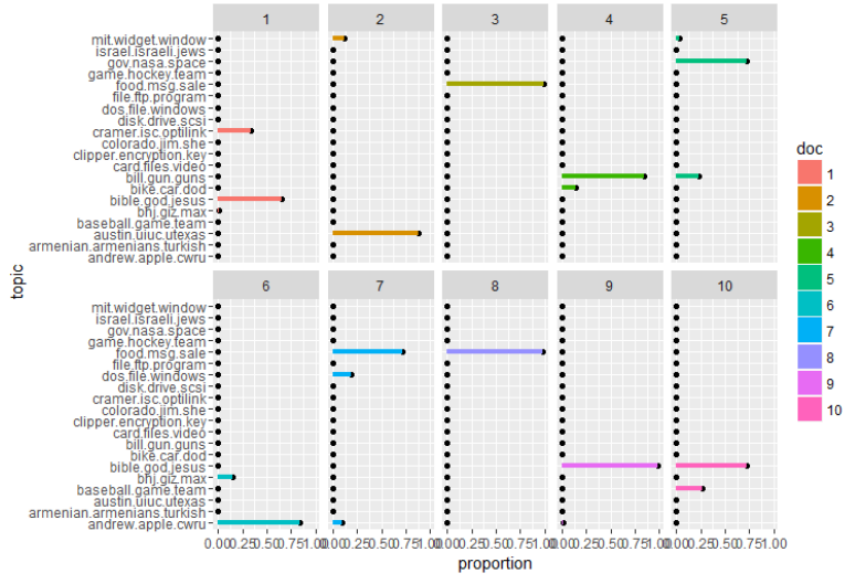


Figure 8: **Topic weights to 10 samples in testing data**