Unpacking sts: An R package for Structural Topic and Sentiment-Discourse Models

Shawn Mankad
North Carolina State University

Li Chen
Cornell University

Abstract

This paper demonstrates how to use the Structural Topic and Sentiment-Discourse Model **sts** R package. The Structural Topic and Sentiment-Discourse model allows researchers to estimate topic models with document-level metadata that determine both topic prevalence and sentiment-discourse. The sentiment-discourse is modeled as a document-level latent variable for each topic that modulates the word frequency within a topic. These latent topic sentiment-discourse variables are controlled by the document-level metadata. Estimation is accomplished through a fast variational inference method with Laplace approximation. The **sts** package can be useful for regression analysis with text data in addition to topic modeling's traditional use of descriptive analysis.

Keywords: topic prevalence, topic sentiment, topic discourse, text analysis, **sts**, R.

1. Introduction

Text data continues to proliferate in social science research, stemming from sources like emails, social media posts, surveys, generative texts from large language models, and more. The vast availability of such data along with document-level metadata (e.g., author demographics, time-stamps) led to the development of the Structural Topic Model (STM) (Roberts, Stewart, Tingley, Lucas, Leder-Luis, Gadarian, Albertson, and Rand 2014; Roberts, Stewart, and Airoldi 2016), which was distinctive in its ability to incorporate metadata into topic models to better summarize the content within text documents. The model, along with the **stm** R package (Roberts, Stewart, and Tingley 2019), allows researchers to discover topics and estimate their relationship to document metadata through regression analysis of latent topic prevalence (the proportion of a document devoted to a topic).

In this paper, we present an extension of the STM called the Structural Topic and Sentiment-Discourse (STS) model (Chen and Mankad 2024), implemented in the **sts** R package.[1] The STS model assumes that people express in written texts their sentiment and discourse, or "sentiment-discourse" (a term coined in Chen and Mankad 2024), on a certain topic by their choice of words, including topic content, tone, and style-related words. Compared to the STM, the STS model further estimates document-level latent sentiment-discourse variables for each topic, in addition to estimating how the topic sentiment-discourse and prevalence are driven by the document-level metadata.

The goal of the STS model is to enable researchers to uncover the underlying themes present

[1]We thank our research assistant Nala Peng for her help in translating portions of our code to C++.

in the text while additionally detecting the sentiment-discourse associated with each topic, through regression analysis of latent topic prevalence and latent topic sentiment-discourse based on the document-level metadata covariates. Applications of this model are wide-ranging. For example, STS can be used to analyze emails, news media, blog posts, online reviews, texts generated from large language models, and more, so long as document-level metadata is available alongside the text. When combined with other packages such as **tm** and **stm**, the **sts** package offers a streamlined workflow for text data analysis, including ingesting and processing raw text, estimating the model, and examining and visualizing results.

The outline of this paper is as follows. In Section 2, we introduce the technical aspects of the STS, including the data generating process and an overview of estimation. In Section 3, we provide examples of how to use the model and the **sts** package, including implementing the model and plots to visualize model output. We also cover in Section 3.5 settings to control details of estimation. Section 4 concludes with a brief discussion of the model and software.

2. The Structural Topic and Sentiment-Discourse Model

To set up the STS model, below we reproduce the model description and notation from the main reference work by the same authors (Chen and Mankad 2024). Let D be the corpus of observed documents (e.g., user reviews). The text in each document d is vectorized into a bag of words representation, denoted as $\boldsymbol{w}_{d,n}$, where n refers to the index of the word sequence in the document. Additionally, for each document, certain characteristics are observed, such as the star rating, reviewer status or publication date in the case of online reviews. These document-specific variables are organized into a row vector $\boldsymbol{x}_d$ of length $1 \times (i_{\boldsymbol{x}} + 1)$, where $i_{\boldsymbol{x}}$ is the number of features in $\boldsymbol{x}_d$ plus one for an intercept. This vector $\boldsymbol{x}_d$ influences both the prevalence of topics and sentiment-discourse associated with them, as will be elaborated below.

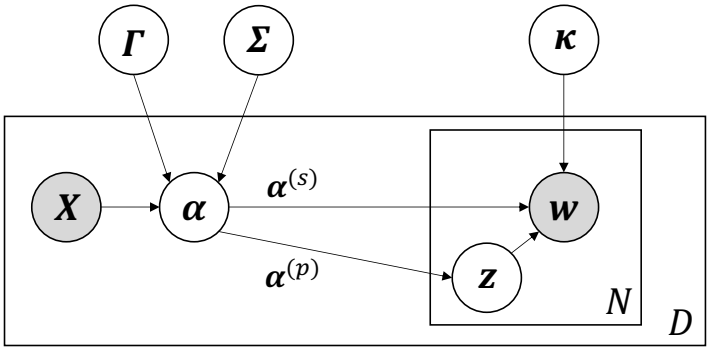

Figure 1: Plate notation diagram of the STS model.

The model assumes the existence of K topics in the corpus. For each document d, the STS model posits a latent $2K \times 1$ vector $\boldsymbol{\alpha}_d = \left[(\boldsymbol{\alpha}_d^{(p)})^T, (\boldsymbol{\alpha}_d^{(s)})^T\right]^T$, where $\boldsymbol{\alpha}_d^{(p)}$ is a $K \times 1$ vector that controls the *latent topic prevalence* vector $\boldsymbol{\theta}_d = [\theta_{d,1}, \ldots, \theta_{d,K}]^T$, which is a K-dimensional probability vector which captures the expected proportion of words in document d associated with each topic. Meanwhile, $\boldsymbol{\alpha}_d^{(s)}$ is a $K \times 1$ vector of the *latent topic sentiment-discourse* that modulates the word frequency within each topic. The following steps outline the generative

process, as illustrated in the plate diagram (see Figure 1), with key notation defined in Table 1.

Table 1: List of notation used in the STS model.

Symbol	Description
$\boldsymbol{w}_{d,n}$	Observed word frequencies
K	Number of latent topics
$\boldsymbol{x}_d$	Observed covariates that control topic prevalence and topic sentiment-discourse
$\boldsymbol{\alpha}_d^{(p)}$	Latent variables that determine topic prevalence
$\boldsymbol{\alpha}_d^{(s)}$	Latent variables that determine topic sentiment-discourse
$\boldsymbol{\gamma}_k^{(p)}$	Regression coefficients that determine $\boldsymbol{\alpha}_d^{(p)}$
$\boldsymbol{\gamma}_k^{(s)}$	Regression coefficients that determine $\boldsymbol{\alpha}_d^{(s)}$
$\boldsymbol{\theta}_d$	Latent topic prevalence
$\boldsymbol{\alpha}_d$	The collection of latent variables for each document with $\boldsymbol{\alpha}_d = [\boldsymbol{\alpha}_d^{(p)}, \boldsymbol{\alpha}_d^{(s)}]$
$\boldsymbol{\Gamma}$	Regression coefficient matrix with $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1^{(p)}, \ldots, \boldsymbol{\gamma}_K^{(p)}, \boldsymbol{\gamma}_1^{(s)}, \ldots, \boldsymbol{\gamma}_K^{(s)}]$
$\boldsymbol{\Sigma}$	Covariance matrix for $[\boldsymbol{\alpha}_d^{(p)}, \boldsymbol{\alpha}_d^{(s)}]$
$\boldsymbol{z}_{d,n}$	Word-topic assignment
$\boldsymbol{\beta}_{d,k}$	Topic-word distributions
$\kappa_{k,v}^{(t)}$	Topic baseline coefficients that control $\boldsymbol{\beta}_{d,k}$
$\kappa_{k,v}^{(s)}$	Topic sentiment-discourse coefficients that control $\boldsymbol{\beta}_{d,k}$
$\boldsymbol{\kappa}$	The collection of $\kappa_{k,v}^{(t)}$ and $\kappa_{k,v}^{(s)}$ for each topic k across the entire vocabulary

Step 1: The latent topic vector $\boldsymbol{\alpha}_d$ is drawn from a multivariate normal distribution with external covariates $\boldsymbol{x}_d$:

$$\boldsymbol{\alpha}_d = [\alpha_{d,1}^{(p)}, \ldots, \alpha_{d,K}^{(p)}, \alpha_{d,1}^{(s)}, \ldots, \alpha_{d,K}^{(s)}]^T \sim \text{Normal}((\boldsymbol{x}_d\boldsymbol{\Gamma})^T, \boldsymbol{\Sigma}),$$

where $\boldsymbol{\Gamma} = [\boldsymbol{\gamma}_1^{(p)}, \ldots, \boldsymbol{\gamma}_K^{(p)}, \boldsymbol{\gamma}_1^{(s)}, \ldots, \boldsymbol{\gamma}_K^{(s)}]$ is a matrix of size $(i_{\boldsymbol{x}}+1) \times 2K$ and $\boldsymbol{\Sigma}$ is the $2K \times 2K$ covariance matrix that captures possible correlations between topic prevalence and sentiment-discourse across different topics. The latent topic prevalence vector $\boldsymbol{\theta}_d = [\theta_{d,1}, \ldots, \theta_{d,K}]$ is derived by the softmax transformation

$$\theta_{d,k} = \frac{\exp(\alpha_{d,k}^{(p)})}{\sum_k \exp(\alpha_{d,k}^{(p)})}. \tag{1}$$

Step 2: For each word in a document d, indexed by n, the model selects a topic assignment, $\boldsymbol{z}_{d,n}$, via a multinomial distribution over the topic probabilities $\boldsymbol{\theta}_d$:

$$\boldsymbol{z}_{d,n} \sim \text{Multinomial}_K(\boldsymbol{\theta}_d), \text{ for } n = 1, \ldots, N_d,$$

where N_d is the total number of words in the document.

Step 3: Finally, each observed word in the document is generated from another multinomial distribution over the corpus vocabulary:

$$\boldsymbol{w}_{d,n} \sim \text{Multinomial}_V(\boldsymbol{B}_d\boldsymbol{z}_{d,n}), \text{ for } n = 1, \ldots, N_d,$$

where $\boldsymbol{B}_d = [\boldsymbol{\beta}_{d,1}, \ldots, \boldsymbol{\beta}_{d,K}]$ is a $V \times K$ matrix encoding the topic-word distributions. For each word v in the vocabulary, the probability of its occurrence in a topic is modeled as:

$$\beta_{d,k,v} = \frac{\exp(m_v + \kappa_{k,v}^{(t)} + \kappa_{k,v}^{(s)} \alpha_{d,k}^{(s)})}{\sum_v \exp(m_v + \kappa_{k,v}^{(t)} + \kappa_{k,v}^{(s)} \alpha_{d,k}^{(s)})}, \tag{2}$$

where m_v is the baseline log-transformed occurrence rate of word v in the corpus. The coefficients $\kappa_{k,v}^{(t)}$ and $\kappa_{k,v}^{(s)}$ control the topic baseline and topic sentiment-discourse, respectively. This generative process introduces two main innovations compared to the STM (Roberts *et al.* 2016). First, the latent topic vector $\boldsymbol{\alpha}_d$ is extended to include a sentiment-discourse component $\boldsymbol{\alpha}_d^{(s)}$ absent from the STM. Second, the topic-word distribution $\beta_{d,k,v}$ is now influenced by the new latent *topic* sentiment-discourse variable $\alpha_{d,k}^{(s)}$ through $\kappa_{k,v}^{(t)} + \kappa_{k,v}^{(s)} \alpha_{d,k}^{(s)}$, whereas STM only allows $\beta_{d,k,v}$ to be modulated by a small number of *observed* (categorical) covariates.

As is common with topic model estimation, the posterior distribution for the STS model is intractable. To avoid lengthy computing times associated with sampling from the posterior via Markov chain Monte Carlo techniques (Blei, Kucukelbir, and McAuliffe 2017), we turn to a fast variant of expectation-maximization (EM) method with nonconjugate variational inference. For the E-step, we apply the Laplace approximation method (also used in STM; see Wang and Blei 2013; Roberts *et al.* 2016), to overcome the intractability and estimate the latent topic prevalence and topic sentiment variables $\boldsymbol{\alpha}_d$ jointly. In the M-Step, we provide multiple options for estimating $\boldsymbol{\kappa}$, including estimation with ℓ_1 lasso regularization as well as a sample aggregation approach for Poisson regression. In this paper, we provide brief interpretations of results, and we direct readers to the companion paper (Chen and Mankad 2024) for full technical details.

3. Using the Structural Topic and Sentiment-Discourse Model

In this section we demonstrate the basics of using the **sts** package. Figure 2 presents an overview of the package and its main functions in context of a typical workflow. For each step, we list different functions in the **sts** and related packages that accomplish each task. First, users ingest the data and prepare it for analysis using tools from **stm** and **tm**. Next a structural topic and sentiment-discourse model is estimated using the core `sts` function. As we demonstrate below, the ability to estimate the structural topic and sentiment-discourse model allows for the evaluation (`topicExclusivity`, `topicSemanticCoherence`, `heldoutLikelihood`), understanding (`printTopWords`, `findRepresentativeDocs`, `estimateRegns`, `printRegnTables`), and visualization of results (`plot`, `plotRepresentativeDocs`). All functions also come with help files and examples, which can be accessed by typing ? and then the function name.

To illustrate how to use the **sts** package, we will use online reviews data from Chen and Mankad (2024) for restaurants on the Yelp platform from two cities, Phoenix, AZ and Charlotte, NC, from September 1, 2019 to September 15, 2020. The dataset consists of 681 restaurants and 36,015 total reviews that span several sub-periods during the COVID-19 pandemic: (i) "Pre-COVID', which includes all observations in 2019; (ii) "Buildup", which spans January 1 - February 29, 2020; (iii) "Onset", which spans March 1 - 30, 2020; (iv) "Stay-at-Home", which spans April 1, 2020 to when Charlotte and Phoenix were subject to

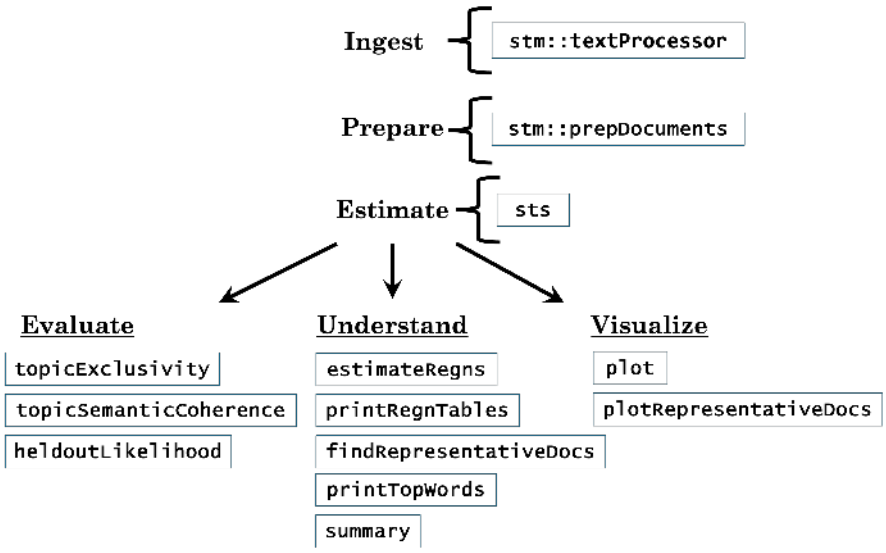

Figure 2: Overview of the **sts** package.

their respective state-wide lockdown orders (May 15, 2020 for Charlotte and May 22, 2020 for Phoenix); (v) "New-Normal" for all reviews thereafter until the observation period ends on September 15, 2020. The data also include an indicator of whether the reviewer is an elite user (a designation that Yelp bestows on its users who are highly active influencers on the platform), in addition to the star rating of the review, and the daily total rainfall, average temperature, and average wind speed for the restaurant city.

3.1. Ingest and Prepare: Reading and Organizing Data

The first step is to load data into R. Our Yelp reviews are stored in a .csv file, with the textual data stored alongside associated metadata in separate columns. We first read this data into a data frame, then invoke `textProcessor` and `prepDocuments` from **stm** to load, clean, and organize the data. Note that we use the `lower.thresh` argument to remove infrequent words from the analysis.[2] Then we can use the `prepDocuments` function to process the loaded data and ensure they are in the correct format; importantly, the function also re-indexes the data if any changes occur due to processing. For example, if a document is removed because it contained only rare words, then `prepDocuments` will drop the corresponding row in the metadata as well.

```
yelp_data <- read.csv("YelpData.csv")

processed <- textProcessor(yelp_data$text, metadata = yelp_data)
out <- prepDocuments(processed$documents, processed$vocab, processed$meta,
       lower.thresh = 30)
```

We will include interactions between the sub-period and elite status indicator variables in

[2]For data not in a spreadsheet format, we refer the interested reader to the discussion in Roberts *et al.* (2019).

our model. We adjust their levels to set an appropriate baseline value. We are now ready to estimate the STS model.

```
out$meta$SubPeriod <- factor(out$meta$SubPeriod,
        levels = c("Pre-COVID", "Buildup", "Onset", "Stay_at_Home", "New_Normal"))
out$meta$eliteStatus <- factor(out$meta$eliteStatus, levels = c("Non-Elite", "Elite"))
```

3.2. Estimate: Fitting the STS Model

Estimation the STS model proceeds with the workhorse `sts` function. In this example, we use the sub-period, elite status, their interaction, as well as the review rating, weather controls, and city indicator as covariates that determine both topic prevalence and topic sentiment-discourse. To illustrate, we estimate an STS model with 10 topics. The first argument specifies the formula for covariates that explain both topic prevalence and topic sentiment-discourse. The second argument is used during initialization and specifies a variable over which topic sentiment-discourse varies. In our dataset, review rating is an ideal choice because we expect the tone to differ between high- and low-rating reviews. In other contexts, this argument might represent something like experimental versus control group assignments.

```
sts_result <- sts(~ SubPeriod*eliteStatus + stars  + meanTemp + sumPrecip +
        meanWind + city, ~ stars, out, K = 10)
```

The model is set to run by default for a maximum of 100 EM iterations. Convergence is monitored by the change in the approximate variational lower bound. Once the bound has a small enough change ($< 10^{-5}$ by default) between iterations, the model is considered converged. To reduce compiling time, in this paper we do not run the models and instead load the model already estimated.

```
sts_result <- readRDS("sts_yelp.RDS")
```

3.3. Understand: Interpreting the STS by Inspecting Results

Understanding topics through words and example documents

We describe several complimentary ways for users to explore the estimated topics, understand the underlying latent themes and the sentiment-discourse around them. First, users can examine the top most-likely words associated with each topic by using the `plot` and `printTopWords` functions. Both functions print the top `n` words (set to 10 by default). The `plot` function plots the top words along with their estimated probability as a graphic barplot, whereas `printTopWords` prints the words to the console. Note that the probability of topic-words depends on the value of the sentiment-discourse variable $\alpha_{d,k}^{(s)}$. Both functions therefore print the top words for when $\alpha_{d,k}^{(s)}$ equals its average value, the 10th percentile over all documents reflecting negative sentiment-discourse, and the 90th percentile reflecting positive sentiment-discourse. These default percentile thresholds can be modified by adjusting arguments `lowerPercentile` and `upperPercentile`.

Figure 3: The top most likely words for each topic. Average, positive, and negative sentiment are defined by setting $\alpha_{d,k}^{(s)}$ equal to its average, 90th percentile, and 10th percentile values, respectively.

We generate the plot shown in Figure 3, displaying the 10 most likely words for each topic using `plot(sts_result, n = 10)`.

From the figure, while most topics seem to focus on food, Topics 2, 3, 6, and 8 seem to focus on other service quality dimensions. For example, we see "covid" as a keyword for Topic 3. To complement the keywords and get a better sense of the theme captured by this topic, we examine the reviews with the highest Topic 3 prevalence (likelihood of Topic 3 content) in Figure 4.

```
docs <- findRepresentativeDocs(sts_result, out$meta$text, topic = 3, n = 3)
plotRepresentativeDocs(docs, text.cex = 0.6)
```

Juxtaposed with the keywords, the top prevalence-value reviews provide evidence that Topic 3 focuses on COVID-19 and hygiene. Next, we examine how the prevalence and sentiment-discourse around this topic relates to the metadata.

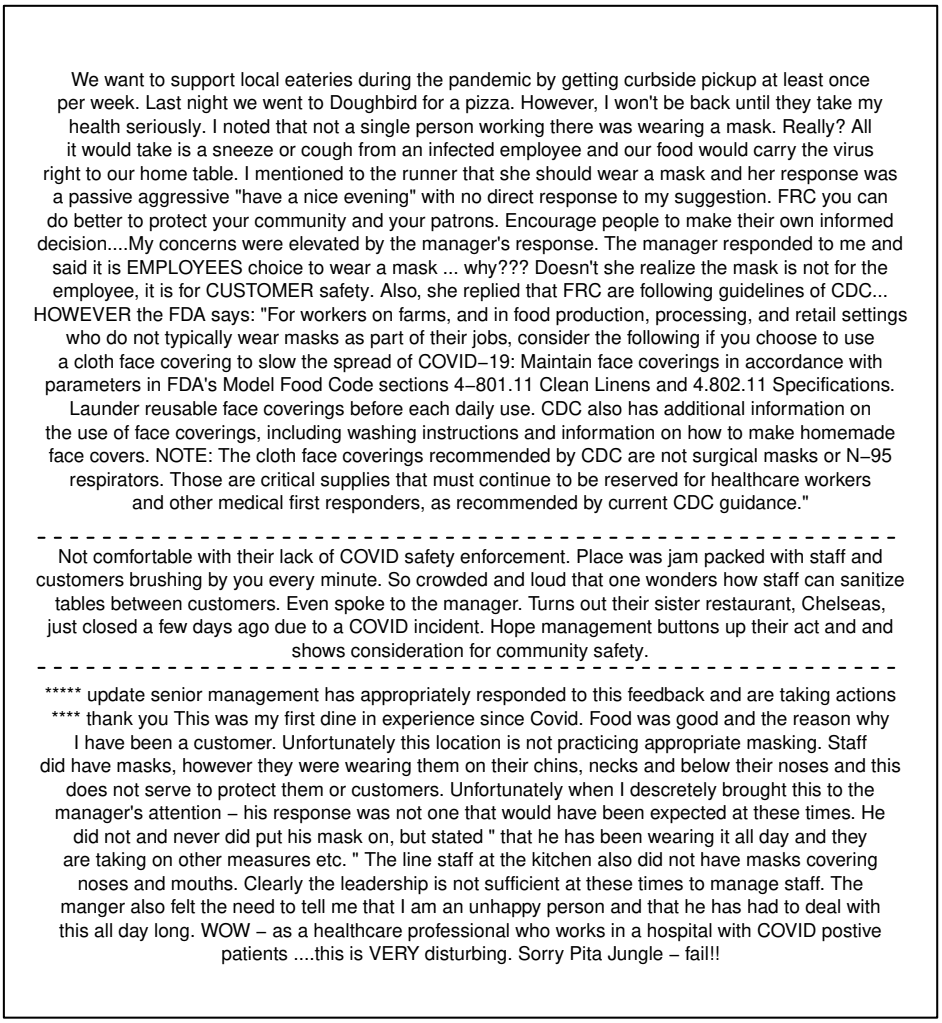

Figure 4: The three reviews that have the largest prevalence for Topic 3.

Estimating metadata/topic relationships

Estimating the relationship between metadata and topic prevalence and sentiment-discourse is accomplished using the `estimateRegns` function. The syntax of the function is designed such that users specify a formula for the metadata and topics of interest. Following the closely related `estimateEffect` function from the **stm** library, the `estimateRegns` function incorporates estimation uncertainty of the topic proportions into the uncertainty estimates using the method of composition. Calling `printRegnTables` on the `estimateRegns` object will generate a regression table.

Below we estimate and plot the regression of latent prevalence and sentiment-discourse on the metadata. We focus on Topic 3 through the `topics` argument which may include multiple topics, e.g., `topics = c(3,6)` to plot the results for Topics 3 and 6. From the prevalence results, we learn that that discussion of COVID-19 started during the Onset sub-period and grew in importance in each subsequent sub-period. The negative and significant coefficient for stars indicates that reviews with lower ratings tended to focus on COVID-19. We also see

evidence that elite reviewers tended to focus on this topic less than non-elite reviewers.

The sentiment-discourse results show that the discussion around this topic was particularly negative during the Stay-at-Home sub-period. Elite reviewers were more positive when discussing this topic than non-elite reviewers except during the final New-Normal sub-period in our data.

```
regns <- estimateRegns(sts_result, ~ SubPeriod*eliteStatus + stars  + meanTemp +
        sumPrecip + meanWind + city, out)
printRegnTables(regns, topics = 3)

Topic 3 prevalence:

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
Intercept                                1.287e-01  5.402e-03  23.831  < 2e-16 ***
SubPeriodBuildup                        -2.780e-03  2.061e-03  -1.349 0.177329
SubPeriodOnset                           6.055e-03  3.252e-03   1.862 0.062574 .
SubPeriodStay_at_Home                    3.269e-02  3.900e-03   8.384  < 2e-16 ***
SubPeriodNew_Normal                      4.671e-02  2.386e-03  19.573  < 2e-16 ***
eliteStatusElite                        -9.386e-03  2.523e-03  -3.720 0.000199 ***
stars                                   -1.726e-02  5.029e-04 -34.312  < 2e-16 ***
meanTemp                                -1.045e-04  6.881e-05  -1.519 0.128778
sumPrecip                                4.478e-05  1.891e-04   0.237 0.812832
meanWind                                -1.684e-04  3.382e-04  -0.498 0.618529
cityPhoenix                              6.438e-03  3.871e-03   1.663 0.096305 .
SubPeriodBuildup:eliteStatusElite       -1.317e-03  3.761e-03  -0.350 0.726338
SubPeriodOnset:eliteStatusElite         -2.236e-03  6.735e-03  -0.332 0.739848
SubPeriodStay_at_Home:eliteStatusElite -1.032e-04  8.429e-03  -0.012 0.990231
SubPeriodNew_Normal:eliteStatusElite     6.982e-03  4.732e-03   1.476 0.140042
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Topic 3 sentiment:

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
Intercept                               -16.578320   0.267841 -61.896  < 2e-16 ***
SubPeriodBuildup                         -0.084683   0.094438  -0.897  0.36988
SubPeriodOnset                            0.079721   0.145208   0.549  0.58300
SubPeriodStay_at_Home                    -0.526391   0.172318  -3.055  0.00225 **
SubPeriodNew_Normal                       0.108382   0.110081   0.985  0.32485
eliteStatusElite                          0.531743   0.134485   3.954  7.7e-05 ***
stars                                     2.765251   0.023490 117.720  < 2e-16 ***
meanTemp                                 -0.009331   0.003394  -2.750  0.00597 **
sumPrecip                                -0.023164   0.010149  -2.282  0.02247 *
meanWind                                 -0.014429   0.016899  -0.854  0.39320
cityPhoenix                              -0.525776   0.185593  -2.833  0.00461 **
```

```
SubPeriodBuildup:eliteStatusElite         0.259085   0.212301   1.220  0.22233
SubPeriodOnset:eliteStatusElite          -0.284937   0.338928  -0.841  0.40052
SubPeriodStay_at_Home:eliteStatusElite   -0.397608   0.393964  -1.009  0.31286
SubPeriodNew_Normal:eliteStatusElite     -0.724383   0.229914  -3.151  0.00163 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.4. Evaluate: Parameter and Model Selection

To assess several fitted candidate models, one can compare semantic coherence and exclusivity for each model and topic. Coherence is a criterion developed by Mimno, Wallach, Talley, Leenders, and McCallum (2011) that is maximized when the most probable words in a given topic frequently co-occur together. Let $D(v, v')$ be the number of times that words v and v' appear together in a document. Then for the top M most probable words for topic k, coherence is defined as

$$C_k = \sum_{i=2}^{M} \sum_{j=1}^{i-1} \log \left(\frac{D(v_i, v_j) + 1}{d(v_j)} \right). \tag{3}$$

Mimno *et al.* (2011) also show that the metric correlates well with human judgment of topic quality. However, Roberts *et al.* (2014) noted that attaining high average coherence can be achieved by having a few topics dominated by very common words. Thus, Roberts *et al.* (2019) suggest additionally measuring exclusivity, defined as the harmonic mean of the word rank in terms of uniqueness to a topic and frequency of usage

$$FREX_{k,v} = \left(\frac{\omega}{\text{ECDF}(\beta_{d,k,v} / \sum_{j=1}^{K} \beta_{d,j,v})} + \frac{1-\omega}{\text{ECDF}(\beta_{d,k,v})} \right)^{-1}, \tag{4}$$

where ECDF is the empirical CDF and $\omega = 0.7$. Higher scores on the coherence and exclusivity indicate better topic quality.

As shown below, in the **sts** package, the `topicSemanticCoherence` and `topicExclusivity` functions compute coherence and exclusivity based on the topic word distribution, given the average estimated $\alpha^{(s)}$ for each topic over all documents.

```
topicExclusivity(sts_result)
topicSemanticCoherence(sts_result, out)
```

For a data-driven approach to selecting K, the number of topics, one can compare how well the model predicts word occurrence within a document for different values of K. Specifically, to help assess the model's predictive performance, the document-completion heldout log-likelihood method (Asuncion, Welling, Smyth, and Teh 2009; Wallach, Murray, Salakhutdinov, and Mimno 2009; Hoffman, Blei, Wang, and Paisley 2013; Roberts *et al.* 2019) estimates the probability of words appearing within a document when those words have been removed from the document. We utilize the `make.heldout` function in the **stm** package to randomly remove words in documents for model estimation. The **sts** package includes `heldoutLikelihood` to evaluate the heldout log-likelihood for missing words.

```
out_ho <- make.heldout(out$documents, out$vocab)
out_ho$meta <- out$meta
sts_result_ho <- sts(~ SubPeriod*eliteStatus + stars  + meanTemp + sumPrecip +
        meanWind + city, ~ stars, out_ho, K = 10)
heldoutLikelihood(sts_result_ho, missing = out_ho$missing)
```

3.5. Configure: Adjusting Estimation Defaults

In this section, we discuss how to change default arguments in the main `sts` estimation function. We will cover different initialization settings, two options for estimating the $\boldsymbol{\kappa}$ coefficients, how to set and evaluate the convergence criteria, and the use of CPU parallelization to speed up the overall estimation.

Model initialization

As with most topic models, the initialization strategy for the variational EM algorithm can have a large impact on the final results. In the `sts` function, we provide two methods of initialization that can be accessed with the argument `initialization`. In both cases, initialization of $\alpha_{d,k}^{(s)}$ depends on a document variable, such as review ratings in the online reviews context, which we identified earlier with the second argument (named `initializationVar`) of the `sts` function.

By default, the initialization for the STS is set to `initialization = "anchor"`, which implements the setup discussed in detail in Section 4.3 of Chen and Mankad (2024). It uses the spectral initialization strategy of Roberts *et al.* (2019) to initialize $\alpha_{d,k}^{(p)}$. For $\alpha_{d,k}^{(s)}$, an initial score that is the same for all topics within a document is assigned based on the variable identified in `initializationVar`. Initial $\boldsymbol{\kappa}$ values are estimated as described below. If $\hat{\kappa}_{k,v}^{(s)} = 0$, we randomly assign it ± 0.001 to ensure all values are nonzero.

Users can also set to `initialization = "stm"`, which fits an STM model with the variable identified in `initializationVar` as a content covariate. The estimates of the fitted model are then used to initialize all parameters and variables. To avoid having to fit an STM model when running `sts`, users can additionally pass in a prefit STM model via the argument `stmSeed`.

Estimation of word-choice modulating parameters

Estimation of the word-choice modulating parameters $\boldsymbol{\kappa}$ is controlled with the `kappaEstimation` argument and performed using the distributed multinomial regression framework of Taddy (2015), which the STM method (Roberts *et al.* 2016, 2019) also adopts. The idea is to estimate the $\boldsymbol{\kappa}$ parameters of the single multinomial logistic regression through V independent Poisson regressions (one for each element of the vocabulary), where the samples are aggregated within each level of `initializationVar`. This estimation strategy allows for the Poisson regressions to be parallelized over the vocabulary, which we discuss below in Section 3.5.4. When `kappaEstimation = "lasso"`, we use the **glmnet** (Friedman, Hastie, and Tibshirani 2010) to encourage sparsity. The default option sets `kappaEstimation = "adjusted"`, which further adjusts the weights of the aggregated samples before estimating each regularized Poisson regression – see Section 4.2 of Chen and Mankad (2024) for detailed discussion. With either setting, the regularization parameter is set automatically according to the AIC.

Convergence criteria

Convergence of the variational EM algorithm is controlled by relative change in the approximate evidence lower bound (ELBO), measured by $\frac{\text{ELBO}_i - \text{ELBO}_{i-1}}{|\text{ELBO}_{i-1}|}$, where i denotes the iteration number. The default tolerance is 10^{-5} and can be adjusted using the `convTol` argument. The maximum number of iterations is by default set to 100 and can be adjusted using the `maxIter` argument.

Following in the footsteps of the **stm** package, the `sts` function prints the status of iterations to the console (the boolean `verbose` argument turn this on or off). For each E-step and M-step, the algorithm prints one dot for every 1% of the corpus it completes and announces completion along with timing information. By default every 10th iteration will print a report of top topic and covariate words using the `printTopWords` function. Once a model has been fit, convergence can be assessed by plotting the approximate ELBO as in Figure 5 using the code below.

```
df <- data.frame(elbo = sts_result$elbo, iteration = 1:length(sts_result$elbo))
ggplot(df, aes(iteration, elbo)) + geom_point() + geom_line() + theme_bw()
        + ylab("Approximate ELBO") + xlab("Iteration Number")
```

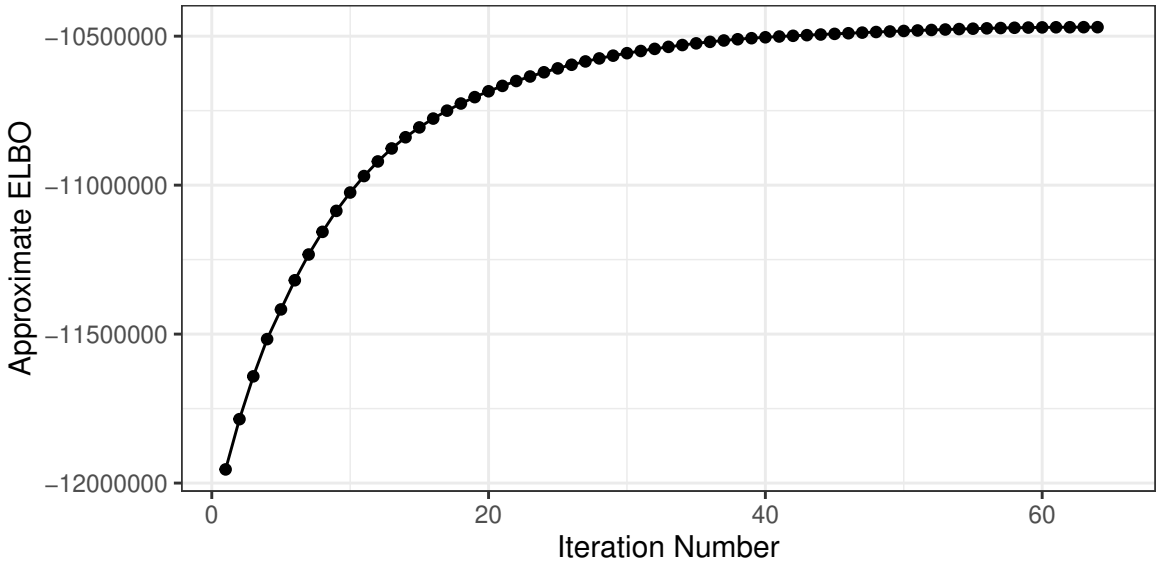

Figure 5: The approximate ELBO for each step of the variational EM estimation algorithm.

CPU parallel computing

For the variational E-step, we apply the Laplace approximation method (also used in STM; see Wang and Blei 2013; Roberts *et al.* 2016) to overcome intractability and estimate the latent topic prevalence and topic sentiment-discourse variables jointly. Details can be found in Chen and Mankad (2024). While use of the Laplace approximation is a significant advancement in computational speed (as discussed in Roberts *et al.* (2019)), the optimization for the variational E-step remains as the main bottleneck for performance. Following the architecture

of **stm**, we code the objective function and gradient in the C++ Armadillo library using the RcppArmadillo package (Eddelbuettel and Sanderson 2014). To compute the optimal $\boldsymbol{\alpha}_d^{(p)}$ and $\boldsymbol{\alpha}_d^{(s)}$ for each document, we calculate the corresponding Hessian matrix (using the analytical formula given by Chen and Mankad 2024) in C++ and then invert it via the Cholesky decomposition.

To further improve performance, the boolean `parallelize` argument controls whether to parallelize the estimation over CPU cores on a local machine. Specifically, when it is turned on, estimation for both the E-step and M-step is parallelized using the **doParallel** and **parallel** libraries. For the Yelp dataset, on an M1 MacBook Air, setting `parallelize = TRUE` reduces the time per iteration from approximately 600 seconds to 180 seconds.

4. Conclusion

The STS model of Chen and Mankad (2024) extended the STM model of Roberts *et al.* (2014, 2016) by further estimating latent topic sentiment (polarity or tone) and/or discourse (writing style and slant) variations driven by document-level metadata covariates, important features possessed by many corpuses in the social sciences. Likewise, the **sts** package builds on the foundational **stm** library to provide a state-of-the-art general-purpose topic modeling library, allowing social scientists to rigorously perform regression analysis and potential causal inference of topic prevalence and sentiment-discourse based on document metadata. This paper provides an overview of how to use the main functions of the **sts** and related libraries to ingest and prepare documents, estimate the STS model, and interpret the fitted model through visualizations and summary tables. We also discussed tools and strategies for parameter and model selection. It is our sincere hope that users will develop more advanced packages based on baseline **sts** package by, for example, further improving computational performance or initializing $\boldsymbol{\kappa}$ with a labeled dictionary in certain application context.

References

Asuncion A, Welling M, Smyth P, Teh YW (2009). "On smoothing and inference for topic models." In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 27–34.

Blei DM, Kucukelbir A, McAuliffe JD (2017). "Variational inference: A review for statisticians." *Journal of the American statistical Association*, **112**(518), 859–877.

Chen L, Mankad S (2024). "A Structural Topic and Sentiment-Discourse Model for Text Analysis." *Management Science*.

Eddelbuettel D, Sanderson C (2014). "RcppArmadillo: Accelerating R with high-performance C++ linear algebra." *Computational statistics & data analysis*, **71**, 1054–1063.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of statistical software*, **33**(1), 1.

Hoffman MD, Blei DM, Wang C, Paisley J (2013). "Stochastic variational inference." *Journal of Machine Learning Research*.

Mimno D, Wallach H, Talley E, Leenders M, McCallum A (2011). "Optimizing semantic coherence in topic models." In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pp. 262–272.

Roberts ME, Stewart BM, Airoldi EM (2016). "A model of text for experimentation in the social sciences." *Journal of the American Statistical Association*, **111**(515), 988–1003.

Roberts ME, Stewart BM, Tingley D (2019). "stm: An R package for structural topic models." *Journal of Statistical Software*, **91**(1), 1–40.

Roberts ME, Stewart BM, Tingley D, Lucas C, Leder-Luis J, Gadarian SK, Albertson B, Rand DG (2014). "Structural topic models for open-ended survey responses." *American Journal of Political Science*, **58**(4), 1064–1082.

Taddy M (2015). "Distributed multinomial regression." *Annals of Applied Statistics*, **9**(3), 1394–1414.

Wallach HM, Murray I, Salakhutdinov R, Mimno D (2009). "Evaluation methods for topic models." In *Proceedings of the 26th annual international conference on machine learning*, pp. 1105–1112.

Wang C, Blei DM (2013). "Variational inference in nonconjugate models." *Journal of Machine Learning Research*, **14**(Apr), 1005–1031.

Affiliation:

Shawn Mankad
North Carolina State University
Department of Information Technology, Analytics, and Operations
3122 Nelson Hall
2801 Founders Drive
Raleigh, NC, 27695
E-mail: smankad@ncsu.edu
URL: https://mankad-research.github.io

Li Chen
Cornell University
Samuel Curtis Johnson Graduate School of Management
Cornell SC Johnson College of Business
327 Sage Hall
Ithaca, NY, 14853
E-mail: li.chen@cornell.edu