Word Space Creator: A Visual Tool for Semantic Space Visualization

Erin M. Buchanan1 [](https://orcid.org/0000-0002-9689-4189)

Amber Gillenwaters2  [](https://orcid.org/0000-0002-7580-3591)

William E. Padfield2 [](https://orcid.org/0000-0003-1662-6895)

Abigail Van Nuland2 [](https://orcid.org/0000-0001-7954-6003)

Arielle Cunningham2 [](https://orcid.org/0000-0002-5496-1398)

Addie Wikowsky2 [](https://orcid.org/0000-0002-4031-0033)

1Harrisburg University of Science and Technology, 326 Market St, Harrisburg, PA 17101

2Missouri State University, 901 S. National Ave, Springfield, MO, 65897

Corresponding Author: Erin M. Buchanan

[ebuchanan@harrisburgu.edu](mailto:ebuchanan@harrisburgu.edu)

4092911095

Abstract

Semantic spaces are used as a representation of language, capturing the meaning between linguistic units. These spaces are often built in large corpora requiring advanced equipment, specialized computational skills, and considerable effort. This project note will introduce and demonstrate the use of an accessible Shiny graphical interface allowing users to create semantic space models easily. Shiny is an *R* package in which one can program interactive web applications in *R* for others to interact with data or analyses. The advantage to Shiny applications is that naïve users can explore data without understanding the programming, and open sharing of code with the application can aid in learning the programming for one’s own use in their research. Within the application, users will be able to load popular semantic spaces or their own corpus for semantic space creation utilizing their preferred modeling technique, including LSA and TOPICS. A variety of user-friendly graphical tools, such as *n*-nearest neighbors or topic weighted graph, will further aid data visualization of the semantic network. Additionally, the application provides the calculation of cosine or simple co-occurrence, among other popular-relatedness values. This tool is intended for researchers who may not be programming-savvy, or as a teaching extension for psycholinguistics courses.

*Keywords*: psycholinguistics, semantic spaces, LSA, visualization

Word Space Creator: A Visual Tool for Semantic Space Visualization

A semantic space model is a statistical representation of a given set of textual documents. Semantic spaces have been used to simulate a multitude of learning tasks, including priming, comprehension, word discrimination, ranking, and information retrieval [(Padó & Lapata, 2007)](https://paperpile.com/c/5HxrxN/lvoP). Semantic spaces have been used to provide insight on episodic memory retrieval indicating that a word space model of semantic similarity was correlated with human similarity ratings and recall [(Steyvers, Shiffrin, & Nelson, 2005)](https://paperpile.com/c/5HxrxN/u0eW). Computations based in semantic space modeling have proven useful in evaluating the content of written work and assigned essay grades [(Kakkonen, Myller, Timonen, & Sutinen, 2005)](https://paperpile.com/c/5HxrxN/G69C). Semantic vectors have been used to improve keyword searches [(Huang et al., 2013)](https://paperpile.com/c/5HxrxN/QOFd) and movie recommendation systems [(Bergamaschi & Po, 2015)](https://paperpile.com/c/5HxrxN/jk7c). Semantic space modeling may improve diagnosis in individuals with verbal communication, including individuals with autism or patients experiencing psychosis [(de Boer et al., 2018)](https://paperpile.com/c/5HxrxN/y9ml). The technique has also been applied in evaluating speech disorganization in individuals with schizophrenia [(Elvevåg, Foltz, Weinberger, & Goldberg, 2007)](https://paperpile.com/c/5HxrxN/ptU5). Overall, it is apparent that these models have many research applications. Next, we provide a brief overview of two models: Latent Semantic Analysis and Topics Models to help the reader orient to the necessity of user-friendly tools as a learning scaffold for those who wish to engage in research in this field.

The most commonly used semantic space model is latent semantic analysis (LSA) by Landauer and Dumais [(1997)](https://paperpile.com/c/5HxrxN/9W9e/?noauthor=1). To begin building a semantic space model using LSA, you must start with a large body of discourse, often multiple text-based documents. These documents are then transformed into a document by term matrix which represents local frequency, global frequency, and regularity. Each instance of the word undergoes a mathematical transformation by the log of its contextual frequency, which serves to provide differentiation between words that appear in the same text and links words across texts. The next transformation involves division by entropy, which provides an indication of how much semantic information is present. Singular value decomposition (SVD) is then used to generate principal components (akin to factor analysis) to optimize dimensionality. Landauer and Dumais suggest that this transformation may approximate the actual mental process, and may help explain the ability of young learners to accurately form broad conceptualizations and understand new information with minimal prompting. LSA provides a close approximation of the learnability, cohesiveness, and quality of a given document [(Landauer, Foltz, & Laham, 1998)](https://paperpile.com/c/5HxrxN/8yuj).

A popular alternative to LSA includes topic models. Similarities between topics models and LSA include a reliance on semantic information based in matrix factorization of co-occurring terms (through the process described above), reduction of dimensions, and are based on the “bag of words” model [(Steyvers & Griffiths, 2007)](https://paperpile.com/c/5HxrxN/GV6A). This form of model evaluates features without considering word order. However, rather than generating a space of words represented by single points, words are categorized into latent variables or topics. A given document can be reduced to a combination of topics. Generative topic models improve upon the simple use statistical likelihood in a text, using probabilistic sampling to accommodate for words with multiple meanings [(Blei, 2012)](https://paperpile.com/c/5HxrxN/oRNP).

Latent dirichlet allocation (LDA) models are a type of topic model representing the allocation of topics within a document, which can be expanded into more complex models [(Blei, 2012)](https://paperpile.com/c/5HxrxN/oRNP). Probabilistic topic modeling includes a posterior distribution of latent variables, based on conditions derived from the observed documents. While LDA operates under the bag of words assumption, adjustments can include the appearances of words before and after a given term. A Gaussian variational expectation-maximization (VEM) algorithm can be used for temporal data clustering to obtain maximum likelihood, with or without a fixed alpha [(Marlin, 2003)](https://paperpile.com/c/5HxrxN/jPrr). VEM can be used while allowing correlation between topics, known as correlated topics model (CTM). Gibbs sampling is a form of Markov chain Monte Carlo (MCMC), an iterative algorithmic process in which a higher-dimensional format is obtained through sequential sampling of subsetted lower-dimensions until convergence is reached [(Steyvers & Griffiths, 2007)](https://paperpile.com/c/5HxrxN/GV6A). Therefore, there are many options to creating a topics model, often involved complex math and coding procedures.

**Coding and Quantitative Reasoning**

It can be difficult for some students in the social sciences to begin working with semantic space models because the work often requires knowledge of computer programming. Many students who are interested in computational or psycho- linguistics may not yet have the needed background in coding. Learning to code can be a particularly difficult, though worthwhile, undertaking. In the modern world, the ability to code has become a highly marketable skill in the job market and a valuable learning tool for students. Unfortunately, this highly valuable skill is not easy to acquire for many students. Many students are deterred from this valuable experience because it is very difficult for them to learn, and many instructors find it difficult to teach. Computer programming itself is a way for students to exercise quantitative reasoning and problem solving strategies.

One important skill computer coding teaches students is quantitative reasoning, however, this reasoning is also coupled with math anxiety. Math anxiety is operationally defined as the panic, helplessness, paralysis, and mental disorganization that arises among some people when they are required to solve a mathematical problem [(Tobias, 1993)](https://paperpile.com/c/5HxrxN/cjLr). Approximately 30% of middle school aged students report feelings associated with math anxiety [(Suárez-Pellicioni, Núñez-Peña, & Colomé, 2016)](https://paperpile.com/c/5HxrxN/MziA). Math is something that students deal with throughout their entire academic careers, and it follows them into their professional lives no matter what paths they take.

Teaching a student quantitative reasoning is a very difficult undertaking. Traditional mathematics teaching methods have been shown not to improve students’ self-efficacy in the domain of math [(Smith, 1996)](https://paperpile.com/c/5HxrxN/sLtC). Raising the self-efficacy of students in any domain is very difficult, and it is difficult to teach a new quantitative skill to someone who already has low self-efficacy in that domain. There are modern day efforts to teach elementary school students computer programming, and to teach them these skills early on [(Lye & Koh, 2014)](https://paperpile.com/c/5HxrxN/lf1p).

It has also been shown that successful and unsuccessful quantitative problem solvers approach problems differently. Successful problem solvers change their problem solving strategies depending on the type of problem and prior experience [(Hegarty, Mayer, & Monk, 1995)](https://paperpile.com/c/5HxrxN/7ROu). This means that when they are met with failure, they will change their strategies. Unsuccessful problem solvers have been shown not to change their strategies at all [(Hegarty et al., 1995)](https://paperpile.com/c/5HxrxN/7ROu). If they are met with failure, they will continue to use the same approach. When students learn computer programming, they will use their quantitative reasoning skills and problem solving strategies. It is very difficult to break the pattern of unsuccessful problem solving, thus making it difficult to teach problem solving through computer programming.

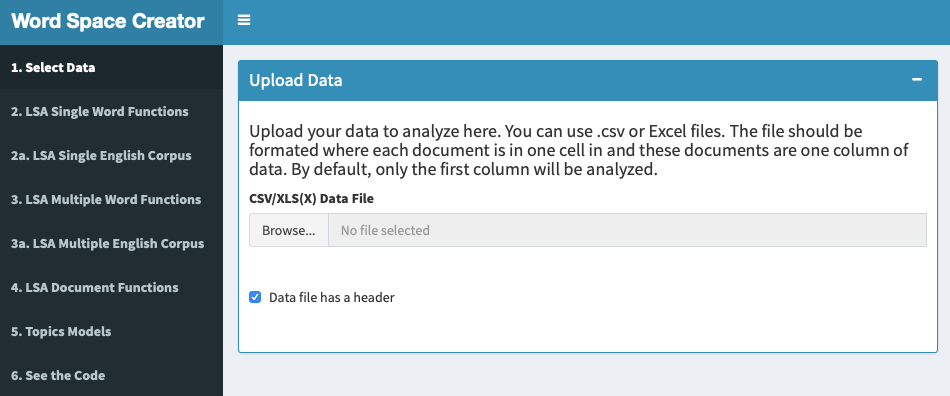
Teaching students successful problem solving strategies and quantitative reasoning is also a difficult task. When teachers were interviewed about teaching computer science to students, they reported that students and teachers seemed to have different perspectives on computer science and on what is considered a beneficial approach to problem-solving and what is considered a successful solution [(Kolikant, 2011)](https://paperpile.com/c/5HxrxN/e4eU). When students and teachers disagree on what the definition of perceived success in the course is, it makes actual learning difficult. A student who is considered a successful problem solver may not be considered so by their instructor because they used a different strategy to acquire that success.

Computer coding has shown to be a difficult area for some students to learn. Even after a semester or two of programming courses, many computer science students fail to reach expected level of competency in programming [(McCracken et al., 2001)](https://paperpile.com/c/5HxrxN/fBVg). Students without specific training in computer science are unlikely to fare any better. Therefore, it is important to provide resources to these students that foster learning, yet still emphasize the skills that computer programming provides. One type of resource to foster this learning is visualization of quantitative and programming concepts. In one study, a visual programming language made coding easier to understand, especially for students with lower self-efficacy [(Tsai, 2018)](https://paperpile.com/c/5HxrxN/xOM7). This format might be particularly beneficial for students lacking a background in computer science. We designed the Word Space Creator as a visualization tool for semantic space models, in part to aid comprehension in students studying computational and psycho- linguistics. Here, we provide a tutorial for using the Word Space Creator.

**Word Space Creator Tutorial**

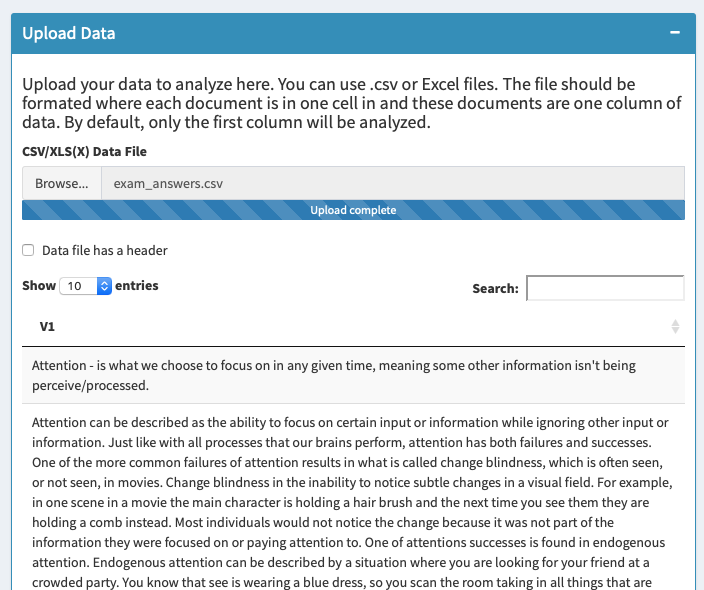
The Word Space Creator can be accessed via the internet at: <https://doomlab.shinyapps.io/wordspace/>. All materials can be downloaded from GitHub: <https://github.com/doomlab/shiny-server/tree/master/scip_wordspace>, which would allow those who were interested in learning *R* code to view and edit the code themselves. We have organized an Open Science Framework page that also contains this information: <https://osf.io/kgn87/>.

**Analyzing Uploaded Corpus**

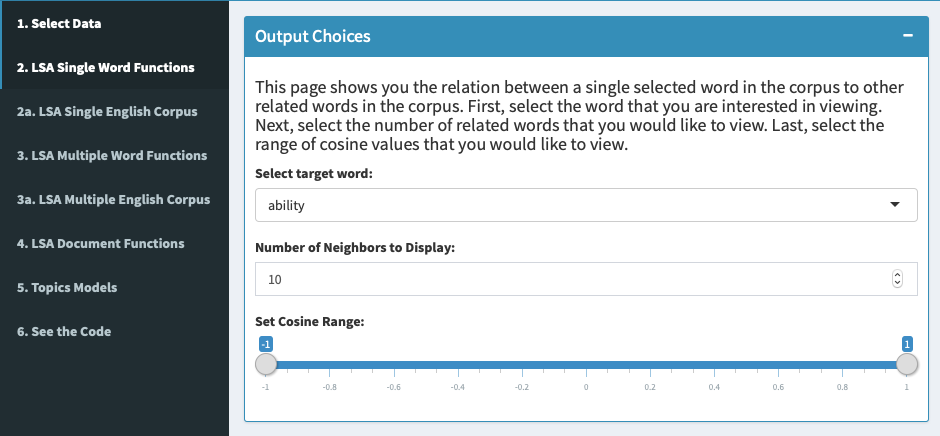


The user accesses the different parts and functions of the app by navigating the tab list on the left. By default, the menu should be displayed upon launch, but the user can toggle its appearance by clicking the hamburger button  next to “Word Space Creator” on the blue bar at the top of the app window. When using the app, users generally should proceed through the tabs displayed in the left side menu in the order listed, starting with “1. Select Data” and continuing through the other tabs in numerical order.

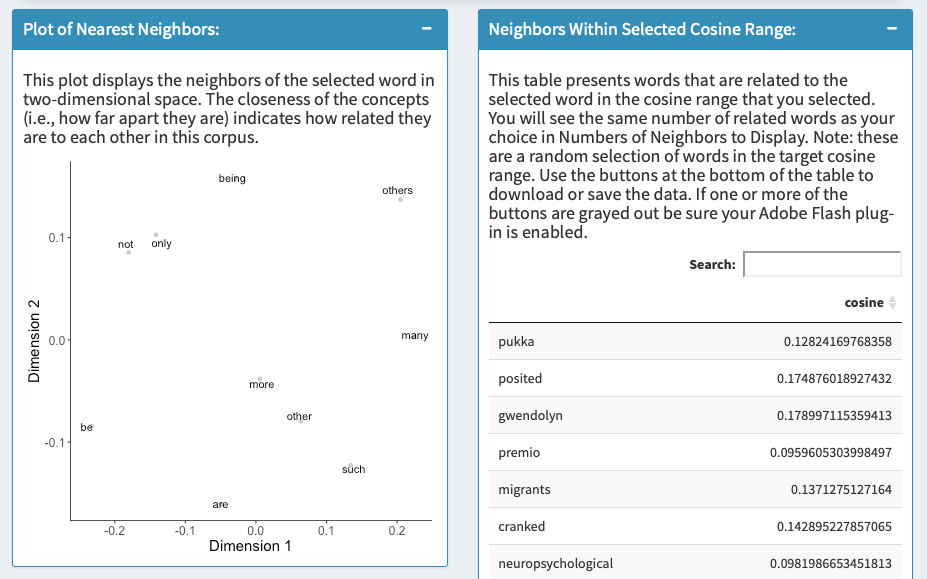
In the “1. Select Data” pane, the user starts by clicking “Browse.” The app will open a dialog box where the user can select a corpus to upload for analysis. The app accepts .csv and .xls(x) files. In order to ensure full compatibility with the app, users should structure their data files such that each individual document in the corpus occupies one cell and that those cells are arranged in one column. By default, the app only analyzes the first column in a given dataset. By clicking the checkbox underneath the “Browse” button, the app will treat the top row as a header.



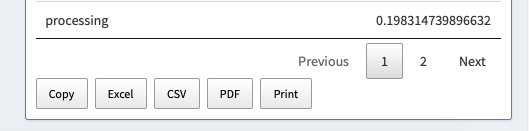
Once the app loads the data, the corpus will be displayed within the “Upload Data” pane. The user can set the number of entries displayed per page using the dropdown menu. Other entries can be accessed at the bottom of the pane. Additionally, the “Search” field allows users to locate specific elements within the corpus. Finally, users should note every box within the app can be collapsed by clicking the minus button at the top right of the box. To expand the box, users can click the plus button. This option allows users to toggle only those boxes they wish to view at a given time.



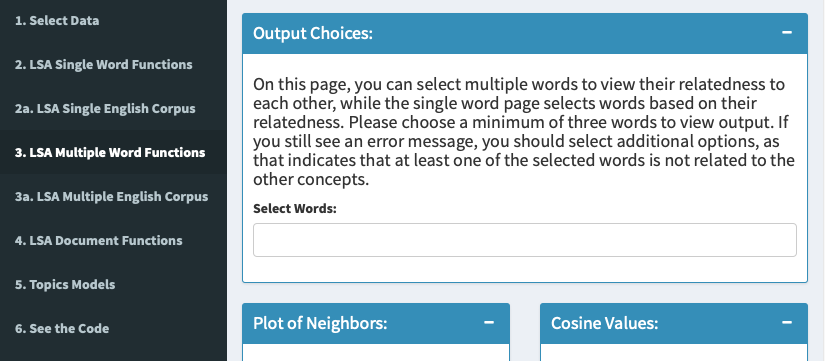
After uploading a corpus, the user can move on to the “2. LSA Single Word Functions” pane. This pane allows for the visual examination of elements obtained through a Latent Semantic Analysis of the uploaded corpus. This pane focuses on analyses involving one target word. In the “Output Choices” box, the user can select the target word from a drop down menu including each unique word in the corpus. The number of LSA neighbors displayed can be set using the next menu down. Finally, the user can adjust the range of cosine values in order to set the criterion for which words appear in the output.



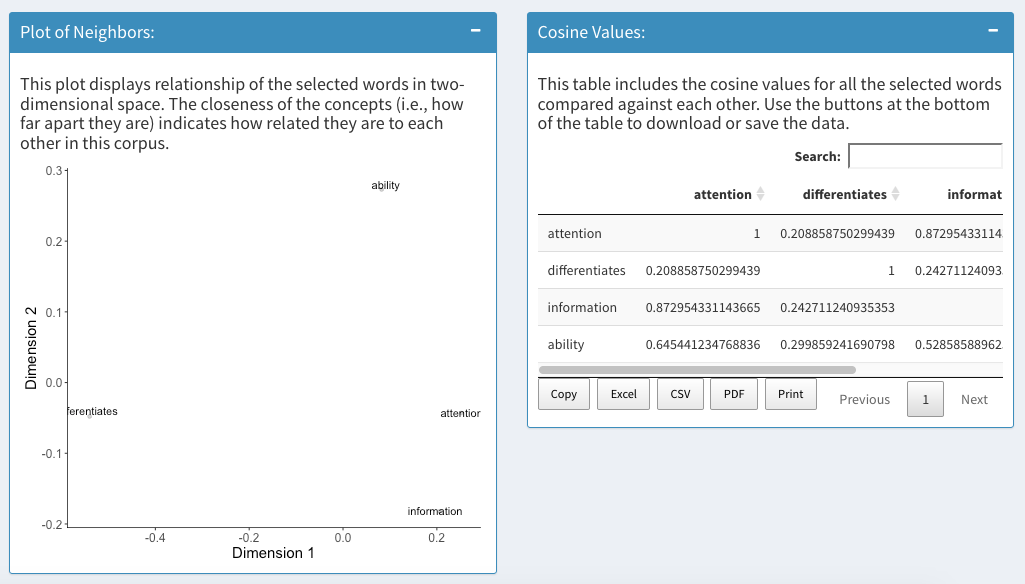
The “Plot of Nearest Neighbors” box displays the nearest neighbors of the selected target word in accordance to the parameters set by the user in the “Output Choices” box. Neighbors are displayed as a function of their semantic relatedness to the target word along Dimensions 1 and 2. The “Neighbors Within Cosine Range” box displays the words from the corpus that are within the cosine range set in the “Output Choices” box. Likewise, the number of words in this output mirrors the number of neighbors the user selected in the “Output Choices” box. Users should note the words are by default displayed at random. The user can display cosines from low to high or high to low by clicking the arrows next to “cosine” at the top of the table. Additionally, users can use the “Search” field to find a specific word and its cosine value.



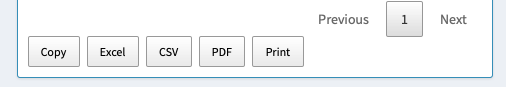
Finally, users can export the output from the “Neighbors Within Selected Cosine Range” box in several formats. Users can copy the contents of the output to the Clipboard. They can also export the output in .xls(x), .csv, and .pdf formats as well as print the output.



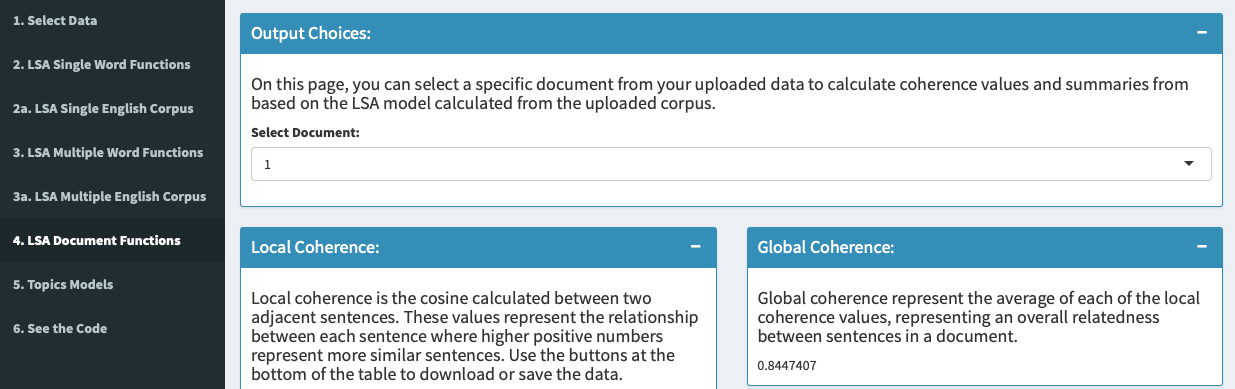
The “3. LSA Multiple Word Functions” pane works much in the same way as “2. LSA Single Word Functions,” except that it accommodates the selection of multiple specific words. In the “Output Choices” box, the user should select at least three words to examine in order to obtain error-free output. The user might still see an error even when three or more words were selected. This simply indicates that less than three words are semantically related. In other words, the app only produces useable output when three or more words with a nonzero relatedness are selected.



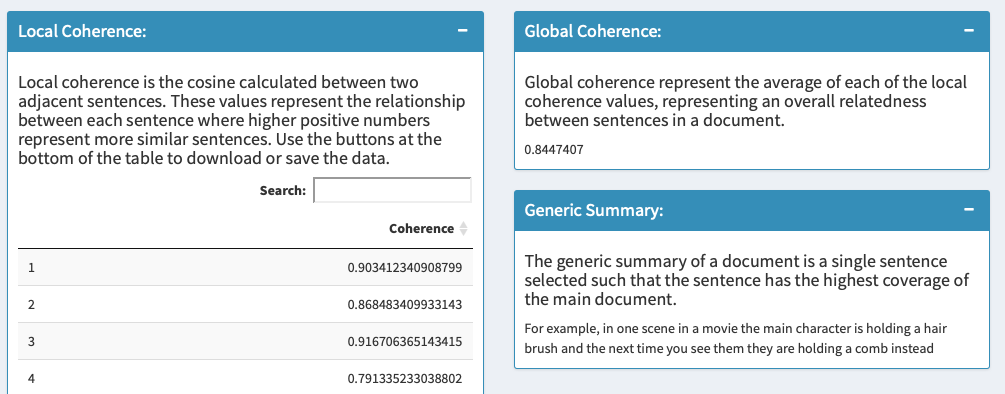
As in the “2. LSA Single Word Functions” pane, the “Plot of Neighbors” box displays the degree of semantic relatedness between the selected words along Dimensions 1 and 2. In this pane, the user does not set a cosine range. Rather, the app produces a graph that scales the axes according to the cosines of the words selected in the “Output Choices” box. The “Cosine Values” box displays each word’s cosine with the others in the selection.



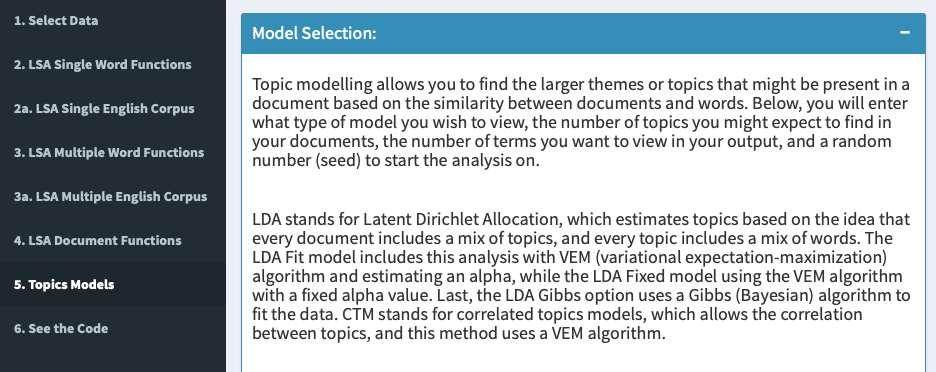
Once again, users can click the buttons at the bottom of the “Output Choices” box to export the displayed data to the Clipboard, to a printer, and several file formats.



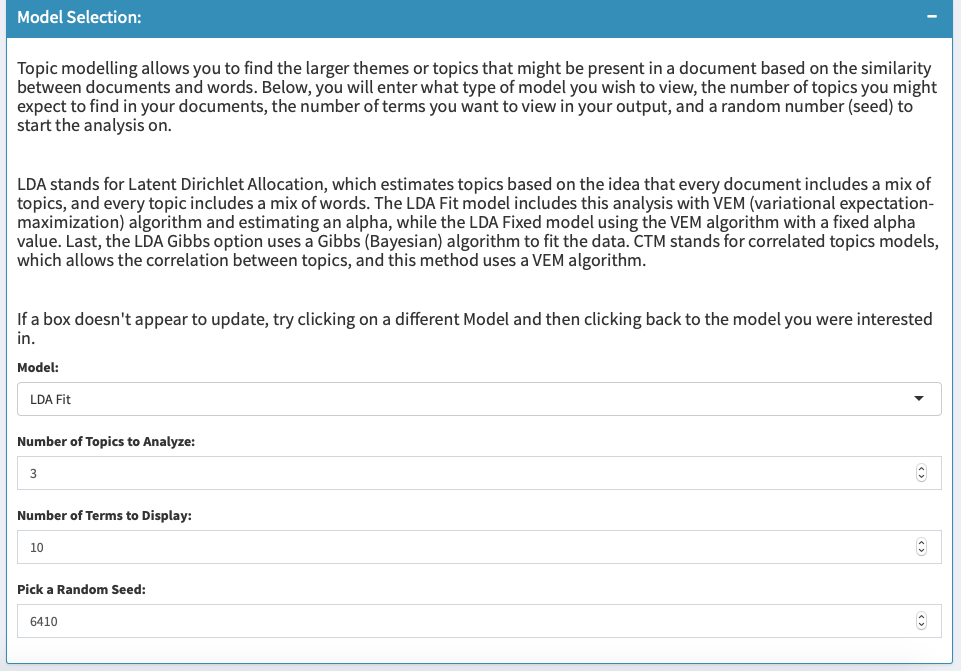
In the “4. LSA Document Functions” pane, the user can examine various elements specific to individual documents in the corpus. Measures are obtained from the LSA performed on the corpus. Using the drop down menu in the “Output Choices” box, the user can select any one of the documents included in the uploaded corpus.

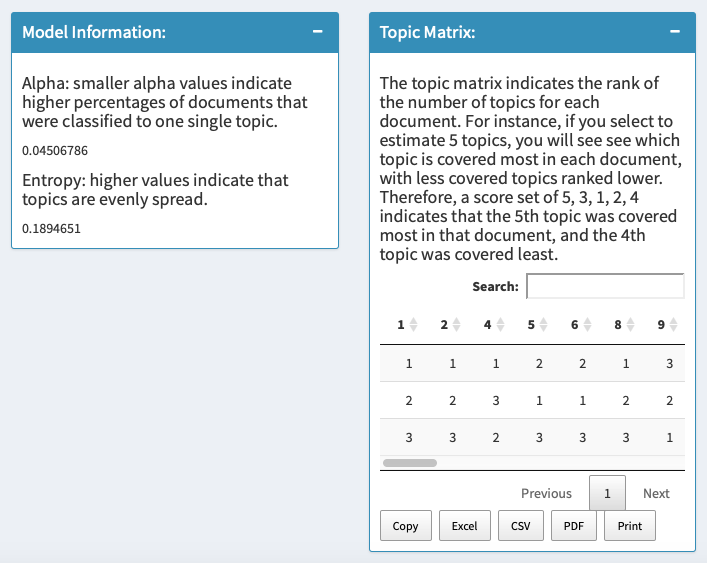


Once a document is selected, the app presents three outputs. The “Local Coherence” box displays the cosine between each sentence in the corpus. This measure attempts to quantify the semantic relatedness between adjacent sentences in the document. Users can export the data in this box in all of the aforementioned formats. The “Global Coherence” box displays a cosine representing the average of all local coherence values. This measure attempts to quantify the document’s relatedness between sentences, and thus the document’s cohesion as a whole. Finally, the “Generic Summary” box extracts and displays the sentence that has the highest relatedness to all other sentences in the document. Ideally, this sentence should be a summary or statement of the document’s main idea.

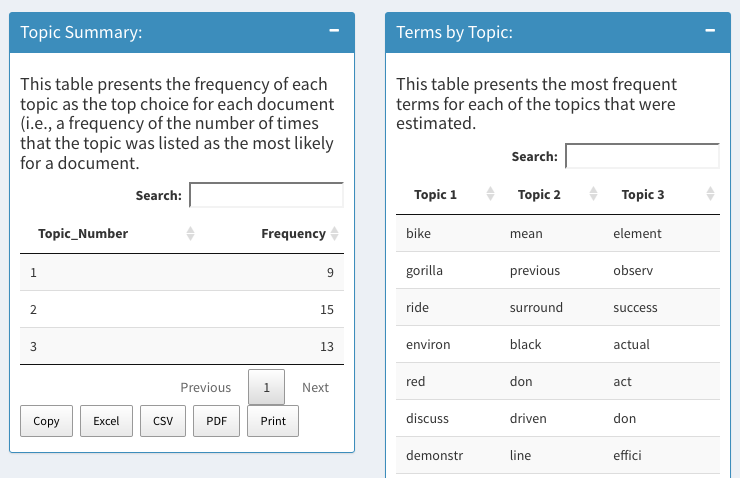


Users can utilize the “5. Topics Models” pane in order to examine themes of documents as approximated by the similarity between words and documents. The “Model Selection” box allows the user to select from four models from a drop down menu, including three Latent Dirichlet Allocation (LDA) methods and one correlated topics model (CTM). The differences in the models are explained in the “Model Selection” box. From the numerical entry fields, the user then selects the number of topics to analyze, number of terms to display, and a random seed value. These values are used to construct a model according to the method specified in the “Model” drop down menu.

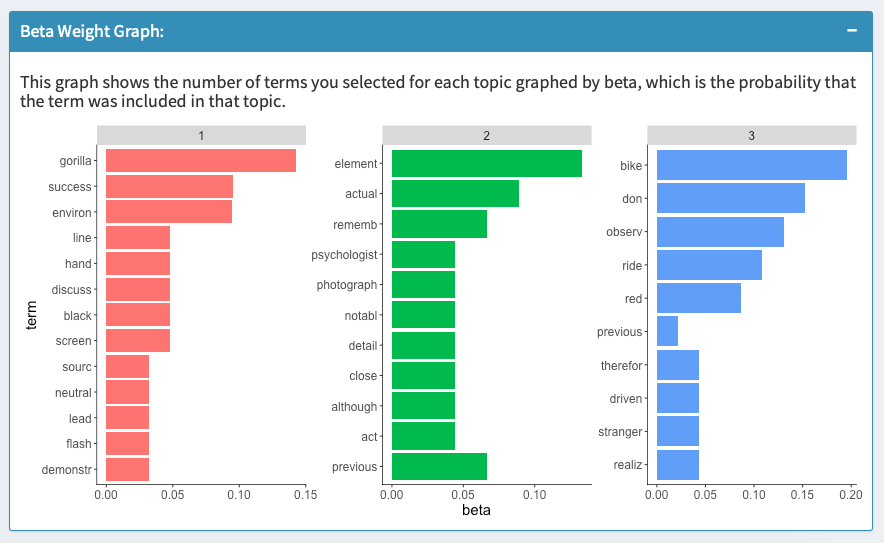




The “Model Information” box displays global information regarding the model, including alpha and entropy values. The “Topic Matrix” box ranks the number of topics per document in order of which topic is covered most in each document. The output from this box can be exported in the formats listed at the bottom.



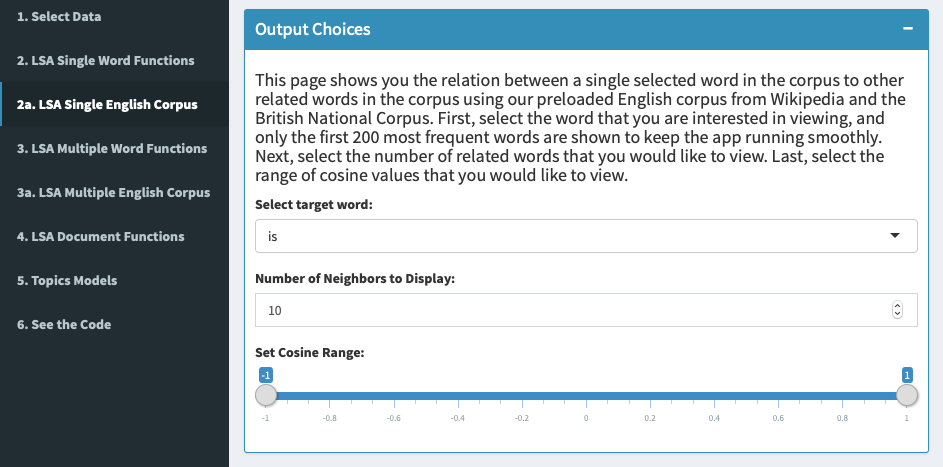
The “Topic Summary” box displays the frequency that each topic ranks as the most likely for a particular document. The output can be exported in the formats listed at the bottom of the box. The “Terms by Topic” box displays the most frequently occurring terms for each estimated topic.



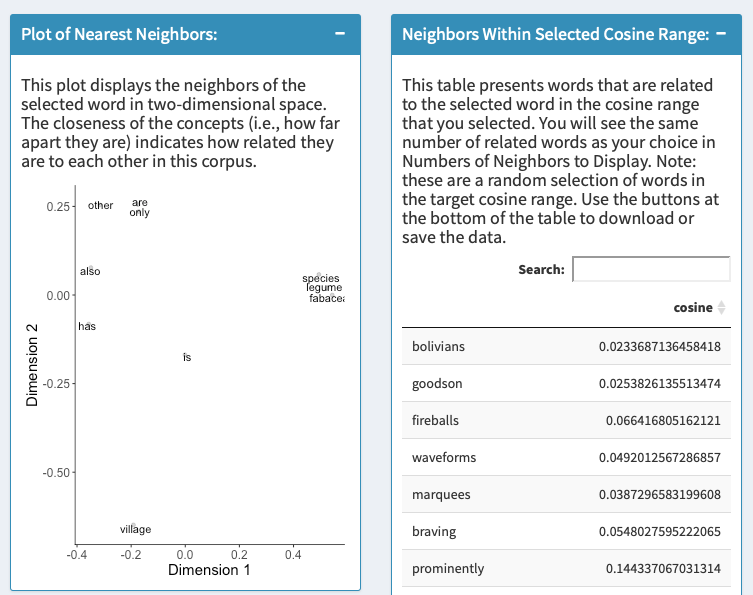
Finally, the “Beta Weight Graph” box displays the number of selected terms for each topic sorted by beta (probability the term was included in the topic).

**Analyzing Included Corpora**

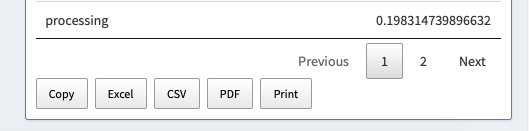
Upon launch, the app preloads a corpus (the English Wikipedia Corpus combined with the British National Corpus) that has been processed with LSA [(Günther, Dudschig, & Kaup, 2014)](https://paperpile.com/c/5HxrxN/wHsF). Users can analyze these corpora in one of two panes: “2a LSA Single English Corpus” and “3a LSA Multiple English Corpus.



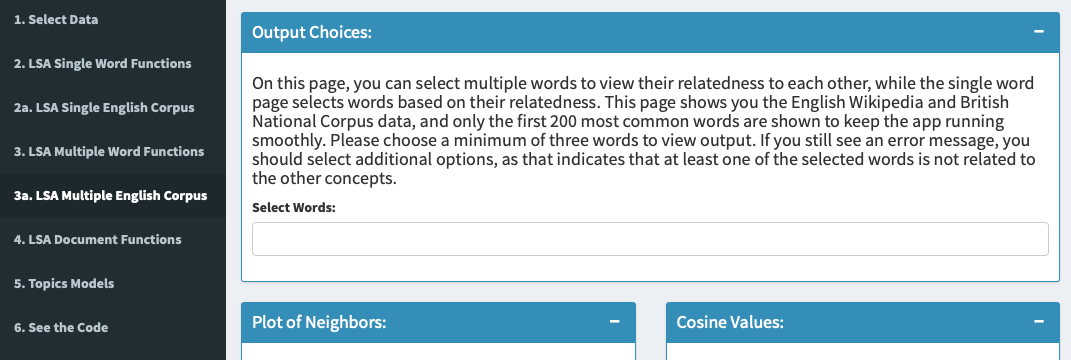
In the “Output Choices” box, the user selects a target word from the drop down menu, the number of neighbors to be displayed in the output, and the range of acceptable cosine values of the displayed neighbors. Users should note only the 200 most frequent words from the corpus are selectable. This restriction is in place so that the app is able to run smoothly without the burden of loading hundreds of thousands of words.



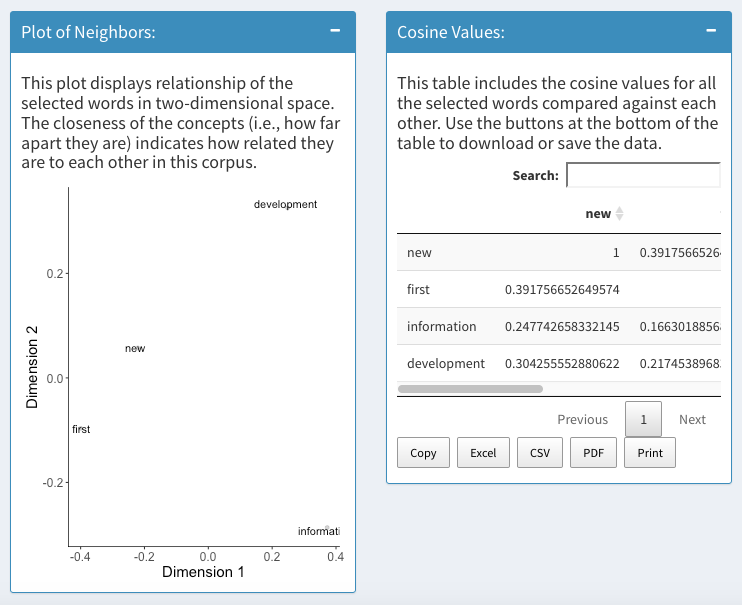
The “Plot of Nearest Neighbors” box displays the selected word’s nearest neighbors arranged by relatedness in a two-dimensional semantic space.



The “Neighbors Within Selected Cosine Range” box displays the cosines of the neighbors for the selected word. The searchable output is displayed in random order by default, but users can arrange cosines from high to low or low to high by clicking the button underneath the search bar. Users can export the data from this box into several formats by clicking the buttons at the bottom.

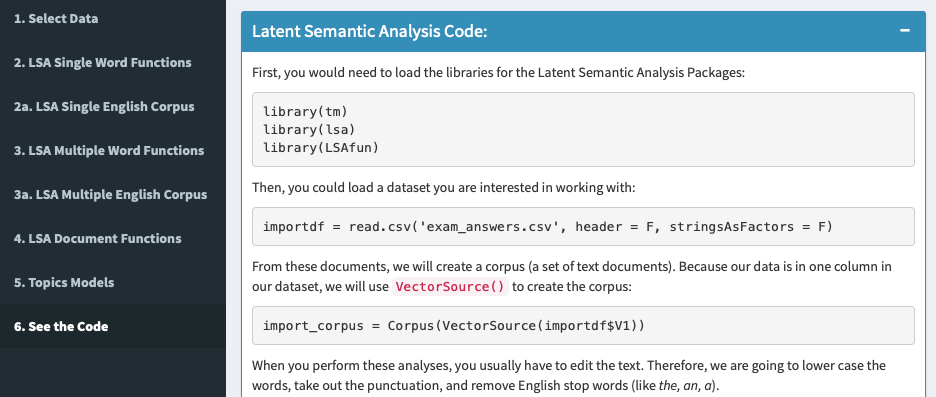


The “3a. LSA Multiple English Corpus” pane allows users to perform LSA on multiple specific words from the two preloaded corpora. In the “Output Choices” box users select at least three of the 200 most frequent words used in the corpora. Sometimes the app displays an error message even when three or more words are selected. This indicates less than three of the words are semantically related. The user should select more words until at least three represent related concepts.



The “Plot of Neighbors” box displays the selected words with their semantic relatedness arranged along Dimensions 1 and 2. The graph scales the axes according to the cosines of the words selected in the “Output Choices” box, as the user does not set a cosine range. The “Cosine Values” box displays each word’s cosine with the others from the selection. The output in this box can be exported in several formats by clicking the buttons at the bottom of the box.

**See the Code**

****

The “6. See the Code” pane displays the *R* code running in the background. The code can be used for running LSA and topics models outside of the Word Space Creator app environment. The code is accompanied by a step-by-step guide to running most analyses for both word space analytic techniques. This pane is not reactive and displays the same information no matter what corpus (or lack thereof) has been uploaded.

**Conclusion**

In this project note, we have detailed an interactive app for creating word space models, along with an integrated teaching component through the tutorial. We believe that this app will be useful for courses that teach these models and as a stepping stone for those who are interested in learning the internal mechanisms to coding a semantic space model. Additionally, the implementation of coded models includes a practical application for future research: reproducibility and replication. The social sciences have undergone a “renaissance” and have turned their focus to implementing procedures that improve the methodological rigor of their studies, such as transparency through open materials and data [(Nelson, Simmons, & Simonsohn, 2018)](https://paperpile.com/c/5HxrxN/lhuB). As more researchers learn coding skills, their work will be easily verifiable, along with the choices made during design and analysis. In conclusion, we believe that supporting these skill sets provides an excellent opportunity to not only train future scientists, but also to impact the reliability of our results through open sharing of methods and data analysis.

References

Ben-David Kolikant, Y. (2011). Computer science education as a cultural encounter: A socio- cultural

framework for articulating teaching difficulties. *Instructional Science*, *39*(4), 543–559.

<https://doi.org/10.1007/s11251-010-9140-7>

[Bergamaschi, S., & Po, L. (2015). Comparing LDA and LSA Topic Models for content-based movie recommendation systems. In *Lecture Notes in Business Information Processing* (pp. 247–263). https://doi.org/](http://paperpile.com/b/5HxrxN/jk7c)[10.1007/978-3-319-27030-2\_16](http://dx.doi.org/10.1007/978-3-319-27030-2_16)

[Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77. https://doi.org/](http://paperpile.com/b/5HxrxN/oRNP)[10.1145/2133806.2133826](http://dx.doi.org/10.1145/2133806.2133826)

[de Boer, J. N., Voppel, A. E., Begemann, M. J. H., Schnack, H. G., Wijnen, F., & Sommer, I. E. C. (2018). Clinical use of semantic space models in psychiatry and neurology: A systematic review and meta-analysis. *Neuroscience and Biobehavioral Reviews*, *93*, 85–92. https://doi.org/](http://paperpile.com/b/5HxrxN/y9ml)[10.1016/j.neubiorev.2018.06.008](http://dx.doi.org/10.1016/j.neubiorev.2018.06.008)

[Elvevåg, B., Foltz, P. W., Weinberger, D. R., & Goldberg, T. E. (2007). Quantifying incoherence in speech: an automated methodology and novel application to schizophrenia. *Schizophrenia Research*, *93*(1-3), 304–316. https://doi.org/](http://paperpile.com/b/5HxrxN/ptU5)[10.1016/j.schres.2007.03.001](http://dx.doi.org/10.1016/j.schres.2007.03.001)

[Günther, F., Dudschig, C., & Kaup, B. (2014). LSAfun - An R package for computations based on Latent Semantic Analysis. *Behavior Research Methods*, *47*(4), 930–944. https://doi.org/](http://paperpile.com/b/5HxrxN/wHsF)[10.3758/s13428-014-0529-0](http://dx.doi.org/10.3758/s13428-014-0529-0)

[Hegarty, M., Mayer, R. E., & Monk, C. A. (1995). Comprehension of arithmetic word problems: A comparison of successful and unsuccessful problem solvers. *Journal of Educational Psychology*, *87*(1), 18–32. https://doi.org/](http://paperpile.com/b/5HxrxN/7ROu)[10.1037//0022-0663.87.1.18](http://dx.doi.org/10.1037//0022-0663.87.1.18)

[Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management - CIKM ’13*. https://doi.org/](http://paperpile.com/b/5HxrxN/QOFd)[10.1145/2505515.2505665](http://dx.doi.org/10.1145/2505515.2505665)

[Kakkonen, T., Myller, N., Timonen, J., & Sutinen, E. (2005). Automatic essay grading with probabilistic latent semantic analysis. In *Proceedings of the second workshop on Building Educational Applications Using NLP - EdAppsNLP 05*. https://doi.org/](http://paperpile.com/b/5HxrxN/G69C)[10.3115/1609829.1609835](http://dx.doi.org/10.3115/1609829.1609835)

[Kolikant, Y. B.-D. (2011). Computer science education as a cultural encounter: A socio-cultural framework for articulating teaching difficulties. *Instructional Science*, *39*(4), 543–559. https://doi.org/](http://paperpile.com/b/5HxrxN/e4eU)[10.1007/s11251-010-9140-7](http://dx.doi.org/10.1007/s11251-010-9140-7)

[Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, *104*(2), 211–240. https://doi.org/](http://paperpile.com/b/5HxrxN/9W9e)[10.1037//0033-295x.104.2.211](http://dx.doi.org/10.1037//0033-295x.104.2.211)

[Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, *25*(2-3), 259–284. https://doi.org/](http://paperpile.com/b/5HxrxN/8yuj)[10.1080/01638539809545028](http://dx.doi.org/10.1080/01638539809545028)

[Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/](http://paperpile.com/b/5HxrxN/lf1p)[10.1016/j.chb.2014.09.012](http://dx.doi.org/10.1016/j.chb.2014.09.012)

[Marlin, B. M. (2003). Modeling user rating profiles for collaborative filtering. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)* (pp. 627–634).](http://paperpile.com/b/5HxrxN/jPrr)

[McCracken, M., Wilusz, T., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., … Utting, I. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *Working group reports from ITiCSE on Innovation and Technology in Computer Science Education - ITiCSE-WGR ’01*. https://doi.org/](http://paperpile.com/b/5HxrxN/fBVg)[10.1145/572134.572137](http://dx.doi.org/10.1145/572134.572137)

[Nelson, L. D., Simmons, J., & Simonsohn, U. (2018). Psychology’s Renaissance. *Annual Review of Psychology*, *69*, 511–534. https://doi.org/](http://paperpile.com/b/5HxrxN/lhuB)[10.1146/annurev-psych-122216-011836](http://dx.doi.org/10.1146/annurev-psych-122216-011836)

[Padó, S., & Lapata, M. (2007). Dependency-based construction of semantic space models. *Computational Linguistics*, *33*(2), 161–199. https://doi.org/](http://paperpile.com/b/5HxrxN/lvoP)[10.1162/coli.2007.33.2.161](http://dx.doi.org/10.1162/coli.2007.33.2.161)

[Smith, J. P., III. (1996). Efficacy and teaching mathematics by telling: A challenge for reform. *Journal for Research in Mathematics Education*, *27*(4), 387. https://doi.org/](http://paperpile.com/b/5HxrxN/sLtC)[10.2307/749874](http://dx.doi.org/10.2307/749874)

[Steyvers, M., & Griffiths, T. (2007). Probabilistic topics models. In Danielle S. McNamara, Thomas Landauer, Simon Dennis, Walter Kintsch (Ed.), *Handbook of Latent Semantic Analysis* (pp. 427–448). Mahwah, NJ: Lawrence Erlbaum Associates. https://doi.org/](http://paperpile.com/b/5HxrxN/GV6A)[10.4324/9780203936399.ch21](http://dx.doi.org/10.4324/9780203936399.ch21)

[Steyvers, M., Shiffrin, R. M., & Nelson, D. L. (2005). Word association spaces for predicting semantic similarity effects in episodic memory. In A. Healy (Ed.), *Decade of behavior. Experimental cognitive psychology and its applications* (pp. 237–249). Washington, DC: American Psychological Association. https://doi.org/](http://paperpile.com/b/5HxrxN/u0eW)[10.1037/10895-018](http://dx.doi.org/10.1037/10895-018)

[Suárez-Pellicioni, M., Núñez-Peña, M. I., & Colomé, À. (2016). Math anxiety: A review of its cognitive consequences, psychophysiological correlates, and brain bases. *Cognitive, Affective & Behavioral Neuroscience*, *16*(1), 3–22. https://doi.org/](http://paperpile.com/b/5HxrxN/MziA)[10.3758/s13415-015-0370-7](http://dx.doi.org/10.3758/s13415-015-0370-7)

[Tobias, S. (1993). *Overcoming math anxiety*. W. W. Norton & Company. Retrieved from](http://paperpile.com/b/5HxrxN/cjLr) <https://books.google.com/books/about/Overcoming_Math_Anxiety.html?hl=&id=mgzqpucKF3sC>

[Tsai, C.-Y. (2018). Improving students’ understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior*. https://doi.org/](http://paperpile.com/b/5HxrxN/xOM7)[10.1016/j.chb.2018.11.038](http://dx.doi.org/10.1016/j.chb.2018.11.038)