Teaching Literary Text Analysis and Visualization with R

Tassie Gniady and Eric Wernert

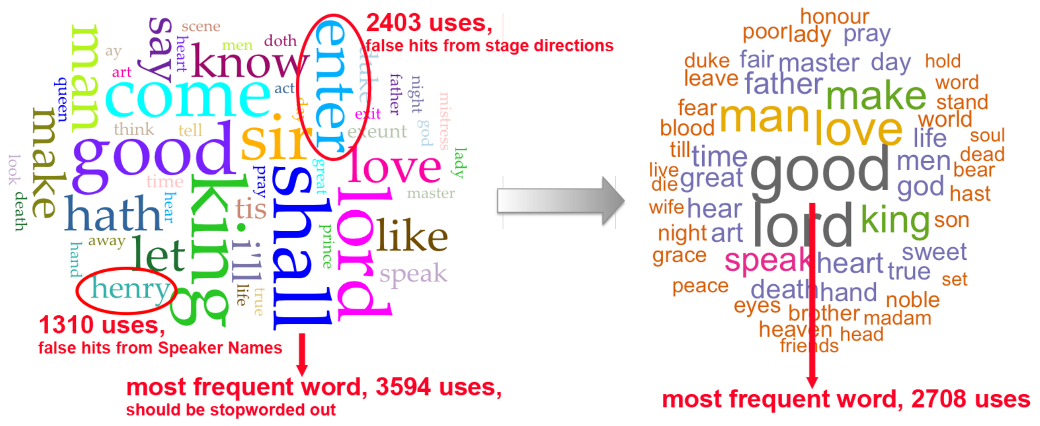

2403 uses,
false hits from stage directions
1310 uses,
false hits from Speaker Names
most frequent word, 3594 uses,
should be stopworded out
most frequent word, 2708 uses

Fig. 1. Comparison of the default result acheived when clicking "Reveal" on the included Shakespearean Corpus in Voyant (left) vs. using an RScript with the Folger Shakespeare Digital Texts "dramatic" extraction (right) wherein speaker names and stage directions have been removed. An early modern stopword list has also been applied in addition to a standard English one.

Abstract—Over the past year, our group has been developing an open instructional workflow for text analysis that aims to build algorithmic understanding and basic coding skills before scaling up analyses. We have chosen to bootstrap in R because of its statistical and graphical capabilities and because of its wealth of domain-specific packages. Moreover, the open source and scripting nature of R allows for methods that are repeatable, extensible, scalable, and sustainable. The aim is to provide code templates that can be adapted, remixed, and scaled to fit a wide range of text analysis tasks. Here we will step through one algorithm from initial engagement via a Shiny web app to a highly annotated RNotebook to a customizable RScript and its output.

Index Terms—digital humanities, text analysis, visualization, r, repeatable workflows

1 Introduction

Text analysis is an important aspect of literary corpus research in the digital humanities. Popular tools like Voyant [10] and AntConc [2] perform significant computational processing for users while hiding most or all of the implementation details. Voyant 2.0 has a wide array of analysis tools, but all of the computation is "black boxed", meaning that the details of the implementation cannot be examined unless one downloads and parses the entire Java code set. The idea behind AntConc is similar, although it runs as a desktop app and is less visualization-oriented than Voyant with dispersion graphs being the only graphical output. We believe that, for scholars and students doing original work in this area, an understanding of fundamentals of the coding behind text analysis is necessary for them to be full participants in the research and to be able to question results adequately. They may also engage in a self-directed interactive dialog with their analysis, rather than following a fixed workflow prescribed by GUI-driven analysis tools.

In fact, Voyant developers Stéfan Sinclair and Geoffrey Rockwell presented a poster at the Digital Humanities 2016 Conference this past July entitled "Voyant Notebooks: Literate Programming, Programming Literacy." On the poster, Sinclair and Rockwell document the next phase of expansion for Voyant: notebooks that will combine text, code, and visualizations. They write that the digital humanities is the perfect place to investigate "the balance between essayist and coder" creating a "natural blend of the expression of intellectual process with the exposition of technical methodologies." They then point to the benefits of explanation for teaching and reproducibility.However, they also note the obstacles of having dynamic notebooks which include: intellectual property rights, potentially malicious code, and browser freezes during data intensive executions [9].

Over the past year, the Cyberinfrastructure for Digital Humanities (CyberDH) Group at Indiana University has been developing an open instructional workflow for text analysis that aims to build algorithmic understanding and basic coding skills before scaling up analyses [4]. We have chosen to bootstrap in R, a high level and high productivity language, with methods that are open, repeatable, and sustainable. The aim is to provide code templates that can be adapted, remixed, and scaled to fit a wide range of text analysis tasks. This position paper presents our approach to teaching computational text analysis and presents a representative hypothetical case study in which two different users are able to start with the same corpus and adapt code to achieve very different end results in a way not currently possible with black box tools.

2 Background

Black box tools with GUIs that hide computation are currently very popular for introducing new practitioners of text analysis in the digital humanities to basic algorithms and outputs. In 2012, AntConc was downloaded 120,000 times by users in 80 different countries [2]. Voy-

- *Tassie Gniady, Indiana University. E-mail: ctgniady@iu.edu.*
- *Eric Wernert, Indiana University. E-mail: ewernert@iu.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.
Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

ant 1.0 had 113 sites linking to it actively in 2012 [10], and the week Voyant 2.0 was released the server went down multiple times from excess traffic [8]. However, of the two default corpora available for exploration on Voyant, one is Shakespeare's plays—with speaker names and stage directions included. This means that all default frequency analysis will be skewed, not a fact that this is immediately evident when simply clicking "Reveal." It was only when noticing the word "henry" in the word cloud (left of Fig. 1) that the authors realized that speaker names and stage directions were a major problem. Because of the Henry plays (*Henry IV Part 1 2, Henry V, Henry VI Parts 1, 2, 3* and *Henry VIII*), Voyant's default view counts every time a King Henry speaks because lines are prefaced "KING HENRY." Thus, any dramatic analysis of frequency from simple word clouds to sophisticated topic modelling will be incorrect. When these false speaker hits are removed, the count of Henry (and its contexts) goes from 1310 uses to 215.

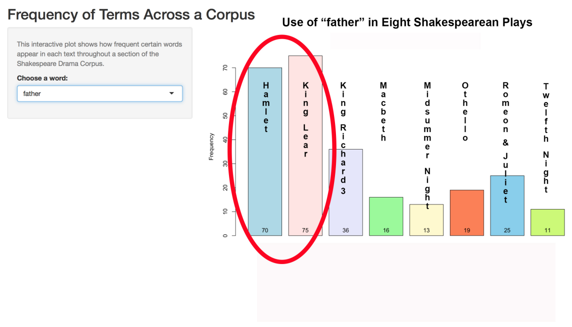
On the other hand, the Visualizing English Print (VEP) project offers "default" and "dramatic" text extractions. "Dramatic" text extractions yield "only the text that is meant to be spoken"–no speaker names, and no stage directions (see the use of "enter" in the word cloud on the left of Fig. 1) [11]. For the researcher who has thought about the implications of speaker names and stage directions (as has VEP member Michael Whitmore, Director of the Folger Shakespeare Library) on text processing algorithms and their visualizations involving frequency such as word clouds, co-occurrence, and dispersion plots, and even topic modeling, the difference in results is dramatic. We downloaded the dramatic files from VEP and placed them into a basic RScript, resulting in the word cloud on the right of Fig. 1 [4]. At this juncture stopword lists were adjusted to the early modern lexicon as words like "hath," "tis," and "doth" need to be excluded. When the authors tried to upload a custom stopword list to Voyant to replicate this workflow, the web page crashed multiple times. However, this is the perfect place to delve deeper into our R workflow and why we think it will yield better results for humanists in the long run.

3 An R Workflow for Humanists

Having looked at one of the most popular "plug-and-play" tools for corpora visualization, it becomes evident that even simple visualizations like wordclouds can lead to inaccurate results if the user is not thinking through how a corpus is being processed. We believe that if the user understands how the algorithm is generating visualizations, they can contribute more meaningfully to critiques of sophisticated algorithms when partnered with programmers or even go on to bootstrap themselves with awareness of their domain's particular caveats (just as one of this paper's author's is an early modern literature scholar who worked through Matthew Jockers's *Text Analysis with R for Students of Literature* [6] to get started). Thus, we advocate teaching humanists the basics of coding. To this end we have a three-step process of introducing R: web-deployed Shiny apps, highly marked up RNotebooks, and lightly commented RScripts, both in "regular" and higher performance versions. All are available for download on Github (with associated data from Shakespeare and Twitter) [4]. What follows is a sample workflow in which two different users begin with the same visualization and follow different paths to achieve different conclusions in a way that would be difficult with a black box tool.

3.1 Shiny Web App

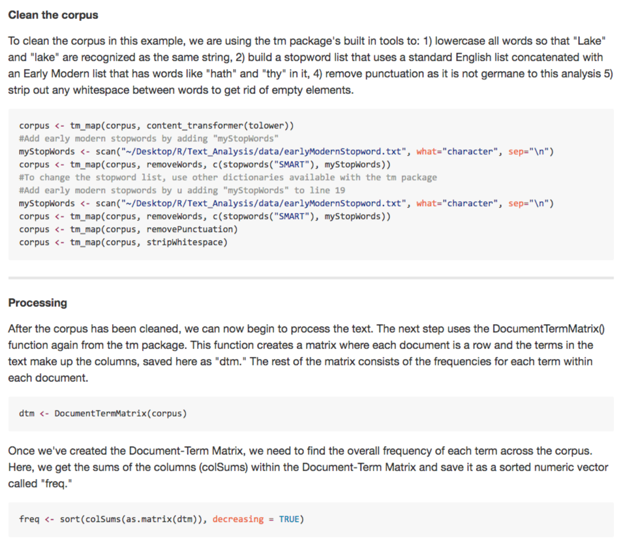
We begin with a comparison of strategically chosen words across eight popular Shakespeare plays. Shiny apps serve the same role as tools like Voyant in that they are meant to introduce users to an algorithm and draw the user in through visualizations that make trends or outliers in a given text or corpora immediately apparent. Voyant would then facilitate the uploading of different text for the re-application of a given visualization technique or the application of another visualization tool on the same text. However, our Shiny apps are not intended to be the actual analysis tool, but rather are meant to interest the user in how an algorithm works, leading them into examining the code that drives the output before facilitating adoption and/or further customization.



This interactive plot shows how frequent certain words appear in each text throughout a section of the Shakespeare Drama Corpus.

Fig. 2. Shiny Web App of Frequent Terms in Eight Popular Shakespearean Plays.

To illustrate, we follow the paths of User A and User B as they work through the same RNotebook that allows them to inspect word frequencies (Fig. 3), but then apply the RScripts based on the notebook in completely different ways. The RNotebooks are heavily annotated and explain both why lines of code are used and how they function–hereby fostering an understanding of how data is analyzed and visualizations are created. This understanding affects humanists' ability to determine if the output is an accurate reflection of the corpus because they are more involved in its production. Even though we are dealing with a simple algorithm–counting the instances of one word in a series of texts–the beginnings of a sophisticated research question can be embarked upon. In Fig. 2 "father" is chosen from the drop down menu as the jumping off point for Users A and B [3]. User A notices that there are 70 instances of the word "father" in Hamlet, only five less than in King Lear, a quintessential play about the role of fatherhood. Thus, she speculates that the spectral father in Hamlet, who dies before the play begins, is almost as present as the physical father (Lear himself) whose entire dilemma revolves around how to best pass on his role as king/father to an entire kingdom. On the other hand, User B is more interested in kinship in general, and he begins to speculate about relationships between mothers, fathers, sons, and daughters in the Shakespearean corpus.



Clean the corpus

To clean the corpus in this example, we are using the tm package's built in tools to: 1) lowercase all words so that "Lake" and "lake" are recognized as the same string, 2) build a stopword list that uses a standard English list concatenated with an Early Modern list that has words like "hath" and "thy" in it, 4) remove punctuation as it is not germane to this analysis 5) strip out any whitespace between words to get rid of empty elements.

```
corpus <- tm_map(corpus, content_transformer(tolower))
#Add early modern stopwords by adding "myStopWords"
myStopWords <- scan("~/Desktop/R/Text_Analysis/data/earlyModernStopword.txt", what="character", sep="\n")
corpus <- tm_map(corpus, removeWords, c(stopwords("SMART"), myStopWords))
#To change the stopword list, use other dictionaries available with the tm package
#Add early modern stopwords by u adding "myStopWords" to line 19
myStopWords <- scan("~/Desktop/R/Text_Analysis/data/earlyModernStopword.txt", what="character", sep="\n")
corpus <- tm_map(corpus, removeWords, c(stopwords("SMART"), myStopWords))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, stripWhitespace)
```

Processing

After the corpus has been cleaned, we can now begin to process the text. The next step uses the DocumentTermMatrix() function again from the tm package. This function creates a matrix where each document is a row and the terms in the text make up the columns, saved here as "dtm." The rest of the matrix consists of the frequencies for each term within each document.

```
dtm <- DocumentTermMatrix(corpus)
```

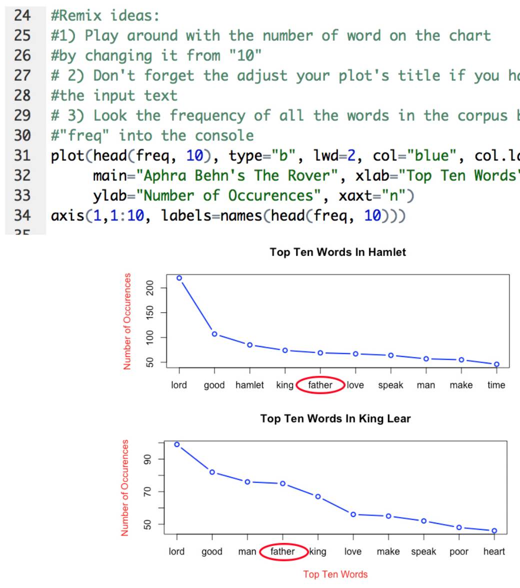
Once we've created the Document-Term Matrix, we need to find the overall frequency of each term across the corpus. Here, we get the sums of the columns (colSums) within the Document-Term Matrix and save it as a sorted numeric vector called "freq."

```
freq <- sort(colSums(as.matrix(dtm)), decreasing = TRUE)
```

Fig. 3. A Portion of an Annotated RNotebook on Word Frequency.

3.1.1 User A

User A proceeds to the RScript and proceeds to change input texts and the number of words displayed so that she now has a sense of how "father" plays out in both Hamlet and King Lear. She sees that "king" and "father" rank right next to each other in terms of frequency in both plays suggesting an entwining of fatherhood and monarchy, as suspected. While she is now looking at a line graph of the frequency of top terms, this is simply a drilled-down look into the frequency bar chart she encountered in the Shiny app. She can now further control the input text, how many terms appear in the chart, or generate a list of the most frequent terms and how often they are used (Fig. 4). In fact, the top ten words in each play bear a striking resemblance to one another. "Good" and "lord" are honorifics–but the use of "love" indicates another way in which fealty is discussed, tested, and ultimately broken down in each of these tragedies.



```
#Remix ideas:
#1) Play around with the number of word on the chart
#by changing it from "10"
# 2) Don't forget the adjust your plot's title if you have changed
#the input text
# 3) Look the frequency of all the words in the corpus by typing
#"freq" into the console
plot(head(freq, 10), type="b", lwd=2, col="blue", col.lab="red",
     main="Aphra Behn's The Rover", xlab="Top Ten Words",
     ylab="Number of Occurences", xaxt="n")
axis(1,1:10, labels=names(head(freq, 10)))
```

Fig. 4. A portion of a lightly commented RScript with remix ideas.

3.1.2 User B

After exploring the notebook, User B decides to look at the top terms in the entire Shakespearean dramatic corpus. Because "good" and "lord" are not useful to his analysis, he adds them to the stopword list. Immediately, it becomes apparent that Shakespeare writes a lot about men as "man," "men", "king", and "father" all appear in the top ten words in the corpus. Even looking at the next ten words only adds "master" to the category of male words. Inspecting the frequency list shows that "lady" appears 23rd on the list with 675 usages–about a third as many as "man" at 1931 and 10% of all the male words preceding this usage. A cursory glance at a female playwright who wrote about seventy years after Shakespeare, Aphra Behn, known as the first woman to make her living by her pen, reveals a much different story. "Man" is the third most used word, but it is the only male term in the top ten and "woman" is the sixth most used term.

4 Conclusion

While Voyant, AntConc, and the like can make possible similar work to that done here, we think that bootstrapping humanists into the coding side of text analysis is worthwhile as it teaches them about how the computer is parsing and analyzing the text. Even using the R Text Mining package [5] requires the user step through different commands necessary to clean a corpus—making them think about how character strings like "lake" and "Lake" need to be computationally equivalent to render good results, or why a custom stopword list might be needed. Sinclair and Rockwell, building on over a decade of evolving black box tools, presented the next evolution of Voyant just this summer at the international DH conference in Krakow, Poland. They want to build upon the popularity of exploratory blogs that make code and detailed analysis available such as those by Ted Underwood, Benjamin Schmidt, Lisa Rhody, and Scott Weingart [9]. By keeping our workflows simple and relying on users to download our scripts and remix them on their own computers, we have avoided these pitfalls and created the base for our text analysis algorithms from Shiny app to RNotebook to RScript in a little under a year.

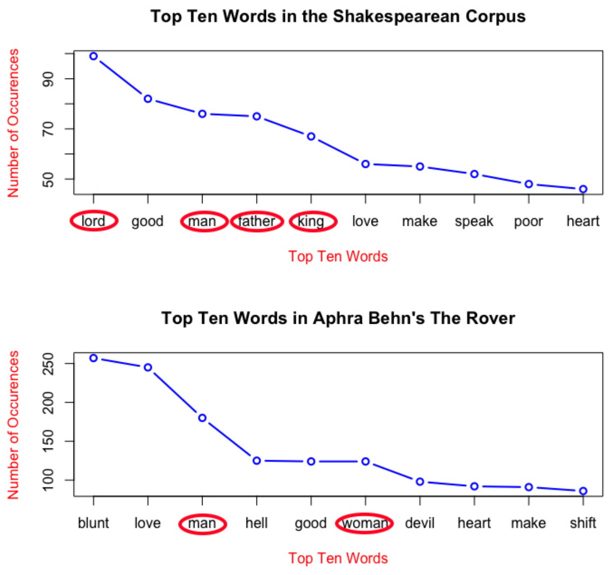


Fig. 5. Comparing terms from two different playwrights.

That said, our development has been driven from the perspective of digital humanists, not as visualization experts. Thus, our visualizations have been relatively simple, even though more sophisticated visualizations are supported in R. We welcome critiques or ideas for enhancement of our visualizations that could lead to more sophisticated outcomes, as long as the code remains relatively straightforward and understandable to digital humanities researchers and students. Likewise, we encourage visualization developers to expose their innovative tools and techniques to languages like R to make them more accessible to the growing community of coding-capable digital humanists.

Acknowledgments

The authors wish to thank Grace Thomas, the CyberDH graduate assistant, who helped to develop the RNotebooks and Shiny apps.

References

[1] L. Anthony. The past, present, and future of software tools in corpus linguistics. *Linguistic Research*, 30(2):141–161, 2013.
[2] L. Anthony. Antconc 3.4.3. Software., 2014.
[3] Cyberinfrastructure for Digital Humanities Team. *Term Frequency Shiny App*, 2016. https://cyberdh.shinyapps.io/TermFreq/.
[4] Cyberinfrastructure for Digital Humanities Team. *Text Analysis Github Repository*, 2016. https://github.com/cyberdh/Text-Analysis.
[5] I. Feinerer. Introduction to the tm package: Text mining in r, July 15 2015.
[6] M. Jockers. *Text Analysis with R for Students of Literature*. Quantitative Methods in the Humanities and Social Sciences. Springer International Publishing, 2014.
[7] S. Sinclair and G. Rockwell. Examples gallery: Examples of voyant tools in research, 2012. http://docs.voyant-tools.org/about/examples-gallery/.
[8] S. Sinclair and G. Rockwell. *Voyant Tools Documentation*, 2012. http://www.sitelinks.info/docs.voyant-tools.org/.
[9] S. Sinclair and G. Rockwell. *Voyant Notebooks: Literate Programming, Programming Literacy*. DH, Krakow, Poland, 2016.
[10] S. Sinclair and G. Rockwell. *Voyant Tools*, 2016. http://voyant-tools.org/.
[11] Visualizing English Print, 2016. http://graphics.cs.wisc.edu/WP/vep/workflow/.
[12] @VoyantTools. Twitter, April 8 2016.