Lang Resources & Evaluation (2019) 53:707–733
https://doi.org/10.1007/s10579-019-09456-6

ORIGINAL PAPER

Beyond lexical frequencies: using R for text analysis in the digital humanities

Taylor Arnold[1] · Nicolas Ballier[2] ·
Paula Lissón[3] · Lauren Tilton[1]

Published online: 8 April 2019

Abstract This paper presents a combination of R packages—user contributed toolkits written in a common core programming language—to facilitate the humanistic investigation of digitised, text-based corpora. Our survey of text analysis packages includes those of our own creation (cleanNLP and fasttextM) as well as packages built by other research groups (stringi, readtext, hyphenatr, quanteda, and hunspell). By operating on generic object types, these packages unite research innovations in corpus linguistics, natural language processing, machine learning, statistics, and digital humanities. We begin by extrapolating on the theoretical benefits of R as an elaborate gluing language for bringing together several areas of expertise and compare it to linguistic concordancers and other tool-based approaches to text analysis in the digital humanities. We then showcase the practical benefits of an ecosystem by illustrating how R packages have been integrated into a digital humanities project. Throughout, the focus is on moving beyond the bag-of-words, lexical frequency model by incorporating linguistically-driven analyses in research.

Keywords Digital humanities · Text mining · R · Text interoperability

✉ Taylor Arnold
tarnold2@richmond.edu

[1] University of Richmond, Virginia, USA

[2] UFR Études Anglophones, Université Paris Diderot, Paris, France

[3] Department of Linguistics, Universität Potsdam, Potsdam, Germany

1 Introduction

Textual data has long been a privileged form of evidence for argumentation within many humanities disciplines. Traditional research methods for historians, literary scholars, theologians, and other humanities scholars include travelling between and exploring within institutional and governmental archives. These archives are filled with newspapers, letters, memos, and other files of printed text. Extensive work by archivists and librarians, along with technological advances and declining hardware costs, have allowed many archives to offer searchable, digital archives directly to researchers on their own computer. The increase in access afforded by digital archives helps to democratise the field, accelerate research, and cross-pollinate ideas across both national and disciplinary boundaries. The digitization of archival materials provide potential benefits to many scholars. However, the fundamental process of building humanistic arguments remains unchanged for most researchers. Evidence in the form of quotations or direct line references to specific passages in primary and secondary textual sources is used to challenge, support, or extend scholarship.

Digital humanities, an emerging set of methods that applies computational techniques to questions in the humanities, offers a fundamentally different way of engaging with digitised documents (Berry 2011). Due to the large scale of available materials, traditional humanities methods, most commonly referred to as close reading, can only uncover a small portion of a given archive or corpus at once. The selection process unavoidably privileges a preselected canon of works at the expense of the rest of the corpus or forces humanists to pick evidence from a set of sources. Digital humanists address this concern by looking for patterns over a much larger collection of texts. When combined with traditional methods, this *distant reading* approach has the potential to both challenge old assumptions and uncover new patterns (Underwood 2017). In order to apply distant reading techniques and to share results with other scholars, new digital tools are needed to extend traditional research workflows in the humanities.

Modern linguists also deal with large corpora of digitised texts and have developed software for exploring and extracting information from them. Concordancers, for example, allow linguists to quickly search for complex linguistic constructs and to see where these occur within a given text. Parallel concordancer tools extend this to multiple texts that have been matched together, such as the translation of a text and its original format. Other corpus linguistic tools, such as Iramuteq (Camargo and Justo 2013), TRAMEUR (Fleury and Zimina 2014), TMX (Heiden 2010) and SketchEngine (Kilgarriff et al. 2014), compute statistical averages to describe broader trends over and within a corpus, while maintaining deep links to the text itself. Tools meant for corpus linguistics have often been provided as either stand-alone applications or served through a web-browser. Natural language processing (NLP), which sits at the intersection of linguistics and computer science, has its own set of tools for working with a textual corpus. These generally take the form of compiled software libraries. While the core of most libraries require running scripts in a terminal, many user friendly front-ends also exist for the most popular libraries. NLP software such as OpenNLP (Morton et al. 2005), CoreNLP (Manning et al. 2014), spaCy (Honnibal and Johnson 2015), and NLTK (Bird 2006) tend to focus on the automated estimation of annotations. At a low level this would include tokenisation, part of speech tagging and building dependency trees. Higher level semantic annotations such as sentiment analysis and named entity recognition use these building blocks to perform automatic annotation, which can in turn retrieve information usable for metadata collection (like placenames or authors derived from named entity recognition, see Sect. 6). In this respect, this type of more elaborated annotations gives a cutting-edge as opposed to standard corpus linguistic tools, where named entity recognition is not typically performed. The needs of digital humanists working with text intersect with those of corpus linguistics and natural language processing. Humanists care deeply about interacting with their raw text; indeed, traditional methods in the humanities chiefly work by closely studying short snippets of text. At the same time, distant reading is fundamentally a study of textual metadata. The ability to extract and study metadata is therefore an important feature in a digital humanist's toolbox. Given this, there is a potential to share many tools and methods from linguistics with the burgeoning field of digital humanities. However, aspects of the tools from these two disparate fields need to be joined together in order to offer the desired functionalities.

In this paper, we present a set of interoperable libraries within the R programming environment that allow for the study of large corpora simultaneously at several levels. These packages promote the loading, cleaning, annotating, searching, and visualizing textual data within a single tool. As R is also a general purpose programming language, this additionally has the benefit of allowing for the application of cutting edge statistical models even when they fall outside of text-specific applications. New R libraries such as **rmarkdown** (Allaire et al. 2017), **knitr** (Xie 2014), and **shiny** (Chang et al. 2017) allow digital humanists to share their results with colleagues unfamiliar with digital methods. This is achieved by the automated building of websites that share results over the web without requiring any special tools on the recipient's machine.

R is already an increasingly popular tool in digital humanities and there have been mature text mining tools available for at least the past decade. The novelty of our argument rests on specific benefits of a particular ecosystem of libraries we are calling for digital humanists to integrate into their workflow. Two of these—**cleanNLP** and **fasttextM**—are of our own creation. The others are contributed by other independent research groups. All of the packages are built around a newly established, common text interchange format. In this paper we argue and demonstrate how such a common format drastically changes the power of using R for the study of textual corpora within digital humanities. Also, while most R text libraries pull from the tradition of NLP-based tools, our specific libraries also incorporate features from corpus linguistics, with a specific focus on discourse analysis.

The remainder of the article is organised as follows. We will first give a brief history of tools for working with textual data from both corpus linguistics and digital humanities. We then discuss former and existing strategies for text mining with R. Next, we discuss our own approach to working with humanistic texts by using a specific ecosystem of packages. This is framed as being in conversation with the tidy data approach and our own work in the field. We conclude by showing several case-studies illustrating how our ecosystem can be used to extract meaningful insights from cultural material.

2 A brief history of tools for textual exploration

2.1 Corpus linguistics software

Concordancers are probably the most well-known tools of analysis in corpus linguistics. These pieces of software allow for the search and visualisation of a word and its context, understood as the words appearing before and after that particular word; a query commonly known as ’key words in context’ (KWIC). The first generation of concordancers appeared between the 1960s and the 1970s. Mainly designed to work in English, these tools could process the ASCII characters and performed very basic tasks, such as the so-called KWIC searches. During the 1980s and 1990s, concordancers incorporated a few other functions such as counting the number of words or generating word lists. A third generation of concordancers includes well-known pieces of software, such as WordSmith (Scott 1996) or AntConc (Anthony 2004), that are still widely used. These concordancers can deal with considerable amounts of data, and thanks to the development of the Unicode system, most languages are supported (Anthony 2013). Broadly speaking, third generation concordancers include keyword search, frequency lists and part-of-the-speech (POS) frequency lists, n-grams queries, as well as user-friendly statistical analysis of texts. This newer wave of tools also includes those specifically designed for statistical exploration of corpora, known as lexicometric tools, such as Lexico3 (Lamalle et al. 2003) and Le Trameur (Fleury and Zimina 2014). These tools usually offer similar features as traditional concordancers, plus more advanced statistical treatment of vocabulary.

Another useful tool in corpus linguistics specifically designed to create, filter and manipulate a corpus, is Le Gromoteur (Gerdes 2014). This piece of software is actually a ’web crawler’, a program that can download websites following user’s preferences, converting them in Unicode and storing them into a database within the Gromoteur. Once the corpus has been compiled from different websites or imported by the user in txt, pdf, or html, classical KWIC searches can be performed. Le Gromoteur also includes a tool called Nexico, based on Lexico3, that computes specificity indices and word co-occurrences based on the cumulative hypergeometric distribution. Graphs of these co-occurrences are automatically generated and easily exported.

2.2 Web-based tools

In the classification proposed by McEnery and Hardie (2011), there is also a fourth generation of concordancers corresponding to web-based tools. Although their functions are similar to those implemented in concordancers of the third generation, these web-based tools store information online, and appear as an alternative for the treatment of larger corpora that cannot be stored in personal computers. Examples of fourth generation concordancers are, among others, Wmatrix (Rayson 2009) or the Sketch Engine (Kilgarriff et al. 2014). Wmatrix, for instance, includes CLAWS part-of-speech tagging (CLAWS 7 tagset), as well as a semantic tagger (SEMTAG), allowing for searches in key semantic domains. Wmatrix also contains 11 different statistical tests that assess the strength of the association of these collocations, such as the Mutual Information Cubed, the log-likehood or the Chi-Square test. Similarly, The Sketch Engine (Kilgarriff et al. 2014) also processes parsed corpora, that is, annotated corpora that contains information about syntactic relations between words. Dependency-based corpora are fully-supported. If the corpus has not been previously parsed, Sketch Engine also offers the possibility to define grammatical relationships manually, mostly with regular expressions, creating a 'grammatical relation set'. Then this set would be used to parse the rest of the corpus. It contains 400 corpora in more than 90 languages, that can be used for comparison purposes. The Sketch Engine also provides aligned parallel corpora, useful for the identification of native-like word combinations in translation. Although both Wmatrix and Sketch Engine are powerful and more sophisticated tools than the previous generation of concordancers and they contain in-built corpora, they are not freely available for the public.

We do not make the claim that state-of-the-art technologies should only be limited to commercial solutions. If anything, we suggest that the current frontier separates shallow parsing and deep parsing. Most concordancers can now analyse POS-tagged linguistic data, but the most important breakthrough in NLP has been achieved by parsing syntax with dependency models. Several libraries have been developed implementing dependency parsing (or constituency parsing). For example, the Stanford NLP suite of tools, includes a part-of-speech (POS) tagger, a named entity recogniser (NER), a parser, a coreference resolution system, sentiment analysis, bootstrapped pattern learning, and the open information extraction tools. Sentences can be processed through an online API or using a command line program. CoreNLP (Manning et al. 2014) exist as a java library that can be used as backend for several tools developed by corpus linguists, allowing the various parameter files to be downloaded according to users' needs. Cases in point are CESAX (Komen 2011) or UAM Corpus Tools (O'Donnell 2008)

2.3 Digital humanities tools

While the early history of textual analysis in the humanities closely followed developments in corpus linguistics, today the research questions and corresponding tools for digital humanists frequently diverge from those of linguistics. This has lead to many PC- and web-based tools that are specifically designed to investigate texts in the spirit of Digital Humanities. Many of these tools, however, continue to making the most of state-of-the-art corpus linguistics, namely fully searchable linguistic corpora, such as the Brown corpus.

One of the earliest DH-focused tools for textual analysis is the Text-Analysis Computing Tools (TACT) programming suite (Lancashire et al. 1996), which was in turn built out of the early frameworks proposed by Bradley and Rockwell (1992). Out of this early work a large collection of tools has emerged from the DH landscape. A case in point of a downloadable tool is the #LancsBox: Lancaster University corpus toolbox (Brezina et al. 2015). Voyant tools, which share a history with TACT, provide a corresponding set of web-based DH tools for text analysis (Sinclair et al. 2016). Recent collected volumes focused entirely on a survey of tools for DH text analysis illustrate the size of the current field and breadth of tool-based options available for analysis (Schreibman et al. 2015). Numerous project examples in Digital Humanities signal how these tools have changed the landscape of research. Prominent examples include the work of Jockers (2013), Bécue-Bertaut and Lebart (2018), and Goldstone and Underwood (2014).

The presence of point-and-click tools for large scale textual analysis in both corpus linguistics and Digital Humanities has made it possible for many scholars to quickly access computational methods to analyze large textual corpora. In the remainder of this article, however, we argue that programming based solutions offer a compelling framework for moving beyond the functionality of one-off tools. Most notably, moving from resource constrained web-based tools allows for the application of methods that go beyond the tokenisation and term-frequency based techniques that are currently popular in both corpus linguistics and Digital Humanities. Accessing more complex features of the text, such as part of speech tags and dependency structures, allows for a nuanced view that the bag-of-words model often hides. The limited usage of such methods in Digital Humanities, such as the work of Klaussner et al. (2015), signals the power of such an approach. Our article here explore how programming tools make such an analysis possible.

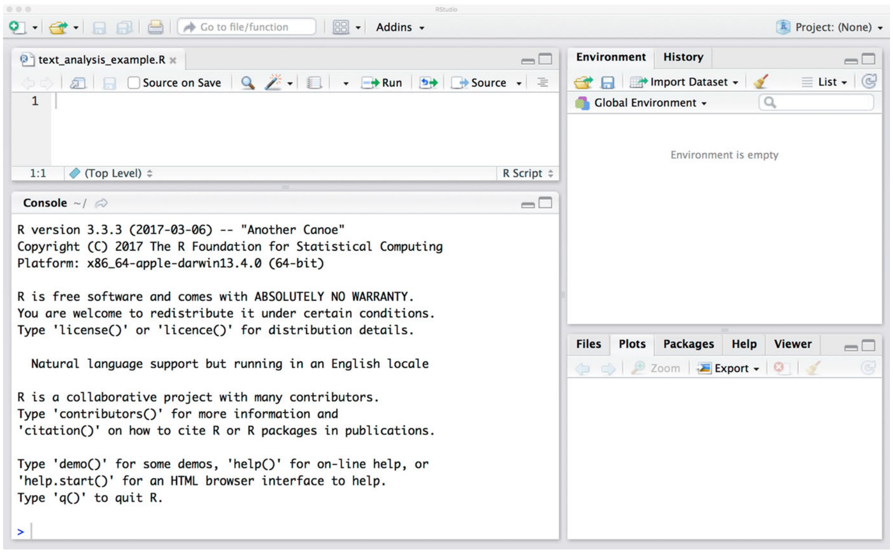
3 R-based tools for textual data

3.1 The R programming language

R is a high-level programming language specifically geared towards the analysis of data (Ihaka and Gentleman 1996). First appearing in 1993, it is an open-source implementation of the S programming language (Becker and Chambers 1984). To run analyses in the language, users write instructions in plain-text code and then enter these into the R interpreter. Output is presented as either plain text, new files, or graphics. While not mandatory, many users of R work within a integrated development environment (IDE)—a standalone application specifically built for writing and running R code. Figure 1 shows an example of the RStudio IDE (RStudio Team 2017). Code is written and saved in one window, run in another, and output is displayed in a third.

There are many benefits to using a programming language in place of stand-alone tools for the analysis of data. By writing procedures in code, we have a complete record of exactly how someone went from raw input to final results.[1] This allows for

[1] The environment can be saved so that the research may still be reproduced even after packages are discontinued and R versions have changed; for details on doing this see Ushey et al. (2016).



```
R version 3.3.3 (2017-03-06) -- "Another Canoe"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Fig. 1 The RStudio environment. Code is written in the upper left-hand corner of the screen and run in the lower left-hand corner. Graphs and current datasets are shown in panels on the right-hand side

the creation of reproducible research that can be validated and understood by other scholars (Peng 2011). Also, using general purpose code allows users, in theory, to conduct any analysis of their data they would like to perform. The choices are not limited to those directly supplied by a particular tool. This is particularly important for scholars working with multimodal data where no particular tool is able to analyse an entire dataset from a plurality of standpoints. Finally, general purpose programming languages benefit from large communities of users. The community around R, for example, has produced hundreds of books and online tutorials to assist new users in learning the language.[2] The active engagement also ensures that the language will be persistently updated and maintained.

While programming languages allow users to built a limitless number of custom analyses and outputs this does not mean that every task must be implemented from scratch. A particular strength of the R programming language is its collection of over eleven thousand packages as of 2017. These packages are open-source, user contributed sets of data and functions for solving particular tasks. Some offer thousands of functions to extend the core language in novel ways with entirely new frameworks. Others provide only a small number of new functions that implement a specific model or feature. Packages are published on the Comprehensive R Archive Network and can be downloaded and installed automatically from within an R IDE such as RStudio (Hornik 2017b). Throughout the remainder of this article we will discuss several packages designed specifically for the analysis of textual data.

[2] A press release from Oracle in 2012, http://www.oracle.com/us/corporate/press/1515738, estimates that at there were at least 2 million users of R. By metrics such as Google searches, downloads, and blog posts, this number has continued to grow over the past 5 years (Hornik et al. 2017).

3.2 Textual frameworks for R

From the earliest releases of the R language there have been basic tools for working with objects containing text. These include functions for reading and writing textual objects to files. There is also basic support for working with regular expressions. Regular expressions in R can be used to filter data, extract particular substrings of an input, or to replace matched substrings with a new value.[3] The primary use case of this functionality was to work with short sections of text from a particular variable in a dataset such as place names, titles, or URLs.

The **NLP** (Hornik 2017a) and **tm** (Feinerer et al. 2008) packages were among the earliest attempts to provide functions specifically built for the analysis of textual corpora. Together they describe a self-contained framework for working with text. Functions such as `VCorpus` create ‘corpus objects’; dozens of other functions provide the ability to plot, print, and save the resulting corpus. A corpus object can be converted into other objects: a `DocumentTermMatrix`, a `TextDocument`, or a `TaggedTextDocument`. Each of these, in turn, has its own custom functions for printing, saving, and other required interactions. New packages have provided additional functionality and frameworks on top of these. The **koRpus** package (Michalke 2017) provides links to **openNLP** to the OpenNLP library (Hornik 2016), and **qdap** to tools for working with transcripts and conducting discourse analysis (Rinker 2013). These packages have been instrumental in allowing users to do meaningful textual analyses with the R language. However, the complex object hierarchies and individual frameworks for each object type become hard to use and integrate into a larger research pipeline. In Sect. 4, we show a modern re-implementation of text analysis tools without the need for the object-oriented structure supplied by **NLP** and **tm**. With several books and custom-built packages, R has become an important tool within the field of linguistics. Harald Baayen has long been advocating the use of R for linguistic analysis (Baayen 2008). Keith Johnson, a phonetician by trade, has also shown the benefits of R for many linguistic domains (Johnson 2008). Stefan Gries, possibly one of the most adamant linguists to push for the use of R in linguistics, has written extensively regarding the use of R for concordance analysis. Examples of Gries’ work include the textbooks *Quantitative Corpus Linguistics with R* (Gries 2009) and *Statistics for Linguistics with R* (Gries 2013). These serve to introduce the R language and to explain the basic principles or corpus analysis. Recent texts by Levshina (2015) further show how to apply modern statistical algorithms in R to the study of corpus linguistics.[4] The benefits of R in corpus linguistics become particularly clear when working with many data sources or while building complex analysis. R largely avoids the issues of stand-alone tools, as Wiedemann and Niekler describe: “Especially when combining several text mining methods in a complex workflow, struggling with data conversion to achieve interoperability between tools undoubtedly creates severe dissatisfaction and even can be a deal breaker for their use at all.” (Wiedemann and

[3] Stefan Gries devised an R script, implementing the function exact.matches(), that literally turns R into a concordancer (Gries 2009).

[4] See (Ballier forthcoming).

Niekler 2017). R-based methods for text analysis from within the linguistic community have recently extended to related fields such as the computational social sciences (Wiedemann 2016) and, as we now discuss, the digital humanities.

3.3 R and the digital humanities

The R language has acquired considerable support within the Digital Humanities. As it has with corpus linguistics, the programming language has begun to compete with field-specific tools such as Gephi (Bastian et al. 2009) and Voyant (Sinclair et al. 2016). R has been taught at various levels during annual workshops such as Humanities Intensive Learning and Teaching (HILT), the Digital Humanities Summer Institute (DHSI), and the European Summer University (ESU). The journal *Programming Historian* has published tutorial articles such as ‘Correspondence Analysis for Historical Research with R’ (Deschamps 2017), ‘Data Wrangling and Management in R’ (Siddiqui 2017), and ‘R Basics with Tabular Data’ (Dewar 2016). These range from teaching the basic principles of the R language to illustrating very particular humanities-oriented applications. Research articles in digital humanities publications may even centre on the creation of new humanities-focused R packages, such as the recent *Digital Humanities Quarterly* article describing the **gender** package (Blevins and Mullen 2015). The popularity of R in digital humanities points to fact that the work of digital humanists is primarily data analysis, the exact task that R was primarily built to support. Digital humanists often work with unstructured data in multiple formats, including spatial data, raw text, and networks. One benefit of using a programming language such as R is that packages exist for dealing with all of these data types within a single environment. This is a central theme in our text *Humanities Data in R: Exploring Networks, Geospatial Data, Images, and Text* (Arnold and Tilton 2015). We illustrate packages for working with these four data types, all of which have rich ecosystems akin to the text packages shown in Sect. 3.2. Having one tool for all of these data types is particularly important when working with a corpus that has different data types intermingled together such as textual captions tagged to image data. Explicit case studies illustrating the flexibility of R to work with multimodal data are shown in Sect. 6.

With the prevalence of textual sources in the humanities, there has been a particular concentration of R-based tools for text analysis in the digital humanities. Matthew Jockers, for example, shows how to use R for text analysis in his book, aptly titled, *Text analysis with R for students of literature* (Jockers 2014). David Mimno wrote an R wrapper to the popular topic modelling software MALLET (Mimno 2013). The R-based **stylo** tool provides a self-contained environment for the study of writing style, with a focus on applications to literature (Eder et al. 2016).

There is clear interest and investment in using R within the digital humanities. Differences exist in the current scholarship regarding whether we should teach R as a general purpose programming language or whether it should be presented as a collection of useful packages. The choice between these two has obvious pedagogical implications (see Ballier and Lissón (2017)). It also influences whether we should focus on building large, all-inclusive frameworks or if R developers should create tools that do one particular task very well. In the following sections we present an ecosystem of packages in R that allows us to capture the benefits of both approaches by creating tools that work *with* rather than *against* the core R programming language. Within this ecosystem users are able to quickly produce interesting analyses, but will be able to iteratively build up to more intricate models as they learn more about the R language itself.

4 An R pipeline for text analysis

Within these guidelines, we specifically chose packages that extend the lexical frequency paradigm—the simple counting of token occurrences from tokenised documents—with cutting edge NLP techniques. This distinguishes our approach most dramatically from other ecosystems of text packages in R.

Our workflow for working with a corpus of textual data involves stitching together an analysis using a set of interoperable R packages. Packages in our toolkit include general purpose libraries for reading (**readr**), cleaning (**tidyr**, **dplyr**), and graphing (**ggplot2**) data. We also use a set of packages specifically built for the study of textual data. The set of *core* packages consists of a particular subset of tools for textual exploration that we have chosen which fulfil the following guidelines:

- store data in tabular forms
- specific to textual data
- where applicable, respect the text interchange format
- limit overlapping duplication of functionality
- sufficiently generic to be applicable to a majority of text analysis problems
- relatively easy to learn

The *text interchange format* is a set of flexible guidelines for how textual data should be structured within R (Arnold and Benoit 2017). These were developed by a committee at the *Text Analysis Package Developers' Workshop*, held at the London School of Economics in April, 2017. There are three distinct formats to capture different levels of analysis:

- *corpus*—a corpus object is described by a table with the full, raw text stored as a single column and a single id of the text stored as another column; other metadata (e.g., author, year, place of publication) may be added as well, with no specific requirements or restrictions on what these may contain
- *term-frequencies*—a term frequency object is stored as a (sparse) matrix, with terms given as columns, documents given as rows, with term counts filled in the individual cells
- *annotations*—an annotation object is a table with one row per object type and a specific column linking back to the corpus table; rows most frequently correspond to tokens, but can be a smaller (ex. syllable) or larger (ex. sentence) linguistic unit

Table 1 R packages forming our core ecosystem for textual analysis

Package name	Functionality	Authorship
stringi	Locale specific string manipulation	Others
readtext	Load text from various file times	Others
cleanNLP	Annotate text; create term frequencies	TA
fasttextM	Apply bilingual word embeddings	TA; NB, PL
hyphenatr	Prescribe hyphenation patterns to text	Others
hunspell	Check spelling and suggest changes	Others

All are provided under an open source license and are available on the Comprehensive R Archive Network (CRAN)

These guidelines were chosen to closely match the way that many R packages were already processing text. Several maintainers of text packages have since adapted their code to fit within these guidelines.

The packages in our core ecosystem are described in Table 1. Included are the low-level tools **stringi** (Gagolewski 2017) and **readtext** (Benoit and Obeng 2017) that assist in loading and cleaning raw textual data. Arnold's **cleanNLP** package, which could be considered the core within our proposed set of packages, takes raw text and returns a set of linguistic annotations (Arnold 2017). Annotations include tokenisation, lemmatisation, part of speech tagging, dependency tagging, named entity recognition, and coreferences. These tasks are available for a number of natural languages and are provided by internally calling one of two cutting-edge NLP libraries: the Java-based CoreNLP or the Python-based spaCy. Our core set of packages is completed by tools that take tokenised text and return additional information or annotations. We can find spelling errors and potential corrections using **hunspell** (Ooms 2017), predict hyphenation patterns with **hyphenatr** (Rudis et al. 2016), and produce multilingual word embeddings with **fasttextM** (Arnold et al. 2017).

How does our proposed pipeline differ from other approaches to textual analysis in R? As described in Sect. 3.2, much of the functionality is also offered in package ecosystems such as **stylo** (Eder et al. 2016) and the suite of R scripts provided by Th Gries and Hilpert (2008). These ecosystems provide a single point of reference that is able to accomplish many common tasks in textual analysis. For research projects that only require the tools within a particular framework—particularly for users with less experience using the R programming language—this can be a time-saving and beneficial approach. However, it is much more difficult to integrate other packages into analysis because the results within **stylo** and the Gries scripts are wrapped in package-specific classes and formats rather than interoperable formats that can be easily passed to other visualization and modeling functions. The flexibility described by our pipeline is particularly important for scholars in DH because textual approaches are often only part of the computational work done with a corpus. Other methods, such spatial and network analysis or the construction of interactive visualizations, are often needed before addressing research questions of interest.

As mentioned throughout Sect. 3, a primary benefit of doing textual analysis in R is the access it gives to other graphical and modelling functionalities developed by a large community of statisticians and data scientists. Our goal in this section is not to give an exhaustive list of models that have been implemented in R. Instead, we focus on a small set of models that have been particularly influential in our own work.

Topic modelling involves the discovery and description of abstract themes within a corpus. A Bayesian model, latent Dirchlet allocation (LDA), is a very popular method for conducting topic modelling. While originally used to describe theme in texts, there is nothing specific in the mathematical model restricting it to textual data. Recent applications have used it in other application domains including topic LDA over images, sound, and space (Wang and Grimson 2008; Lienou et al. 2010). In some fields, such as within the humanities, LDA has such a dominant presence that it is often erroneously described as *the* way to do topic modelling. The **topicmodels** package in R provides a fast implementation of LDA (Grün and Hornik 2011). Many tuning parameters and tweaks of the basic model are included; complementary packages such as **LDAtools** provide nice visualizations of the output (Sievert and Shirley 2015).

A common representation of textual data is a term-frequency matrix. This is a table with one row per document and one column per token. Even with aggressive filtering rules, such a table can easily exceed over 1000 columns. Including all tokens over a large corpus can quickly lead to a term-frequency matrix with several million columns. Within statistics, tables with a large number of columns are described as ‘high-dimensional’. Tables with many columns occur in fields such as genetics, image analysis, and sound studies. One simple problem that arises in the study of high-dimensional data is how to build exploratory plots summarizing the data. Even with clever uses of colour, shape, and facets, typical plots can show at most six variables at once. A solution to this problem is to conduct *dimensionality reduction*, where many variables are synthesised by a small set of algorithmically generated summary statistics. Principal component analysis (PCA) is one common way of doing this, and is provided for small datasets by functions in R without the need for external packages (Wold et al. 1987). The **irlba** package gives access to significantly faster algorithms for PCA over sparse matrices, such as those resulting from term-frequency matrices (Baglama et al. 2017). A more complex dimensionality reduction scheme is given by the **tsne** package (Donaldson 2016). While requiring a longer runtime, the results generally produce more interesting summary statistics compared to PCA.

In text mining, users typically have a metadata variable of interest associated with each document in a corpus. In authorship detection this would be a key describing the writer of each piece and in sentiment analysis it would be a score describing the mood of each document. Typically the goal is to figure out which combination of terms in the corpus are associated with each output. The highdimensionality of the term-frequency matrix once again causes a problem for many traditional predictive modelling algorithms. The elastic net, provided by the **glmnet** package in R, offers a solution to this issue (Simon et al. 2011). As with traditional regression models, it finds a linear model that minimises the sum of squared residuals, but also penalises models for including too many large regression coefficients. The output, when given a high-dimensional model, returns a linear regression where most of the slope coefficients are set to exactly zero. The resulting model is therefore useful from both a predictive standpoint and also as an exploratory technique for indicating which tokens (or any other linguistic elements included in the data matrix) are most associated with a given response.

5 Using the R text analysis pipeline

The description of our R package ecosystem for text analysis has so far been relatively abstract. While this article is not meant as a tutorial on R or the specific packages in our ecosystem, we find it instructive to offer a brief demonstration of a potential workflow in order to best demonstrate the potential benefits to researchers working with textual corpora.[5] In the following code snippets any text following the 'greater than' symbol (>) or plus sign (+) is code meant to be typed or copied into the R interpretor. Output from R is given without such a prefix. Here, we will use a verbose calling scheme to make clear which package each function is being pulled from. The package name is given first followed by the function name, along with two separating colons (`::`) to distinguish each.

The example here shows how to use R to perform information extraction. The specific task is to extract the capital cities of the 28 member nations currently in the European Union. This task requires extending the lexical frequency paradigm to include information about dependencies and named entity recognition. As a corpus to use for this task, we take the English language Wikipedia country pages.[6] While R was used to grab the raw data from the web, for simplicity here we assume that the text files are already downloaded and saved within a directory named `wiki_eu_en_GB`.

The first step in our workflow is to use **readtext** to load a corpus object, stored as a data table, into R.

[5] A full set of code and data for replication can be found at https://github.com/statsmaths/beyond-lexical-frequencies.

[6] In this case, Wikipedia includes an infobox at the top of the page listing the country's capital city. Our analysis ignores this box, using only the raw text to illustrate how information can be extracted from completely unstructured text.

```
> corpus <- readtext::readtext("wiki_eu_en_GB/")
> corpus
readtext object consisting of 28 documents and 0 docvars.
# data.frame [28 x 2]
              doc_id                text
               <chr>               <chr>
1        Austria.txt "\"The origin\"..."
2        Belgium.txt "\"Historical\"..."
3       Bulgaria.txt "\"Organised \"..."
4        Croatia.txt "\"The Croats\"..."
5         Cyprus.txt "\"Cyprus was\"..."
6 Czech_Republic.txt "\"Following \"..."
# ... with 22 more rows
```

Notice that results are given as a data frame along with some additional attributes. Unlike corpus objects in older packages such as **NLP** and **tm**, we can choose to treat this object as a normal data frame. As per the text interchange format specifications, **readtext** has produced a corpus object with one row per document and columns for a document id and another containing the raw text.

Next, we need to preprocess the text before running it through our annotation engine. While HTML tags have already been removed in the download process, Wikipedia-style references still exist. These consist of either the phrase 'citation needed' in square brackets or a number contained in square brackets. We use the **stringi** package to replace both of these with an empty string. The functions from **stringi** that we use all accept vectors (a single column from a table) as inputs and return a new vector of the same length as an output.

```
> stringi::stri_locale_set("en_GB")
> corpus$text <- stringi::stri_replace_all(corpus$text, "",
+                                 regex = "\\[[0-9]+\\]")
> corpus$text <- stringi::stri_replace_all(corpus$text, "",
+                                 fixed = "[citation needed]")
> corpus$lang <- stringi::stri_sub(corpus$doc_id, 9, 13)
> corpus$country <- basename(corpus$doc_id)
> corpus$country <- stringi::stri_sub(corpus$country, 1, -5)
> corpus$country <- stringi::stri_replace_all(
+                               corpus$country, " ", fixed = "_")
```

Here, we manually set the locale to illustrate that **stringi** has the ability to properly parse text in non-Latin scripts. After running these lines of code, the text now contains only the written material itself.

Our next step is to create an annotation object, with one row for every token in the original text. We set up **cleanNLP** to internally use the spaCy library for text parsing.

```
> cleanNLP::init_spaCy()
> anno <- cleanNLP::tif_annotation(corpus)
> anno
# A tibble: 396,160 x 15
       doc_id   sid   tid    word   lemma  upos   pos
        <chr> <int> <int>   <chr>   <chr> <chr> <chr>
 1 Austria.txt     1     1     The     the   DET    DT
 2 Austria.txt     1     2 origins  origin  NOUN   NNS
 3 Austria.txt     1     3      of      of   ADP    IN
 4 Austria.txt     1     4  modern  modern   ADJ    JJ
 5 Austria.txt     1     5       -       - PUNCT  HYPH
 6 Austria.txt     1     6     day     day  NOUN    NN
 7 Austria.txt     1     7 Austria austria PROPN   NNP
 8 Austria.txt     1     8    date    date  NOUN    NN
 9 Austria.txt     1     9    back    back   ADV    RB
10 Austria.txt     1    10      to      to   ADP    IN
# ... with 396,150 more rows, and 8 more variables:
#   cid <int>, source <int>, relation <chr>,
#   word_source <chr>, lemma_source <chr>,
#   entity_type <chr>, entity <chr>, spaces <dbl>
```

The output now has split the text into tokens and assigned learned annotations to each row. The default view in R hides some columns in order to fit the table in the output, but these can be viewed when running the code interactively in RStudio or selecting a subset of the columns.

The first step in performing information extraction about the capital cities in this dataset is to filter the annotation dataset to include only those sentences containing the lemma “capital”.

```
> anno_capital <- dplyr::filter(anno, lemma == "capital")
> cc <- dplyr::semi_join(anno, anno_capital,
+                        by = c("doc_id", "sid"))
```

With this reduced dataset, the next step is to determine exactly which rows correspond to the name of the capital city. We will do those by making use of both the dependencies and named entity recognition annotations produced by the **cleanNLP** package. The algorithm we use to identify capital cities applies the following rules to identify cities:

- the word must be tagged as a spaCy entity type that is not an organization name
- the word has an apostrophe dependency with the lemma ‘capital’ or ‘city’, or

- the word is a direct object, attribute, or subject of the verb form ‘be’ or ‘make’.

Additionally, we throw away any cities with the same name as the country and, when multiple cities match the same city, prefer the construction ‘capital is [city]’ over ‘the capital [city]’ as the first is more explicit. To encode these rules we create a new table that describes all of the possible combinations of the variables that should be filtered for:

```
> df_match <- dplyr::data_frame(
+              upos = c("PROPN", "PROPN", "PROPN", "PROPN",
+                       "PROPN", "PROPN", "PROPN", "PROPN"),
+              lemma_source = c("be", "be", "be", "make",
+                               "make", "make", "capital",
+                                "city"),
+              relation = c("dobj", "attr", "nsubj", "dobj",
+                           "attr", "nsubj", "appos",
+                           "appos"),
+              rank = 1:8)
```

With this table, we now filter the data `cc` to include only those rows that match the object `df_match`.

```
> cc <- dplyr::inner_join(cc, df_match)
> cc <- dplyr::filter(cc, entity != country)
> cc <- dplyr::filter(cc, lemma != country)
> cc <- dplyr::filter(cc, !(entity_type %in% c("NORP", "ORG")))
> cc <- dplyr::filter(cc, word_source != "was")
> cc <- dplyr::arrange(cc, country, rank)
> cc <- dplyr::select(cc, entity, country)
> cc <- cc[!duplicated(cc$country),]
```

Ties are broken using the rank variable in the matching table and duplicates are removed.

The results of our information extraction routing are reasonably accurate, as we can see by printing out the country and city pairs.

```
> sprintf("%s => %s", cc$country, cc$entity)
 [1] "Austria => Vienna"         "Belgium => Brussels"
 [3] "Bulgaria => Preslav"       "Croatia => Zagreb"
 [5] "Cyprus => Nicosia"         "Czech Republic => Prague"
 [7] "Denmark => Copenhagen"     "Estonia => Tallinn"
 [9] "Finland => Helsinki"       "France => Paris"
[11] "Germany => Berlin"         "Greece => Athens"
[13] "Hungary => Budapest"       "Ireland => Dublin"
[15] "Italy => Milan"            "Latvia => Riga"
[17] "Lithuania => Vilnius"      "Malta => Valletta"
[19] "Netherlands => Amsterdam" "Poland => Warsaw"
[21] "Romania => Bucharest"      "Slovakia => Bratislava"
[23] "Slovenia => Ljubljana"     "Spain => Madrid"
[25] "Sweden => Stockholm"       "United Kingdom => London"
```

There are only two false positive. One is Milan (Italy), which is caused because of a sentence stating that Milan is one of the "main fashion capitals". Bulgaria is erroneously linked to the historic capital of Preslav. Two countries are missing from the list: Luxembourg and Portugal. The capital of Luxembourg is Luxembourg City, and therefore gets filtered out when we remove cities with the same name as the country. For Portugal, the only sentence that describes the capital is this one:

> Portugal has been a semi-presidential representative democratic republic since the ratification of the Constitution of 1976, with Lisbon, the nation's largest city, as its capital.

The spaCy parser correctly parses this sentence, but the grammatical relationship between Lisbon and capital are too far up the dependency tree to be found by our relatively simple algorithm.

This simple algorithm implements linguistic queries resting on a very limited set of syntactic constructions. The 'Lisbon' example shows that apposition was underestimated and in our simplified detection algorithm. Our simple example shows the perspectives opened for Digital Humanities by this kind of R-based implementations of queries relying on linguistic knowledge which can be informed with discourse analysis.

6 Case study—photogrammar

Photogrammar is a digital humanities project under the direction of two of the present paper's authors, Taylor Arnold and Lauren Tilton, and Laura Wexler. The project's main object of study is a collection of 170 thousand photographs taken under the direction of the Historic Division of the United States Farm Security Administration and then Office of War Information between 1935 and 1945 (Levine

1988). The archive of study is often referred to as the FSA photographic collection, in reference to the two organizations that housed the Historic Division. There are two distinct aspects to the *Photogrammar* project. A public-facing website, photogrammar.org, presents interactive visualisations and faceted search mechanisms for the the photographic corpus. It has received over half a million visitors in the past 3 years and has been featured in publications such as *LeMonde*, *Slate*, and *BBC News*. The project also engages in an active research agenda using computational tools to make novel insights in the collection.

While the FSA–OWI archive is first and foremost about visual material, there is also a substantial amount of textual data. In this section, we present three examples where we have used our core R ecosystem for textual analysis to study the FSA–OWI archive. These examples are particularly illustrative of why digital humanists are interested in both the raw textual sources as well as the associated textual metadata. In the interest of brevity only short code snippets are included to describe the main aspects of each application. The full datasets and replication scripts are available as supplemental materials.

In the 1960's, the Archives of American Art conducted oral interviews with many of the staff photographers that had worked for the Photographic Division. We have acquired audio files and text transcripts of these and integrated them into the public-facing *Photogrammar* website. The material from the interviews has also influenced our research into the collection. While nowhere as massive in scale as the photographs themselves or the Federal Writer's project documents, the collection of interviews spans over 12 hours of audio material. Our goal was to create ways of better navigating, close reading, and understanding the interviews. This required a deeper analysis of the corpus than available through lexical frequency counts.

We used topic models in order to understand the main themes within and across each interview. As a preprocessing step, we split the interviews up into individual segments. Each segment includes 7 questions by the interviewer and 7 response by each photographer. This was done in R through manipulation of the raw transcripts using the **stringi** package. We were able to split apart questions and answers using the coded metadata in the input. These segments were then annotated using **cleanNLP** and filtered to include only nouns and verbs. With the annotated data, we then created a term frequency matrix and decomposed this using the **irlba** package using the following code:

```
> tf <- cleanNLP::get_tfidf(anno_obj, doc_var = "doc_id",
+                           min_df = 0.01, max_df = 0.3,
+                           type = "tfidf")
> X <- tf$tfidf; vocab <- tf$vocab
> X_svd <- irlba::irlba(X, nv = 8)
> top_words <- apply(X_svd$v, 2, function(v) {
+  id <- order(v, decreasing = TRUE)[1:4]
+  stri_paste(vocab[id], collapse = "; ")
+ })
```

Table 2 Primary topic using SVD topic modelling from interview segments

	AR	BS	DL	JC	JV	MPW	RT	RS	WE
Art; artist; painting; painter	3	6	2	3	0	0	5	7	48
Book; county; month	0	4	3	1	2	3	0	9	15
Camera; approach; assignment	10	5	5	12	2	2	0	12	7
County; problem; place; country	2	6	14	1	2	3	5	12	8
Dust; storm; county; community	3	2	3	1	3	2	0	0	9
File; kind; photography; project	11	20	19	20	9	14	20	44	24
Guy; man; office; job	0	8	4	0	1	5	2	78	16
Storm; dust; story; photography	2	3	5	0	2	2	0	17	11

Segments were defined as occurring after every 8 questions asked by the interviewer. Interviewees are Arthur Rothstein (AR), Ben Shahn (BS), Dorothea Lange (DL), John Collier (JC), John Vachon (JV), Marion Post Wolcott (MPW), Rexford Tugwell (RT), Roy Stryker (RS), and Walker Evans (WE). Topics are described by the top most influential words within each topic

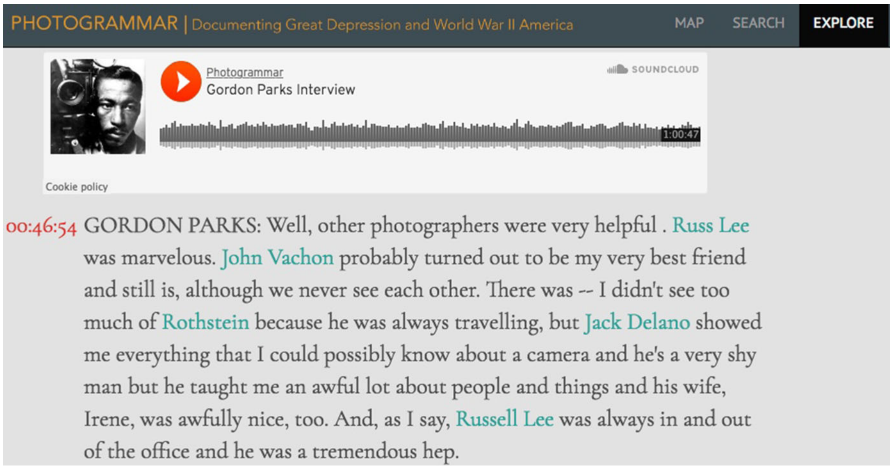
The output of the `top_words` vector describe the most representative words from each of the 8 topics in our model:

```
> top_words
[1] "file; kind; photography; project"
[2] "art; artist; painting; painter"
[3] "guy; man; office; job"
[4] "dust; storm; county; community"
[5] "storm; dust; story; photography"
[6] "book; photography; county; month"
[7] "camera; photography; approach; assignment"
[8] "county; problem; place; country"
```

In Table 2, we see the number of segments from each topic associated with a give interview. By linking together segments from different interviewers categorised by the same dominant topic, we can compare how different photographers felt about similar aspects of their work.

At the macroscopic level, we also see interesting patterns in the overall counts and subject matters of the learned topics. Walker Evans, the photographer most associated with the art world, has a majority of segments focused on the art and painting topic. The one interview with a non-photographer, Roy Stryker, who headed the Historic Division, is dominated by the topic ‘guy; man; office; job’. We also see the prominence and lasting memories for the photographers around the difficulties of covering dust storms.

As a final task for the interviews, we wanted to create automated links from the interviews to guide users to third-party sites explaining concepts that might be unfamiliar to them. During the interviews, the photographers often reference people, places, and events that are likely unfamiliar to many modern readers. The references are also often subtle enough that a simple web search might not be helpful enough



00:46:54 GORDON PARKS: Well, other photographers were very helpful . Russ Lee was marvelous. John Vachon probably turned out to be my very best friend and still is, although we never see each other. There was -- I didn't see too much of Rothstein because he was always travelling, but Jack Delano showed me everything that I could possibly know about a camera and he's a very shy man but he taught me an awful lot about people and things and his wife, Irene, was awfully nice, too. And, as I say, Russell Lee was always in and out of the office and he was a tremendous hep.

Fig. 2 Portion of a transcript of the Archives of American Art interview with the FSA–OWI photographer Gordon Parks. The highlighted text has been hyperlinked to secondary materials to assist readers in understanding the context of the interviews. The time marker on the left was add as well to give a deep link to the specific part of the audio file referenced by the transcript

(e.g., a town name without a state or a relatively common last name without a first name or title). We rectified this issue by passing the entire interview through the **cleanNLP** package and extracting named entities. We filtered entities referencing people, places and events, constructed a look-up table of unique occurrences, and exported this to a .csv file. In a spreadsheet program, we manually went through the entities and linked them to related Wikipedia pages. Finally, we read this sheet back into R and reconstructed the original text with the entities. For example, here is an excerpt of the interview with Gordon Parks coded with related HTML links:

> Well, other photographers were very helpful. < a href='https://en.wikipedia.org/wiki/Russell_Lee_(photographer)'>RussLee</a> was marvelous. < a href='https://en.wikipedia.org/wiki/John_Vachon'> John Vachon</a> probably turned out to be my very best friend and still is, although we never see each other.

Users can now click on the links referencing each photographer and be directed to the appropriate page for more information about them. We were able to easily reconstruct the text from the annotated text and embed links into the output using the **cleanNLP** package in combination with **stringi**. Figure 2 shows how the formatted HTML is rendered the public website.

7 Federal writers project

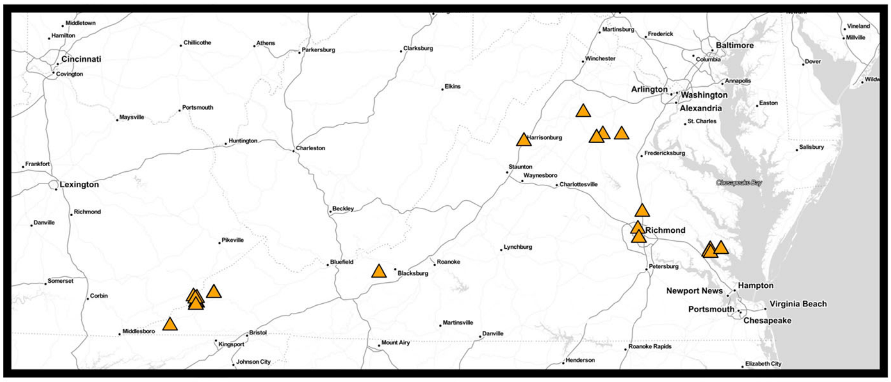
The Federal Writers Project (FWP) was another project of the United States Federal government that ran from 1935 to 1943. Like the Photographic Division, its broad goal was to document and capture the stories of Americans across the country.

Where the FSA–OWI captured this primarily through photography, the FWP documented life stories through written transcripts of oral interviews. Given the overlap in time periods and goals, both our research agenda and the public-facing website has begun to include references to the FWP in addition to the FSA–OWI collection. At the moment, our focus has been on a specific subset of the FWP: the life histories produced by the Southern Life History Program. Here, for simplicity, we will show just the collection from the state of Virginia.

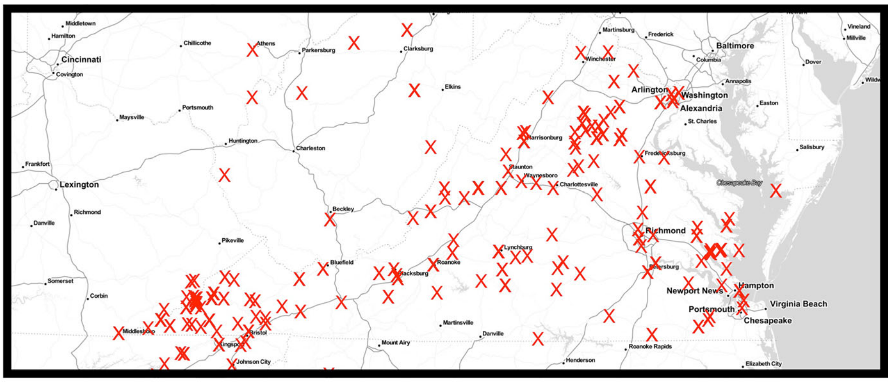
The Federal Writers project conducted 84 oral history interviews in the state of Virginia. Our dataset consists of a short 1–2 sentence description of the interview as well as a digitised version of the interview text. There was no additional structured metadata associated with the interviews. A primary research interest we have about the collection is the presentation and representation of place and space in the interviews. As a first step, we wanted to know where the interviews occurred. Almost all of the summary descriptions list this information and we were able to quickly grab all 84 locations by looking for locations mentioned in the description of each text. With only a few exceptions, which were dealt with manually, this yielded a clean textual description of locations where interviews were given. Knowing that all interviews were given in Virginia and because almost every listing gave both a city and county name, looking up the few unique locations and adding the longitude and latitude to the data was straightforward.

Next, we wanted to find all locations referenced by the interviewee in the text. This task was much more complex than grabbing locations from the descriptions. In this case, there was far too much text to fall-back on manual methods when automated approaches failed. Also, we did not have a consistent description of location with city and county information nor did we know that all locations should be in Virginia. As with the photogrammar example, the task requires more than counting lexical frequencies with the text. To address these issues we built a workflow around our core text packages and one additional R package specifically created for working with geospatial data. The first step involved running all of the text through **cleanNLP** and extracting all named entities referring to locations. After inspecting the list, several common false positives were removed (primarily terms of the era such as ‘Negro’). With the remaining locations, we had R identify and remove locations that were more fully referenced with other locations in a given interview. For example, if an interview referred to the location ‘Wise’ and latter mentioned ‘Wise County’, we included only the more specific reference. Those most complete references made up our final list of locations.

The textual references to locations, even after removing shortened forms, rarely gave a complete, unambiguous place name. There is, for example, a Wise County in both Virginia and Texas. Our solution to this was to use a geolocation service to find the most likely coordinates referenced by all locations mentioned in the texts in three different ways: in the raw form given in the text, with the state ‘Virginia’ added on to the string, and with the country ‘USA’ added on to the string. The geolocation was done within R using the **ggmap** package (Kahle and Wickham 2013). For 72% of locations, the same result showed up in all three searches and this



(a) "Locations where Federal Writers Project interviews occurred in the state of Virginia."



(b) "Locations mentioned in Federal Writers Projects interviews conducted in Virginia (and additional 180 locations are truncated due to the plot boundaries)."

Fig. 3 Locations associated with Federal Writers Project interviews conducted in the state of Virginia, as learned from text extraction from the interview texts

was considered an accurate estimate of what a text was referencing.[7] When there was disagreement, we used the Virginia result if one existed and the USA result otherwise. Ten interviews were manually checked using this logic. We found no falsely determined or located places, but did uncover 6 locations that were missed by the named entity recognition algorithm and 2 that were not found by the geolocation service.

In Fig. 3, we show both the locations where interviews in Virginia occurred as well as locations that were mentioned in the text. The plots were produced in R using the plotting functions in the **ggmap** package. In addition, nearly 200 points referencing other locations across the United States are excluded in the zoomed in version shown here. From the plot we see that restricting our understanding of the

[7] Interestingly, this was not always in Virginia. When the original string was, for example, 'Dallas, Texas' all three locations pointed to the Dallas in Texas regardless of what was tacked onto the end.

FWP in Virginia to only those locations where interviews were conducted is a misrepresentation of the cultural coverage of those interviews. While interviews were mostly given in dense areas of concentration—such as Richmond, Norfolk, and Harrisonburg—many of the oral interviews discuss rural locations such as the Blue Ridge Mountains in the western half of the state and the large farmland region along the border with North Carolina.

8 Challenges and conclusions

We have presented a potential ecosystem of R packages for text analysis and demonstrated how it is able to perform meaningful analyses of humanities data. There are, however, still some challenges and questions that remain to be addressed.

Currently available R packages on CRAN lack several functions that would be helpful for the analysis of text in DH. One particular challenge is the lack of full support for most natural languages. Good models exist for English, Spanish, French, Chinese, and a handful of other languages. Outside of these, tokenisation is often the best that is supported by general purpose libraries.[8] This is not a particular problem of a lack in R support but a general issue with all NLP libraries. There is also need for more tools to work with parallel texts, a common feature of corpus linguistics that has not been fully implemented in R. Nevertheless, bilingual copora can be analysed with R, see the R-based analysis of the bilingual corpus INTERSECT (Salkie 1995) in Gries and Wulff (2012). Their regression analysis uses the two subsets of the corpus which are loaded separately in the analysis. Text specific add-ons for the **shiny** package, which allows users to create dynamic websites from within, are also lacking. The ability to share analyses over the web is particularly important in the humanities where many non-DH scholars are not otherwise familiar with how to run code examples. As we mentioned in Sect. 4, our privileged set of packages is not uniquely defined by our criteria for choosing packages. Other collections of packages could provide very similar functions while still maintaining the interoperability described by the text interchange format. For instance, we could replace **cleanNLP** with a combination of **quanteda** (Benoit et al. 2017), **spacyr** (Benoit and Matsuo 2017) and **tidytext**. A tutorial-oriented paper describing an alternative set of related tools is, for example, given in Welbers et al. (2017). At a larger level, most of our arguments for the use of our R ecosystem could also apply to a collection of modules within the Python programming language. Our goal has been to describe a specific example of a text analysis pipeline for the digital humanities rather than weighing the specific technical benefits of our particular approach compared to other alternative choices.

Finally, while clearly there are benefits of working with a programming language over one-off tools, there is also an added up-front cost in learning how to code. The extent to which digital humanists, and humanities scholars in general, should be expected to program is a hotly debated topic (O'Sullivan et al. 2015). Our argument

[8] In the past year the **udpipe** package has done an admirable job of extending lemmatisation and dependency parsing to a larger set of target languages (Wijffels 2018).

here has been to show the benefits of programming in the context of text analysis. It is not our position here to specifically argue for whom these benefits outweigh the need to learn new skills. Point and click tools may be the right choice for many humanities scholars; we only hope to illustrate the kinds of nuanced analysis precluded by this approach. Despite these open questions and challenges, we feel that our ecosystem of R packages provides an excellent starting point for moving beyond the lexical-frequency paradigm of analysis with the digital humanities.[9]

Using these tools, digital humanists can build both straightforward and complex analyses using data sources involving components with raw text. Our suggested ecosystem balances the ability to easily run some common analyses with little effort with the ability to extend the core functionality to arbitrarily complex workflows. This flexibility, in fact, makes our ecosystem an excellent tool for researchers across the computational social sciences. We expect future work extending the work presented here will further enable scholars from many backgrounds to make novel discoveries and arguments from raw, unstructured textual-based corpora.

References

Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., Wickham, H., Atkins, A., Hyndman, R., & Arslan, R. (2017). rmarkdown: Dynamic documents for R. R package version 1.6. https://cran.r-project.org/package=rmarkdown.

Anthony, L. (2004). Antconc: A learner and classroom friendly, multi-platform corpus analysis toolkit. In *Proceedings of IWLeL* (pp. 7–13).

Anthony, L. (2013). A critical look at software tools in corpus linguistics. *Linguistic Research*, *30*(2), 141–161.

Arnold, T., & Benoit, K. (2017). tif: Text interchange format. R package version 0.2. https://github.com/ropensci/tif/.

Arnold, T., Lissón, P., & Ballier, N. (2017). fasttextM: Work with bilingual word embeddings. R package version 0.0.1. https://github.com/statsmaths/fasttextM/.

Arnold, T. (2017). A tidy data model for natural language processing using cleannlp. *The R Journal*, *9*(2), 1–20.

Arnold, T., & Tilton, L. (2015). *Humanities data in R*. New York: Springer.

Baayen, R. H. (2008). *Analyzing linguistic data: A practical introduction to statistics using R*. Cambridge: Cambridge University Press.

Baglama, J., Reichel, L., & Lewis, B. W. (2017). irlba: Fast truncated singular value decomposition and principal components analysis for large dense and sparse matrices. R package version 2.2.1. https://cran.r-project.org/package=irlba.

Ballier, N., & Lissón, P. (2017). R-based strategies for DH in English Linguistics: A case study. In Bockwinkel, P., Declerck, T., Kübler, S., Zinsmeister, H. (eds), *Proceedings of the Workshop on Teaching NLP for Digital Humanities, CEUR Workshop Proceedings*, Berlin, Germany (Vol. 1918, pp. 1–10). http://ceur-ws.org/Vol-1918/ballier.pdf.

Ballier, N. (2016). R, pour un écosystème du traitement des données? L'exemple de la linguistique. In P. Caron (Ed.), *Données, Métadonnées des corpus et catalogage des objets en sciences humaines et sociales*. Rennes: Presses universitaires de Rennes.

Bastian, M., Heymann, S., Jacomy, M., et al. (2009). Gephi: An open source software for exploring and manipulating networks. *International Conference on Web and Social Media*, *8*, 361–362.

Becker, R. A., & Chambers, J. M. (1984). *S: An interactive environment for data analysis and graphics*. Boca Raton: CRC Press.

[9] Mutatis mutandis, this also applies to corpus linguistics where Stefan Gries has advocated more complex modelling of L1 to investigate L2 production, promoting what he calls MuPDAR (Multifactorial Prediction and Deviation Analysis Using R, (Gries and Deshors 2014).

Bécue-Bertaut, M., & Lebart, L. (2018). *Analyse textuelle avec R*. Rennes: Presses universitaires de Rennes.

Benoit, K., & Matsuo, A. (2017). spacyr: R Wrapper to the spaCy NLP Library. R package version 0.9.0. https://cran.r-project.org/package=spacyr.

Benoit, K., & Obeng, A. (2017). readtext: Import and handling for plain and formatted text files. R package version 0.50. https://cran.r-project.org/package=readtext.

Benoit, K., Watanabe, K., Nulty, P., Obeng, A., Wang, H., Lauderdale, B., & Lowe, W. (2017). Quanteda: Quantitative analysis of textual data. R package version 0.99.9. https://cran.r-project.org/package=quanteda.

Berry, D. M. (2011). The computational turn: Thinking about the digital humanities. *Culture Machine*, *12*, 1–22.

Bird, S. (2006). NLTK: The natural language toolkit. In *Proceedings of the COLING/ACL on interactive presentation sessions, Association for Computational Linguistics* (pp. 69–72).

Blevins, C., & Mullen, L. (2015). Jane, John ... Leslie? A historical method for algorithmic gender prediction. *Digital Humanities Quarterly* 9(3).

Bradley, J., & Rockwell, G. (1992). Towards new research tools in computer-assisted text analysis. In *Canadian Learned Societies Conference*.

Brezina, V., McEnery, T., & Wattam, S. (2015). Collocations in context: A new perspective on collocation networks. *International Journal of Corpus Linguistics*, *20*(2), 139–173.

Camargo, B. V., & Justo, A. M. (2013). Iramuteq: um software gratuito para análisede dados textuais. *Temas em Psicologia*, *21*(2), 513–518.

Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2017). shiny: Web application framework for R. R package version 1.0.4. https://cran.r-project.org/package=shiny.

Deschamps, R. (2017). Correspondence analysis for historical research with R. The Programming Historian. https://programminghistorian.org/en/lessons/correspondence-analysis-in-R.

Dewar, T. (2016). R basics with tabular data. The Programming Historian. https://programminghistorian.org/en/lessons/r-basicswith-tabular-data.

Donaldson, J. (2016). tsne: T-distributed stochastic neighbor embedding for R (t-SNE). R package version 0.1-3. https://cran.r-project.org/package=tsne.

Eder, M., Rybicki, J., & Kestemont, M. (2016). Stylometry with R: A package for computational text analysis. *R Journal*, *8*(1), 107–121.

Feinerer, I., Hornik, K., & Meyer, D. (2008). Text mining infrastructure in R. *Journal of Statistical Software*, *25*(5), 1–54.

Fleury, S., & Zimina, M. (2014). Trameur: A framework for annotated text corpora exploration. In *COLING* (Demos) (pp. 57–61).

Gagolewski, M. (2017). R package stringi: Character string processing facilities. https://cran.r-project.org/package=stringi.

Gerdes, K. (2014). Corpus collection and analysis for the linguistic layman: The Gromoteur. http://gromoteur.ilpga.fr/.

Goldstone, A., & Underwood, T. (2014). The quiet transformations of literary studies: What thirteen thousand scholars could tell us. *New Literary History*, *45*(3), 359–384.

Gries, S. (2009). *Quantitative corpus linguistics with R: A practical introduction*. London: Routledge.

Gries, S. (2013). *Statistics for linguistics with R: A practical introduction*. Berlin: Walter de Gruyter.

Gries, S. T., & Deshors, S. C. (2014). Using regressions to explore deviations between corpus data and a standard/target: Two suggestions. *Corpora*, *9*(1), 109–136.

Gries, S. T., & Wulff, S. (2012). *Regression analysis in translation studies. Quantitative methods in corpus-based translation studies: A practical guide to descriptive translation research* (pp. 35–52). Amsterdam: Benjamins.

Grün, B., & Hornik, K. (2011). topicmodels: An R package for fitting topic models. *Journal of Statistical Software*, *40*(13), 1–30. https://doi.org/10.18637/jss.v040.i13.

Heiden, S. (2010). The txm platform: Building open-source textual analysis software compatible with the tei encoding scheme. In *24th Pacific Asia conference on language, information and computation, Institute for Digital Enhancement of Cognitive Development*, Waseda University (pp. 389–398).

Honnibal, M., & Johnson, M. (2015). An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 conference on empirical methods in natural language processing, Association for Computational Linguistics, Lisbon, Portugal* (pp. 1373–1378).

Hornik, K. (2016). openNLP: Apache OpenNLP tools interface. R package version 0.2-6. https://cran.r-project.org/package=openNLP.

Hornik, K. (2017a). NLP: Natural language processing infrastructure. R package version 0.1-11. https://cran.r-project.org/package=NLP.

Hornik, K. (2017b). R FAQ. https://cran.r-project.org/doc/FAQ/R-FAQ.html.

Hornik, K., Ligges, U., & Zeileis, A. (2017). Changes on CRAN. *The R Journal*, *9*(1), 505–507.

Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, *5*(3), 299–314.

Jockers, M. L. (2013). *Macroanalysis: Digital methods and literary history*. Champaign: University of Illinois Press.

Jockers, M. L. (2014). *Text analysis with R for students of literature*. New York: Springer.

Johnson, K. (2008). *Quantitative methods in linguistics*. London: Wiley.

Kahle, D., & Wickham, H. (2013). ggmap: Spatial visualization with ggplot2. *The R Journal*, *5*(1), 144–161.

Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., et al. (2014). The sketch engine: Ten years on. *Lexicography*, *1*(1), 7–36.

Klaussner, C., Nerbonne, J., & Çöltekin, Ç. (2015). Finding characteristic features in stylometric analysis. *Digital Scholarship in the Humanities*, *30*(suppl 1), i114–i129.

Komen, E. R. (2011). *Cesax: Coreference editor for syntactically annotated xml corpora. Reference manual Nijmegen*. Nijmegen: Radboud University Nijmegen.

Lamalle, C., Martinez, W., Fleury, S., Salem, A., Fracchiolla, B., Kuncova, A., & Maisondieu, A. (2003). Lexico3–outils de statistique textuelle. manuel d'utilisation. SYLED–CLA2T, Université de la Sorbonne nouvelle–Paris 3:48.

Lancashire, I., Bradley, J., McCarty, W., Stairs, M., & Wooldridge, T. (1996). *Using tact with electronic texts*. New York: MLA.

Levine, L. W. (1988). *Documenting America* (Vol. 2, pp. 1935–1943). Berkeley: University of California Press.

Levshina, N. (2015). *How to do linguistics with R: Data exploration and statistical analysis*. Amsterdam: John Benjamins Publishing Company.

Lienou, M., Maitre, H., & Datcu, M. (2010). Semantic annotation of satellite images using latent dirichlet allocation. *IEEE Geoscience and Remote Sensing Letters*, *7*(1), 28–32.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *ACL (system demonstrations)* (pp. 55–60).

McEnery, T., & Hardie, A. (2011). *Corpus linguistics: Method, theory and practice*. Cambridge: Cambridge University Press.

Michalke, M. (2017). koRpus: An R package for text analysis. (Version 0.10-2). https://cran.rproject.org/package=koRpus.

Mimno, D. (2013). mallet: A wrapper around the Java machine learning tool MALLET. R package version 1.0. https://cran.r-project.org/package=mallet.

Morton, T., Kottmann, J., Baldridge, J., & Bierner, G. (2005). Opennlp: A java-based nlp toolkit. In *EACL*.

O'Donnell, M. (2008). The uam corpustool: Software for corpus annotation and exploration. In *Proceedings of the XXvI congreso de AESLA*, Almeria, Spain (pp. 3–5).

Ooms, J. (2017). hunspell: High-performance Stemmer, Tokenizer, and spell checker for R. R package version 2.6. https://cran.r-project.org/package=hunspell.

O'Sullivan, J., Jakacki, D., & Galvin, M. (2015). Programming in the digital humanities. *Digital Scholarship in the Humanities*, *30*(suppl 1), i142–i147.

Peng, R. D. (2011). Reproducible research in computational science. *Science*, *334*(6060), 1226–1227.

Rayson, P. (2009). Wmatrix: A web-based corpus processing environment. http://ucrel.lancs.ac.uk/wmatrix/.

Rinker, T. W. (2013). *qdap: Quantitative discourse analysis package*. Buffalo, NY: University at Buffalo/SUNY. 2.2.8.

RStudio Team. (2017). *RStudio: Integrated development environment for R*. Boston, MA: RStudio Inc.

Rudis, B., Levien, R., Engelhard, R., Halls, C., Novodvorsky, P., Németh, L., & Buitenhuis, N. (2016). hyphenatr: Tools to Hyphenate Strings Using the 'Hunspell' Hyphenation Library. R package version 0.3.0. https://cran.r-project.org/package=hyphenatr.

Salkie, R. (1995). Intersect: A parallel corpus project at brighton university. *Computers and Texts*, *9*, 4–5.

Schreibman, S., Siemens, R., & Unsworth, J. (2015). *A new companion to digital humanities*. London: Wiley.

Scott, M. (1996). WordSmith tools, Stroud: Lexical analysis software. https://lexically.net/wordsmith/.

Siddiqui, N. (2017). Data wrangling and management in R. The Programming Historian. https://programminghistorian.org/en/lessons/data_wrangling_and_management_in_R.

Sievert, C., & Shirley, K. (2015). LDAtools: Tools to fit a topic model using Latent Dirichlet Allocation (LDA). R package version 0.1. https://cran.r-project.org/package=LDAtools.

Simon, N., Friedman, J., Hastie, T., & Tibshirani, R. (2011). Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, *39*(5), 1–13.

Sinclair, S., Rockwell, G., et al. (2016). Voyant tools. http://voyant-tools.org/. Accessed 4 Sept 2018.

Th Gries, S., & Hilpert, M. (2008). The identification of stages in diachronic data: Variability-based neighbour clustering. *Corpora*, *3*(1), 59–81.

Underwood, T. (2017). A genealogy of distant reading. Digital Humanities Quarterly. http://digitalhumanities.org/dhq/vol/11/2/000317/000317.html.

Ushey, K., McPherson, J., Cheng, J., Atkins, A., & Allaire, J. (2016). packrat: A dependency management system for projects and their R package dependencies. R package version 0.4.8-1. https://cran.r-project.org/package=packrat.

Wang, X., & Grimson, E. (2008). Spatial latent Dirichlet Allocation. In: *Advances in neural information processing systems 20* (pp. 1577–1584). Curran Associates, Inc. http://papers.nips.cc/paper/3278-spatial-latent-dirichlet-allocation.pdf.

Welbers, K., Van Atteveldt, W., & Benoit, K. (2017). Text analysis in R. *Communication Methods and Measures*, *11*(4), 245–265.

Wiedemann, G., & Niekler, A. (2017). Hands-on: A five day text mining course for humanists and social scientists in R. In *Proceedings of the 1st workshop teaching NLP for digital humanities*.

Wiedemann, G. (2016). *Text mining for qualitative data analysis in the social sciences*. New York: Springer.

Wijffels, J. (2018). udpipe: Tokenization, parts of speech tagging, lemmatization and dependency parsing with the 'UDPipe' 'NLP' Toolkit. R package version 0.6.1. https://cran.r-project.org/package=udpipe.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, *2*(1–3), 37–52.

Xie, Y. (2014). knitr: A comprehensive tool for reproducible research in R. In: Stodden, V., Leisch, F., & Peng, R. D. (eds), *Implementing reproducible computational research*. Chapman and Hall/CRC. ISBN: 978-1466561595.