Hands-on: a five day text mining course for humanists and social scientists in R

Gregor Wiedemann **Andreas Niekler**
Natural Language Processing Group
Department of Computer Science
Leipzig University, Germany
gregor.wiedemann@uni-leipzig.de
aniekler@informatik.uni-leipzig.de

Abstract

The article introduces our concept and experiences of teaching text mining in R to humanists and social scientists in a one week course. We teach methods to support the entire analysis workflow, from data import and conversion, basic (linguistic) preprocessing to actual analysis such as key-term extraction, co-occurrence statistics, topic models and text classification. The use of the statistical programming language R for the workshop enables participants to learn about methodical backgrounds at different depths levels, as well as to meet their own specific analysis requirements. Among other things, this is possible by relying on a vast supply of community-developed extensions for (textual) data analysis, visualization, and presentation. In detail, we introduce R Markdown tutorial sheets as a didactic core component of the course, and how they can be extended to alternative teaching formats such as a text book for self-learners.

1 Motivation and background

With the emergence of digital humanities (DH) and computational social science (CSS), large digital text collections have become a primary source of data for empiric analyses. During the last decade, humanists and social scientists often cooperated with computer scientists in interdisciplinary pioneer projects to approach such new large-scale data sets. By opening up qualitative data analyses for "big data",[1] such projects paved the way for acceptance of these new technologies in the traditional disciplines. Text mining, defined as a set of statistical and computer-linguistic methods to (semi-)automatically extract semantic structures from very large amounts of texts, started to become a major innovation in various disciplines from political science and economics to modern history (Lemke and Wiedemann, 2016).

Since more and more humanist scholars became aware of the potential of computer-assisted text analysis, the demand for skills and resources to take text mining into their own hands increased. Gesis, a research infrastructure facility for the social sciences in Germany, reacted to this development by setting up a text mining course in 2014, especially targeted to humanists and social scientists. The course was integrated into the periodical education program of the institution. Participants usually are post-graduate scholars, predominantly working in the (public) science sector. The goal was not only to provide them with a theoretical overview of text mining foundations and potential applications, but to educate them to conduct analyses using their own data. Instantly, this raised the question whether the course should be based on (a variety of) publicly available standalone tools[2] or if it should take the effort to introduce participants to coding. A few years back, in DH this question culminated in the famous debate of 'more hack' versus 'less yack'. Pioneers of the field demanded that protagonists of DH should engage more in actual analysis and educating themselves by getting hands on data (Nowviskie, 2014).

We were guided by the premise that for many humanities scholars willing to engage in DH it is actually no problem to acquire theoretical knowledge. Their major challenge is to get on track with the multiverse of potentially useful technical infrastructures. Additionally, we expected many partici-

[1] The term "big data" is associated with a variety of meanings. In social science and humanities contexts, it is often used as reference to data collections which significantly exceed close reading capabilities of a single researcher.

[2] Commonly used standalone tools are AntConc (Anthony, 2014) or WordSmith (Scott, 2016). Examples for web-based infrastructures for text are Voyant tools (Sinclair and Rockwell, 2012) or the Leipzig Corpus Miner (Niekler et al., 2014).

pants with concrete research projects in mind and, at least in some cases, also already in possession of their datasets. To fulfill their needs and in acknowledgement of 'hack vs. yack', we opted to focus on the coding approach. Relying on standalone tools would have had the disadvantage of limitations in analysis capabilities. To our knowledge, there is no single tool covering all major analysis methods from lexicometric analyses to machine learning. Even more problematic, we expected format issues of getting data in and out of a specific analysis tool. 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.

In contrast, teaching basics of coding in a simple and coherent scripting environment allows scholars to create individual solutions tailored to their data formats and specific analysis requirements. At the same time, it poses the challenge to participants to learn to think in terms of a programming language. But especially in social science, many students and scholars already have had contact with statistical analysis software such as SPSS, STATA or R. Hence, writing syntax to manipulate data objects in a procedural way is something which many of them are familiar with. By choosing a convenient coding environment and a selection of mature libraries for text analysis, coding proved to be a manageable endeavor for the vast majority of the participants (see section 4).

Since 2014, the course was taught five times reaching an audience up to 30 scholars per course, among others political scientists, sociologists, economists, historians and philologists. Over the years, it underwent constant refactoring to adapt to the specific requirements of the target group and to integrate new technologies from the growing R community. In the next section, we describe the structure of the course as well as the technologies we use. The third section introduces the different text mining methods taught. In the fourth section, we discuss our experiences of teaching the course. The fifth section elaborates on our future plans to extend the course for alternative teaching formats.

2 (Infra-)structure

The course is a five day, full-time workshop where students are present in class. Two teachers, ideally one with computer science background and one with social science background, give lectures and guide through tutorial sessions throughout the entire week. The didactic concept relies on three major pillars: 1. eight lectures on text mining and its applications in DH projects (30 % of course time), 2. eight tutorials on writing and discussing text mining scripts in R (50 % of course time), 3. presentation and discussion of user projects (20 % of course time).

Lectures contain theoretical and methodological foundations of text mining (1), example studies from DH contexts (2), data acquisition (3), text preprocessing (4), lexicometric analysis (5), unsupervised machine learning (6), supervised machine learning (7) and integration with conventional text analysis methodologies (8) such as grounded theory (Glaser and Strauss, 2005), discourse analysis (Baker et al., 2008) or frame analysis (Waldherr et al., 2016).

Tutorial sessions are the didactic core of the course. In a preparatory phase to the workshop coordinated through an e-Learning platform (ILIAS Core Team, 2017), the students are requested to setup a programming environment with the statistical programming language R and the IDE R-Studio. Beginners in R are additionally requested to acquire some basic coding skills in the language, in particular with respect to R-specific data types, data selection and manipulation syntax, program blocks and function definitions. Additionally, we provide optional readings to the participants to introduce them to the course topics theoretically. Tutorial sessions are based upon the following technologies:

R (R Core Team, 2016): The R project is an open source programming language primarily intended for statistical analysis. Among scientists of all disciplines interested in digital data, it gained popularity over the last years. A huge community of developers has created libraries (so-called packages) for specific purposes which suit almost every imaginable analysis interest. Moreover, the R base functions as well as extension libraries are documented excellently which makes it easy for learners. For textual data a variety of mature packages exists, which provide functionality for reading file formats, creating corpora, running linguistic preprocessing methods and creating document-term matrices (DTM).[3] Many of the packages are well

[3] DTMs encode counting of unique words (types) in documents of a collection. They are a core concept in statistical text mining. For more information about the vector space

prepared to deal with large data sets of tenths of thousands of documents. By using data objects inherited from the same R base objects many packages are interoperable by default. For instance, a DTM created by one package can be given to a machine learning (ML) algorithm of a second package, or results of a clustering algorithm from that ML package can be visualized by a third package.

R-Studio (RStudio Team, 2015): is a user-friendly integrated development environment (IDE) for R. It provides a code editor with auto-completion, syntax highlighting, simple code checking, spell checking, integration with the *git* version control system, a viewer for the state of variables in the current R session environment, as well as easy access to the R console, plotting and help windows. For beginners as well as for advanced users, these tools facilitate the development of new R code immensely.

Swirl (Kross et al., 2017): is an R package to learn R, in R. We do not use it during the course, but promote it during the preparatory e-Learning phase to those participants who had not much contact with R before. With swirl anybody can provide interactive R tutorials on any topic of interest and share them easily by a single install command. For preparation of the course, we show how to install and run a range of beginner tutorials which introduce to R programming. Depending on the background of coding experience and willingness to learn a new programming language, spending a few hours with these tutorials usually is enough to prepare the course.

Packages for text analysis: For the actual analysis we rely on a selection of mature packages providing most of the functionality of text mining covered in the course. For management of documents as corpus objects and text preprocessing, we use the *tm* package (Feinerer et al., 2008). For data acquisition, we introduce web crawling and scraping with *rvest* (Wickham, 2016) and import of PDF- and Word-documents with *readtext* (Benoit and Obeng, 2017). Advanced preprocessing such as POS-tagging is performed with *openNLP* (Hornik, 2016). For unsupervised clustering the *topicmodels* (Grün and Hornik, 2011) package is used. For supervised classification we rely on *LiblineaR* (Helleputte, 2017).

model of semantics see (Turney and Pantel, 2010).

Packages for visualization: Results from analysis processes are visualized in various ways using some plot functions from the R base package, but also additional libraries such as *wordcloud* (Fellows, 2014), *ggplot2* (Wickham, 2009) for time series and topic model visualizations, and *igraph* (Csardi and Nepusz, 2006) for co-occurrence networks.

knitr (Xie, 2014): is a package which provides convenient methods for reproducible research purposes. With knitr one can create HTML, PDF or WORD documents from R analysis scripts. We use this functionality to prepare the tutorial sheets for the courses. See Fig. 1 for a code example and Fig. 2 for the rendered result. The tutorial sheets contain scripts which run an entire analysis chain from reading data, preprocessing, to the actual analysis and result visualization. For better understanding, program scripts of the entire workflow are divided into separate blocks. The purpose of each block is explained in textual form. Text explanations can be formatted easily using "markdown"-syntax. Further, literature references to main algorithms are given. For this, KnitR is able to automatically render references lists from bibtex files. The final rendered scripts contain nicely formatted code blocks, textual explanations, console outputs of evaluated code blocks, and visualization plots. In printed form, provided to the students as hand-out, they can be read as an analog document entirely comprehensible on its own and as a familiar way to take notes. Additionally provided in digital form (preferred as HTML file), they serve as a starting point to re-produce the analysis in an own R-script.

3 Contents

Participants should learn not only about single text mining applications, but also how to combine several applications to complex analysis workflows. For this, we decided to use the same data source for each single tutorial. By building up the tutorials from simple to more complex applications, students get an idea how different approaches may be combined with each other. Students are writing and running the scripts on their own machines.[4]

[4]Of course, the institution offering the course could also provide computer infrastructure. But, by using their own computers participants are guaranteed to have a working setup in their own hands. Another option would be to provide a central installation of a scripting environment such as R-Studio Server or Jupyter notebook on a dedicated classroom server.

Figure 1: Example code of Rmarkdown (Tutorial 2).

```
38  **How many speeches do we have per president?** This can easily be counted
    with the command `table`, which can be used to create a cross table of
    different values. If we apply it to a column, e.g. *president* of our data
    frame, we get the counts of the unique *president* values.
39
40  ```{r eval=T, echo=T}
41  table(textdata[, "president"])
42  ```
```

Figure 2: Rendered output by knitR (Tutorial 2).

How many speeches do we have per president? This can easily be counted with the command `table`, which can be used to create a cross table of different values. If we apply it to a column, e.g. *president* of our data frame, we get the counts of the unique *president* values.

```
table(textdata[, "president"])
```

```
##
##       Abraham Lincoln         Andrew Jackson         Andrew Johnson
##                     4                      8                      4
##          Barack Obama      Benjamin Harrison        Calvin Coolidge
##                     8                      4                      6
##     Chester A. Arthur        Donald J. Trump   Dwight D. Eisenhower
##                     4                      1                      9
## Franklin D. Roosevelt        Franklin Pierce       George H.W. Bush
##                    12                      4                      4
```

Since R is distributed for various operating systems in pre-compiled binary versions, we only experienced minor problems due to different platforms so far.[5]

3.1 Data and resources

As data set we utilize the "State of the Union" addresses (SOTU) of the 45 presidents of the United States published between 1790 and 2017. The speeches[6] are freely available via Wikisource and come with the president's name and a publication date as metadata. The long-term, diachronic nature of the data is ideal for time series analysis and observation of characteristic changes over time. The corpus consists of 231 documents, containing roughly 28,000 types and 1,400,000 tokens. This is certainly not a very large corpus. But for the practical work during the course it has several advantages. The size is large enough for statistical analysis, but not too large to exclude participants working on their own machines with potentially low computational power. Further, certain preprocessing steps or text mining applications such as creating DTMs or topic modeling do not take too much time during tutorials. This allows teaching of concepts such as model selection and parameter optimization relying on repeated runs of an algorithm with changing parameters.

Additionally to the SOTU addresses, we utilize freely available language resources. Sentence segmentation and POS-tagging is realized with openNLP and publicly available pre-trained models (Morton et al., 2005). As reference corpora for key-term extraction we use data from the Leipzig Corpora Collection (Biemann et al., 2007).

3.2 Tutorials

During the course, we provide printed and digital versions of tutorial sheets and an R project skeleton.[7] The tutorial sheets introduce complete analy-

[5] Windows and Mac users are sometimes encountering encoding issues, since these operating systems do not operate with UTF-8 as default encoding. Then, this problem provides a good opportunity to talk about encoding standards and conversion issues in class.

[6] President Woodrow Wilson started the tradition to deliver SOTU addresses as a speech. Before, they were published as written message.

[7] The project skeleton contains a folder structure with the SOTU data and linguistic resources such as reference corpora

sis scripts for several text mining tasks and explain newly introduced functionality. Users can read through the sheets, type code lines (or copy and paste from the digital version of the sheet). Step by step, they are guided through different stages of the workflow. During half time and at the end of each tutorial session, parts of script are explained by an instructor, performing the analysis as a live demonstration on the projector in the class room. During the tutorial session both instructors help with individual problems of participants.

For fast learners or students with R experience, each tutorial sheet provides optional exercises. For these exercises, only output of result computations are given, but no code how to obtain them.

Contents of the individual tutorial sessions are covering a wide range of text mining techniques popular throughout DH and CSS. We begin with data management and preprocessing tasks, which are essential basics for any computer-assisted text analysis. Lexicometric analyses such as frequency and co-occurrence statistics or key term extraction are introduced as a selection of simple approaches on the word level. In the final three tutorials we introduce advanced preprocessing and text analyses based on machine learning. These methods become increasingly popular among social scientists, since they allow for insights into more complex semantic structures beyond isolated words (Wiedemann, 2013).

1. Data acquisition: We start with two basic questions. Where can I get data from and how can I get it into my analysis environment? For this, we introduce basics of web crawling and scraping using the rvest package. The package allows for downloading and parsing HTML pages from the web and extracting elements via XPATH. In two steps the idea of crawling is built up: First, we show how to extract single document texts and metadata such as publication date from a single page. Second, we demonstrate how to extract links to document URLs from an overview page to prepare the crawling of multiple documents. Additionally to web crawling, this tutorial introduces a procedure to extract text from files already stored locally on the hard disk. For this, we provide a sample set of document files in various formats (PDF, HTML, DOC, and TXT) and show how to read them into an R data object for further text processing.

and pre-trained machine learning models.

2. Text processing: the second tutorial introduces basic text processing with the tm package. Concepts of a corpus, metadata, tokenization and DTMs are explained. Characteristics of distributions in language data (Zipf's law), stop words and low-frequent words are explored both, statistically and visually. Further preprocessing concepts such as lowercase reduction, stop word removal, vocabulary pruning and stemming are introduced step by step throughout the following tutorials.

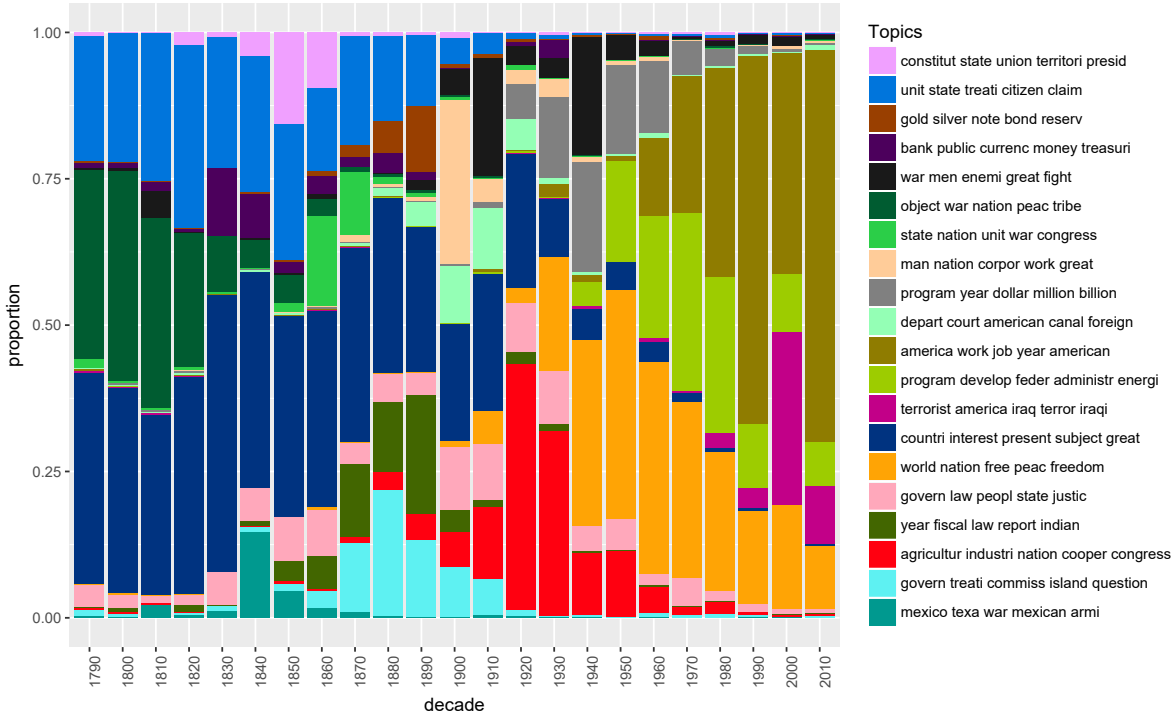
3. Frequency analysis: The goal of this tutorial is to introduce to the potential of word counts for time series or dictionary analysis. The frequency of selected key-terms is observed in the speeches aggregated by decades. As a very basic approach to sentiment analysis, positive/negative terms from SentiWordNet (Esuli and Sebastiani, 2006) are used to count sentiment categories per president. As an alternative visualization for frequencies, we introduce heat maps to trace term usage over time.

4. Key term extraction: The previous tutorials showed that mere frequency is not necessarily a good indicator of the significance of a term. This tutorial introduces TF-IDF and the log-likelihood ratio test (Rayson et al., 2004) in comparison to a reference corpus as ways to identify key terms. First, key terms are extracted from a single document. In a second step, key terms are extracted from all speeches of each president separately and visualized as wordcloud. Visualizations are exported to disk in PDF-format.

5. Co-occurrence analysis: Following the notion of distributional semantics, we extract statistically significant co-occurrences of words from the president's speeches. For this, speeches are first separated into single sentences using the openNLP sentence segmentation. Then, word co-occurrences are measured by different statistics (frequency, Dice, mutual information, log-likelihood). Effects of different statistical measures are compared. Finally, a graph network of co-occurring terms is drawn to visualize the semantic environment of terms.

6. Unsupervised machine learning: Topic models became pretty popular in various fields of the humanities and social sciences (Schmidt, 2012). We apply the initially introduced and most widely used LDA topic model (Blei et al., 2003). Students

Figure 3: Topic shares in SOTU data over time (Tutorial 6).



learn how to compute a model, visualize term and topic distributions for interpretation and tune parameters to influence the model outcome. For further application of model results, we show how to filter the document collection with regard to certain topics and how to track topic proportions over time. As an example, Fig. 3 visualizes topic proportions in SOTU speeches throughout the centuries.

7. Supervised machine learning: Since social scientists often do not look for content patterns inductively, but actually want to observe theoretically derived categories in a deductive setting, supervised machine learning can be very useful. We introduce the use of linear SVM classification for automatic content analysis by categorizing single paragraphs of the speeches as related to domestic or foreign affairs. For this, we provide a manually annotated training data set containing of 300 paragraphs related to either of the two categories. With this, a classification model to infer categories for all 21,300 paragraphs from all speeches is built. The tutorial shows how to optimize parameters of the SVM classification as well as the features used (e.g. stop word removal, stemming, bi-grams).

8. Advanced preprocessing: The last tutorial introduces to Part-of-Speech (POS) tagging and Named Entity (NE) recognition as more complex preprocessing methods. We use openNLP methods and pre-trained models to tag syntactical word types (nouns, verbs, adjectives) and NEs (persons, places, organizations). POS-tags appended to tokens allow for disambiguation between homographs (e.g. *state_NOUN* $\neq$ *state_VERB*), and filtering of word types for subsequent steps such as key term extraction or co-occurrence analysis. As an example, we extract NEs to create a network visualization of associations between persons and places occurring together in speech paragraphs.

4 Teaching experience

The last section made clear that the course is committed to introduce to a broad variety of computational text analysis approaches instead of focusing deeply on a few single methods. Evaluations regularly done by Gesis after the course show that participants extremely welcome the broad variety of the course contents (see Table 1 for evaluation results of the 2016 course). The idea is to provide beginners with a broad overview in a way that enables them to decide which approach fits best to their research problem. Once they can identify what is actually possible and how that may help them for their analysis, they have enough material at hand to get deeper into a single method on their own. To support this endeavor, we introduced user

project discussions into our concept. One day at the end of the week is reserved for participants to introduce their research ideas. Together with the audience, operationalization strategies using appropriate text mining methods are discussed. By this, students learn from illustrative examples how to approach a text mining based analysis, even if they do not present their own project.

More than 95 % of the participants considered the structure of the course as good or very good. Beyond learning achievements of each single tutorial, the concept of investigating a single text resource with multiple methods allows to grasp the potentials for combining single analysis instruments to complex workflows. In such workflows output results of one process could be utilized as input for the next. For instance, document selection via topic models allows for very specific thematic selection of sub corpora. Single words in such sub corpora can then be investigated further with the help of co-occurrence analysis. In this regard, more than 85 % highlighted the successful combination of theoretical and practical knowledge transfer.

The high level of satisfaction is also achieved by constant improvements of the tutorial sheets. Over the years, we reduced complexity of the presented R code to concentrate on only a few new functions to introduce in each unit. This allows learners to successfully get acquainted with the new functions by reading the provided documentation on the tutorials sheets and the integrated R documentation. The tutorial sheets are designed in a way that allows reproduction of the analysis scripts in everyone's own pace. Each tutorial sheet is conceptualized as independent from the others. If one is not able to finish a sheet during the designated session, no frustration is produced for following tutorials. Participants which finish a sheet early can start to figure out the (slightly more complex) optional exercises provided for each session. These exercises are extensions of the analysis workflows which further clarify concrete usage scenarios of the specific text mining applications. The evaluation shows that most participants not only gain theoretical knowledge from the course, but actually feel prepared to engage in their own analysis. While more than two thirds at least partly agree that after finishing the course they are able to apply text mining methods on their own, 100 % of the participants (strongly) agree that the course materials were useful to advance in that direction.

5 Adaptations and future work

Offering the course within the Gesis training program leads to a highly skilled and motivated target audience consisting of scholars mostly at the Ph.D. or post-doc level. For other target audiences, course contents could be reduced or requirement levels could be lowered. Such a variant of the course was taught during a DH summer school in 2015. For this, the format had to be adapted to a half-day course. We left out the parts about machine learning in lectures as well as tutorial sessions. By focusing on lexicometric measures such as frequency and co-occurrence analysis, we were able to easily fit to the time constraints.

Other types of adaptations are realizable in an easy manner, too. Especially in university contexts using R in combination with rendering outputs and textual descriptions via knitr, provides interesting new possibilities for teaching. To modify the workshop to a semester course, alternating sessions of lectures and tutorials can be held in weekly manner. We further suggest excluding the optional exercises from the work sheets. More advanced students who finish the tutorial sessions early can then be motivated to help other students who need more time or have problems getting familiar with the code. The concept of optional exercises from our course material (describing tasks and providing result outputs, but not the code how to produce them) can be used at the end of the semester for in-class exams to assess student's ability to produce certain measurements or visualizations from given input data. Alternatively, they can be used as a basis for homework assignments where students are requested to write a paper including text mining analysis, result interpretations and theoretical embedding of their work. By requesting students to hand in papers as HTML files rendered from Rmarkdown scripts, teachers are able to fully reproduce the student's work. On a voluntary basis, student papers could be published to provide alternative solutions to the class.

As a follow-up to the 2017 course, we plan to publish the tutorial sheets under an open source license on Github.[8] We further plan to publish the course material as an open source textbook for self-learners. Since the code and its textual explanation already exists in Rmarkdown syntax, rendering the course as a digital and/or printed book project comes with (almost) no extra cost for layout.

[8] https://tm4ss.github.io

Table 1: Course evaluation 2016 (N = 21)

Survey question / scale	**1**	**2**	**3**	**4**	**5**
The course is well structured.*	-	4.7	-	38.1	57.1
The knowledge transfer between theory and practice works well.*	-	4.7	9.5	28.6	57.1
I feel enabled to approach my own text mining analysis.*	4.7	19.1	33.3	23.8	19.1
The course materials were useful.*	-	-	-	23.8	76.2
I have learned a lot in the course.*	-	-	4.7	47.6	47.6
How do you assess the quantity of the course contents?**	-	-	38.1	47.6	14.3
How do you assess the amount of time for discussion?**	-	9.5	90.5	-	-
How do you assess the amount of time for practical work?**	4.7	28.6	66.7	-	-

* scale: strongly disagree (1), rather disagree (2), neither/nor (3), rather agree (4), strongly agree (5)
** scale: way too low (1), rather too low (2), just right (3), rather too much (4), way too much (5)

For the text book version, an extended theoretical introduction to the course is planned.

6 Conclusions

In this paper, we share our experience of four years teaching a text mining course for social scientists and humanities scholars. We highlight the use of the R programming language as a flexible and easy to learn environment for many complex text analysis tasks. We further recommend using R extensions such as knitr to create tutorial sheets for gaining practical experience in hands-on sessions which make up to 50 % of the course time. The course material will be made publicly available and can easily be adapted for alternative teaching formats such as semester courses or as text book for self-learners.

References

Laurence Anthony. 2014. Antconc.

Paul Baker, Costas Gabrielatos, Majid KhosraviNik, Michael Krzyzanowski, Tony McEnery, and Ruth Wodak. 2008. A useful methodological synergy? combining critical discourse analysis and corpus linguistics to examine discourses of refugees and asylum seekers in the uk press. *Discourse & Society*, 19(3):273–306.

Kenneth Benoit and Adam Obeng. 2017. readtext: Import and handling for plain and formatted text files.

Chris Biemann, Gerhard Heyer, Uwe Quasthoff, and Matthias Richter. 2007. The leipzig corpora collection: Monolingual corpora of standard size. In *Proceedings of Corpus Linguistic 2007*, Birmingham.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Gabor Csardi and Tamas Nepusz. 2006. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC'06*, pages 417–422.

Ingo Feinerer, Kurt Hornik, and David Meyer. 2008. Text mining infrastructure in r. *Journal of Statistical Software*, 25(5):1–54.

Ian Fellows. 2014. wordcloud: Word clouds.

Barney G. Glaser and Anselm L. Strauss. 2005. *Grounded theory: Strategien qualitativer Forschung*. Huber, Bern, 2 edition.

Bettina Grün and Kurt Hornik. 2011. Topicmodels: an r package for fitting topic models. *Journal of Statistical Software*, 40(13):1–30.

Thibault Helleputte. 2017. Liblinear: Linear predictive models based on the liblinear c/c++ library.

Kurt Hornik. 2016. opennlp: Apache opennlp tools interface.

ILIAS Core Team. 2017. Ilias: Open source e-learning.

Sean Kross, Nick Carchedi, Bill Bauer, and Gina Grdina. 2017. swirl: Learn r, in r. R package version 2.4.3.

Matthias Lemke and Gregor Wiedemann, editors. 2016. *Text Mining in den Sozialwissenschaften: Grundlagen und Anwendungen zwischen qualitativer und quantitativer Diskursanalyse*. Springer VS, Wiesbaden.

Thomas Morton, Joern Kottmann, Jason Baldridge, and Gann Bierner. 2005. Opennlp: A java-based nlp toolkit.

Andreas Niekler, Gregor Wiedemann, and Gerhard Heyer. 2014. Leipzig corpus miner: A text mining infrastructure for qualitative data analysis. In *Terminology and Knowledge Engineering (TKE '14)*.

Bethany Nowviskie. 2014. On the origin of "hack" and "yack". *Journal of Digital Humanities*, 3(2).

R Core Team. 2016. R: A language and environment for statistical computing.

Paul Rayson, Damon Berridge, and Brian Francis. 2004. Extending the cochran rule for the comparison of word frequencies between corpora. In *Proceedings of JADT'04*, pages 926–936.

RStudio Team. 2015. Rstudio: Integrated development environment for r.

Benjamin M. Schmidt. 2012. Words alone: dismantling topic models in the humanities. *Journal of Digital Humanities*, 2(1).

Mike Scott. 2016. Wordsmith tools.

Stéfan Sinclair and Geoffrey Rockwell. 2012. Voyant tools (web application).

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Annie Waldherr, Gerhard Heyer, Patrick Jähnichen, Gregor Wiedemann, and Andreas Niekler. 2016. Mining big data with computational methods. In Gerhard Vowe and Philipp Henn, editors, *Political communication in the online world: Theoretical approaches and research designs*, pages 201–217. Routledge, New York.

Hadley Wickham. 2009. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Hadley Wickham. 2016. rvest: Easily harvest (scrape) web pages.

Gregor Wiedemann. 2013. Opening up to big data: computer-assisted analysis of textual data in social sciences. *Historical Social Research*, 38(4):332–357.

Yihui Xie. 2014. knitr: A comprehensive tool for reproducible research in r. In Victoria Stodden, Friedrich Leisch, and Roger D. Peng, editors, *Implementing reproducible research*. Taylor and Francis, Boca Raton.