ReadMe: Software for Automated Content Analysis[1]

Daniel Hopkins[2] Gary King[3] Matthew Knowles[4] Steven Melendez[5]

Version 0.996
August 31, 2010

[1]Available from http://GKing.Harvard.Edu/readme under the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 License, for academic use only. Special thanks to Anton Strezhnev for assistance with making ReadMe compatible with Python 3.0.

[2]Institute for Quantitative Social Science, 1737 Cambridge Street, Harvard University, Cambridge MA 02138; www.danhopkins.org,dhopkins -at- fas.harvard.edu

[3]Institute for Quantitative Social Science, 1737 Cambridge Street, Harvard University, Cambridge MA 02138; http://GKing.Harvard.Edu, King -at- Harvard.Edu, (617) 495-2027.

[4]Institute for Quantitative Social Science, 1737 Cambridge Street, Harvard University, Cambridge MA 02138; MKnowles -at- Fas.Harvard.Edu, (617) 384-5747.

[5]Institute for Quantitative Social Science, 1737 Cambridge Street, Harvard University, Cambridge MA 02138; melend -at- Fas.Harvard.Edu, (617) 384-5747.

Contents

1 Introduction

The ReadMe software package for R takes as input a set of text documents (such as speeches, blog posts, newspaper articles, judicial opinions, movie reviews, etc.), a categorization scheme chosen by the user (e.g., ordered positive to negative sentiment ratings, unordered policy topics, or any other mutually exclusive and exhaustive set of categories), and a small subset of text documents hand classified into the given categories. The hand classified subset need not be a random sample and can differ in dramatic but specific ways from the population of documents. If used properly, ReadMe will report, normally within sampling error of the truth, the proportion of documents within each of the given categories among those not hand coded.

ReadMe computes the proportion of documents in each category without the more error-prone intermediate step of classifing individual documents. This is an important limitation for some purposes, but not for most social science applications. For example, we have been unable to locate many published examples of content analysis in political science where the ultimate goal was individual-level classification rather than the generalizations provided by the proportion of documents within each category, or perhaps the proportion within each category in subsets of the documents (such as policy areas or years). It appears that a similar point also applies to the most social sciences and related academic areas. Thus, for example, our method cannot be used to classify letters to a legislative representative by policy area, but it could accurately estimate the distribution of letters by policy areas — which makes the method useless in helping the legislator

route letters to the most informed employee to draft a response, but useful for a political scientist tracking the intensity of this form of constituency expression by policy.

The specific procedures implemented in ReadMe are described in

> Daniel Hopkins and Gary King. 2007. "Extracting Systematic Social Science Meaning from Text," `http://GKing.Harvard.edu/`.

2 Installation

ReadMe requires the current version of `R`, available for free from `http://cran.r-project.org/`. Installation differs slightly by operating system.

2.1 Python (Required)

ReadMe requires an interpreter for the Python programming language. Python is free and open-source software and is available for Windows, Mac OS X, Linux and many other common platforms.

If Python is not installed on your computer, you can download source or executable packages for free at `http://www.python.org/download/`. Standard installations for Windows, Mac or Linux should require no further configuration for use with ReadMe.

If you receive a message indicating that Python is not on your system path, check that the interpreter is installed and the directory in which it is installed is on your system path. If Python is installed with the default options on Windows, it is usually unnecessary to change your system path. Default installations for other operating systems normally place Python in a directory already on your system path. Please see the excellent documentation on the Python site for more information about installing and running the Python interpreter.

The system path is a list of directories in which the operating system will search for a given program when you type its name. In both Windows and Unix, it is defined in an environment variable generally called PATH. In Unix, the path is specified in the appropriate configuration file for your login shell. In recent versions of Windows, you may change your system path by right clicking on the "My Computer" icon on your desktop, clicking properties, clicking the "Advanced" tab and, within "Advanced," clicking the Environment Variables button. Find the path variable and click "edit." Notice that the path is a list of directories, separated by a semicolon. Add or delete directories as appropriate while adhering to this format.

If you do not wish to change your system path for any reason, you can specify the full path of the python binary using the pyexe argument to the `undergrad` function.

ReadMe has been tested with Python versions 2.3, 2.4 and 2.5 and should work with earlier versions as well. If you are running an earlier version of Python and experience any difficulty with the Python portion of the program, please upgrade to a more recent version.

2.2 Windows

Launch `R` and then at the `R` command prompt, type either:

```
> install.packages("ReadMe", repos= "http://gking.harvard.edu", type="source")
```

Alternatively, you may download the Windows bundle from `http://GKing.Harvard.Edu/bin/windows/contrib` and use the `R` pull-down menu commands for installing a package from a zip file.

2.3 Linux/Unix

You initially need to create both local `R` and local `R` library directories if they do not already exist. At the Unix command prompt in your home directory, do this by typing:

```
> mkdir ~/.R ~/.R/library
```

Then open the '`.Renviron`' file that resides in your home directory, creating it if necessary, and adding the line:

```
R_LIBS = "~/.R/library"
```

using your preferred text editor (e.g. pico, VI, Emacs, etc.). These steps only need to be performed once. After starting R, install ReadMe by typing at the R command prompt, either:

```
> install.packages("ReadMe", repos = "http://gking.harvard.edu")
```

You can ignore warning messages.

Alternatively, you may download the Unix bundle 'ReadMe_XX.tar.gz', available from http://gking.harvard.edu/src/contrib/, and place it in your home directory. Note that 'XX' is the current version number. Then, at the Unix command line from your home directory, type

```
> R CMD INSTALL ReadMe_XX.tar.gz
```

to install the package.

2.4 Updating VA

ReadMe also requires VA, which may need updating if previously installed:

```
update.packages("VA",repos="http://gking.harvard.edu",lib="~/.R/library", type="source")
```

If VA has not been previously installed, it will be installed automatically during the ReadMe installation.

3 Examples

To use ReadMe, you must always begin by issuing a library command: `library(ReadMe)`

3.1 Estimating Blogger Sentiment Toward Senator Hillary Clinton

This example uses a training set of size 500 to estimate sentiment toward Senator Hillary Rodham Clinton in a test set of size 1438 blog posts.

A control file is given in comma-separated form, along with the 1938 posts comprising the training and test sets. These can be found in demofiles/clintonposts within the package's install directory.

The command `demo(clinton)` executes the following R code:

```
oldwd <- getwd()
setwd(system.file("demofiles/clintonposts", package="ReadMe"))

undergrad.results <- undergrad(sep = ',')

undergrad.preprocess <- preprocess(undergrad.results)

readme.results <- readme(undergrad.preprocess)
setwd(oldwd)
```

The first two lines save the current working directory for the user's convenience, then switch the working directory to "demofiles/clintonposts" in the ReadMe directory.

The next line calls the "undergrad" function to process the texts based on the control file, storing the data in undergrad.results; default parameters are used, except that we specify that the control file is comma-separated with the "sep" argument.

Next, we use the "preprocess" function to remove columns with variance 0.

Finally, we store the results of the readme function in readme.results and restore the initial working directory.

After the demo is complete, its results can be viewed in readme.results and the intermediate data in undergrad.results and undergrad.preprocess.

4 R Function Reference

4.1 Function `undergrad()`

The data input function is called `undergrad` in honor of the undergraduates who commonly perform this task in content analyses. It translates a set of texts stored in a single folder into a data set where each text is represented by a row and each word is a column, with 1s indicating if that word appears in the text.

4.1.1 Usage

```
undergrad(control ="control.txt", stem=T,strip.tags=T,ignore.case=T,
          table.file="tablefile.txt",threshold=0.01, python3=F,
          pyexe=NULL, sep=NULL,printit=TRUE)
```

4.1.2 Inputs

control Specifies a control file to load in, specifying filenames and binary classifications for the texts. The file should contain be three sep-separated (or whitespace-delimited, if sep is NULL as in the default case) columns, one headed "filename" providing a list of filenames, one headed "truth" providing the classifications for a subset and missing values (NA or "." for the others), and a third headed "trainingset" and having a 1 for each element of the training set and a 0 for elements of the test set. When `trainingset`=1, `truth` should not be missing or it will be deleted. The function will compute the distribution of documents across categories for all documents with `trainingset`=0 (if `truth` is not missing for some these observations, it will not be used during estimation but will be used for printing and graphics on output to compare to the estimates). Defaults to "control.txt".

Alternatively, one can provide a data frame in the same three-column format. This will be written to readmetmpctrl.txt in the working directory during program operation.

stem Should the Porter stemmer be used to stem the individual words? Please note: the Porter stemmer relies on case-insensitivity and will only function properly when ignore.case is set to TRUE. See details at `http://www.tartarus.org/~martin/PorterStemmer/`. Defaults to TRUE.

strip.tags Indicates whether or not HTML/XML/SGML "head," tags and JavaScript should be stripped from the input. Defaults to TRUE.

table.file Path of file in which table of word frequencies should be stored. Defaults to "table-file.txt". Of course, user must have read and write access to this file, and prior contents of file will be overwritten.

threshold A floating-point number between 0 and 1. Only words occuring in more than `threshold` (and less than 1-`threshold`) times the number of texts will be included as features. To include all words, set threshold to 0. Default = 0.01, which includes all words occuring in more than 1% of texts.

pyexe Path to use for Python interpreter. If NULL, ReadMe will first search the system path and then, if on Windows, default installation directories. If ReadMe is unable to locate your Python interpreter or you wish to use a different interpreter than that which lies on your system path set this variable. Defaults to NULL.

python3 Python versions 3.0 and greater require that different syntax be passed to Python, so set this to TRUE if you are using Python 3.0 or greater. Defaults to FALSE.

sep String variable indicating column separators in control file. Defaults to NULL, in which case whitespace separates columns.

printit Boolean variable indicating whether or not progress of text processing module should be output to screen. Defaults to TRUE.

fullfreq Boolean variable indicating whether data returned should be frequency data giving the number of times a word occurs rather than the usual binary (word occurs or not). Defaults to FALSE, meaning binary data are returned.

4.1.3 Value

A list containing the following elements, to be passed to the ReadMe function. To call ReadMe with your own choice of values, edit this list or pass changes separately as arguments.

trainingset Binary table indicating which words, among those words that satisfy `threshold`, appear in which text in the training set. Words appearing in all or none of the texts are omitted regardless of `threshold`.

testset Same format as trainingset, but for test set of texts.

formula Formula to use in the call to function VA, where the main computation is done. Defaults to include all words that statisfy `threshold` and appear less than 100% of the time as dependent variables. "truth" from control file is the explanatory variable.

features Number of features to use in each subset in VA. Corresponds to `nsymp` in VA. Defaults to 15. See VA or ReadMe function documentation for details.

n.subset Number of subsets to use in VA. Defaults to 300. See VA or ReadMe function documentation for details. Larger numbers produce more precision.

prob.wt Vector of probability weights for the features to be employed by VA. Must have length equal to the number of features in the formula. Defaults to 1. See VA or ReadMe function documentation for details.

boot.se Use bootstraping in VA to compute standard errors? Defaults to FALSE. Bootstrapping produces standard errors but is time-intensive. See VA or ReadMe function documentation for details.

nboot How many bootstrapping samples for VA? Defaults to 300. Ignored unless boot.se set to TRUE. See VA or ReadMe function documentation for details.

printit Print progress of VA function? Defaults to TRUE. See VA or ReadMe function documentation for details.

4.1.4 Details

When a text file is used as the control file, it should be a comma-delineated table in which each line refers to a text to be included in the test set or training set. The first column, `filename`, is the path to the text (either absolute or relative paths will work). The second column, `truth`, defines the category of each text. The third column, `trainingset`, contains a binary value which indicates whether the text should be included training set.

If the `trainingset` value is 0 or is omitted, ReadMe will treat the text as part of the test set. Likewise, the `truth` value is typically omitted for texts in the test set. However, if a `truth` value is included for a test set text, it will not be used during estimation but will be used for printing and graphics on output to compare to ReadMe's estimate of the distribution.

Note that there is no numerical significance to the values used in the `truth` column to identify the categories; these values serve only as labels.

Consider the following example control file:

```
filename,truth,trainingset
/users/m/readme/example/file1.txt,1,1
/users/m/readme/example/file2.txt,2,1
/users/m/readme/example/file3.txt,2,1
/users/m/readme/example/file4.txt,3,1
/users/m/readme/example/file5.txt,,
```

```
/users/m/readme/example/file6.txt,,
/users/m/readme/example/file7.txt,,
/users/m/readme/example/file8.txt,,
/users/m/readme/example/file9.txt,,
```

ReadMe always disregards the first line of the control file, which can be used to label the columns. In this example, ReadMe will use the text documents in `file1.txt` through `file4.txt` as training texts, and will compute the distribution of across the categories `1,2,3` for the remaining texts.

When working with large control files, it may be useful to build and manage the control file in a spreadsheet program and then export the resulting file in the CSV comma-delineated format (both Microsoft Excel and OpenOffice.org Calc support this feature). Likewise, on systems which support a UNIX shell, the `ls -1` command can be used to build a list of the texts in a given folder, which can then be copy-and-pasted into a spreadsheet or directly into a text control file.

4.2 Function `preprocess`

This function takes the inputted data matrix from `undergrad()` and prepares it for analysis by `readme()` by removing invariant columns.

4.2.1 Usage

```
preprocess(undergrad.results)
```

4.2.2 Inputs

undergrad.results The output from `readme`

4.2.3 Value

undergrad.preprocessed The preprocessed list with invariant columns from either data set (the test or training set) removed

4.3 Function `readme()`

The main function in ReadMe is called `readme`. It computes the proportion of text documents within each of the user-specified categories. It can also optionally computes bootstrap-based standard errors. `readme` requires the R function `VA` (by Gary King and Ying Lu), which after extensive processing does the final computation.

4.3.1 Usage

```
readme(undergradlist = list (), trainingset = NULL,
       testset = NULL, formula = NULL,
       n.subset=NULL, prob.wt=NULL, boot.se=NULL,
       nboot=NULL, printit=NULL)
```

4.3.2 Inputs

There are two ways to provide input to the `readme` function: as the list object as returned by the `undergrad` function using the undergradlist parameter, or through the individual parameters corresponding to the list elements. Any non-null parameter will override the corresponding list element. If unmodified, the function will use the default parameters for `VA`.

undergradlist A list, as defined above, having an element for each parameter below. Specify parameter values either through the list or through the individual arguments. Non-null individual arguments will supersede list values.

features A positive integer specifying the number of words to be subset from all words for estimation at each iteration. For the choice of `features`. Default=16. (This option is `nsymp` in `VA`).

formula By default, the formula specifies all the possible features (see `undergrad()`). To modify it, the new formula must be specified as

```
formula=cbind(WORD.the+WORD.formula)~TRUTH
```

n.subset A positive integer specifying the total number of draws of different subsets of features. Default=300.

prob.wt A positive integer or a vector of weights that determines how likely a feature should be when selecting subsets of features. When `prob.wt` is a vector, it must be a vector of probabilities that sum up to 1 with length equal to the total number of features. When `prob.wt=1`, binomial weights which are proportion to the inverse of variances of the each reported binary feature variable. When `prob.wt=0`, all features will be selected with equal probability. Default=1.

boot.se A Logical value. If `TRUE`, bootstrap standard errors of the CSMF are estimated. This option typically takes a lot of computing time. The default is `FALSE`.

nboot A positive integer. If `boot.se=TRUE`, it specifies the number of bootstrapping samples taken to estimate the standard errors of CSMF. The default is `300`.

printit Logical value. If `TRUE`, the progress of the estimation procedure is printed on the screen.

4.3.3 Value

An object of class "VA", a list containing the following elements:

est.CSMF The estimated proportion in each category

true.CSMF The observed proportion in each category whenever available.

est.se The bootstrap standard errors of `est.CSMF` when `boot.se=TRUE`.

true.CSMF.bootmean The bootstrap mean of the observed category proportions when they are available and when `boot.se=TRUE`.

true.bootse The bootstrap standard errors of the observed category proportions when they are available and when `boot.se=TRUE`.

4.4 Examples

```
oldwd <- getwd()
setwd(system.file("demofiles", package="ReadMe"))

###Load in word only if it appears in more than 20% of documents
output<-undergrad(threshold=.2)

###Modify number of subsets
results<-readme(undergradlist=output,n.subset=30)

###Modify Training Set
newtrainingset <- output$trainingset[1:7,]
results<-readme(undergradlist=output,traingset=newtrainingset,n.subset=30)
setwd(oldwd)
```

5 Hand-Coding Procedures

As with many methods, many of the most critical decisions in analyzing texts come prior to using the computational and statistical procedures outlines above. Here, we provide step-by-step guidelines on hand-coding which could improve the quality of the data available for analysis with `ReadMe`.

5.1 Define Objectives

First, consider what information will be extracted from the coding results, and how this information will address the research objectives. Define the unit of analysis and the quantity of interest for the study. What will be your fundamental unit—documents, paragraphs, webpages, sentences, etc.? Although ReadMe lets you choose among these, we recommend that you choose the unit that makes the most substantive sense. For our work, the blog post was the relevant unit and sentences within it relatively meaningless; other applications might lead to different choices.

5.2 Develop the Corpus

Define the scope of the corpus Identify any structural limitations (e.g., no texts under n characters should be included, non-English texts should be excluded, etc). Consider whether the entire corpus or merely a subset will be hand-coded. In the latter case, identify how the subset should be selected (randomly, representatively, etc.)

Format the corpus In some cases it is necessary to remove information which should not be distributed to the hand-coders, such as personal details and other such data. Likewise, some corpora might contain tags or labels which would interfere with the coders making objective analyses of the text.

5.3 Develop Coding Procedures

Codebook Compile a codebook which includes an overview of the project, its goals, and methods. It should also include an outline of the coding task: how to identify relevant text, select the appropriate dimension(s), score the text, and how to deal with 'tough cases' and other uncertainty. Ideally, all information the coders need should be in the codebook; in this way, all coding rules can be made public and your study can be replicated without your personal participation.

Coding Models

1. Categorical Classification: Either mutually exclusive coding where coders will select the single best classification for the unit of analysis, or non-exclusive coding where coders will select all categories relevant to the unit of analysis

2. Dimension / Affect Coding: Coders will score each unit of analysis on a relevant dimension (scale). Scores are specific to the dimensions on which they are evaluated. Any numerical value can be used to establish the affect scale, but it is essential to carefully define what each discrete score represents. For example, on an affect scale $\{-2, -1, 0, 1, 2\}$ coders should be provided with specific examples of what constitutes a -1 as opposed to a -2 score, and so on. Coders should be given a single dimension, a choice of dimensions, or multiple dimensions on which to code each document. Each dimension should be a single, exhaustive scale containing all relevant positions. Some examples of dimensions: "Economic liberalism / conservatism," "Leadership ability," "Hawk/Dove," etc. In single dimension coding, coders are only given one dimension on which to code, so each unit of analysis will be coded on that dimension or coded as irrelevant. In multiple dimension coding, coders are given a list of dimensions, and code the unit of analysis on all relevant dimensions.

3. Develop coding infrastructure: Consider whether hand coding done with computer assistance would be helpful, or whether keeping track in a spreadsheet or on paper is sufficient. Commercial options include programs like Atlas.ti, Nud*ist, Xsight, or EZ-text. Some projects develop their own computer programs for collecting coding data. The selection of a coding interface can have a dramatic effect on the results. For example, see discussion in Kwon, Shulman and Hovy (2006).

5.4 Coder Training and Evaluation

Training There are at least three important aspects to coder training: use of controlled, prescored practice cases, periodic collaboration among coders as a means of calibrating their work and identifying problems with the coding system, and periodic retraining of coders to ensure consistency of their work over time.

Evaluation The coders work should be evaluated by analyzing inter-coder reliability rates. Likewise, it is important to let coders know how they are doing; many coders see inter-coder data as a competitive challenge to improve their work.

Feedback from Coders On the coding sheet or interface, provide the coders with an opportunity to note any additional issues specific to the text or the coding process. This can be a valuable tool for identifying problems with the coding system.

5.5 Assembling and Reporting Results

One issue the coding system must address is how to deal with ties and disagreement among coders. One option is to have any disputed codings re-coded by additional coders. However, this may significantly increase the among of coder-hours required for a project. Alternatively, ties can be broken at random. It is essential to report intercoder reliability statistics in a clear way, prior to implementing these fixes. The Kappa statistic described in Kwon, Shulman and Hovy (2006) is helpful, but may not be provide sufficient detail. Craggs and Wood (2005) offers an in-depth review of various methods of reporting intercoder reliability; they suggest that "only chance-corrected measures that assume a common distribution of labels for all coders are suitable for measuring agreement in reliability studies" (Craggs and Wood, 2005). In our work, we find that the whole inter-coder reliability matrix, without summary, is useful, since it provides specific feedback about where the coding scheme can be improved.

References

Craggs, Richard and Mary McGee Wood. 2005. "Evaluating Discourse and Dialogue Coding Schemes." *Computational Linguistics* 31(3):289–295.

Kwon, Namhee, Stuart W. Shulman and Eduard Hovy. 2006. "Collective Text Analysis for eRulemaking." *7th Annual International Conference on Digital Government Research* .