An Introduction to Austin

Will Lowe
MZES
University of Mannheim

January 6, 2012

This vignette describes how to get word frequency data into `austin` and how to scale it using Wordfish or Wordscores. Text scaling methods rely on the 'bag of words' representation of documents, so it's useful to say something first about counting words and turning them into a form `austin` can deal with.

1 Counting Words

Austin's document scaling functions operate with word frequency matrices with rows representing documents and columns representing words. You can populate these matrices and read them into R however you like. However, if you do not yet have a prefered method try JFreq, particularly if you a need for multiple language support, stop word removal, lemmatisation, a choice of sparse output formats, and speed. JFreq offers three formats for word frequency data:

LDA-C an SVMlight-like sparse format used in many topic models packages, e.g. `lda`. It is described in section B.2 of the documentation from Blei (2006)[1]. (SVMlite format is described at (Joachims, 2008)).

Matrix Market a sparse matrix format common in numerical analysis applications, and described by Boisvert, Pozo and Remington (1996). JFreq generates and Austin reads only the 'coordinate' (sparse) format for integer elements because this is most suitable for representing sparse word count data.

CSV the comma-separated value format as understood by R's `read.csv` function with `row.names=1`. This is not generally suitable for word count data and does not scale well to large document collections.

If you have generated word frequencies using JFreq and specified the output folder to be `folder` then reading the data back into austin is straightforward, whichever output format was chosen:

```
> wfm <- read.jfreq(folder)
```

You can also use the helper functions in Austin to read individual files in the supported formats. See the documentation for `read.ldac` and `read.mtx` for details.

Austin works with any two dimensional matrix-like object with documents as rows and words as columns[2]. Such objects may be of any class that indexes like a matrix. We recommend the `Matrix` package's sparse matrix class for word frequency data. All the example data sets in the package are in this form. Word frequency matrices should have a full set of row and column names. Ideally these should also be labelled 'docs' and 'words' respectively, so we remember what we're doing.

Once a matrix has been constructed the function `trim` can be used to remove low frequency words and words that occur in only a few documents. It can also be used to sample randomly from the set of words. This can be helpful to speed up analyses and check the robustness of scaling results to different vocabulary choices.

[1] Not noted on this webpage is that word indices start from 0, not 1.

[2] Previous versions of this package used a set of accessor functions and a custom word frequency matrix object `wfm`. This turned out to be more trouble than it was worth. The orientation is now fixed: documents are rows. Dammit

2 Scaling Documents using Wordfish

Austin implements the one dimensional text scaling model Wordfish (Slapin and Proksch, 2008; Proksch and Slapin, forthcoming). Under a slightly different parametrisation this is Goodman's RC(M) model with M=1 (e.g. Goodman, 1985). When document positions are random variables rather than unknown parameters the model has been called Rhetorical Ideal Points (Monroe and Maeda, 2004), which is in turn equivalent to a form of multinomial Item Reponse Theory or Latent Trait model with Poisson link (see e.g. Moustaki and Knott, 2000).

2.1 Model and Estimation

Austin implements Goodman's model in Slapin and Proksch's parameterisation: The number of times word j occurs in document i is modeled as

$$P(Y_{ij} \mid \theta_i) = \text{Poisson}(\mu_{ij})$$
$$\mu_{ij} = \psi_j + \beta_j \theta_i + \alpha_i.$$

where θ is the document position parameter of primary interest for scaling applications.

The model is identified by setting $\alpha_1 = 0$ and constraining θ to have zero mean and unit variance. The package's `wordfish` estimation routine then resolves the remaining sign ambiguity by forcing one document position to be large than another using the `dir` argument.

The four sets of parameters are estimated using a Conditional Maximum Likelihood procedure that alternates the estimation of the word parameters (j) and document parameters (i). Word parameter estimation is stabilised by ridge regularizing the word parameters β. This is equivalent to putting an independent Normal$(0, \sigma^2)$ prior on each β, ensuring they do not get too large due to small numbers of documents or low frequency words. Standard errors for θ are generated from the profile likelihood.

2.2 Example

Here we show an analysis using simualted data. We start by loading the package

```
> library(austin)
```

and generate an small set of simulated data according to the model assumptions above using the `sim.wordfish` function

```
> dd <- sim.wordfish(docs=10, vocab=12)
```

The true document positions are `dd$theta` and the data is `dd$Y`. Next we try to recover these position from the generated word counts by fitting a wordfish model

```
> wf <- wordfish(dd$Y, dir=c(1,10))
```

```
2 iterations: deviation 2.842171e-14
```

The `dir` argument identifies the sign of θ. Here $\hat{\theta}_1 < \hat{\theta}_{10}$. You can also hand the `wordfish` function a set of starting parameters using `params` or specify the initialization function to use instead using `init.fun`. The actual estimation will be done using `fit.fun`. You can also set the estimation tolerance with `tol`, the regularization parameter for β with `sigma`. Finally, if you want a running commentary on the estimation progress, set `verbose=TRUE`. All these have sensible default values, although you may wish to set `dir` to order two documents that you think probably have opposing positions.

If you write your own initialization or estimation functions just adhere to the calling interface and return values of `classic.wordfish` or `initialize.wordfish`, and hand `wordfish` the name of your new function as the value of `init.fun` or `fit.fun` respectively.

After estimation the model's estimated document positions can be summarized using

```
> summary(wf)

Call:
        wordfish(wfm = dd$Y, dir = c(1, 10))

Document Positions:
    Estimate Std..Error    Lower   Upper
D01  -1.4644    0.10840 -1.67683 -1.2519
D02  -1.2571    0.07070 -1.39570 -1.1186
D03  -0.7885    0.06507 -0.91605 -0.6610
D04  -0.5504    0.06802 -0.68376 -0.4171
D05  -0.3127    0.06518 -0.44044 -0.1849
D06   0.1451    0.06863  0.01061  0.2797
D07   0.5102    0.07012  0.37271  0.6476
D08   0.7754    0.06554  0.64694  0.9038
D09   1.3280    0.07493  1.18115  1.4749
D10   1.6145    0.07029  1.47673  1.7522
```

The remaining parameters examined using the `coef` function. A fitted word counts are available using `fitted`. New documents can be scaled using `predict` although the confidence intervals do not currently take into account estimation uncertainty in ψ, β and α.

Estimated document positions and 95% confidence intervals can also be graphed[3]. Any second argument to the plot function is taken to be a vector of true document positions. These are then plotted over the original plot, as shown in Figure 1. The red dots are the true document positions.

3 Scaling Documents using Wordscores

Wordscores (Laver, Benoit and Garry, 2003) is a non-parametric method for scaling texts closely related to both correspondence analysis by implementing an incomplete reciprocal averaging algorithm, and to quadratic ordination as an approximation to an unfolding model (see Lowe, 2008, for discussion).

3.1 Model and Estimation

Wordscores makes a low rank matrix decomposition of a word frequency matrix in essentially the same way as correspondence analysis, except that an initial set of row scores (θ) are assumed to be the known positions of 'reference' documents. From these positions, column scores π are computed and referred to as 'wordscores'.

Out of sample 'virgin' documents are assigned positions on the basis of the initial wordscores by taking the average of the scores of the words in the new document. Several ad-hoc weighting procedures have been suggested for making out of sample document positions comparable to those of the original documents. Standard error formulae have also been offered, although the meaning of standard errors in a model with no explicit probability model is unclear. For completeness Austin implements the weighting and standard error computations described in Laver, Benoit and Garry (2003).

3.2 Example

We replicate the analysis in the original papers by loading a simulated data set

```
> data(lbg)
> lbg
```

[3] For more than a few tens of words the confidence intervals will probably be 'implausibly' small. They are nevertheless asymptotically correct given the model assumptions. It is those assumptions you might doubt.

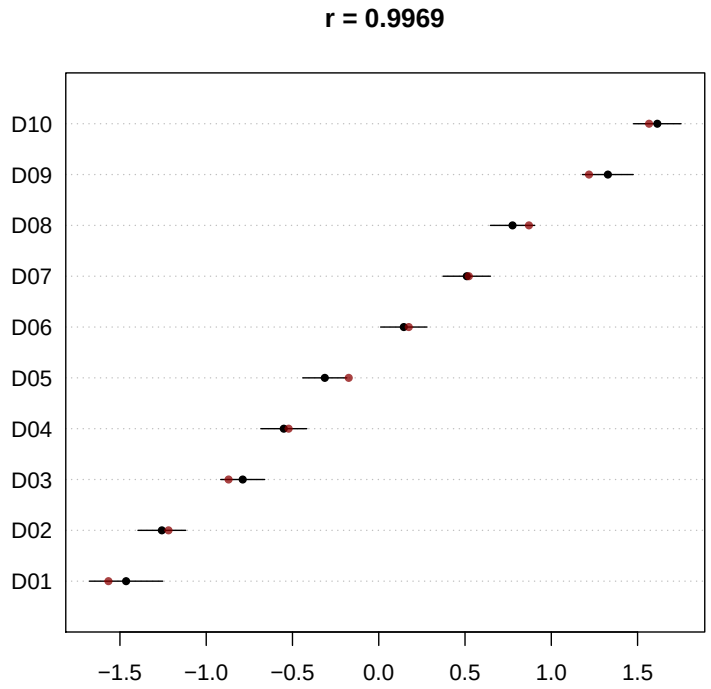

Figure 1: Wordfish position estimates on simulated data using the `plot(wf, dd$theta)`.

```
6 x 37 sparse Matrix of class "dgCMatrix"

R1  2  3 10 22 45 78 115 146 158 146 115  78  45  22  10   3   2   .   .   .   .
R2  .  .  .  .  .  2   3  10  22  45  78 115 146 158 146 115  78  45  22  10   3
R3  .  .  .  .  .  .   .   .   .   .   2   3  10  22  45  78 115 146 158 146 115
R4  .  .  .  .  .  .   .   .   .   .   .   .   .   .   .   2   3  10  22  45  78
R5  .  .  .  .  .  .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   2
V1  .  .  .  .  .  .   .   2   3  10  22  45  78 115 146 158 146 115  78  45  22

R1   .   .   .   .   .   .   .   .   .   .  .  .  .  .  .  .
R2   2   .   .   .   .   .   .   .   .   .  .  .  .  .  .  .
R3  78  45  22  10   3   2   .   .   .   .  .  .  .  .  .  .
R4 115 146 158 146 115  78  45  22  10   3  2  .  .  .  .  .
R5   3  10  22  45  78 115 146 158 146 115 78 45 22 10 3  2
V1  10   3   2   .   .   .   .   .   .   .  .  .  .  .  .  .
```

For this data we assume that scores for documents R1-R5 are known, and V1 is an out of sample document whose position we require. We first separate the in and out of sample data:

```
> ref <- lbg[1:5,]
> vir <- lbg[6,,drop=FALSE]
```

fit the model using the reference documents and positions from the paper, and then summarise the result

```
> ws <- wordscores(ref, scores=seq(-1.5, 1.5, by=0.75))

> summary(ws)

Call:
        wordscores(wfm = ref, scores = seq(-1.5, 1.5, by = 0.75))

Reference Document Statistics:

   Total Min Max Mean Median Score
R1  1000   0 158   27      0 -1.50
R2  1000   0 158   27      0 -0.75
R3  1000   0 158   27      0  0.00
R4  1000   0 158   27      0  0.75
R5  1000   0 158   27      0  1.50
```

The summary presents details about the reference documents. Like the `wordfish` function `wordscores` also accepts a `fit.fun` argument.

If we want to see the wordscores that were generated we look for the model's coefficients

```
> coef(ws)

         A          B          C          D          E          F          G
-1.5000000 -1.5000000 -1.5000000 -1.5000000 -1.5000000 -1.4812500 -1.4809322
         H          I          J          K          L          M          N
-1.4519231 -1.4083333 -1.3232984 -1.1846154 -1.0369898 -0.8805970 -0.7500000
         O          P          Q          R          S          T          U
-0.6194030 -0.4507576 -0.2992424 -0.1305970  0.0000000  0.1305970  0.2992424
         V          W          X          Y          Z         ZA         ZB
 0.4507576  0.6194030  0.7500000  0.8805970  1.0369898  1.1846154  1.3232984
```

```
       ZC         ZD         ZE         ZF         ZG         ZH         ZI
1.4083333  1.4519231  1.4809322  1.4812500  1.5000000  1.5000000  1.5000000
       ZJ         ZK
1.5000000  1.5000000
```

which can also be plotted.

To get a position for the new document we use the predict function

```
> predict(ws, newdata=vir)

37 of 37 words (100%) are scorable

    Score Std. Err. Rescaled  Lower  Upper
V1 -0.448    0.0119   -0.448 -0.471 -0.425
```

When more than one document is to be predicted, an ad-hoc procedure is applied by default to the predicted positions to rescale them to the same variance as the reference scores. This may or may not be what you want.

References

Blei, D. 2006. "LDA-C: Software for Latent Dirichlet Allocation.".
URL: *http://www.cs.princeton.edu/∼blei/lda-c/*

Boisvert, R. F., R. Pozo and K. A. Remington. 1996. The Matrix Market exchange formats: Initial design. Interagency Report 5935 National Institute of Standards and Technology (NIST).

Goodman, L. A. 1985. "The analysis of cross-classified data having ordered and/or unordered categories: Association models, correlation models, and asymmetry models for contingency tables with or without missing entries." *The Annals of Statistics* 13(1):10–69.

Joachims, T. 2008. "SVMlight Software.".
URL: *http://svmlight.joachims.org*

Laver, M., K. Benoit and J. Garry. 2003. "Extracting policy positions from political texts using words as data." *American Political Science Review* 97(2):311–331.

Lowe, W. 2008. "Understanding Wordscores." *Political Analysis* 16(4).

Monroe, B. and K. Maeda. 2004. "Talk's Cheap: Text-Based Estimation of Rhetorical Ideal-Points." MS.

Moustaki, I. and M. Knott. 2000. "Generalized latent trait models." *Psychometrika* 65(3):391–411.

Proksch, S.-O. and J. B. Slapin. forthcoming. "Position taking in the European Parliament speeches." *British Journal of Political Science* .

Slapin, J. B. and S.-O. Proksch. 2008. "A scaling model for estimating time-series party positions from texts." *American Journal of Political Science* 52(3):705–722.