Class: Linguistics introduction

Dean Foster

October 30, 2013

Admistrivia

• discuss $F$ vs $t$
  
  – $F$ is spherically symmetric—all directions count equally
  
  – $t$ is coordinate based—directions with mostly zeros count more
  
  – $F$ is traditional statistics
  
  – $t$ is more modern (Risk Inflation)
  
  – Both are useful in the right setting

Background on statistical linguistics

• Big problems
  
  – machine translation
– Google queries
– machine reading (i.e. language understanding)
– author identification

• Applications in science:
  – finding related articles on a topic (MEDLINE)

• Business applications:
  – sentiment analysis
  – email sorting (filter email to billing vs. sales)

• Baby steps:
  – Medical records have human diagnostics attached
  – Billing records have transcripts of previous exchanges
  – Purchase patterns have comments on who purchased what
  – reputation isn’t just measured in “stars” but also in short sentences

**Todays Topic: Linguistic data analysis**

**Methodolgy**

Let’s replicate Mosteller and Wallace. First we need to read in all the words:
> federal <- read.csv("federalist.csv",header=TRUE)

{NOTE: Notice: I spent an hour preprocessing this in an editor to make it a bit easier for R to deal with. So for example I added titles for the 3 columns.}

> names(federal)

[1] "number" "author" "text"

> typeof(federal$text)

[1] "integer"

> federal$text <- as.character(federal$text)
> typeof(federal$text)

[1] "character"

> typeof(federal$text)

Types: R thinks of the text as an integer since it doesn’t really know what to do with it. So let’s help it out.

If you don’t do this, many things will run, but run incorrectly. So check your results as you go along.

Now let’s use R to split this into “words.” There are smarter ways of doing this, but for now we will do the most trivial:
> federal$words <- strsplit(federal$text," ")
>

How long are these documents? We want the length of each vector of words. So federal$words is a vector of vectors, but it isn’t a matrix!

> sapply(federal$words,length)

[1] 1773 1836 1606 1795 1471 2271 2546 2297 2196 3308 2757 2398 1050 2346 3362
[16] 2220 1728 2372 2281 1708 2204 3928 1947 2217 2198 2606 1591 1747 2372
[31] 1882 1615 1850 2395 2470 2980 2985 3654 2912 3311 3886 3069 3802 3302
[46] 2870 3127 2084 1827 1252 2104 2049 2427 2247 2271 1730 2383 2307 2372
[61] 1663 2629 3368 2555 2207 2508 1814 1655 3309 3454 1926 2244 2604 1252
[76] 2542 2155 3370 1135 2718 4311 1694 6390 4633 2993
>

Or as a histogram:

> num.words <- sapply(federal$words,length)
> hist(num.words)
>

>
Putting these into our data frame:

```r
> federal$num.words <- sapply(federal$words,length)
> federal$num.char <- c(lapply(as.character(federal$text),nchar),recursive=TRUE)
```

It is always good to check that what you see is actually what you want:

```r
> typeof(federal$words)
```
Now a messy way of generating a variable to do the regression on. We know some of the authors but we don’t know them all:

```r
> federal$training = rep(NA, length(row.names(federal)))
> federal$training[federal$author == "HAMILTON"] = "HAM"
> federal$training[federal$author == "MADISON"] = "mad"
> federal$Y <- (federal$training == "HAM")
>
> Let’s check that we did it correctly:

> cbind(federal$training, as.character(federal$author))[c(1,2,10,18,19,49,64,85),]
 [,1]   [,2]
[1,] "HAM" "HAMILTON"
[2,] NA    "JAY"
```

Ok, finally we can try a trivial regression. If we think like a statistician, about all we can do is count the number of characters / words in the document. These provide significant statistics:
And a R-squared of 0.4.

Let’s come up with some color codes for the various authors:

```r
> colors = rep("gray",length(row.names(federal)))
> colors[federal$author == "HAMILTON"] = "red"
> colors[federal$author == "MADISON"] = "blue"
> colors[federal$author == "JAY"] = "yellow"
> colors[federal$author == "HAMILTON OR MADISON"] = "black"
> colors[federal$author == "HAMILTON AND MADISON"] = "orange"
> shapes = rep(17,length(colors))
> shapes[federal$author == "HAMILTON"] = 16
> shapes[federal$author == "MADISON"] = 15
> shapes[federal$author == "HAMILTON OR MADISON"] = 5

This relies on the fact that R stores these as `level`. So with a bit of futzing we can figure out that 1 = Hamilton, etc. So now we can make our predictions for each article.
Thinking like a linguist

So let’s think like a linguist. Content words (“freedom”, “liberty”, etc) are related to content. But function words (“the”, “an”, etc) are related to style. So let’s try the frequency of these words to predict authorship.

First we grab all the words together in one place:

```r
> all.words = c(federal$words,recursive=TRUE)
>
```

Now let’s count them.

```r
> counts = table(all.words)
```

Finally, let’s look at the most common such words

```r
> sort(counts[which(counts > 1600)],decreasing=TRUE)
```

```
all.words
      the .c. of to .p. and in be a that is which
    16365 13178 11695  6800  5310  4890  4104  3820  3809  2731  2122  2046
by
    1709

```

Putting this in a pretty latex table, we see the most common words are:
Now for some R magic. Let’s make a function that will identify whether a word is "the" or not.

```r
> is.the <- function(t){sum(t == "the")}
> is.the(3)

[1] 0

> is.the("three")

[1] 0

> is.the("the")

[1] 1
```
We can use this function now to pick out all the words that are infact “the” in the text. Because we are statisticians, we normalize by the number of words in the document.

```r
> federal$freq.the <- c(lapply(federal$words,is.the),
+ recursive=TRUE
+ ) / federal$num.words
```

We can repeat this for the other most common words.

{NOTE: We don’t actually have to give these new functions names. This makes the file look a bit prettier–albeit a bit more confusing.}

For the “adult” way of doing this, we would use a loop:

```r
> sorted <- sort(counts[which(counts > 3)],decreasing=TRUE)
> for(i in 1:100)
+   {
+     count <- sorted[i]
+     word <- rownames(sorted)[i]
+     variable.name <- paste("freq",word,sep=".")
+     if(variable.name != "freq.")
+       {
+         federal[[variable.name]] <- c(lapply(federal$words,function(t){sum(t == word)}),recursive=TRUE) / federal$num.words
+       }
+   }
```

Running a regression on the 5 most common symbols we see that the function words are useful, but commas and sentence boundaries aren’t useful.
|                     | Estimate | Std. Error | t value | Pr(>|t|) |
|---------------------|----------|------------|---------|----------|
| (Intercept)         | -1.2791  | 0.5320     | -2.40   | 0.0193   |
| num.words           | -0.0000  | 0.0000     | -1.06   | 0.2941   |
| freq.the            | -13.4613 | 3.4981     | -3.85   | 0.0003   |
| freq.of             | 31.3729  | 5.8299     | 5.38    | 0.0000   |
| freq.to             | 41.9370  | 6.0874     | 6.89    | 0.0000   |

The R-squared has improved to 0.6. We seem to be on a roll. Let’s get rid of the punctuation and add a few more words:

|                     | Estimate | Std. Error | t value | Pr(>|t|) |
|---------------------|----------|------------|---------|----------|
| (Intercept)         | -1.0109  | 0.9009     | -1.12   | 0.2669   |
| freq.the            | -11.9891 | 3.7947     | -3.16   | 0.0026   |
| freq.of             | 25.6606  | 6.8167     | 3.76    | 0.0004   |
| freq.to             | 32.1581  | 6.5057     | 4.94    | 0.0000   |
| freq.and            | -15.9606 | 8.7769     | -1.82   | 0.0746   |
| freq.in             | 18.3245  | 8.2769     | 2.21    | 0.0312   |
| freq.a              | 5.6307   | 9.7073     | 0.58    | 0.5643   |
| freq.be             | 0.7842   | 8.6547     | 0.09    | 0.9281   |
| freq.that           | 9.3265   | 9.9125     | 0.94    | 0.3510   |
| freq.is             | 3.5276   | 10.4993    | 0.34    | 0.7382   |
| freq.which          | 14.0545  | 12.9579    | 1.08    | 0.2830   |
| freq.it             | -6.8455  | 15.5692    | -0.44   | 0.6620   |
| freq.by             | -28.6133 | 13.5325    | -2.11   | 0.0392   |

Ah, now we have a respectable R-squared of 0.72. Let’s see how well it predicts.
## Other models

|                        | Estimate | Std. Error | t value | Pr(>|t|) |
|------------------------|----------|------------|---------|----------|
| (Intercept)            | 0.3907   | 0.6371     | 0.61    | 0.5420   |
| freq.the               | -13.9578 | 3.1257     | -4.47   | 0.0000   |
| freq.of                | 22.0047  | 5.8730     | 3.75    | 0.0004   |
| freq.to                | 29.4070  | 6.3981     | 4.60    | 0.0000   |
| freq.and               | -24.1498 | 6.9560     | -3.47   | 0.0010   |
| freq.by                | -33.6095 | 12.7982    | -2.63   | 0.0109   |

With fewer variables the R-squared is a bit lower: 0.68.

Adding a lot of variables improves it:

|                        | Estimate   | Std. Error | t value | Pr(>|t|) |
|------------------------|------------|------------|---------|----------|
| (Intercept)            | -25.0200   | 14.7096    | -1.70   | 0.0952   |
| freq.the               | -146.7136  | 74.9299    | -1.96   | 0.0558   |
| freq.of                | -167.2712  | 186.2622   | -0.90   | 0.3735   |
| freq.to                | -16.0921   | 98.4739    | -0.16   | 0.8709   |
| freq.and               | -68.8252   | 90.4418    | -0.76   | 0.4502   |
| freq.by                | -124.1622  | 78.0896    | -1.59   | 0.1181   |
| I(sqrt(freq.the))      | 76.2059    | 42.5062    | 1.79    | 0.0791   |
| I(sqrt(freq.of))       | 92.5611    | 90.5097    | 1.02    | 0.3114   |
| I(sqrt(freq.to))       | 14.5797    | 36.2043    | 0.40    | 0.6889   |
| I(sqrt(freq.and))      | 18.0623    | 27.1489    | 0.67    | 0.5089   |
| I(sqrt(freq.by))       | 17.4639    | 12.5061    | 1.40    | 0.1688   |
| num.words              | -0.0034    | 0.0037     | -0.90   | 0.3707   |
| I(num.words^2)         | 0.0000     | 0.0000     | 0.17    | 0.8656   |
| num.char               | 0.0006     | 0.0007     | 0.83    | 0.4079   |
| I(num.char^2)          | 0.0000     | 0.0000     | 0.08    | 0.9341   |
| I(num.words * num.char)| -0.0000    | 0.0000     | -0.13   | 0.9000   |

R-squared is 0.77. Let’s see how well it predicts.