Using R to Compare Two Groups

Michael A. Covington
Institute for Artificial Intelligence
The University of Georgia

2011 December 9

Introduction

In what follows I will demonstrate statistical analysis of an experiment that compares two groups of texts, using Excel to edit and prepare the data and R to analyze it.

My "Using R" tutorials partly repeat each other's content so that each one will be complete by itself.

About R

R is freeware, downloadable from www.r-project.org . It is good for calculation, matrix arithmetic, statistics, and various kinds of machine learning.

Suggested reading: Peter Dalgaard (2002) *Introductory Statistics with R*. Berlin: Springer.

R is derived from an earlier language called S. To learn more about its advanced statistical functions see Venables and Ripley, *Modern Applied Statistics with S* (which includes R). To learn more about R as a programming language, see Braun and Murdoch, *A First Course in Statistical Programming with R*.

R will mystify you until you learn something about its data types and syntax. The data types are:

Numbers
Strings
Vectors (1-dimensional arrays of numbers or strings)
Factors (vectors of discrete named values, like an array of enums in C)
Lists (sequences whose components are named, not numbered)

Matrices (2-dimensional arrays)
Arrays (with any number of dimensions)
Data frames (2-dimensional arrays with named columns, for statistical data)

The syntax of R is unusual. Note that:

There are no semicolons between statements.
Vectors and arrays are indexed from 1, not 0.
All names are case-sensitive.
`<-` is assignment
`.` within a name is just an ordinary character
`$` picks out parts of lists or frames (such as **`a$b`**, which means the part of **`a`** named **`b`**)
`#` makes the rest of the line a comment
`%` marks special operators, such as **`%/%`** (integer division), **`%*%`** (matrix multiplication)

Vectors (sequences) have to be created, usually with the function **`c()`**.

Compare:	In Prolog:	**`X = [1,2,3]`**
	In R:	**`X <- c(1,2,3)`**

The help system is accessed with commands such as **`help(t.test)`** (for finding out about the function named **`t.test`**).

About data files

Excel's own file formats, .xls and .xlsx, are generally not understood by other software. Instead, we exchange spreadsheets with other software by using two other formats, **tab-delimited text** and **comma-separated values (CSV)**. These are text files that use, respectively, tab characters (Unicode 0009) and commas to separate the columns.

R reads tab-delimited text with the function **`read.delim()`** and CSV with **`read.csv()`**. In each case what you get is a data frame with the first line used as column labels.

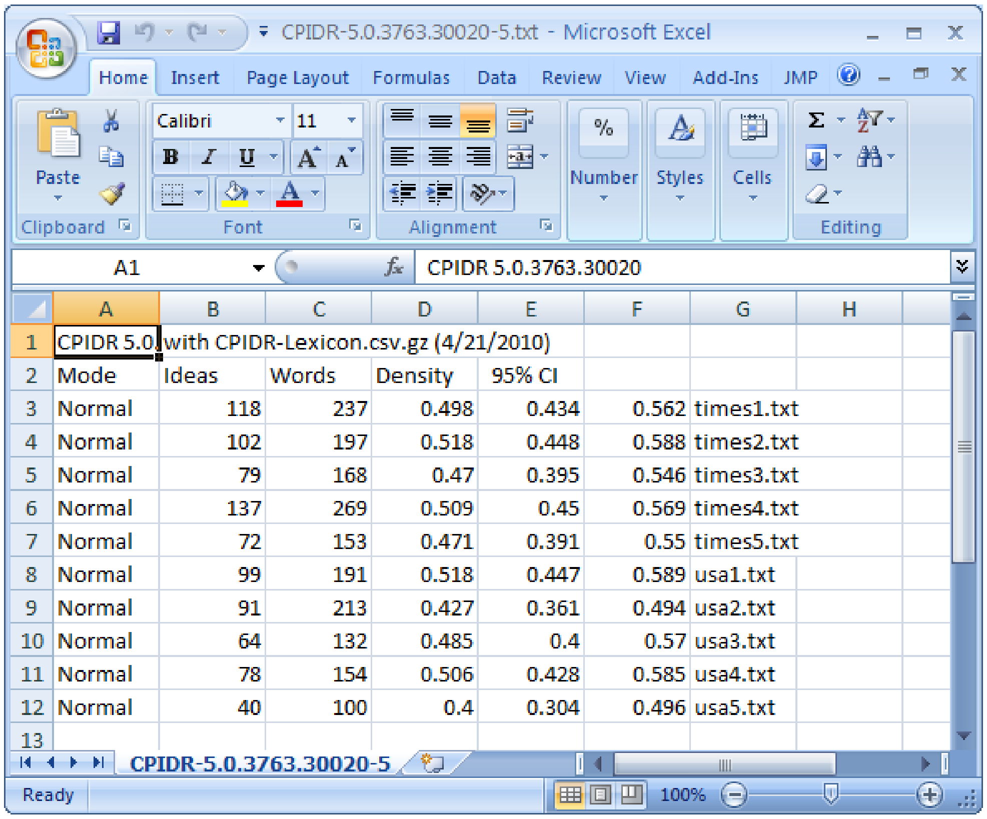
A sample experiment

I wanted to find out whether news stories in *USA Today* or *The New York Times* have higher idea density. To find out, I gathered the first 5 stories on each newspaper's web site, pasted them into text files, cleaned them up (removing fragments of advertisements and other irrelevant material), and ran them through CPIDR to measure the idea density.

A sample of 5 stories from each newspaper is not big enough; as we will see, the statistical analysis will tell us so. But it's interesting as an example.

Editing the data file

CPIDR produced a tab-delimited text file. I opened it with Excel, and this is what I saw:


CPIDR-5.0.3763.30020-5.txt - Microsoft Excel
Home Insert Page Layout Formulas Data Review View Add-Ins JMP
Paste Clipboard Calibri 11 Font Alignment Number Styles Cells Editing
A1
CPIDR-5.0.3763.30020-5
Ready 100%

	A	B	C	D	E	F	G	H
1	CPIDR 5.0.3763.30020 with CPIDR-Lexicon.csv.gz (4/21/2010)							
2	Mode	Ideas	Words	Density	95% CI			
3	Normal	118	237	0.498	0.434	0.562	times1.txt	
4	Normal	102	197	0.518	0.448	0.588	times2.txt	
5	Normal	79	168	0.47	0.395	0.546	times3.txt	
6	Normal	137	269	0.509	0.45	0.569	times4.txt	
7	Normal	72	153	0.471	0.391	0.55	times5.txt	
8	Normal	99	191	0.518	0.447	0.589	usa1.txt	
9	Normal	91	213	0.427	0.361	0.494	usa2.txt	
10	Normal	64	132	0.485	0.4	0.57	usa3.txt	
11	Normal	78	154	0.506	0.428	0.585	usa4.txt	
12	Normal	40	100	0.4	0.304	0.496	usa5.txt	
13								

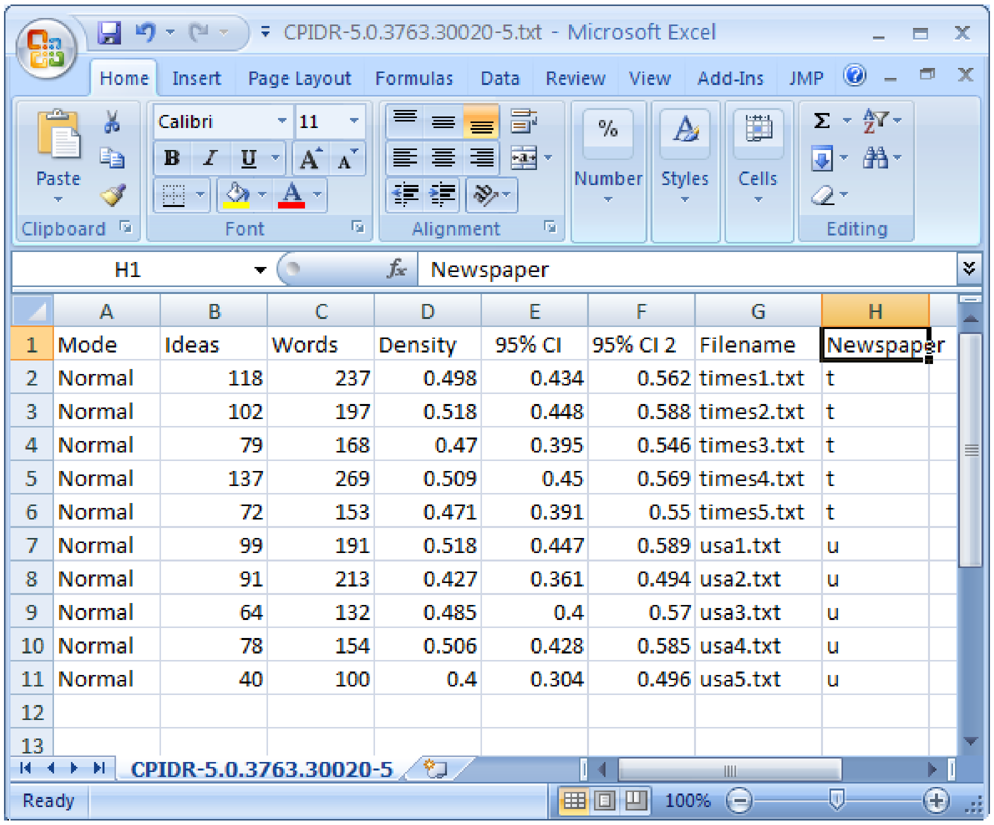
Three things are wrong:

(1) The column labels are not in the first row. I'll delete row 1.

(2) There is not a column that says which newspaper each story came from. I'll add one.

(3) Not all of the columns have labels. I'll add some.

After making these changes, here's what I have:



	A	B	C	D	E	F	G	H
1	Mode	Ideas	Words	Density	95% CI	95% CI 2	Filename	Newspaper
2	Normal	118	237	0.498	0.434	0.562	times1.txt	t
3	Normal	102	197	0.518	0.448	0.588	times2.txt	t
4	Normal	79	168	0.47	0.395	0.546	times3.txt	t
5	Normal	137	269	0.509	0.45	0.569	times4.txt	t
6	Normal	72	153	0.471	0.391	0.55	times5.txt	t
7	Normal	99	191	0.518	0.447	0.589	usa1.txt	u
8	Normal	91	213	0.427	0.361	0.494	usa2.txt	u
9	Normal	64	132	0.485	0.4	0.57	usa3.txt	u
10	Normal	78	154	0.506	0.428	0.585	usa4.txt	u
11	Normal	40	100	0.4	0.304	0.496	usa5.txt	u
12								
13								

Note that I used character strings ("u" and "t") to identify the newspapers, rather than numbers. This will cause R to interpret that column as a **factor**, i.e., a series of discrete labels that can be used to divide data up into groups. (This is what other statistics packages call *categorical data*. It is unrelated to "factors" in factor analysis.)

Now I'll save the data **onto a tab-delimited file, but *not* the original one**. (For research integrity, I always want to keep the original data file that came out of CPIDR!) The new file will be called **`newspapers.txt`**.

The research question

We are interested in the columns called **Density** and **Newspaper**. The question is, if we split the values in **Density** into two groups by **Newspaper**, are the two groups alike?

Loading the data into R

This is how to read the whole data set into R and store it in a variable called **d**:

```
> d <- read.delim("c:\\Users\\mc\\Desktop\\newspapers.txt")
```

Note the doubled backslashes. If you want R to put up an open file dialog box and let you pick a file, you can do it in this somewhat clumsy way:

```
> require(tcltk)    # do this just once, to bring in the tcltk library
> d <- read.delim(tclvalue(tkgetOpenFile()))
```

To confirm that it was read in correctly, we can immediately display it:

```
> d
     Mode Ideas Words Density X95..CI X95..CI.2   Filename Newspaper
1  Normal   118   237   0.498   0.434     0.562 times1.txt         t
2  Normal   102   197   0.518   0.448     0.588 times2.txt         t
3  Normal    79   168   0.470   0.395     0.546 times3.txt         t
4  Normal   137   269   0.509   0.450     0.569 times4.txt         t
5  Normal    72   153   0.471   0.391     0.550 times5.txt         t
6  Normal    99   191   0.518   0.447     0.589   usa1.txt         u
7  Normal    91   213   0.427   0.361     0.494   usa2.txt         u
8  Normal    64   132   0.485   0.400     0.570   usa3.txt         u
9  Normal    78   154   0.506   0.428     0.585   usa4.txt         u
10 Normal    40   100   0.400   0.304     0.496   usa5.txt         u
```

Note that two of the column labels have been changed a bit in order to make them valid names in R.

The columns we are interested in are **Density** and **Newspaper**.

We can look at the individual columns as vectors (sequences of values):

```
> d$Density
 [1] 0.498 0.518 0.470 0.509 0.471 0.518 0.427 0.485 0.506 0.400
> d$Newspaper
 [1] t t t t t u u u u u
```

We can split **Density** into a pair of vectors according to **Newspaper**:

```
> s <- split(d$Density,d$Newspaper)
> s
$t
[1] 0.498 0.518 0.470 0.509 0.471
$u
[1] 0.518 0.427 0.485 0.506 0.400
```

Here **s** is a list with two elements: **s$t** is the vector of densities for the *Times* and **s$u** is the vector of densities for *USA Today*.

We can ask for summary statistics on these:

```
> summary(s$t)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.4700  0.4710  0.4980  0.4932  0.5090  0.5180 
> summary(s$u)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.4000  0.4270  0.4850  0.4672  0.5060  0.5180 
```

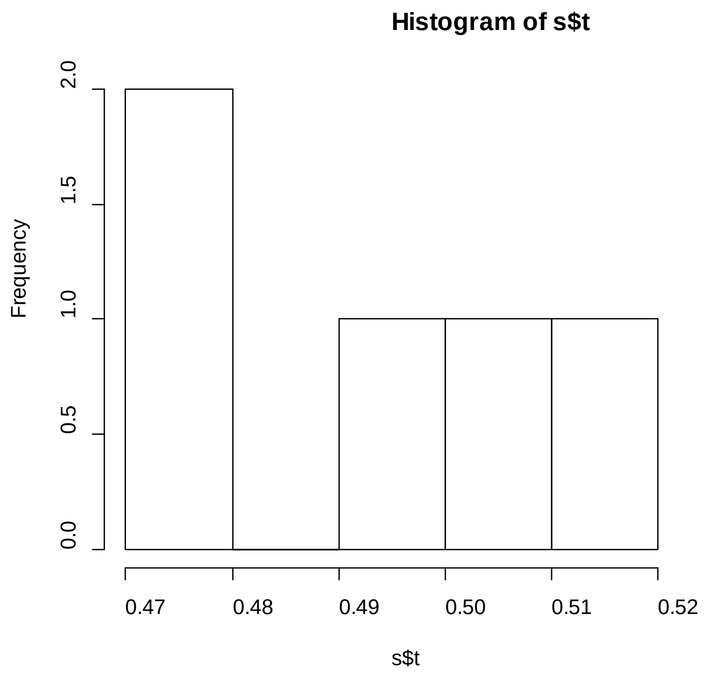
We see that the *Times* has a higher mean density than *USA Today*. But is the difference enough to show anything? Let's explore further.

Histograms

We can plot a histogram of one vector like this:

```
> hist(s$t)
```

Here's what pops up in the graphics window:



Nice, but we'd like to compare the two vectors, and right now, if we plot another histogram, the first one disappears. What's more, we'd like to display histograms of both vectors with the *same* horizontal and vertical scale. Doing this requires four commands:

```
> b <- c(0.4,0.425,0.45,0.475,0.5,0.525,0.55,0.575,0.6)
> par(mfrow=c(2,1))
> hist(s$t,breaks=b,xlim=c(0.4,0.6),ylim=c(0,3))
> hist(s$u,breaks=b,xlim=c(0.4,0.6),ylim=c(0,3))
```

We're using the important function **`c()`** to create vectors (sequences).

First we make a list of where the histogram columns should break, and we store it in the variable **`b`**. Ordinarily, R would choose the breaks for us, but it might not choose the same breaks on different histograms. Instead of

```
> b <- c(0.4,0.425,0.45,0.475,0.5,0.525,0.55,0.575,0.6)
```

we could have written

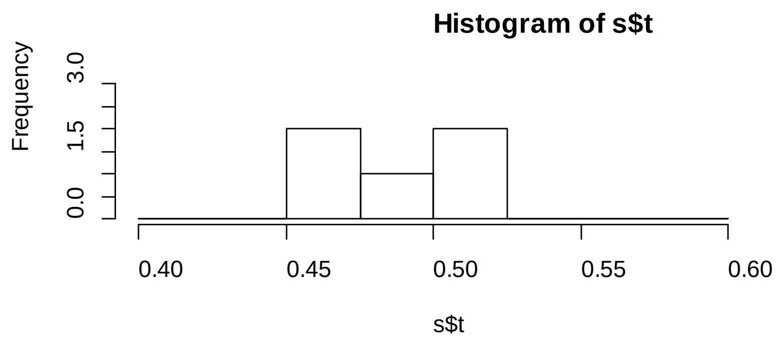
```
> b <- seq(0.4,0.6,0.125)
```

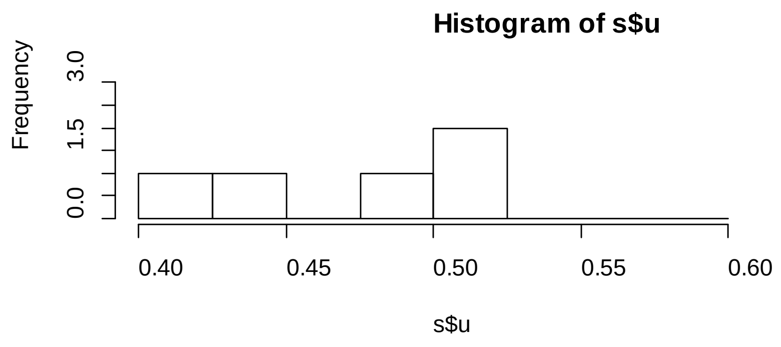
which means "give me a sequence from 0.4 to 0.6 stepping by 0.125."

The **`xlim`** and **`ylim`** parameters of course say "the x-axis goes from 0.4 to 0.6 and the y-axis goes from 0 to 3." Here's **`c()`** again, creating each of them as 2-element vector.

Then we plot two histograms with the same list of breaks and the same vertical and horizontal limits (again, vectors created with **`c()`**).

Here's the result:





Look at what's going on – both newspapers have the same maximum idea density, but *USA Today* ranges much lower (below 0.45). That's an important clue.

Significance testing

Can we conclude that the two newspapers are different? To find out, let's do a *t*-test.

```
> t.test(s$t,s$u)

        Welch Two Sample t-test

data:  s$t and s$u
t = 1.0419, df = 5.41, p-value = 0.3417
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.03670918  0.08870918
sample estimates:
mean of x mean of y
   0.4932    0.4672
```

Here we are testing the hypothesis that **s$t** and **s$u** have the same mean, i.e., that the difference between their means is 0.

The test gives us a p-value of 0.3417. That means there's a 34% chance we'd see this much difference, or more, in the sample means even if there were no difference in the real means (of the whole newspapers). We want a p-value less than 0.05 in order to draw a conclusion. So: No, we cannot conclude (from this sample) that the newspapers are different.

The test also tells us that we can be 95% sure that the difference of means is between -0.037 and +0.089. That's a wide range! A smaller sample would give a smaller confidence interval.

We conclude either of two things: **(1) there is no real difference between the newspapers, or (2) we need to try a larger sample in order to detect the difference.** The histograms of the two newspapers looked different enough to make me think (2) is the right choice.

But wait a minute… Was that the right test?

If we had known in advance that one particular newspaper *had* to have lower idea density than the other one, we could have done a one-tailed *t*-test and gotten an even lower *p*-value. But in this case that is not appropriate.

The *t*-test assumes that the two populations have bell-shaped (normal) distributions and, in most statistical packages, assumes that have the same variance. Looking at those histograms, we can't justify either of those assumptions. It looks like *USA Today* has considerably more variance.

So, for starters, let's change the *t*-test so it doesn't assume equal variances… Surprise! The Welch test (unequal variances) is already the default for R. Type **help(t.test)** to learn more about this.

If we don't want to assume bell-shaped distributions either, we need a Wilcoxon test instead of a *t*-test:

```
> wilcox.test(s$t,s$u)

        Wilcoxon rank sum test with continuity correction

data:  s$t and s$u
W = 15.5, p-value = 0.6004
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(s$t, s$u) : cannot compute exact p-value with ties
```

This comes out even worse, with $p = 0.6004$. We also get a warning that the Wilcoxon test could not distinguish all the values – it assumes that no value ever recurs in the data, and in our case there are some duplicates.

Conclusion: We need a bigger sample.

Skipping some steps

The t-test and Wilcoxon test could have been done on the original data frame **d**, without splitting it in advance. Here's how:

```
> t.test(d$Density ~ d$Newspaper)
> wilcox.test(d$Density ~ d$Newspaper)
```

Here **~** means "divided up according to a factor."

But then we wouldn't have had all that fun with histograms. The **hist()** function does not know how to divide up data in this way, so we had to divide up the data (and the graphical display) for it.

If you find yourself needing to do all of this frequently, you can of course write your own R function (program) to do it.