A crosstab and a test for independence

Ben Klemens

1 August 2013

In this commentary on the Amash amendment[1] ("To end authority for the blanket collection of records under the Patriot Act."), Josh Tauberer disputes the claim that "defense cash was a better predictor of a member's vote on the Amash amendment than party affiliation."

Next time, we'll try some simple regressions to more directly address the question, but with the intent of starting simple, this time I'll cover a relatively simple null hypothesis: $H_P^0 =$ *the congressperson's party affiliation and his/her vote on this amendment are independent*. To give away the not-surprising ending, this was indeed a partisan vote.

Hypothesis H_P is especially easy to work with because it is about two binary categories. The claim of independence means that then P(Republican and vote Aye) = P(Republican) · P(vote Aye), and similarly for P(Democrat and vote Aye), P(Republican and vote Nay), P(Democrat and vote Nay).

This claim neatly fits the form of two common tests of independence. The first is the χ^2 test. [Pronounced 'chi squared', so named because the test statistic theoretically has a χ^2 distribution, so named because a 1900 paper by Pearson used a Taylor expansion where the terms where named χ, χ^2, χ^3, ..., and showed that only the second term was important in the given context.] As per the name, this test is based on a distribution that holds perfectly as $N \to \infty$. The second test is named the Fisher exact test [after a famous eugenicist, RA Fisher], and is based on explicitly counting the options and their odds under a claim that vote and party affiliation are independent.

The crosstab Simply printing the crosstab does show a clear enough correlation between vote and party affiliation that we can expect the Null will be rejected. Let's look at the crosstab now.

The default form for a data set is that each row is an observation, and each column is some meaningful variable, whose units may or may match those of the variable in the next column over. That is, the x axis of the matrix and the y-axis are entirely asymmetric. A *crosstab* is about a single variable, typically a count, and the x-axis represents one categorization (like *voted Aye* and *voted Nay*) and the y-axis another categorization (like *Democrat* or *Republican*).

So, we'll read in the data and generate a crosstab. I'll print the output first (cleaned slightly, and with a title that I just added to Apophenia's code base):

[1] `http://razor.occams.info/blog/2013/07/27/defense-dollars-arent-a-better-predictor-of-the-amash-v`

```
#include <apop.h>

int main(){
    int readin_status = apop_text_to_db("amash_vote_analysis.csv", .tabname="amash");
    Apop_stopif( readin_status== -1, exit(1), 0, "Trouble reading in the data. "
                        "Have you downloaded it to this directory?");

    apop_query("create table pv_xtab as "
                " select party, vote, count(*) as ct "
                " from amash "
                " group by party, vote ");

    apop_data *xtab = apop_db_to_crosstab("pv_xtab", "party", "vote", "ct");
    apop_data_show(xtab);
    apop_data_show(apop_test_fisher_exact(xtab));
}
```

```
[          ] Aye             No
   Democrat 111              83
 Republican  94             134

        Fisher Exact test

 probability of table 0.000366298
              p value  0.00125049
```

We reject the null that vote and party are independent with 99.874% certainty. This is not surprising, but the fact that the crosstab looks sane and a simple test gave us the simple result we were expecting gives us confidence that the data is what it says it is and that we haven't (yet) overlooked any blatant flaws.

Here's the program to generate this output, and some subsequent commentary. As usual, this could be reduced to two lines of code, but fewer lines of code typically means less readable and less extensible. Also, as usual, most of the work is in getting the data in the right place. Once it's in the right form, the modeling and testing is a line or two.

How it works:

- Mr Tauberer posted the data at this location[2] (CSV file hotlinked to his site), based on a query from his govtrack.org[3] website. Save it to the directory where you're working (on Linux, Mac, or Cygwin, `wget` will do this from the command line).
- `apop_text_to_db` does what it says. We didn't open an on-disk database using `apop_db_open`, so this will create a table named `amash` in an in-memory

[2] `http://razor.occams.info/pubdocs/amash_vote_analysis.csv`
[3] `govtrack.org`

database.

- That call is followed by an `Apop_stopif` macro, which is intended to take action on errors. I'm not nearly paranoid enough to wrap every function call in such a wrapper, but reading in a text file is one of those things that fails with great frequency.
- SQL is the right place to do data shunting. Here, we create a spare table with a count of how many people are in each party/vote combination. The documentation for `apop_db_to_crosstab` suggests some ways to avoid making this spare table, but doing so makes clear what's going on. The spare table isn't quite a crosstab yet; you could view it with, e.g.

  apop_data_print(apop_query_to_text("select * from pv_xtab"));

  and you'll see that each row is an observation listing party, vote, count, so we need one more step to turn it into an XY-symmetric sort of table.
- `apop_db_to_crosstab` does this conversion. We save the result to `xtab`.
- Finally, run `apop_test_fisher_exact` on the resulting table.

Are defense contributions independent of vote? Of course not. The program is very similar; merging this script and the above is left as an exercise for the reader.

Following the Tauberer analysis, we split contributions down the middle, between high and low contributions, which brings us back to the binary case.

Once again, we'll use SQL to do the data prep, which in this case is a little awkward because we need the median, which is not a built-in SQL function. But it's easy to calculate: if you select a sorted list of contributions, but limit its length to half the total count of rows, then the max of that sorted list is the median.

By the way, the Mehta & Patel code (the core of the Fisher Exact test; see below) works for an $N \times M$ matrix for any N, M, not just 2×2. The median-calculating method could be used, probably with a C-side `for` loop, to split the congresspack into thirds or quartiles.

In code, the median is first calculated and saved to a C-side scalar with `apop_query_to_float`, and then the next query uses a `printf`-style substitution to make use of that number.

After that calculation, the rest of the program is the same story.

The output also looks pretty similar:

```
[] Aye     No
 0 129     85
 1  76    132
        Fisher Exact test

 probability of table 4.96527e-07
              p value 1.09589e-06
```

Yes, the p-value is 1e-6 instead of 1e-3, but comparing p-values across tests is basically meaningless.

```
#include <apop.h>

int main(){
    int readin_status = apop_text_to_db("amash_vote_analysis.csv", .tabname="amash");
    Apop_stopif( readin_status== -1, exit(1), 0, "Trouble reading in the data. "
                        "Have you downloaded it to this directory?");

    double median_contrib = apop_query_to_float("select max(contribs) from "
                                " (select contribs from amash order by contribs "
                                " limit (select count(*)/2 from amash))");
    apop_query("create table pv_xtab as "
                " select vote, (contribs+0.0 > %g) as hi_money, count(*) as ct "
                " from amash "
                " group by vote, hi_money ", median_contrib);

    apop_data *xtab = apop_db_to_crosstab("pv_xtab", "hi_money", "vote", "ct");
    apop_data_show(xtab);
    apop_data_show(apop_test_fisher_exact(xtab));
}
```

Next time, a logit and a probit, more directly addressing the original question regarding the Amash vote.

Scaling the test I'll conclude with a digression about the Fisher Exact test itself.

The code to evaluate the FExact test hasn't really changed: as far as I can tell, everybody uses the same code written in 1993 by Cyrus R. Mehta and Nitin R. Patel[4], perhaps modified by running it through an automated FORTRAN-to-C converter. I have received multiple emails from people who wanted to test the limits and try very large integers, like $N = 100,000$. [Evidently, R throws away the workspace overflow errors that Mehta & Patel's algorithm emits and returns in-expected-range but garbage values, which gave these readers false hope that this should work without modification.]

But both the Fisher Exact test and the χ^2 test are very likely to reject the claim of no correlation for real-world data with N in the thousands. For a one-dimensional bell curve, the standard error $\sigma \propto 1/\sqrt{N}$, so if $N = 10,000$, then $\sigma \propto \mu/100$—if your observed mean μ_{obs} is more than about $3\mu_{obs}/100$ away from μ_{obs}, then a typical one-dimensional hypothesis test will reject the claim that $\mu_{obs} = \mu$. That intuition carries over to the χ^2 and Fisher Exact tests: for real-world data with N in the thousands, don't even bother.

But don't take my word for it, here's some code, which fills a 2×2 matrix with a few values, and then re-fills it with the same values multiplied by 100. In eight days, I'll show how to simplify even this:

In both cases, these are independent except for 2.4% of the observations—shift the first column to 10 and 25 (or 1100 and 2500) and the row and column are fully

[4] https://dl.acm.org/citation.cfm?doid=6497.214326

```
#include <apop.h>

int main(){
    apop_data *d = apop_data_alloc(2,2);
    apop_data_fill(d,
            11, 14,
            24, 35);

    apop_data_show(apop_test_fisher_exact(d));

    apop_data_fill(d,
            1100, 1400,
            2400, 3500);

    printf("\n Scaled 100X:\n");
    apop_data_show(apop_test_fisher_exact(d));
}
```

independent. Here's the output, showing how the scaling affects the p-value and the probability the table could happen at random given the independence assumption:

```
Fisher Exact test

 probability of table 0.182861
              p value 0.812263

 Scaled X100:
        Fisher Exact test

 probability of table 0.00036128
              p value 0.00498963
```