Interrater agreement

Ben Klemens

22 September 2013

[This and the next entry are blog-level discussions of this Journal of Official Statistics paper[1].]

I first met Cohen's Kappa maybe three years ago. I did a joint project with a friendly coworker who had spent decades dealing with the sort of data that requires such measures. After some amused discussion of our different paths and how I could spend years dealing with data and never meet it, yet she had to use agreement measures on every project, she apologized. Cohen's Kappa and the many related measures (Fleiss's Kappa, Scott's Pi, Krippendorff's Alpha, . . .) don't make a lot of sense, she told me, and don't seem to have much theoretical basis, but it's what everybody uses.

The basic problem: you have two sequences of observations about the same data. The typical situation is that there is non-categorical data, like free-response questions on a survey or a set of chest X-rays, and two or more raters independently decide into which category each observation falls, like a check-box summary or a diagnosis. We'd like to know how often the raters are in agreement.

If there are two categories, two raters flipping coins would agree 50% of the time. Given two categories that have an expected 90-10 split, we expect raters to be in agreement 82% of the time. Given 100 equiprobable categories, the expected agreement entirely at random is 1%. So it seems inequitable to compare raw categories, and it seems we should have some sort of 'complexity adjustment' to the raw observed rate of agreement to take into account how much room there is for agreement over the at-random case.

From here, complications ensue. Below, I'll discuss the difference in baseline calculations between Cohen's Kappa and Scott's Pi (and define both at the same time). Or what if the categories are ordered; should a near-miss count for something? What if some categories are really hard to identify and others are very easy to identify? And, for that matter, why are we taking two people drawing values at random as a baseline at all?

That's the state of the literature: everybody has a different baseline concept of randomness, which implies that everybody has a different measure of the distance to randomness. For example, this paper[2] (PDF), sent to me by Dr MH of SF, CA, characterizes the entire interrater agreement literature as a big mess, with too many answers to the same question.

[1]`http://www.jos.nu/Articles/abstract.asp?article=283395`
[2]`http://business.illinois.edu/shavitt/BA_531/Grayson-week14.pdf`

The math The calculations for these things all begin by generating what I've been referring to as a crosstab, but many in this literature call the *confusion matrix*, (from the Latin prefix *con-*, meaning *with*, melded with the root word *fusion*). Cell (1,1) is the percent of times both raters chose category one, (1, 2) is the percent of times the row rater chose 1 and the column rater chose 2, and so on.

I'll be assuming a normalized confusion matrix from here on in, where the grand total of all cells is exactly one. That means that having large or small N is irrelevant to the measurements.

The basic formula for both Cohen's Kappa and Scott's Pi is

$$\frac{P_o - P_e}{1 - P_e}.$$

Here, P_o is the observed percent agreement—the sum of the main diagonal of the confusion matrix. Cohen finds the expected percent agreement for category one by multiplying the percent of times the row rater chose category one by the percent of times the column rater chose category one (i.e., the total weight in row one times the total weight in column one). Then P_e is the sum of the percent agreement for every category. For Scott, we find the agreement for category one by finding the percent of all classifications (both for row and column) that went to category one, and squaring that.

That is, for Cohen, the row rater and column rater are distinct random number generators, with distinct odds of picking category one, and the at-random odds of agreement is the product of the two. For Scott, the row rater and column rater are both instances of a generic rater, and our best estimate of the generic rater's odds of picking category one is from pooling both row and column categorizations together; the at-random odds of agreement is then the square of the odds the generic rater picks a category.

So we already have two plausible models of the at-random null hypothesis. I have yet to see an argument advocating for one over the other that did not reduce to *this one just seems more plausible to me than the other one*. Within the two, the literature I have read seems to have a preference for Cohen's Kappa. I suspect this is because Cohen's Kappa is always larger than Scott's Pi, so you look like you have better raters using it. [Proving this is left as an exercise for the reader.]

The intuition for both measures largely works out: if there is full agreement, then $P_o = 1$ and both measures are $(1 - P_e)/(1 - P_e) = 1$ (given raters that used more than one category). If agreement at random is more likely, so P_e is larger, then the same P_o produces a smaller measure, so we are successfully complexity-adjusting P_o. If there is worse-than-random agreement, then these measures are negative, which is a little awkward; the usual advice in the literature (and I agree) is that if you get a negative Kappa, then your protocol is seriously broken and you should focus on why your raters can't agree rather than fretting over the exact value of a statistic.

The sample code Below is code calculating both Scott's Pi and Cohen's Kappa. I'll use the Amash amendment data again, which I first used and described in this earlier entry (entry #158). Here, the 'raters' are (1) votes cast on the amendment, and (2) the party affiliation of the voter. A claim that this was a party vote is a claim that the agreement between these two ratings should be high.

When you run this, you'll see that the rate of agreement between the first and second rating schemes is not all that much further from the coin-flip odds of 50%, so all of the rating schemes give the pair of rating systems a low agreement score.

What about campaign contributions by defense industry members? The second half of `main` divides voters into high- and low-contribution recipients. To improve the comparison with the above, we'd like a division that produces roughly the same bin counts as the split by party did. So the SQL first counts the Democrats (there are 194), and then finds a contribution cutoff that would produce 194 low-contribution voters. [A complication: the correct contribution cutoff is $13,000, but there are four people who got that level of contributions, so there's no way to get a perfect match between counts. To two decimal places, it makes no difference in the statistics into which bin we put these four voters.]

Then, we try the same measures using these two systems of categorization. It turns out that the campaign contributions give a marginally greater agreement rate on all measures calculated, though it is not enough to be a significant difference. [To get variances for these measures, by the way, I recommend bootstrapping.]

Next time, I'll present the alternative measure of intecoder agreement that I developed based on entropy measures. As a preview, it's already calculated in the example code (as P_I).

```
#include ”169−kappa_and_pi.h”

apop_data *kappa_and_pi(apop_data const *tab_in){
    apop_data *out = apop_data_alloc();
    Apop_stopif(!tab_in, out−>error=’n’; return out, 0, ”NULL input. Returning output with
          ’n’ error code.”);
    Apop_stopif(!tab_in−>matrix, out−>error=’m’; return out, 0, ”NULL input matrix.
          Returning output with ’m’ error code.”);
    Apop_stopif(tab_in−>matrix−>size1 != tab_in−>matrix−>size2, out−>error=’s’;
          return out, 0, ”Input rows=%zu; input cols=%zu; ”
                      ”these need to be equal. Returning output with error code ’s’.”, tab_in
                          −>matrix−>size1, tab_in−>matrix−>size2);

    apop_data *tab = apop_data_copy(tab_in);
    double total = apop_matrix_sum(tab−>matrix);
    gsl_matrix_scale(tab−>matrix, 1./total);
    double p_o = 0, p_e = 0, scott_pe = 0, ia = 0, row_ent = 0, col_ent = 0;
    for (int c=0; c< tab−>matrix−>size1; c++){
        double this_obs = apop_data_get(tab, c, c);
        p_o += this_obs;

        Apop_matrix_row(tab−>matrix, c, row);
        Apop_matrix_col(tab−>matrix, c, col);
        double rsum = apop_sum(row);
        double csum = apop_sum(col);
        p_e += rsum * csum;
        scott_pe += pow((rsum+csum)/2, 2);

        ia += this_obs * log2(this_obs/(rsum * csum));
        row_ent −= rsum * log2(rsum);
        col_ent −= csum * log2(csum);
    }
    apop_data_free(tab);

    asprintf(&out−>names−>title, ”Scott’s  and Cohen’s ”);
    apop_data_add_named_elmt(out, ”total count”, total);
    apop_data_add_named_elmt(out, ”percent agreement”, p_o);
    apop_data_add_named_elmt(out, ””, ((p_e==1)? 0: (p_o − p_e) / (1−p_e) ));
    apop_data_add_named_elmt(out, ””, ((p_e==1)? 0: (p_o − scott_pe) / (1−scott_pe)));
    apop_data_add_named_elmt(out, ”P_I”, ia/((row_ent+col_ent)/2));
    apop_data_add_named_elmt(out, ”Cohen’s p_e”, p_e);
    apop_data_add_named_elmt(out, ”Scott’s p_e”, scott_pe);
    apop_data_add_named_elmt(out, ”information in agreement”, ia);
    apop_data_add_named_elmt(out, ”row entropy”, row_ent);
    apop_data_add_named_elmt(out, ”column entropy”, col_ent);
    return out;
}
```

```
#include "169-kappa_and_pi.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 summed as select vote, party, count(*) as ct from amash group by
        vote, party");
    apop_data *confusion = apop_db_to_crosstab("summed", "vote", "party", "ct");
    apop_data_show(confusion);
    apop_data_show(kappa_and_pi(confusion));

//Find the contribution level that matches the count of democrats;
//use that cutoff to produce a binary small-contrib|big-contrib categorization.
//There are 194 dems. The contribution cutoff is thus $13,000. There are four
    double dem_count = apop_query_to_float("select count(*) from amash where party='
        Democrat'");
    apop_query("create table contrib_sums as select vote, "
            "contribs> (select max(contribs) from "
                    "(select contribs from amash order by contribs limit %g)) "
            "as big_dollars, "
            "count(*) as ct from amash group by vote, big_dollars", dem_count);

    confusion = apop_db_to_crosstab("contrib_sums", "vote", "big_dollars", "ct");
    apop_data_show(confusion);
    apop_data_show(kappa_and_pi(confusion));
}
```