Paper 1825-2014

Calculate All Kappa Statistics in One Step

Matthew Duchnowski, Educational Testing Service

ABSTRACT

Cohen's kappa statistic has enjoyed a growing popularity in the social sciences as a way of evaluating rater agreement on a common scale.

Currently, SAS® users can produce the kappa statistic of their choice through PROC FREQ and the use of relevant AGREE options. Complications arise, however, when the data do not possess a completely square cross-tabulation of ratings. That is, this method requires that both raters have at least one data point for every available category on the scale.

Many solutions have been offered for this predicament. Most solutions include inserting dummy records into the data and then assigning a weight of zero to those records through an additional class variable. The result is a multistep macro that often creates extraneous observations, variables and datasets that present potential data integrity issues.

The author offers a more elegant solution by producing a segment of code that uses brute force to calculate Cohen's kappa as well as all popular variants. The code comes in two forms: a multistep approach and a single body of nested PROC SQL statements. By using this method, users may generate kappa statistics of all types – even those that they wish to define for themselves.

INTRODUCTION

One obvious way to measure agreement between two independent raters is to calculate the proportion of records that have identical ratings. This method is slightly flawed because there is bound to be some chance agreement between two independent judges. In 1960, Cohen devised the kappa statistic to tease out this chance agreement by using an adjustment with respect to expected agreements that is based on observed marginal frequencies.

The kappa statistic can be calculated as Cohen first proposed or by using any one of a variety of weighting schemes. The most popular among these are the "linear" weighted kappa and the "quadratic" weighted kappa. The unweighted (or "simple") kappa can be viewed as a weighted kappa that has a trivial weighting scheme.

$$1 - \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij} o_{ij}}{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij} e_{ij}}$$

The formula above yields all kappa statistics where o_{ij} is the observed score frequency and e_{ij} is the expected score frequency for Rater 1 issuing the i^{th} score on the rating scale and Rater 2 issuing the j^{th} score on an identical scale. The weight w_{ij} is applied to the score combination, determined by the weighting scheme being used and by how many scale categories the two ratings differ. Throughout this paper, weights are generated as a function of the absolute difference between two ratings, and so each rating is assumed to be on a numeric integer scale.

STANDARD CODE

As previously mentioned, SAS provides a method for generating agreement statistics, and many times this will work sufficiently.

```
proc freq data = scores;
   tables rater1 * rater2 / nopercent norow nocol expected agree (WT=FC) ;
   output agree out=stats;
   test kappa wtkap ;
run;
```

The code above will output a dataset “stats” containing the simple kappa coefficient, _KAPPA_, as well as a weighted kappa, _WTKAP_. Quadratic weights are specified with the option (WT=FC), as shown above. If that specification is omitted, the weighting scheme used is linear. SAS documentation refers to the linear and quadratic weighting schemes formally as Cicchetti-Allison and Fleiss-Cohen, respectively. Tests of symmetry and confidence intervals are also produced by the AGREE option. Users should refer to SAS documentation if they require such tests.

There are cases, however, when the agreement statistics are not generated; namely, when the data are not “fully crossed.” This condition produces the following output:

```
NOTE: AGREE statistics are computed only for tables where the number of rows equals the number of
      columns. To include zero-weight observations in the analysis, use the ZERO option in the
      WEIGHT statement.
WARNING: No OUTPUT data set is produced because none of the requested statistics are available.
WARNING: Data set WORK.STATS was not replaced because new file is incomplete.
```

Output 1. Log Output from a Failed PROC FREQ Statement

While the kappa statistics may not have been computed, SAS will execute and display the cross-tabulation matrix.

Table 1 shows our cross-tabulation of such a data set.

The SAS System

The FREQ Procedure

Frequency Expected	Table of rater1 by rater2				
	rater2				
rater1	1	2	3	4	Total
1	10 3.6667	1 2.9333	0 1.1	0 3.3	11
2	0 2.3333	6 1.8667	1 0.7	0 2.1	7
4	0 4	1 3.2	2 1.2	9 3.6	12
Total	10	8	3	9	30

Table 1. Two-Way Frequency with Expected Frequencies

For ease of viewing, the only values displayed within the cells are the observed frequency count (on top) and the expected value for each cell according to the marginal probabilities (on bottom). The expected values were requested above by using the EXPECTED option within the TABLE statement.

As we can see from the table, there are no ratings of “3” issued by rater1, and SAS is not able to assume that there is an “empty” marginal distribution at this score point. This prohibits the generation of agreement stats.

One popular way around this issue is to determine which records are needed to make the table square, add these records, and then have SAS disregard those records through the use of a WEIGHT statement and a ZERO option.

This approach is very unsatisfying and also raises data integrity issues. Adding dummy records to a dataset inevitably means that you must manage those records and ensure they are properly labeled and handled. Processing dummy records might also entail making an entire copy of your data set at run time, which further increases processing time. In either case, the solution is sloppy and the potential for human error is increased.

If these concerns do not apply to you, then the following multistep approach might suffice.

A 3-STEP SOLUTION

To calculate our statistic, we will need to capture observed frequencies and expected frequencies like those found in the PROC FREQ table shown above. Luckily, PROC FREQ enables us to capture this output into a dataset “OE”.

```
proc freq data=scores;
   tables rater1*rater2 /
         out=OE (rename=(COUNT=O EXPECTED=E) drop=PERCENT)
         outexpect nopercent sparse;
run;
```

The dataset “OE” contains observed frequencies O and expected frequencies E. The expected frequencies were requested through the OUTEXPECT option and the SPARSE option helped to ensure that all cross-tabulation cells, even empty ones, were considered. This is critical because even cells with no observed frequency almost always have an expected frequency that is nonzero, and therefore need to contribute to our calculation.

Now, weighting schemes can be created by operating on the scores themselves.

```
data OEW;
    set OE;
    if rater1 = rater2 then W_Sim = 0; else W_Sim = 1;
    W_Lin = ABS(rater1-rater2);
    W_Qua = W_Lin**2;
run;
```

The only thing that remains to be done is to apply the mathematical formula stated earlier.

```
proc sql;
    create table stats as
    select
           1- Sum(O*W_Sim)/Sum(E*W_Sim) As Kappa_Simple,
           1- Sum(O*W_Lin)/Sum(E*W_Lin) As Kappa_Linear,
           1- Sum(O*W_Qua)/Sum(E*W_Qua) As Kappa_Quadratic
    from
           OEW;
quit;
```

The “stats” dataset now contains all three kappa statistics for our data. Performing a PROC PRINT on this dataset shows the following result:

Kappa_Simple	Kappa_Linear	Kappa_Quadratic
0.37759	0.46588	0.54631

Table 2. Resulting Kappa Statistics

ONE-STEP SOLUTION

The code in Appendix A combines all of the components of the multistep approach, but in a single nested PROC SQL step. This approach involves combining all of the summands for the kappa formula numerator (NUMPARTS) and the summands for the kappa formula denominator (DENPARTS). All weights are calculated in the same manner discussed in the 3-step approach.

As a bonus, the code segment allows for the processing of records by group. This additional “by” variable could be used if, for example, two raters are judging a group of dancers, but on multiple metrics (e.g. style, agility, and grace).

Let us consider such a dataset, which contains pairs of ratings for three dancers across three metrics.

Dancer	Metric	Rater 1	Rater 2
Laney	Style	3	3
Laney	Agility	2	2
Laney	Grace	3	3
Penny	Style	3	3
Penny	Agility	1	1
Penny	Grace	1	2
Elody	Style	2	1
Elody	Agility	3	3
Elody	Grace	2	2

Table 3. Raw Dancer Ratings

It might be interesting to see how well the two raters agree on particular metrics and so we would choose to produce kappas using the variable “Metric” as the by-group. The code segment in Appendix A would then produce kappa statistics for each metric as shown below.

Metric	Kappa_Simple	Kappa_Linear	Kappa_Quadratic
Agility	1.0	1.00000	1.00000
Grace	0.5	0.57143	0.66667
Style	0.4	0.57143	0.72727

Table 4. Kappa Statistics with By-Grouping

CONCLUSION

Applying the single-step method of producing kappa statistics might not be beneficial in some situations.

First of all, some of the SQL queries contained in the PROC SQL statement make use of Cartesian joins, which might be problematic for large data sets with many score points. It also might be the case that statistical testing of the kappa statistic is required for your application, and those tests are not offered in the approaches described here. Additionally, the one-step solution contains a multitude of nested queries that some programmers might find cumbersome, especially if they have a need to adapt the code regularly.

If any of these cases apply, the author encourages the user to adapt the 3-step approach as needed to produce desired results.

REFERENCES

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational & Psychological Measurement, 20*(1), 37. doi:10.1177/001316446002000104

Fleiss, J. L.; Cohen, J., & Everitt, B. S. (1969). Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin, 72*, 323–327. doi:10.1037/h0028106.

SAS Institute. (2014). *Test and measures of agreement.* Retrieved from http://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm#statug_freq_a0000000665.htm

ACKNOWLEDGMENTS

I would like to thank all friends and colleagues at ETS who are endlessly willing to discuss topics such as the ones discussed here. I would like to especially thank my wife Robin, who provides me with unending support in all that I do.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Matthew Duchnowski
Educational Testing Service
Rosedale Rd, MS 20T
Princeton, NJ 08541
(609) 683-2939
mduchnowski@ets.org

APPENDIX A

```
/*
KAPPA STATISTICS IN ONE STEP

    ds – data set being analyzed
    by - by grouping
    score1 - the first rating
    score2 - the second independent rating
*/

%let ds = scores;
%let by = group;
%let score1 = rater1;
%let score2 = rater2;

proc sql;
   create table stats as

   select &by.,
      1- Sum(NumParts_Sim)/ Sum(DenParts_Sim) As Kappa_Simple,
      1- Sum(NumParts_Lin)/ Sum(DenParts_Lin) As Kappa_Linear,
      1- Sum(NumParts_Qua)/ Sum(DenParts_Qua) As Kappa_Quadratic
   from
    (
          select distinct Observed.&score1., Observed.&score2., Observed.&by.,
            (Observed.O * Weights.W_Sim) as NumParts_Sim,
            (Expected.E * Weights.W_Sim) as DenParts_Sim,
            (Observed.O * Weights.W_Lin) as NumParts_Lin,
            (Expected.E * Weights.W_Lin) as DenParts_Lin,
            (Observed.O * Weights.W_Qua) as NumParts_Qua,
            (Expected.E * Weights.W_Qua) as DenParts_Qua
          from

          /*OBSERVED SCORES*/
          (
                select A.&by., A.&score1., A.&score2., Max(A.O,B.O) as O from (
                      select *, 0 as O
                      from
                      (select distinct &by. from &ds.) As X,
                      (select distinct &score1. from &ds.) As A,
                      (select distinct &score2. from &ds.) As B
                      ) as A

                      left join(
                      select &by., &score1., &score2., (Count(*)) As O from &ds.
                                  group by &by., &score1., &score2.
                      ) as B
                on
                A.&by. = B.&by.
                and
                A.&score1. = B.&score1.
                and
                A.&score2. = B.&score2.
          ) as Observed,
```

```
            /*EXPECTED SCORES*/
            (
            select
                  A.&by., &score1., &score2., ((A.Freq_1 * B.Freq_2)/(C.N_Total)) as E
            from
            (select &by., &score1., Count(*) As Freq_1
                  from &ds. group by &by., &score1.) As A,
            (select &by., &score2., Count(*) As Freq_2
                  from &ds. group by &by., &score2.) As B,
            (select &by., Count(*) As N_Total
                  from &ds. group by &by.) As C
            where
                  A.&by. = B.&by.
                  and
                  A.&by. = C.&by.
            ) as Expected,

            /*WEIGHTS*/
            (
                  select
                        x.&by., &score1., &score2.,
                        case
                              when &score1.=&score2. then 0
                              else 1
                        end as W_Sim,
                        ABS(&score1.-&score2.) as W_Lin,
                        ABS(&score1.-&score2.)**2 as W_Qua
                  from
                      (
                              select distinct &by.
                              from &ds.
                        ) As X,
                        (
                              select &score1., Count(*) As Freq_1
                              from &ds.
                              group by &score1.
                        ) As A,
                        (
                              select &score2., Count(*) As Freq_2
                              from &ds.
                              group by &score2.) As B
            ) as Weights

      where
      Weights.&by.=Observed.&by. and Weights.&by.=Expected.&by.
      and
      Weights.&score1.=Observed.&score1. and Weights.&score1.=Expected.&score1.
      and
      Weights.&score2.=Observed.&score2. and Weights.&score2.=Expected.&score2.
      )

    group by &by.
;
quit;
```