Calculating Chance-corrected Agreement Coefficients (CAC)

Kilem L. Gwet, Ph.D.

2019-08-25

```
library(irrCAC)
```

Abstract

The **irrCAC** is an R package that provides several functions for calculating various chance-corrected agreement coefficients. This package closely follows the general framework of inter-rater reliability assessment presented by Gwet (2014). A similar package was developed for STATA users by Klein (2018).

The functions included in this package can handle 3 types of input data: (1) the contingency table, (2) the distribution of raters by subject and by category, (3) the raw data, which is essentially a plain dataset where each row represents a subject and each column, the ratings associated with one rater. The list of all datasets containined in this package can be listed as follows:

```
  data(package="irrCAC")
```

Computing Agreement Coefficients

Computing agreement Coefficients from Contingency tables

cont3x3abstractors is one of 2 datasets included in this package and that contain rating data from 2 raters organized in the form of a contingency table. The following r script shows how to compute Cohen's kappa, Scott's Pi, Gwet's AC_1, Brennan-Prediger, Krippendorff's alpha, and the percent agreement coefficients from this dataset.

```
  cont3x3abstractors
#>         Ectopic AIU NIU
#> Ectopic      13   0   0
#> AIU           0  20   7
#> NIU           0   4  56
  kappa2.table(cont3x3abstractors)
#>      coeff.name coeff.val   coeff.se     coeff.ci coeff.pval
#> 1 Cohen's Kappa 0.7964094 0.05891072 (0.68,0.913)      0e+00
  scott2.table(cont3x3abstractors)
#>   coeff.name coeff.val   coeff.se      coeff.ci coeff.pval
#> 1 Scott's Pi 0.7962397 0.05905473 (0.679,0.913)      0e+00
  gwet.ac1.table(cont3x3abstractors)
#>   coeff.name coeff.val   coeff.se      coeff.ci coeff.pval
#> 1 Gwet's AC1 0.8493305 0.04321747 (0.764,0.935)      0e+00
  bp2.table(cont3x3abstractors)
#>         coeff.name coeff.val   coeff.se      coeff.ci coeff.pval
#> 1 Brennan-Prediger     0.835 0.04693346 (0.742,0.928)      0e+00
  krippen2.table(cont3x3abstractors)
```

```
#>             coeff.name coeff.val   coeff.se     coeff.ci coeff.pval
#> 1 Krippendorff's Alpha 0.7972585 0.05905473 (0.68,0.914)      0e+00
  pa2.table(cont3x3abstractors)
#>          coeff.name coeff.val   coeff.se      coeff.ci coeff.pval
#> 1 Percent Agreement      0.89 0.03128898 (0.828,0.952)      0e+00
```

Suppose that you only want to obtain Gwet's AC_1 coefficient, but don't care about the associated precision measures such as the standard error, confidence intervals or p-values. You can accomplish this as follows:

```
  ac1 <- gwet.ac1.table(cont3x3abstractors)$coeff.val
```

Then use the variable ac1 to obtain $AC_1 = 0.849$.

Another contingency table included in this package is named **cont3x3abstractors**. You may use it to experiment with the r functions listed above.

Computing agreement coefficients from the distribution of raters by subject & category

Included in this package is a small dataset named **distrib.6raters**, which contains the distribution of 6 raters by subject and category. Each row represents a subject (i.e. a psychiatric patient) and the number of raters (i.e. psychiatrists) who classified it into each category used in the inter-rater reliability study. Here is the dataset and how it can be used to compute the various agreement coefficients:

```
distrib.6raters
#>    Depression Personality.Disorder Schizophrenia Neurosis Other
#> 1           0                    0             0        6     0
#> 2           0                    3             0        0     3
#> 3           0                    1             4        0     1
#> 4           0                    0             0        0     6
#> 5           0                    3             0        3     0
#> 6           2                    0             4        0     0
#> 7           0                    0             4        0     2
#> 8           2                    0             3        1     0
#> 9           2                    0             0        4     0
#> 10          0                    0             0        0     6
#> 11          1                    0             0        5     0
#> 12          1                    1             0        4     0
#> 13          0                    3             3        0     0
#> 14          1                    0             0        5     0
#> 15          0                    2             0        3     1
gwet.ac1.dist(distrib.6raters)
#>   coeff.name     coeff     stderr      conf.int      p.value        pa
#> 1 Gwet's AC1 0.4448007 0.08418757 (0.264,0.625) 0.0001155927 0.5511111
#>          pe
#> 1 0.1914815
fleiss.kappa.dist(distrib.6raters)
#>      coeff.name     coeff     stderr     conf.int      p.value        pa
#> 1 Fleiss' Kappa 0.4139265 0.08119291 (0.24,0.588) 0.0001622724 0.5511111
#>          pe
#> 1 0.2340741
krippen.alpha.dist(distrib.6raters)
#>             coeff.name     coeff     stderr      conf.int      p.value
#> 1 Krippendorff's Alpha 0.4204384 0.08243228 (0.244,0.597) 0.0001615721
```

```
#>          pa        pe
#> 1 0.5560988 0.2340741
bp.coeff.dist(distrib.6raters)
#>         coeff.name     coeff     stderr      conf.int   p.value        pa
#> 1 Brennan-Prediger 0.4388889 0.08312142 (0.261,0.617) 0.0001163 0.5511111
#>    pe
#> 1 0.2
```

Once again, you can request a single value from these functions. To get only Krippendorff's alpha coefficient without it's precission measures, you may proceed as follows:

```
alpha <- krippen.alpha.dist(distrib.6raters)$coeff
```

The newly-created alpha variable gives the coefficient $\alpha = 0.4204384$.

Two additional datasets that represent ratings in the form of a distribution of raters by subject and by category, are included in this package. These datasets are **cac.dist4cat** and **cac.dist4cat**. Note that these 2 datasets contain more columns than needed to run the 4 functions presented in this section. Therefore, the columns associated with the response categories must be extracted from the original datasets before running the functions. For example, computing Gwet's AC_1 coefficient using the **cac.dist4cat** dataset should be done as follows:

```
ac1 <- gwet.ac1.dist(cac.dist4cat[,2:4])$coeff
```

Note that the input dataset supplied to the *gwet.ac1.dist* function is **cac.dist4cat[,2:4]**. That is, only columns 2, 3, and 4 are extracted from the original datset and used as input data. We know from the value of the newly created variable *ac1* that $AC_1 = 0.3518903$.

Computing agreement coefficients from raw ratings

One example dataset of raw ratings included in this package is **cac.raw4raters** and looks like this:

```
cac.raw4raters
#>    Rater1 Rater2 Rater3 Rater4
#> 1       1      1     NA      1
#> 2       2      2      3      2
#> 3       3      3      3      3
#> 4       3      3      3      3
#> 5       2      2      2      2
#> 6       1      2      3      4
#> 7       4      4      4      4
#> 8       1      1      2      1
#> 9       2      2      2      2
#> 10     NA      5      5      5
#> 11     NA     NA      1      1
#> 12     NA     NA      3     NA
```

As you can see, a dataset of raw ratings is merely a listing of ratings that the raters assigned to the subjects. Each row is associated with a single subject.Typically, the same subject would be rated by all or some of the raters. The dataset **cac.raw4raters** contains some missing ratings represented by the symbol NA, suggesting that some raters did not rate all subjects. As a matter of fact, in this particular case, no rater rated all subjects.

Here is you can compute the various agreement coefficients using the raw ratings:

```
pa.coeff.raw(cac.raw4raters)
#> $est
```

```
#>          coeff.name        pa pe coeff.val coeff.se  conf.int      p.value
#> 1 Percent Agreement 0.8181818  0 0.8181818  0.12561 (0.542,1) 4.345373e-05
#>       w.name
#> 1 unweighted
#>
#> $weights
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]    1    0    0    0    0
#> [2,]    0    1    0    0    0
#> [3,]    0    0    1    0    0
#> [4,]    0    0    0    1    0
#> [5,]    0    0    0    0    1
#>
#> $categories
#> [1] 1 2 3 4 5
gwet.ac1.raw(cac.raw4raters)
#> $est
#>   coeff.name        pa        pe coeff.val coeff.se  conf.int     p.value
#> 1        AC1 0.8181818 0.1903212   0.77544  0.14295 (0.461,1) 0.000208721
#>       w.name
#> 1 unweighted
#>
#> $weights
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]    1    0    0    0    0
#> [2,]    0    1    0    0    0
#> [3,]    0    0    1    0    0
#> [4,]    0    0    0    1    0
#> [5,]    0    0    0    0    1
#>
#> $categories
#> [1] 1 2 3 4 5
fleiss.kappa.raw(cac.raw4raters)
#> $est
#>      coeff.name        pa        pe coeff.val coeff.se  conf.int
#> 1 Fleiss' Kappa 0.8181818 0.2387153   0.76117  0.15302 (0.424,1)
#>       p.value     w.name
#> 1 0.000419173 unweighted
#>
#> $weights
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]    1    0    0    0    0
#> [2,]    0    1    0    0    0
#> [3,]    0    0    1    0    0
#> [4,]    0    0    0    1    0
#> [5,]    0    0    0    0    1
#>
#> $categories
#> [1] 1 2 3 4 5
krippen.alpha.raw(cac.raw4raters)
#> $est
#>             coeff.name    pa   pe coeff.val coeff.se  conf.int
#> 1 Krippendorff's Alpha 0.805 0.24   0.74342  0.14557 (0.419,1)
```

```
#>        p.value     w.name
#> 1 0.0004594257 unweighted
#>
#> $weights
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]    1    0    0    0    0
#> [2,]    0    1    0    0    0
#> [3,]    0    0    1    0    0
#> [4,]    0    0    0    1    0
#> [5,]    0    0    0    0    1
#>
#> $categories
#> [1] 1 2 3 4 5
conger.kappa.raw(cac.raw4raters)
#> $est
#>       coeff.name        pa        pe coeff.val coeff.se  conf.int
#> 1 Conger's Kappa 0.8181818 0.2334252   0.76282  0.14917 (0.435,1)
#>        p.value     w.name
#> 1 0.0003367066 unweighted
#>
#> $weights
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]    1    0    0    0    0
#> [2,]    0    1    0    0    0
#> [3,]    0    0    1    0    0
#> [4,]    0    0    0    1    0
#> [5,]    0    0    0    0    1
#>
#> $categories
#> [1] 1 2 3 4 5
bp.coeff.raw(cac.raw4raters)
#> $est
#>         coeff.name        pa  pe coeff.val coeff.se  conf.int      p.value
#> 1 Brennan-Prediger 0.8181818 0.2   0.77273  0.14472 (0.454,1) 0.0002375609
#>       w.name
#> 1 unweighted
#>
#> $weights
#>      [,1] [,2] [,3] [,4] [,5]
#> [1,]    1    0    0    0    0
#> [2,]    0    1    0    0    0
#> [3,]    0    0    1    0    0
#> [4,]    0    0    0    1    0
#> [5,]    0    0    0    0    1
#>
#> $categories
#> [1] 1 2 3 4 5
```

Most users of this package will only be interessted in the agreement coefficients and possibly in the related statistics such as the standard error and p-values. In this case, you should run these functions as follows (AC_1 is used here as an example. Feel free to experiment with the other coefficients):

```
ac1 <- gwet.ac1.raw(cac.raw4raters)$est
ac1
```

```
#>   coeff.name        pa        pe coeff.val coeff.se  conf.int     p.value
#> 1        AC1 0.8181818 0.1903212   0.77544  0.14295 (0.461,1) 0.000208721
#>       w.name
#> 1 unweighted
```

You can even request only the AC_1 coefficient estimate 0.77544. You will then proceed as follows:

```
ac1 <- gwet.ac1.raw(cac.raw4raters)$est
ac1$coeff.val
#> [1] 0.77544
```

References:

1. Gwet, K.L. (2014) *Handbook of Inter-Rater Reliability*, 4th Edition. Advanced Analytics, LLC.
2. Klein, D. (2018) "Implementing a general framework for assessing interrater agreement in Stata," *The Stata Journal*, **18**, 871-901.