R FUNCTIONS IN SCRIPT FILE **agree.coeff2.r**

If your analysis is limited to two raters, then you may organize your data in a contingency table that shows the count of subjects by rater and by category. Table B.1 is an example of such data where two neurologists classified 65 patients into 4 diagnostic categories.

Table B.1: Diagnostic Classification of Multiple Sclerosis Patients by Two Neurologists[a]

New Orleans Neurologist	Winnipeg Neurologist 1	2	3	4
1	5	3	0	0
2	3	11	4	0
3	2	13	3	4
4	1	2	4	14

[a]From Landis & Koch (1977)

Here is a list of functions in the script file `agree.coeff2.r`:

(1) `kappa2.table` (for Cohen's unweighted and weighted kappa coefficients)

(2) `scott2.table` (for Scott's unweighted and weighted Pi coefficients)

(3) `gwet.ac1.table` (for Gwet's unweighted and weighted AC_1 coefficients)

(4) `bp2.table` (for Brennan-Prediger unweighted and weighted coefficients)

(5) `krippen2.table` (for Krippendorff alpha coefficients)

All these functions operate the same way. Therefore, only the first of these functions named `kappa2.table` is discussed here in details. The same discussion applies to the other functions as well.

`kappa2.table`: *Cohen's kappa coefficient for 2 raters*

Description

This function calculates the unweighted as well as the weighted Cohen's kappa coefficients for 2 raters whose ratings are summarized in a square contingency table such as Table B.1. Some cells may have 0 values. However, the number of rows and columns must be equal.

Usage

```
kappa2.table(ratings,weights=diag(ncol(ratings)),conflev=0.95,
          N=Inf,print=TRUE)
```

Arguments

Of all arguments that this function takes, only the first one is required. The remaining arguments are all optional.

- `ratings`: A $q \times q$ matrix, where q is the number of categories. This is the only argument you must specify if you want the unweighted analysis,
- `weights`: A $q \times q$ matrix of weights. The default argument is the diagonal matrix where all diagonal numbers equal to 1, and all off-diagonal numbers equal to 0. This special weight matrix leads to the unweighted analysis. You may specify your own $q \times q$ weight matrix here as `weights=own.weights`. If you want to use quadratic weights with Table B.1 data for example, then the weights parameter would be `weights=quadratic.weights(1:4)`. You may want to look at the `weights.gen.r` script for a complete reference of all weight functions.
- `conflev`: The confidence level associated with the agreement coefficient's confidence interval.
- `N`: An optional parameter representing the total number of subjects in the target subject population. Its default value is infinity, which for all practical purposes assumes the target subject population to be very large and will not require any finite-population correction when computing the standard error.
- `print`: An optional logical parameter which takes the default value of TRUE if you also want the function to output the results on the screen. Set this parameter to FALSE if you do not want the results to be displayed on the screen. Setting this parameter to FALSE is recommended if this function is used as part of another routine.

Details

`kappa2.table` can accept data in the form of a matrix or in the form of a data frame as long as the input data supplied can be interpreted as a square matrix. To do the weighted analysis, you may create your own weight matrix, or use one of the many existing weight-generating functions in the `weights.ge.r` script file. Each weight function takes single mandatory parameter, which is a vector containing all categories used in the study. *The weight functions always sort all numeric-type category vectors in ascending order. Consequently, the weighted coefficients are computed properly only if the columns and rows in the input dataset are sorted in ascending order as well. For alphanumeric-type*

category vectors, they are assumed to already be ranked following an order that is meaningful to the researcher. That is adjacent columns and adjacent rows are associated with categories that can be considered as partial agreement.

Value

Calling the function `kappa2.table` returns the following 5 values:

- `pa`: the percent agreement.
- `pe`: the percent chance agreement.
- `kappa`: Cohen's kappa coefficient.
- `stderr`: the standard error of Cohen's kappa.
- `p.value`: the p-value of the kappa coefficient.

Examples

```
>ratings<-matrix(c(5, 3, 0, 0, # creates a matrix with Table B.1 data
+                   3, 11, 4, 0,
+                   2, 13, 3, 4,
+                   1, 2, 4, 14),ncol=4,byrow=T)
```

to compute unweighted kappa, its standard error and more

```
>kappa2.table(ratings)
```

The results displayed on the screen will look like this:

```
Cohen's Kappa Coefficient
=========================
Percent agreement: 0.4782609 Percent chance agreement: 0.2583491
Kappa coefficient: 0.2965166 Standard error: 0.07850387
95% Confidence Interval: ( 0.1398645, 0.4531686)
P-value: 0.0003361083
```

to compute weighted kappa with quadratic weights

```
>kappa2.table(ratings,quadratic.weights(1:4))
```

`#` the above call assumes the script file `weights.gen.r` was read into R, and the results obtained are the following:

```
Cohen's Kappa Coefficient
=========================
Percent agreement: 0.9098229 Percent chance agreement: 0.7591542
Kappa coefficient: 0.6255814 Standard error: 0.07873187
95% Confidence Interval: ( 0.4684744, 0.7826884)
P-value: 2.749756e-11
```

R FUNCTIONS IN SCRIPT FILE **agree.coeff3.dist.r**

If your analysis is based on three raters or more you can no longer summarize the ratings in a contingency table as previously done for the case of two raters. One option is to present that data in the form of a table where each row represents one subject, each column represents one category, and each table cell represents the number of raters who classified the specified subject into the specified category. Such a table shows the distribution of raters by subject and by category. Table B.2 is an example of such data where six raters classified 4 patients into 5 diagnostic categories.

Table B.2: Distribution of 6 Raters by Subject and Category[a]

	Category				
Subject	Depression	Personality Disorder	Schizophrenia	Neurosis	Other
A	0	0	0	6	0
B	0	1	4	0	1
C	2	0	4	0	0
D	0	3	3	0	0

[a]An extract of Table 1 of Fleiss (1971)

The following functions contained in the script file `agree.coeff3.dist.r` are what you will need to analyze rating data such as described in Table B.2:

(1) `fleiss.kappa.dist` (for Fleiss's unweighted and weighted kappa coefficients)

(3) `gwet.ac1.dist` (for Gwet's unweighted and weighted AC_1 coefficients)

(4) `bp.coeff.dist` (for Brennan-Prediger unweighted and weighted coefficients)

(5) `krippen.alpha.dist` (for Krippendorff unweighted and weighted alpha coefficients)

All these functions operate the same way. Therefore, only the first of these functions named `fleiss.kappa.dist` is discussed here in details. The same discussion applies to the other functions as well.

`fleiss.kappa.dist`: *Fleiss' kappa coefficient for multiple raters*

Description

This function calculates the unweighted as well as the weighted Fleiss' generalized kappa coefficients for multiple raters whose ratings are presented in the form of a distribution of raters by subject and category such as in Table B.2. A

table cell may have a 0 value if none of the raters classified the subject into the category associated with that cell. The number of raters may vary by subject leading to a table with different row totals. That will be the case when the experiment generated missing ratings, with subjects being rated by a different number of raters.

Usage

```
fleiss.kappa.dist(ratings,weights="unweighted",conflev=0.95,
          N=Inf,print=TRUE)
```

Arguments

Of all arguments used by this function, only the first one is required, the remaining arguments being all optional. If your goal is limited to unweighted statistics, then the simple function call `fleiss.kappa.dist(ratings)` *is sufficient to produce Fleiss' generalized kappa along with it standard error, confidence interval, and p-value.*

- `ratings`: This is an $n \times q$ matrix or data frame (or matrix), where n is the number of subjects, and q the number of categories. This is the only argument that must be specified if you want an unweighted analysis,
- `weights`: This is a $q \times q$ matrix of weights. The default argument is "unweighted". With this option, the function will create a diagonal weight matrix with all diagonal numbers equal to 1, and all off-diagonal numbers equal to 0. This special weight matrix leads to the unweighted analysis. You may create your own $q \times q$ weight matrix (e.g. `own.weights`) and assign it to the weights parameter as `weights=own.weights`. If you want to use quadratic weights with Table B.2 data for example, then the weights parameter would be `weights=quadratic.weights(1:5)`. You may want to look at the `weights.gen.r` script for a complete reference of all weight functions available.
- `conflev`: The confidence level associated with the agreement coefficient's confidence interval.
- `N`: An optional parameter representing the total number of subjects in the target subject population. Its default value is infinity, which for all practical purposes assumes the target subject population to be very large and will not require any finite-population correction when computing the standard error.
- `print`: An optional logical parameter which takes the default value of TRUE if you also want the function to output the results on the screen. Set this parameter to FALSE if you do not want the results to be displayed on the screen.

Details

`fleiss.kappa.dist` can accept input data in the form of a matrix or in the form of a data frame as long as the input data supplied can be interpreted as a matrix. To do the weighted analysis, you may create your own weight matrix, or use one of the many existing weight-generating functions in the `weights.ge.r` script. Each weight function takes single mandatory parameter, which is a vector containing all categories used in the study. *The weight functions always sort all numeric-type category vectors in ascending order. Consequently, the weighted coefficients are computed properly only if columns and rows in the input dataset are ordered the same way. For alphanumeric-type category vectors, they are assumed to already be ranked following an order that is meaningful to the researcher.*

Value

Calling function `fleiss.kappa.dist` returns the following 5 values:

- `pa`: the percent agreement.
- `pe`: the percent chance agreement.
- `fleiss.kappa`: Fleiss' generalized kappa coefficient.
- `stderr`: the standard error of Fleiss' kappa.
- `p.value`: the p-value of Fleiss' kappa coefficient.

Examples

```
# creates a matrix with Table B.2 data
>ratings<-matrix(c(0, 0, 0, 6, 0,
+                  0, 1, 4, 0, 1,
+                  2, 0, 4, 0, 0,
+                  0, 3, 3, 0, 0),ncol=5,byrow=T)

# to compute unweighted Fleiss' kappa, its standard error and more
>fleiss.kappa.dist(ratings)
```

The results displayed on the screen will look like this:

```
Fleiss' Kappa Coefficient
=========================
Percent agreement: 0.5666667 Percent chance agreement: 0.3090278
Fleiss kappa coefficient: 0.3728643 Standard error: 0.2457742
95% Confidence Interval: ( -0.409299 , 1 )
P-value: 0.2265189
```

```
# to compute weighted kappa with quadratic weights
>fleiss.kappa.dist(ratings,quadratic.weights(1:5))
# the call above assumes the script file weights.gen.r was read into R, and
```

generates the following results:

```
Fleiss' Kappa Coefficient
=========================
Percent agreement: 0.9270833 Percent chance agreement: 0.8854167
Fleiss kappa coefficient: 0.3636364 Standard error: 0.2525845
Weights:
1 0.9375 0.75 0.4375 0
0.9375 1 0.9375 0.75 0.4375
0.75 0.9375 1 0.9375 0.75
0.4375 0.75 0.9375 1 0.9375
0 0.4375 0.75 0.9375 1

95% Confidence Interval: ( -0.4402002 , 1)
P-value: 0.2455769
```

R FUNCTIONS IN SCRIPT FILE **agree.coeff3.raw.r**

If your analysis is based on three raters or more we previously saw that one option is to organize your data as a distribution of raters by subject and by category. Alternatively, you may report the raw ratings in a table where each raw represents a subject, each column a rater, and each table cell the actual rating assigned by the rater to the subject. Table B.3 is an example of such data where 5 raters classified 4 subjects into 3 categories labeled as {1, 2, 3}.

Table B.3: Rating of Four Subjects by Five Raters[a]

	Raters				
Subject	I	II	III	IV	V
A	2	2	3	2	2
B	2	2	2	2	2
C	2	2	2	2	1
D	1	2	2	2	2

[a]This is Table 2 of Finn (1970)

The following functions contained in the script file `agree.coeff3.raw.r` are what you will need to analyze rating data such as described in Table B.3:

(1) `fleiss.kappa.raw` (for Fleiss's unweighted and weighted kappa coefficients)

(3) `gwet.ac1.raw` (for Gwet's unweighted and weighted AC_1 coefficients)

(4) `bp.coeff.raw` (for Brennan-Prediger unweighted and weighted coefficients)

(5) `krippen.alpha.raw` (for Krippendorff unweighted and weighted alpha coefficients)

(5) `conger.kappa.raw` (for Conger's unweighted and weighted kappa coefficients)

All these functions operate the same way. Therefore, only the first of these functions named `fleiss.kappa.raw` is discussed here in details. The same discussion applies to the other functions as well.

`fleiss.kappa.raw`: *Fleiss' kappa coefficient for multiple raters & raw ratings*

Description

This function calculates the unweighted as well as the weighted Fleiss' generalized kappa coefficients for multiple raters whose raw ratings are listed horizontally for each subject such as in Table B.3. A table cell may be missing if a rater did not rate a particular subject. When the ratings are alphanumeric then the blank character is treated as a missing value.

Usage

```
fleiss.kappa.raw(ratings,weights="unweighted",conflev=0.95,
          N=Inf,print=TRUE)
```

Arguments

Of all arguments used by this function, only the first one is required. The remaining arguments being all optional. If your goal is limited to unweighted statistics, then the simple function call ***`fleiss.kappa.raw(ratings)`*** *is sufficient to produce Fleiss' generalized kappa along with its standard error, confidence interval, and p-value.*

- `ratings`: This is an $n \times r$ matrix or data frame (or matrix), where n is the number of subjects, and r the number of raters. This is the only argument that must be specified if you want an unweighted analysis.
- `weights`: This is a $q \times q$ matrix of weights. The default argument is "unweighted", and there is no need to specify it explicitly when the unweighted analysis is what you want. The `weights` parameter can take any of the following values "quadratic", "linear", "ordinal", "radical", "ratio", "circular", or "bipolar". You may refer to the previous chapters for an explicit definition of these different weights. You will need to read the `weights.gen.r` script into R before this function can perform a weighted analysis.

  When the input data is in the form of raw ratings, you may not have a direct way of obtaining a list of all categories involved in the experiment, especially

if the dataset is large. This makes it more difficult although not impossible to define your own weight matrix.

- `conflev`: The confidence level associated with the agreement coefficient's confidence interval.
- `N`: An optional parameter representing the total number of subjects in the target subject population. Its default value is infinity, which for all practical purposes assumes the target subject population to be very large and will not require any finite-population correction when computing the standard error.
- `print`: An optional logical parameter which takes the default value of TRUE if you also want the function to output the results on the screen. Set this parameter to FALSE if you do not want the results to be displayed on the screen.

Details

`fleiss.kappa.raw` can accept data in the form of a matrix or in the form of a data frame as long as the input data supplied can be interpreted as a matrix. The ratings may be of numeric or alphanumeric types. To perform the weighted analysis, you need to assign one the values mentioned above to the weights parameter. If you have the list of categories in your dataset, you may even create your own weight matrix, or use one of the many existing weight-generating functions in the `weights.ge.r` script. Each weight function takes single mandatory parameter, which is a vector containing all categories used in the study. *The weight functions always sort all numeric-type category vectors in ascending order. Consequently, the weighted coefficients are computed properly only if columns and rows in the input dataset are ordered the same way. For alphanumeric-type category vectors, they are assumed to already be ranked following an order that is meaningful to the researcher.*

Value

Calling function `fleiss.kappa.raw` returns the following 5 values:

- `pa`: the percent agreement.
- `pe`: the percent chance agreement.
- `fleiss.kappa`: Fleiss' generalized kappa coefficient.
- `stderr`: the standard error of Fleiss' kappa.
- `p.value`: the p-value of Fleiss' kappa coefficient.

Examples

```
# creates a matrix with Table B.3 data
>table.b.3<-matrix(c(
+                  2, 2, 3, 2, 2,
```

```
+                   2, 2, 2, 2, 2,
+                   2, 2, 2, 2, 1,
+                   1, 2, 2, 2, 2),ncol=5,byrow=TRUE)
```

to compute unweighted Fleiss' kappa, its standard error and more

```
>fleiss.kappa.raw(ratings)
```

The results displayed on the screen will look like this:

```
Fleiss' Kappa Coefficient
=========================
Percent agreement: 0.7 Percent chance agreement: 0.735
Fleiss kappa coefficient: -0.1320755 Standard error : 0.05375461
95% Confidence Interval: ( -0.3031466 , 0.03899568 )
P-value: 1.908890
```

to compute weighted kappa with quadratic weights

```
>fleiss.kappa.dist(ratings,weights="quadratic")
```

the above call assumes that the script file `weights.gen.r` was previously read into R, and generates the following results:

```
Fleiss' Kappa Coefficient
=========================
Percent agreement: 0.925 Percent chance agreement: 0.92625
Fleiss kappa coefficient: -0.01694915 Standard error: 0.06525606
Weights: quadratic
95% Confidence Interval: ( -0.2246230 , 0.1907247 )
P-value: 1.188125
```