APPENDIX B

Software Solutions

This appendix provides a brief discussion of the software solutions available to researchers for computing inter-rater reliability coefficients. The list of software packages presented here is far from being exhaustive. It merely represents my short list of products that I recommend the readers of this book to consider. Many packages offer several options for computing agreement coefficients, although the number of built-in procedures is quite limited. Specialized add-in packages, functions, or macros written by independent researcher-programmers compensate this deficiency to some extent. Among the statistical packages considered here are R, SAS, SPSS, and STATA, with a particular emphasis on R and SAS. I will also mention some freely-available online calculators, which generally have limited capability. For MS Excel, AgreeStat developed by the author of this book, is a user-friendly Excel-based software that is commercially available in Windows and Mac versions. The Mac version of AgreeStat requires Mac Office 2011 or a more recent version.

B.1 The R Software

The R package has become an immensely popular statistical package across the world. If you are going to do statistical analysis on a regular basis for many years, and you do not know which statistical software to learn, this is one you will want to give a very serious consideration to. No doubt. You will enjoy the assistance of an extended online support group where you will be able to ask questions. Moreover, the product is entirely free, and can be downloaded at `http://www.r-project.com`. Numerous quality books have been published to help practitioners and scientists learn how to use it.

R is an interactive computing environment that makes a large collection of statistical functions available to you. Using R is about finding the right function and learning how to use it. R gives you the opportunity to develop your own functions for performing routine tasks as well as develop completely new packages for advanced users. Those who are new to R might be interested in the PDF file entitled “Using R for Introductory Statistics” prepared by John Verzani. It provides a short and friendly introduction to the R package as well as a good overview of its capabilities. It can be downloaded at,

`http://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf`

A key feature of R is the opportunity to write your own functions to perform special analyses. Moreover, several functions aimed at performing similar analyses are often organized in a package. You must install the R package in an R session so that you can access its many functions. I will first review so R packages of interest developed by various authors, before describing a few specific R function that are freely available online.

B.1.1 *R Packages for Computing Inter-Rater Reliability Coefficients*

Several R packages are available to practitioners for computing agreement coefficients. Here is a non-exhaustive list of them:

- `irrCAC`: *Computing Chance-Corrected Agreement Coefficients (CAC)*, by Gwet, K.L.
  This R package was developed by the author of this book and is available on the Comprehensive R Archive Network (CRAN). This package implements most agreement coefficients discussed in this book with the exception of those special conditional coefficients discussed in chapter 8. A detailed description of this package can be obtained from the following URL:

  https://cran.r-project.org/web/packages/irrCAC/index.html

- `irr`: *Various Coefficients of Interrater Reliability and Agreement*, by Matthias Gamer, Jim Lemon, Ian Fellows, and Puspendra Singh.
  This R package offers several functions that implement various agreement coefficients. A few coefficients implemented in `irr` are not implemented in `irrCAC`, and vice-versa, there many coefficients in `irrCAC` are not implemented in the `irr` package. A detailed description of this package can be obtained from the following URL:

  https://cran.r-project.org/web/packages/irr/index.html.

- `rel`: *Reliability Coefficients*, by Riccardo LoMartire.
  This package provides point estimates with confidence intervals for agreement coefficients proposed by Bennett et al. (1954), Cohen (1960), Conger (1980),

Fleiss (1971), Gwet (2008a), Krippendorff (1970) and a few more. Note that this package implemented Fleiss' generalized kappa using the standard error expression proposed by Fleiss et al. (1979). However, the correct expression of this standard error was proposed by Gwet (2020). A detailed description of this package can be obtained from the following URL:

https://cran.r-project.org/web/packages/rel/index.html.

- `icr`: *Compute Krippendorff's Alpha,* by Alexander Staudt, and Pierre L'Ecuyer. This small package focuses on Krippendorff's alpha coefficient and is described in details on the following webpage:

  https://cran.r-project.org/web/packages/icr/index.html.

B.1.2 *Some R Functions for Computing Inter-Rater Reliability Coefficients*

I wrote several stand-alone R functions to compute most chance-corrected agreement coefficients presented in this book. Readers familiar with the R environment, could download these functions and modify them to fit their specific needs or just use them unmodified. Standard errors, confidence intervals, as well as p-values associated with these coefficients are calculated by these functions. These R functions can all be downloaded from the following URL:

`https://agreestat.com/software/default.html#rpackage`

They are organized in three R script files, corresponding to the three ways your ratings must be organized:

- **agree.coeff2.r**
  The functions contained in this script file compute various agreement coefficients and their standard errors when dealing with two raters, and ratings that are organized in a contingency table (or a square matrix) showing counts of subjects by rater and category. You may use this format if each rater rated all subjects. Otherwise subjects rated by one rater and not by the other may not be properly classified. In this case, you should have two columns of raw scores, and use one of the functions in the script file agree.coeff3.raw.r.

- **agree.coeff3.dist.r**
  The functions contained in this script file compute various agreement coefficients and their standard errors when dealing with multiple raters, and ratings that are organized in the form of an $n \times q$ table showing counts of raters by subject and category. Here n represents the number of subjects and q the number of categories.

- **agree.coeff3.raw.r**
  The functions contained in this script file compute various agreement coefficients and their standard errors when dealing with multiple raters, and rating data organized in the form of an $n \times r$ table showing the (alphanumeric) raw ratings that the raters assigned to the subjects. Here n represents the number of subjects and r the number of raters. The data is presented in the form of n rows containing r ratings each.

- **weights.gen.r**
  The functions in this script file generate various weights to be used when computing weighted agreement coefficients.

In order to use any of the functions contained in these script files, you need to read the appropriate script into R. If you want to use the functions contained in "agree.coeff2.r" for example, then you will read this file into R as follows:

```
>source("C:\\AdvancedAnalytics\\R Scripts\\agree.coeff2.r")
```

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.1.1 is an example of such data where two neurologists classified 65 patients who suffer from Multiple Sclerosis into 4 diagnostic categories.

Table B.1.1: Diagnostic Classification of Multiple Sclerosis Patients by Two Neurologists[a]

New Orleans	Winnipeg Neurologist			
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 and G. (1977)

Here is the 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.1.1. 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.1.1 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 a 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 positions of the columns and rows in the input dataset are in the same order as the corresponding categories in the sorted category vector. For alphanumeric-type category vectors, they are assumed to be already 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.1.1 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 experiment involves 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.1.2 is an example of such data where six raters classified 4 patients into 5 diagnostic categories.

Table B.1.2: Distribution of 6 Raters by Subject and Category[a]

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.1.2:

(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.1.2. 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.1.2 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 a 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 column positions in the input dataset match those of the corresponding categories in the sorted category vector. 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.1.2 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 row represents a subject, each column a rater, and each table cell the actual rating assigned by the rater to the subject. Table B.1.3 is an example of such data where 5 raters classified 4 subjects into 3 categories labeled as {1, 2, 3}.

Table B.1.3: 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.1.3:

(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.1.3. 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 is required 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 a 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. I assume here that adjacent categories on the sorted list represent a higher degree of agreement than two categories that are farther apart.*

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.1.3 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(table.b.3)
```

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: 0.09110958
```

# to compute weighted kappa with quadratic weights
```
>fleiss.kappa.raw(table.b.3,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: ( -0.5307952 , 0.1907247 )
P-value: 0.8118745
```

B.2 AgreeStat for Excel

At the time this book was published, there were 2 known Excel solutions that I would recommend practitioners to consider. One of these solutions is non-commercial while the other is commercial. The non-commercial solution is the *Real Statistics Data Analysis Tools* and the commercial solution is *AgreeStat360*.

- *Real Statistics Data Analysis Tools*, by Charles Zaiontz.

  The "Real Statistics Data Analysis Tools" is an Excel add-in, which implements a large number of statistical techniques, including the calculation of several inter-rater reliability coefficients. Interested Excel users can obtain more information regarding the installation of this software from the following URL:

  https://www.real-statistics.com/free-download/real-statistics-resource-pack/

  For a detailed description of the capability of this Excel add-in with respect to the computation of various inter-rater reliability coefficients, you may visit the following page:

  https://www.real-statistics.com/reliability/interrater-reliability/

- *AgreeStat360*, by K. Gwet.

  AgreeStat360 is a commercial Excel-based software for Windows developed by the author of this book. It is menu-driven and very user-friendly. It can compute various chance-corrected agreement coefficients, and many versions of intraclass correlation coefficients. A detailed description of its features can be obtained from the following URL:

  https://agreestat.com/software/default.html#excel

  AgreeStat360 implements all chance-corrected agreement coefficients discussed in this book, including the special conditional agreement coefficients of chapter 8. Moreover, various forms of the intraclass correlation can be calculated as well, including the optimal sample sizes. Intraclass correlation coefficients are discussed in more details by Gwet (2021).

B.3 Online Calculators

There are very few non-commercial online inter-rater reliability calculators. Although most of them are limited to the original two-rater version of Cohen (1960), a few have implemented Fleiss' extension to multiple raters as well. These online calculators are rarely maintained and do not always implement the latest techniques. Nevertheless, readers interested in the non-commercial options may want to consider the following 2 solutions:

- *StatsToDo:* http://www.statstodo.com/ResourceIndex_Subjects.php
- *ReCal:* http://dfreelon.org/utils/recalfront/

The only genuine cloud-based software for inter-rater reliability assessment is the commercial **AgreeStat360.com**. It is regularly maintained, and can be accessed from the URL agreestat360.com. To compute most agreement coefficients discussed in this book, all you need is your browser. No installation is required.

If your goal is to compare the difference between 2 agreement coefficients for statistical significance, you may consider using the free cloud-based **AgreeTest** developed by K. Gwet, and which can be accessed from the following URL:

`https://agreestat.net/agreetest/`

B.4 SAS Software

SAS is one of the major statistical software packages on the market today. It is a massive software system that has been around for many decades, and which is very expensive. It would be unwise to consider acquiring this product for the sole purpose of computing inter-rater reliability coefficients. Those who already have access to it, will certainly want to know about its capability as far as computing inter-rater reliability coefficients is concerned.

One of the many SAS modules is known as SAS/STAT. As of its version 14.2, SAS/STAT offers with the FREQ Procedure, options for computing the AC_1 coefficient (see Gwet, 2008a) as well as the PABAK coefficient[1] (see Byrt et al., 1993), in addition to Cohen's Kappa by Cohen (1960) , which was already implemented in the previous versions. Therefore, SAS users do not need to use another software to obtain theses statistics.

Note the FREQ Procedure will only compute the extent of agreement between 2 raters and presents some limitations regarding its treatment of missing values. By default, the FREQ procedure systematically deletes all observations with a missing rating. Consequently, the results obtained with SAS may differ from those obtained with functions available in several R packages, if your dataset contains missing ratings. An option is available for instructing the FREQ procedure to treat missing values as true categories. However, this option is useless for the analysis of agreement among raters. What would be of interest is for Proc FREQ developers to allow for the marginals associated with both raters to be calculated independently. That is, if a rating is available from one rater, then it should be used for calculating that rater's marginal probability whether or not the other rater rated the same subject or not.

[1]The coefficient often referred to by researchers as PABAK is also known (perhaps more rightfully so) as the Brennan-Prediger coefficient. It was formally studied by Brennan and Prediger (1981) , 13 years earlier.

If you want to compute the extent of agreement among 3 raters or more, then the FREQ Procedure can no longer be used. Fortunately, SAS Institute has provided a very useful Macro program called *MAGREE*, which is well documented and implemented several agreement coefficients used in the literature. For more information about this SAS macro, read *Sample 25006: Compute estimates and tests of agreement among multiple raters* from the following URL:

`https://support.sas.com/kb/25/006.html`

Here is a summary of the purpose of this macro as provided by SAS Institute:

> Compute estimates and tests of agreement among multiple raters when responses (ratings) are on a nominal or ordinal scale. For a nominal response, kappa and Gwet's AC1 agreement coefficient are available. For an ordinal response, Gwet's weighted agreement coefficient (AC2) and statistics based on a cumulative probit model with random effects for raters and items are available. If the response is numerically-coded (and possibly continuous), Kendall's coefficient of concordance is also available.

B.5 SPSS & STATA

STATA is another major statistical software packages, which is more recent than SAS and SPSS, and which specializes in the medical field. I strongly advise STATA users with interest in interrater reliability assessment to start by reading the very interesting article written by Klein (2018). Moreover, the document http://www.stata.com/manuals13/rkappa.pdf summarizes the built-in STATA commands related to inter-rater reliability assessment. Unlike SAS and SPSS, STATA has a built-in procedure for computing the multiple-rater version of kappa proposed by Fleiss (1971). Unless you are already a STATA user, it would be unwise to acquire this major software for the sole purpose of computing inter-rater reliability coefficients.

SPSS is also one of the major statistical software packages on the market today. Just like SAS, SPSS has been around for a while, and specializes in the social sciences. It offers some limited built-in procedures for computing inter-rater reliability coefficients and will be useful to researchers who are primarily interested in Cohen's kappa (see Cohen, 1960) or its generalized version by Fleiss (see Fleiss, 1971).

For computing Cohen's Kappa, one can follow the detailed instructions provided on the following page:

https://statistics.laerd.com/spss-tutorials/cohens-kappa-in-spss-statistics.php

You can compute Fleiss' generalized Kappa with SPSS. However, the procedure for doing it depends on the version of SPSS you are using. Researchers interested in this procedure can obtain more information from the following URL:

https://statistics.laerd.com/spss-tutorials/fleiss-kappa-in-spss-statistics.php

B.6 Concluding Remarks

In this appendix, I reviewed some of what I consider to be among the most interesting software options for researchers involved in inter-rater reliability assessment. The number of software solutions available to researchers for computing inter-rater reliability coefficients has increased dramatically since the first edition of this book. R and Excel users have more options to choose from than others. However, SAS and STATA users can also rely on very useful macro programs and functions developed by independent researchers. SPSS on the other hand, offers very few options, which are all limited to Cohen's Kappa and its generalized Fleiss' version. More details regarding all these software solutions can be obtained from the following URL:

https://agreestat.com/software/default.html

Before using a particular software package for calculating inter-rater reliability coefficients, researchers need to find out how that package handles missing ratings. Many programs made available to the general public do not have a well-defined strategy for dealing with missing ratings, which are known to be an important problem in many inter-rater reliability experiments. Even some existing R functions proposed in some publicly-available R packages such as `‘irr’` or `‘concord’` tend to exclude from analysis any subject that was not rated by all raters. This crude strategy may eliminate a substantial amount of data collected during the experiment. This may not a problem if it is the way you want missing data to be handled. But a better strategy would be to use every single data point that was gathered as recommended throughout this book.