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
```

As a courtesy service to the readers of this book, I decided to write a series of R functions that compute all chance-corrected agreement coefficients presented in this book. 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 webpage `www.agreestat.com/book4/` . 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 is an example of such data where two neurologists classified 65 patients who suffer from Multiple Sclerosis into 4 diagnostic categories.

Table B.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 & Koch (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. 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 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 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.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 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.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.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 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.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
```

B.2 AgreeStat for Excel

AgreeStat is a commercial Excel-based software developed by the author of this book. This software is menu-driven and user-friendly. It can compute various chance-corrected agreement coefficients, and many versions of intraclass correlation coefficients. The current version can be tested by downloading a trial copy from AgreeStat's webpage `http://www.agreestat.com/agreestat.html`, and is available for both Windows and Mac operating systems. For the Mac edition, the user will need to have Excel 2011 or a later version of Microsoft Office. AgreeStat's webpage contains a wealth of information about the different features of this software.

B.3 Online Calculators

Several websites have implemented the kappa coefficient. 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. Only a few online calculators are mentioned here. Interested readers could find many more minor and very limited calculators with a simple Google search.

- `http://www.statstodo.com/CohenKappa_Pgm.php`

  This online program is simple and very intuitive. It only implements two coefficients, which are the original kappa coefficient of Cohen (1960) for two raters, and its multiple-rater extension proposed by Fleiss (1971) . This program gives the user the opportunity to input the data in the form of a raw list of ratings for each subject, or in the form of a summary table of counts. However, this program always uses the summary table of counts to perform all calculations. Even if raw ratings are supplied by the user, the program will convert them to a summary table of counts.

  There are a few issues that users of this program must be aware of:

  ⇒ This program does not appear to use the correct standard error expressions for Cohen's kappa that were published by Fleiss et al. (1969) . Therefore the standard errors associated with kappa that this program produces are incorrect.

  ⇒ For FLeiss' generalized kappa, the program uses the standard error expression of Fleiss (1971). This expression too is incorrect. The correct expression is provided in chapter 5 of this book.

  ⇒ This program assumes that each rater rated all subjects. Therefore, you may need to eliminate from your dataset all subjects that were not rated by all raters. The program will not do it for you.

  ⇒ This program automatically creates the 95% confidence interval with no possibility of specifying another confidence level.

- ReCal ("Reliability Calculator"): `http://dfreelon.org/utils/recalfront/`
  This is an online utility that essentially computes the percent agreement, Fleiss' extension of kappa for multiple raters, Cohen's kappa (for 2 raters), the average of all Cohen's pairwise kappa, and Krippendorff's alpha. This online calculator seems to be used primarily by researchers and students in the field of communication and is very intuitive.

  ReCal does not appear to be flexible regarding the way input ratings must be organized. You must supply a matrix of raw ratings where each row represents the subject and each column the rater. No standard error, confidence interval or p-value are computed, which could be a problem for more sophisticated users.

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. SAS does not offer many built-in procedures for calculating inter-rater reliability coefficients. Actually the FREQ Procedure is the only option available and is limited to the original two-rater version of kappa suggested by Cohen (1960). A SAS macro program is necessary if you need to go beyond the basics.

The use of SAS for the purpose of calculating inter-rater reliability coefficients has been discussed extensively in Gwet (2010*c*). The reader would find in this manuscript a wealth of information regarding the advantages and disadvantages of using SAS to compute inter-rater reliability coefficients, as well as some options for calculating many of the coefficients discussed in this book.

B.5 SPSS & STATA

SPSS is also among the major statistical software packages on the market today. Just like SAS, it has been around for a while, and specializes in the social sciences. SPSS offers more built-in procedures for computing inter-rater reliability coefficients than SAS, although not many alternatives to the two-rater kappa of Cohen (1960) can be found. Not many SPSS macro programs were developed by independent programmers to implement some of well-known agreement coefficients.

STATA is another major statistical software packages, which is more recent than SAS and SPSS, and which specializes in the medical field. The document `http://www.stata.com/manuals13/rkappa.pdf` summarizes well what STATA has to offer in the area inter-rater reliability. Unlike SAS and SPSS, STATA has a built-in procedure for computing the multiple-rater version of kappa proposed by Fleiss (1971). Once again, 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.

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. Numerous researchers have made it known to me over the years that they wanted to have R functions that implement the different coefficients and associated standard errors that I proposed in the earlier editions of this book. It is the main reason why I developed the R functions presented in this appendix. Additional R functions will eventually be developed for the calculation of intraclass correlation coefficients. AgreeStat was developed for Excel users and those who are primarily interested in user-friendly menu-driven software packages. It implements almost all techniques discussed in this book, and additional information can be found about it at `www.agreestat.com/agreestat.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 by the researcher. A better strategy consists of using every single data point that was gathered as recommended throughout this book.