5. EXAMPLES OF VARIOUS ANALYSES, WITH CODE

This section illustrates how to conduct various reliability calculations using, where possible, the statistical software R (R Core Team, 2015). R is an extendable statistics program for which various user-supplied packages have been implemented. Many of the interrater reliability statistics are available in the package “irr” by Matthias Gamer and his colleagues (2012). (The exception is Andres and Marzo’s Delta, which requires special software.) The R Core Team’s website supplies an introduction to the basics, and this section assumes basic familiarity with the program such as can be obtained from that source (2015). In order to run the examples, the user will need to install the irr package, which is easily accomplished using a pull-down menu in R.

5.1 Categorical Variables

Table 10.9 presents initial codes Zelinsky and Shadish produced for a categorical variable representing activity types in single-case-design studies (2016). A fictitious, and less accurate, third rater has been added to allow the illustration of calculations for more than two raters. Although various methods exist in R for reading data from files, for the sake of simplicity the data here are entered from the keyboard by creating a vector of data for each rater and binding the columns of the table together. In all instances of R code, the “>” at the left side of a line is the R prompt, and hence is not typed by the user. We create objects called TwoRaters and ThreeRaters, for which TwoRaters is the first two columns of the table, and ThreeRaters is the complete table. The variable names (such as Rater1, TwoRaters) are supplied by the user.
> Rater1 <- c(1,3,1,1,1,2,2,1,4,1,1,1,1,1,1,1)
> Rater2 <- c(1,1,1,1,1,2,2,1,4,1,1,1,9,1,1,1)
> Rater3 <- c(1,3,1,3,1,2,2,1,4,1,1,4,1,1,1,1)
> TwoRaters <- cbind(Rater1, Rater2)
>
> TwoRaters
Rater1 Rater2
[1,] 1 1
[2,] 3 1
[3,] 1 1
[4,] 1 1
[5,] 1 1
[6,] 2 2
[7,] 2 2
[8,] 1 1
[9,] 4 4
[10,] 1 1
[11,] 1 1
[12,] 1 9
[13,] 1 1
[14,] 1 1
[15,] 1 1
[16,] 1 1
>
> ThreeRaters <- cbind(Rater1, Rater2, Rater3)
> ThreeRaters
Rater1 Rater2 Rater3
[1,] 1 1 1
[2,] 3 1 3
[3,] 1 1 1
[4,] 1 1 3
[5,] 1 1 1
[6,] 2 2 2
[7,] 2 2 2
[8,] 1 1 1
[9,] 4 4 4
[10,] 1 1 1
[11,] 1 1 1
[12,] 1 9 4
[13,] 1 1 1
[14,] 1 1 1
[15,] 1 1 1
[16,] 1 1 1

5.1.1 Agreement Rate
The “agree” function in the irr package calculates percentage of agreement. It can be applied to any number of raters. First, we observe the agreement rate for the two real raters.

```r
> agree(TwoRaters)
Percentage agreement (Tolerance=0)

Subjects = 16
   Raters = 2
 %-agree = 87.5
```

Note that the agreement rate is 87.5 percent. The agreement rate for the three-rater table is somewhat lower, in part because it represents the percentage of cases for which all three raters agree, a more stringent hurdle, and in part because the fictitious rater disagrees more frequently.

```r
> agree(ThreeRaters)
Percentage agreement (Tolerance=0)

Subjects = 16
   Raters = 3
 %-agree = 81.2
```

### 5.1.2 Cohen’s Kappa and Weighted Kappa

Cohen’s kappa for two raters is available in the irr package through the “kappa2” function. This calculates the unweighted form of kappa.

```r
> kappa2(TwoRaters)
Cohen's Kappa for 2 Raters (Weights: unweighted)

Subjects = 16
   Raters = 2
    Kappa = 0.701
           z = 4.35
       p-value = 1.37e-05
```
Note that kappa is .701. If one attempts to apply “kappa2” to more than two raters, the function returns an informative error message.

```r
> kappa2(ThreeRaters)
Error in kappa2(ThreeRaters) :
  Number of raters exceeds 2. Try kappam.fleiss or kappam.light.
```

As the message suggests, Fleiss’s kappa for more than two raters is available:

```r
> kappam.fleiss(ThreeRaters)
Fleiss' Kappa for m Raters

  Subjects = 16
  Raters = 3
  Kappa = 0.691

  z = 7.45
  p-value = 9.66e-14
```

Similarly to agreement rate, coding reliability is lower for the three-rater data, primarily because of the lower accuracy of the fictitious rater. It is also possible to calculate Light’s kappa, which is the average of all possible two-rater kappas:

```r
> kappam.light(ThreeRaters)
Light's Kappa for m Raters

  Subjects = 16
  Raters = 3
  Kappa = 0.694

  z = 1.9
  p-value = 0.0578
```

The “kappa2” function can also calculate weighted kappa by specifying a “weight” argument. The most likely value for weight is “equal” but weights based on squared distance are also available with the value “squared.” For this
data set, weighted kappa is not meaningful because the data are nominal level. We calculate it to illustrate the syntax, but the presence of the missing value code 9 makes the value of weighted kappa quite low:

```r
> kappa2(TwoRaters,weight="equal")
Cohen's Kappa for 2 Raters (Weights: equal)

Subjects = 16
Raters = 2
  Kappa = 0.547
  z = 2.99
  p-value = 0.00279
```

### 5.1.3 Andres and Marzo’s Delta

Andres and Marzo’s (2004) delta is not implemented in any available R package. A program executable on Windows computers is available at [http://www.ugr.es/~bioest/Delta.exe](http://www.ugr.es/~bioest/Delta.exe). The graphical interface involves a number of screens and is not amenable to presentation in text. For the two-rater data set, delta is 0.469 with a standard error of 0.118.

### 5.1.4 Krippendorff’s Alpha

The irr package implements Krippendorff’s alpha. However, the package expects a matrix in which rows represent raters, not columns, so that the data matrix must be transposed using R’s “t” function. The calculation of Krippendorff’s alpha depends on the level of measurement of the variable being coded. The default is nominal, which is appropriate here.

```r
> kripp.alpha(t(TwoRaters))
Krippendorff's alpha

Subjects = 16
Raters = 2
```
alpha = 0.709

Note that the function can be applied seamlessly to multiple-rater problems:

```
> kripp.alpha(t(ThreeRaters))
Krippendorff's alpha

Subjects = 16
Raters = 3
alpha = 0.694
```

### 5.2 Continuous Variables

Table 10.10 presents initial codes Zelinsky and Shadish produced for a continuous variable that represents the proportion of female participants in each of sixteen studies (2016). One again, a fictitious, and less accurate, third rater has been added to allow the illustration of calculations for more than two raters. This time, to avoid confusion with raters 1, 2, and 3 in the previous section, we create variables name RaterA, RaterB, and RaterC:

```r
> RaterA <- c(0, 2/3, 0, 0, 3/5, 1/3, 1, 0, 0, 4/9, 2/3, 2/3, 3/5, 2/3, 0, 47/83)
> RaterB <- c(0, 2/3, 0, 0, 3/5, 1/3, 1, 0, 0, 7/26, 2/3, 2/3, 3/5, 2/3, 0, 47/83)
> RaterC <- c(0, 2/3, 1/2, 0, 4/5, 1/2, 1, 0, 0, 1/2, 2/3, 2/3, 3/5, 2/3, 0, 47/83)
```

Next, we create objects containing the first two columns and all three columns, in order to illustrate analyses with two and more than two raters:

```r
> TwoContinuousRaters <- cbind(RaterA, RaterB)
> TwoContinuousRaters

          RaterA    RaterB
[1,] 0.0000000 0.0000000
[2,] 0.6666667 0.6666667
[3,] 0.0000000 0.0000000
```
Table 10.10 presents initial codes Zelinsky and Shadish produced for a

5.2.1 Intercoder Correlation

The R program contains a built-in function, cor, that calculates the Pearson product-moment correlation:
> cor(RaterA, RaterB)
[1] 0.9916407

> cor(TwoContinuousRaters)
   RaterA  RaterB
RaterA 1.0000000 0.9916407
RaterB 0.9916407 1.000000

With the irr package, it is possible to calculate the mean correlation for more than two raters:

> library(irr)
Loading required package: lpSolve
> meancor(ThreeContinuousRaters)
Mean of bivariate correlations R

Subjects = 16
Raters = 3
   R = 0.961

          z = 3.47
p-value = 0.00053

5.2.2 Krippendorff's Alpha

Krippendorff's alpha, applied to categorical data with the default nominal level of measurement in section 5.1.4, may also be applied to continuous data by specifying a higher level of measurement. In the current example, the proportions of female participants represent ratio level measurement. Once again, the function for Krippendorff’s alpha expects the transposition of the data matrix.

> kripp.alpha(t(TwoContinuousRaters), method="ratio")
Krippendorff's alpha

Subjects = 16
Raters = 2
alpha = 0.993

> kripp.alpha(t(ThreeContinuousRaters), method="ratio")
Krippendorff's alpha

Subjects = 16
Raters = 3
alpha = 0.901

5.2.3 Intraclass Correlation

The irr package also provides a function icc to calculate intraclass correlations. The function can take various arguments to specify different modes of icc calculation. For the present example (and for most applications in the area of coding reliability), the default “oneway” option is correct.

> icc(TwoContinuousRaters)
Single Score Intraclass Correlation

    Model: oneway
    Type : consistency

Subjects = 16
    Raters = 2
    ICC(1) = 0.992

F-Test, H0: r0 = 0 ; H1: r0 > 0
    F(15,16) = 238 , p = 6.76e-16

95%-Confidence Interval for ICC Population Values:
    0.977 < ICC < 0.997

> icc(ThreeContinuousRaters)
Single Score Intraclass Correlation

    Model: oneway
    Type : consistency

Subjects = 16
    Raters = 3
    ICC(1) = 0.934

F-Test, H0: r0 = 0 ; H1: r0 > 0
    F(15,32) = 43.5 , p = 5.3e-17
95% Confidence Interval for ICC Population Values:
0.858 < ICC < 0.974