Appendix: A Pascal program to compute kappa and weighted kappa

This appendix contains the source code for a Pascal program that computes kappa and weighted kappa. The program was compiled using Borland's Pascal 6.0 and should run on IBM-compatible microcomputers in a Windows or DOS environment. The program contains essentially no error checking for the numbers the user enters, so these must be correct. All values entered (the number of rows, weights if any, the tallies themselves) are integers, so entering a letter or even a decimal point instead of a digit, for example, will cause the program to fail (unless error checking code is added).

```
Program Kappa; { A simple Pascal no-bells, no-whistles,}
               { no-errors-permitted-in-input program  }
uses  Crt;     { to compute kappa and weighted kappa.  }
               { (c) Roger Bakeman, September 1995.    }
const Max = 20;
      Enter = chr(13);
      N : word = 0;

var   i, j : word;
      Ch   : char;
      X, W : array [1..Max,1..Max] of word;
      WeightsDefined : boolean;

{- - - - - - - - - - - - - - - - - - - - - - - - - - - - }
Procedure ComputeKappa;
var   i, j : word;
      M : array[1..Max+1,1..Max+1] of real;
      Kappa, Numer, Denom : real;

begin
  for i := 1 to N do    { Set row & col sums to zero.  }
  begin
    M[i,N+1] := 0.0;
```

```
    M[N+1,i] := 0.0;
  end;
  M[N+1,N+1] := 0.0;
                        { Tally row & col totals.       }
  for i := 1 to N do for j := 1 to N do
  begin
    M[i,N+1] := M[i,N+1] + X[i,j];
    M[N+1,j] := M[N+1,j] + X[i,j];
    M[N+1,N+1] := M[N+1,N+1] + X[i,j];
  end;
                         { Compute exp. frequencies.    }
  for i := 1 to N do for j := 1 to N do
    M[i,j] := M[i,N+1] * M[N+1,j] / M[N+1,N+1];

  Numer := 0.0;          { Compute kappa.               }
  Denom := 0.0;
  for i := 1 to N do for j := 1 to N do
  begin
    Numer := Numer + W[i,j] * X[i,j];
    Denom := Denom + W[i,j] * M[i,j];
  end;
  Kappa := 1.0 - Numer/Denom;
  writeln ('  Kappa =',Kappa:7:4);
  writeln;
end;

{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
Procedure DefaultWeights; { Provide standard weights,  }
var i, j : word;          { 0 on diagonal, 1 otherwise.}

begin
  for  i := 1 to N do for j := 1 to N do
  if (i = j) then W[i,j] := 0 else W[i,j] := 1;
end;

{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
Procedure AskForNumbers (Kind : word);
const Message:                {   Kind=1  Kind=2        }
     array [1..2] of string[7] = ('weights', 'tallies');
var i : word;

begin
```

```
  writeln (‘  Enter  ’,N:3,‘  ’,Message[Kind],
           ‘  (with spaces between for) ...’);
  for i :=1 to N do
  begin                                  {Read numbers;     }
    write (‘ row’,i:3,‘> ’);             {weights if Kind=1,}
    for j := 1 to N do read (X[i,j]);{tallies if Kind=2.}
    readln;                              {Must be integers. }
  end;
  if (Kind = 1) then WeightsDefined := true;
end;

{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
Function WantsSame (Kind : word) : boolean;
const Message : array [1..5] of string[12] =
      (‘Same # rows ’,
       ‘Same labels ’,   { Ask Y|N questions. Return   }
       ‘Weighted k  ’,   { TRUE if Y, y, or Enter.     }
       ‘Same weights ’,
       ‘More kappas ’);
var Ch : char;

begin
  write (‘   ’,Message[Kind],‘ (Y|N)? ’);
  Ch := ReadKey; writeln (Ch);
  WantsSame := (UpCase(Ch) = ‘Y’) or (Ch = Enter);
end;

{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
Procedure AskForOrder;    { Ask for order of matrix.   }
begin                     { Must be 1 through 20.      }
  repeat
    write (‘   Number of rows (1-20)? ’);
    readln (N)
  until (N > 0) and (N  <=  20);
  weightsDefined := false;
end;

{- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - }
BEGIN
  TextColor  (Black) ;
  TextBackground (LightGray) ;
  ClrScr;
```

```
  Writeln
  ('Compute kappa or wt kappa; (c) Roger Bakeman, GSU');

  repeat
    if N = 0 then AskForOrder
    else if not WantsSame(1) then AskForOrder;

    if not WeightsDefined
    then begin if WantsSame(3) then AskForNumbers(1) end
    else if not WantsSame(4) then AskForNumbers(1);

    if not WeightsDefined then DefaultWeights;
    AskForNumbers(2) ;
    ComputeKappa;
  until not WantsSame(5);
END.
```