Berechnungen von Multi-Coder-Reliabilitäten mit SPSS

Ein Exkurs zum Beitrag „Verlässlichkeit von Inhaltsanalysedaten. Reliabilitätstest, Errechnen und Interpretieren von Reliabilitätskoeffizienten für mehr als zwei Codierer“ in Medien & Kommunikationswissenschaft 3/2004, S. 335-354

Steffen Kolb

Mithilfe dieses Online-Tutorials soll der Arbeitsaufwand für Reliabilitätsberechnungen minimiert werden, indem eine SPSS-Anleitung (`Syntax ist in Courier gesetzt und jeweils mit Kommentaren versehen, die mit * beginnen und mit . enden`) für die Berechnung von paarweisen Übereinstimmungskoeffizienten, Kappa-Werten und Krippendorffs Alpha für mehr als zwei Codierer gegeben wird.

Voraussetzungen an die Daten:

Die Daten müssen auf der Fallebene in SPSS tatsächlich die Fälle haben und als Variablen die Codierer (vgl. Abbildung 1-1 für 4 Codierer). Dies wird grundlegend für die SPSS-Syntax im Folgenden als Grundlage gesehen. Im Regelfall wird man ohnehin auf Variablenebene Berechnungen anstellen, aber man kann natürlich auch *einen* Datensatz mit allen Fällen, Codierern und Variablen bearbeiten (vgl. Abbildung 1-2 für 4 Codierer und 2 Variablen).

Abbildung 1-1: SPSS-Datentabelle:

	c1	c2	c3	c4
Fall1	1	1	1	1
Fall2	2	2	2	3

Abbildung 1-2: SPSS-Datentabelle:

	C1 V1	C2 V1	C3 V1	C4 V1	C1 V2	C2 V2	C3 V2	C4 V2
Fall1	1	1	1	1	2	2	2	2
Fall2	2	2	2	3	1	2	3	2

Da die Daten meistens nicht in diesem Format vorliegen, sondern jeweils vier Fälle von vier Codierern untereinander stehen haben und in der SPSS-Variablenebene nur die Variablen, sei hier kurz erläutert, wie man den Datensatz anpassen muss.

```
* Vorbereiten für den Reltest.
* Erstellen von Datensätzen mit je nur einer Variable.

SAVE OUTFILE='C:\Eigene Dateiene\Reltest\reltestvar1.sav'
/drop  var2 var3 var4 var5 ...
/COMPRESSED.

* Drehen des Datensatzes - Achsenflip.
FLIP
  VARIABLES=var1
 /NEWNAME=codierer.
```

Es entsteht eine neues File mit einer (ehemaligen) Variablen auf der (SPSS-)Fallebene und jeweils vier hintereinanderstehenden (SPSS-)Variablen, die die Codierungen in einem Fall repräsentieren. Danach muss der neue Datensatz gespeichert werden:

```
SAVE OUTFILE='C:\Eigene Dateiene\Reltest\reltestvar1cas1.sav'
/rename (v1, v2, v3, v4 = c1, c2, c3, c4)
/drop  v11 v21 v31 v41 ...
/COMPRESSED.
```

Die neuen Variablen wurden standardmäßig benannt als v1, v2, v3, v4, v11, v21 usw., wobei sich an erster (Ziffern-)Stelle immer der Codierer und an zweiter immer der Fall (seltsamerweise beginnend bei nichts). Jetzt bleiben nur die Codierungen des ersten Falls übrig. Dieses Vorgehen muss mit allen einzelnen Fällen wiederholt werden:

```
SAVE OUTFILE='C:\Eigene Dateiene\Reltest\reltestvar1cas2.sav'
/rename (v11, v21, v31, v41 = c1, c2, c3, c4)
/drop  v1 v2 v3 v4 ...
/COMPRESSED.
```

Liegen für jeden Fall einzelne SPSS-Datensätze vor, so können diese einfach mit dem Add Files-Befehl zusammengefügt werden, so dass am Ende das gewünschte Datenformat entsteht:

```
GET
  FILE="C:\Eigene Dateien\Reltest\reltestvar1cas1.sav".
EXECUTE .

ADD FILES /FILE=*
 /FILE='C:\Eigene Dateien\Reltest\reltestvar1cas2.sav'.
...

SAVE OUTFILE='C:\Eigene Dateiene\Reltest\reltestvar1all.sav'
/COMPRESSED.
```

Liegen die Daten vor, so kann mit der Auswertung gestartet werden.

Syntax der Berechnung von Übereinstimmungskoeffizienten:

```
* Berechnung paarweiser Übereinstimmungen.
* Codierung in neue Variablen.
compute ue12=c1/c2.
compute ue13=c1/c3.
compute ue14=c1/c4.
compute ue23=c2/c3.
compute ue24=c2/c4.
compute ue34=c3/c4.
exe.
```

Die Division als Bestimmung von Übereinstimmungen bietet sich an, weil eine Zahl durch die übereinstimmende Zahl dividiert 1 ergibt und 1 gleichzeitig der Koeffizient für Übereinstimmung ist. Bei Variablen, die die Ausprägung 0 haben können, muss eine Umcodierung der 0 erfolgen, damit es nicht zu Divisionen durch 0 kommen kann.[1]

```
*Umcodieren der Übereinstimmungsvariablen in dichotome Variablen.
if ue12 ne 1 ue12=0.
if ue13 ne 1 ue13=0.
if ue14 ne 1 ue14=0.
if ue23 ne 1 ue23=0.
if ue24 ne 1 ue24=0.
if ue34 ne 1 ue34=0.
exe.

* Mittelwerte der Übereinstimmungspaare.
FREQUENCIES
  VARIABLES=ue12 ue13 ue14 ue23 ue24 ue34  /FORMAT=NOTABLE
  /STATISTICS=STDDEV MEAN
  /ORDER  ANALYSIS .

* Auszählen der Übereinstimmungen pro Fall.
COUNT
  ueges = ue12 ue13 ue14 ue23 ue24 ue34  (1)  .
VARIABLE LABELS ueges 'Gesamtübereinstimmungen' .
EXECUTE .
compute holsti=ueges/6.

* Auszählen der Nicht-Übereinstimmungen pro Fall.
COUNT
  abwges = ue12 ue13 ue14 ue23 ue24 ue34  (0)  .
VARIABLE LABELS ueges 'Gesamtübereinstimmungen' .
EXECUTE .
compute neholsti=abwges/6.

* Gesamtreliabilität nach Holsti.
FREQUENCIES
```

[1] Die Berechnung kann auch über die Subtraktion der Einzelwerte voneinander passieren. Die nachfolgende Dichotomisierung muss dann allerdings anders erfolgen.

```
  VARIABLES=holsti  /FORMAT=NOTABLE
  /STATISTICS=STDDEV MEAN
  /ORDER  ANALYSIS .
FREQUENCIES
  VARIABLES=neholsti  /FORMAT=NOTABLE
  /STATISTICS=STDDEV MEAN
  /ORDER  ANALYSIS .


* Codierleistungen nach Codierern.
COUNT
  ueges1 = ue12 ue13 ue14 (1)  .
VARIABLE LABELS ueges1 'Gesamtübereinstimmungen Codierer 1' .
EXECUTE .
compute holsti1=ueges1/3.

COUNT
  ueges2 = ue12 ue23 ue24 (1)  .
VARIABLE LABELS ueges2 'Gesamtübereinstimmungen Codierer 2' .
EXECUTE .
compute holsti2=ueges2/3.

COUNT
  ueges3 = ue23 ue13 ue34 (1)  .
VARIABLE LABELS ueges3 'Gesamtübereinstimmungen Codierer 3' .
EXECUTE .
compute holsti3=ueges3/3.

COUNT
  ueges4 = ue24 ue34 ue14 (1)  .
VARIABLE LABELS ueges4 'Gesamtübereinstimmungen Codierer 4' .
EXECUTE .
compute holsti4=ueges4/3.

*Reliabilität nach Codierern.
FREQUENCIES
  VARIABLES=holsti1 holsti2 holsti3 holsti4  /FORMAT=NOTABLE
  /STATISTICS=STDDEV MEAN
  /ORDER  ANALYSIS .
```

Mit Anpassungen sind somit Ermittlungen von Intercoderreliabilitäten nach Holsti für einzelne Variablen, einzelne Codierer und Gesamtbetrachtungen möglich.

Syntax der Berechnung von Kappa:

Die Syntax des Auswertungsprogramms ist ziemlich kurz dargestellt, allerdings beinhaltet sie lediglich die Aufforderung an SPSS, die Makro-Datei mkappasc.sps zu verwenden und dabei die 4 Variablen c1, c2, c3 und c4 einzubeziehen.

```
include "C:\Eigene Dateien\Reltest\mkappasc.sps".
mkappasc vars= c1 c2 c3 c4 .
```

Das Makro zur Kappa-Berechnung ist kostenlos von SPSS erhältlich. Allerdings hat es bei meinen Berechnungen – wie von SPSS programmiert – nie funktioniert. Hier deshalb die veränderte Form (**Änderungen sind fett**):

```
************************************************************
*  MACRO NAME:          MKAPPASC.SPS                       *
*                                                          *
*  README FILE:         MKAPPASC.RM                        *
*                                                          *
*  SPSS REQUIREMENTS:   Release 4.0 or above               *
*                       Advanced Statistics Module         *
*                                                          *
*  AUTHOR:              David Nichols (nichols@spss.com)   *
*                                                          *
*  LAST UPDATED:        04/08/97                           *
************************************************************

preserve.
set printback=off mprint=off.
save outfile='ka__tmp1.sav'.
define mkappasc (vars=!charend('/')).
set mxloops=48000.
set mxmemory=64000.
set workspace=64000.
count ms__=!vars (missing).
select if ms__=0.
matrix.
get x /var=!vars.
compute c=mmax(x).
compute y=make(nrow(x),c,0).
loop i=1 to nrow(x).
loop j=1 to ncol(x).
loop k=1 to c.
do if x(i,j)=k.
compute y(i,k)=y(i,k)+1.
end if.
end loop.
end loop.
end loop.
```

```
compute pe=msum((csum(y)/msum(y))&**2).
compute k=ncol(x).
compute pa=mssq(y)/(nrow(y)*k*(k-1))-(1/(k-1)).
compute kstat=(pa-pe)/(1-pe).
compute num=2*(pe-(2*k-3)*(pe**2)+2*(k-2)*msum((csum(y)/msum(y))&**3)).
compute den=nrow(y)*k*(k-1)*((1-pe)**2).
compute ase=sqrt(num/den).
compute z=kstat/ase.
compute sig=1-chicdf(z**2,1).
save {kstat,ase,z,sig} /outfile='ka__tmp2.sav'
     /variables=kstat,ase,z,sig.
end matrix.
get file='ka__tmp2.sav'.
formats all (f11.8).
variable labels kstat 'Kappa' /ase 'ASE' /z 'Z-Value' /sig 'P-Value'.
report format=list automatic align(center)
  /variables=kstat ase z sig
  /title "Estimated Kappa, Asymptotic Standard Error,"
         "and Test of Null Hypothesis of 0 Population Value".
get file='ka__tmp1.sav'.
!enddefine.
restore.
```

Die Veränderungen betreffen die SPSS-Speichergrenze, den tatsächlich reservierten Arbeitsspeicher und die Anzahl von Loops, die SPSS bei Matrix-Berechnungen höchstens verwendet. Diese Änderungen sind natürlich ja nach Umfang des Datenmaterials unterschiedlich zu machen. Die angegebenen Werte haben aber guten Erfolg gezeigt. Interessanterweise kann SPSS zwar offiziell gar nicht 64000Kbyte Arbeitsspeicherreservierung verarbeiten. Allerdings haben meine Berechnungen mit der angeblichen Speichergrenze, die je nach Version unterschiedlich hoch ist (Version 8 14336Kbyte), z. T. nicht funktioniert, mit der dokumentierten Einstellung aber schon. Das Programm beschwert sich erst beim nächsten Start:

```
Warning # 883 in column 71. Text: 64000
The specified value of WORKSPACE exceeds MXMEMORY (the maximum amount of
memory that can be allocated).
WORKSPACE has been set to MXMEMORY.
Use SHOW MXMEMORY to find the maximum allocation and SET MXMEMORY to change
it.
```

Syntax der Berechnung von Krippendorffs Alpha:

```
*Beispiel 4 Codierer 5 Ausprägungen 50 Fälle.
* Berechnung paarweiser Übereinstimmungen für Ausprägung 1.
* Codierung in neue Variablen.
*1er Übereinstimmungen.
if (c1=1 or c2=1 or c3=1 or c4=1) and c1 ge c2 ue1_12=c1 + c2*10.
if (c1=1 or c2=1 or c3=1 or c4=1) and c1 lt c2 ue1_12=c1*10 + c2.
if (c1=1 or c2=1 or c3=1 or c4=1) and c1 ge c3 ue1_13=c1 + c3*10.
if (c1=1 or c2=1 or c3=1 or c4=1) and c1 lt c3 ue1_13=c1*10 + c3.
if (c1=1 or c2=1 or c3=1 or c4=1) and c1 ge c4 ue1_14=c1 + c4*10.
if (c1=1 or c2=1 or c3=1 or c4=1) and c1 lt c4 ue1_14=c1*10 + c4.
if (c1=1 or c2=1 or c3=1 or c4=1) and c2 ge c3 ue1_23=c2 + c3*10.
if (c1=1 or c2=1 or c3=1 or c4=1) and c2 lt c3 ue1_23=c2*10 + c3.
if (c1=1 or c2=1 or c3=1 or c4=1) and c2 ge c4 ue1_24=c2 + c4*10.
if (c1=1 or c2=1 or c3=1 or c4=1) and c2 lt c4 ue1_24=c2*10 + c4.
if (c1=1 or c2=1 or c3=1 or c4=1) and c3 ge c4 ue1_34=c3 + c4*10.
if (c1=1 or c2=1 or c3=1 or c4=1) and c3 lt c4 ue1_34=c3*10 + c4.
exe.
*2er Übereinstimmungen.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c1 ge c2 ue1_12=c1 + c2*10.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c1 lt c2 ue1_12=c1*10 + c2.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c1 ge c3 ue1_13=c1 + c3*10.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c1 lt c3 ue1_13=c1*10 + c3.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c1 ge c4 ue1_14=c1 + c4*10.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c1 lt c4 ue1_14=c1*10 + c4.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c2 ge c3 ue1_23=c2 + c3*10.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c2 lt c3 ue1_23=c2*10 + c3.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c2 ge c4 ue1_24=c2 + c4*10.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c2 lt c4 ue1_24=c2*10 + c4.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c3 ge c4 ue1_34=c3 + c4*10.
if (c1=2 or c2=2 or c3=2 or c4=2 ) and c3 lt c4 ue1_34=c3*10 + c4.
exe.
*3er Übereinstimmungen.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c1 ge c2 ue1_12=c1 + c2*10.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c1 lt c2 ue1_12=c1*10 + c2.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c1 ge c3 ue1_13=c1 + c3*10.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c1 lt c3 ue1_13=c1*10 + c3.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c1 ge c4 ue1_14=c1 + c4*10.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c1 lt c4 ue1_14=c1*10 + c4.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c2 ge c3 ue1_23=c2 + c3*10.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c2 lt c3 ue1_23=c2*10 + c3.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c2 ge c4 ue1_24=c2 + c4*10.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c2 lt c4 ue1_24=c2*10 + c4.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c3 ge c4 ue1_34=c3 + c4*10.
if (c1=3 or c2=3 or c3=3 or c4=3 ) and c3 lt c4 ue1_34=c3*10 + c4.
exe.
*4er Übereinstimmungen.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c1 ge c2 ue1_12=c1 + c2*10.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c1 lt c2 ue1_12=c1*10 + c2.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c1 ge c3 ue1_13=c1 + c3*10.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c1 lt c3 ue1_13=c1*10 + c3.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c1 ge c4 ue1_14=c1 + c4*10.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c1 lt c4 ue1_14=c1*10 + c4.
```

```
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c2 ge c3 ue1_23=c2 + c3*10.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c2 lt c3 ue1_23=c2*10 + c3.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c2 ge c4 ue1_24=c2 + c4*10.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c2 lt c4 ue1_24=c2*10 + c4.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c3 ge c4 ue1_34=c3 + c4*10.
if (c1=4 or c2=4 or c3=4 or c4=4 ) and c3 lt c4 ue1_34=c3*10 + c4.
exe.
*5er Übereinstimmungen.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c1 ge c2 ue1_12=c1 + c2*10.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c1 lt c2 ue1_12=c1*10 + c2.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c1 ge c3 ue1_13=c1 + c3*10.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c1 lt c3 ue1_13=c1*10 + c3.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c1 ge c4 ue1_14=c1 + c4*10.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c1 lt c4 ue1_14=c1*10 + c4.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c2 ge c3 ue1_23=c2 + c3*10.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c2 lt c3 ue1_23=c2*10 + c3.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c2 ge c4 ue1_24=c2 + c4*10.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c2 lt c4 ue1_24=c2*10 + c4.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c3 ge c4 ue1_34=c3 + c4*10.
if (c1=5 or c2=5 or c3=5 or c4=5 ) and c3 lt c4 ue1_34=c3*10 + c4.
exe.

fre ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34.

* Häufigkeiten von Übereinstimmungen und Abweichungen.
COUNT
  ue11 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (11)  .
VARIABLE LABELS ue11 'Übereinstimmende Codierung von 1' .
EXECUTE .

COUNT
  ue22 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (22)  .
VARIABLE LABELS ue11 'Übereinstimmende Codierung von 2' .
EXECUTE .

COUNT
  ue33 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (33)  .
VARIABLE LABELS ue11 'Übereinstimmende Codierung von 3' .
EXECUTE .

COUNT
  ue44 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (44)  .
VARIABLE LABELS ue11 'Übereinstimmende Codierung von 4' .
EXECUTE .

COUNT
  ue55 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (55)  .
VARIABLE LABELS ue11 'Übereinstimmende Codierung von 5' .
EXECUTE .

COUNT
  abw12 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (12)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 12' .
EXECUTE .
```

```
COUNT
  abw13 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (13)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 13' .
EXECUTE .

COUNT
  abw14 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (14)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 14' .
EXECUTE .

COUNT
  abw15 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (15)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 15' .
EXECUTE .

COUNT
  abw23 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (23)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 23' .
EXECUTE .

COUNT
  abw24 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (24)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 24' .
EXECUTE .

COUNT
  abw25 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (25)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 25' .
EXECUTE .

COUNT
  abw34 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (34)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 34' .
EXECUTE .

COUNT
  abw35 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (35)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 35' .
EXECUTE .

COUNT
  abw45 = ue1_12 ue1_13 ue1_14 ue1_23 ue1_24 ue1_34  (45)  .
VARIABLE LABELS ue11 'Abweichende Codierung von 45' .
EXECUTE .

fre ue11 ue22 ue33 ue44 ue55.
fre abw12 abw13 abw14 abw15 abw23 abw24 abw25 abw34 abw35 abw45.

*Breakvariable erstellen.
compute breaker =1.

* Aggregieren ACHTUNG Dateinamen ändern!.
AGGREGATE
  /OUTFILE='C:\Eigene Dateien\Reltest\aggr.sav'
  /BREAK=breaker
```

```
   /ue11_1 = SUM(ue11) /ue22_1 = SUM(ue22) /ue33_1 = SUM(ue33) /ue44_1 =
  SUM(ue44) /ue55_1 = SUM(ue55) /abw12_1 = SUM(abw12) /abw13_1 = SUM(abw13)
 /abw14_1 = SUM(abw14) /abw15_1 = SUM(abw15) /abw23_1 = SUM(abw23) /abw24_1 =
  SUM(abw24) /abw25_1 = SUM(abw25) /abw34_1 = SUM(abw34) /abw35_1 = SUM(abw35)
  /abw45_1 = SUM(abw45)
  /fallanz=N.

*HIER STOPP! Datei neu laden.
GET FILE='C:\Eigene Dateien\Reltest\aggr.sav'


*Bestimmung der Randsummen von Krippendorffs Matrix.
compute rand1=2*ue11_1+abw12_1+abw13_1+abw14_1+abw15_1.
compute rand2=2*ue22_1+abw12_1+abw23_1+abw24_1+abw25_1.
compute rand3=2*ue33_1+abw13_1+abw23_1+abw34_1+abw35_1.
compute rand4=2*ue44_1+abw14_1+abw24_1+abw34_1+abw45_1.
compute rand5=2*ue55_1+abw15_1+abw25_1+abw35_1+abw45_1.
exe.

*Gesamtsumme.
compute randges=rand1+rand2+rand3+rand4+rand5.
exe.

*Gesamtabweichungen.
compute abwges=2*(abw12_1 + abw13_1 + abw14_1 + abw15_1 + abw23_1 + abw24_1 +
abw25_1 + abw34_1 + abw35_1 + abw45_1).
exe.

* Anteil beobachteter Abweichungen.
compute abwemp=abwges/randges.
exe.

*Randsummen-Abweichungsprodukt (off-diagonal triangle product).
compute  prodrand=rand1*rand2  +  rand1*rand3  +  rand1*rand4  +  rand1*rand5  +
rand2*rand3  +  rand2*rand4  +  rand2*rand5  +  rand3*rand4  +  rand3*rand5  +
rand4*rand5 .
exe.

*Krippendorff's Alpha (mit drei df).
compute alpha=1-(randges-3)*(abwges/2)/prodrand.
for alpha (F8.6).
exe.

*Alternative Berechnung von Krippendorffs Alpha.
compute alpha2=1-(abwemp/(prodrand*2/(randges*(randges-3)))).
for alpha2 (F8.6).
exe.[2]
```

[2] Für ein Nachvollziehen der einzelnen Schritte sei auf Krippendorff (1980: 129ff. und besonders 142) verwiesen.

Syntax der Berechnung des alternativen Alpha:

Vor der Aggregierung müssen – verglichen mit der Syntax für Krippedorffs Alpha wenige zusätzliche Berechnungen gemacht werden:

```
* Beispiel für 4 Codierer 5 Ausprägungen und 50 Fälle.
* Kolbs Reliabilitätsmessung für mehrere Codierer.
* Berechnung paarweiser Übereinstimmungen.
* Codierung in neue Variablen.
compute ue12=c1/c2.
compute ue13=c1/c3.
compute ue14=c1/c4.
compute ue23=c2/c3.
compute ue24=c2/c4.
compute ue34=c3/c4.
exe.

*Umcodieren der Übereinstimmungsvariablen in dichotome Vars.
if ue12 ne 1 ue12=0.
if ue13 ne 1 ue13=0.
if ue14 ne 1 ue14=0.
if ue23 ne 1 ue23=0.
if ue24 ne 1 ue24=0.
if ue34 ne 1 ue34=0.
exe.

* Codierleistungen nach Codierern.
COUNT
  ueges1 = ue12 ue13 ue14 (1)  .
VARIABLE LABELS ueges1 'Gesamtübereinstimmungen Codierer 1' .
EXECUTE .
compute holsti1=ueges1/3.

COUNT
  ueges2 = ue12 ue23 ue24 (1)  .
VARIABLE LABELS ueges2 'Gesamtübereinstimmungen Codierer 2' .
EXECUTE .
compute holsti2=ueges2/3.

COUNT
  ueges3 = ue23 ue13 ue34 (1)  .
VARIABLE LABELS ueges3 'Gesamtübereinstimmungen Codierer 3' .
EXECUTE .
compute holsti3=ueges3/3.

COUNT
  ueges4 = ue24 ue34 ue14 (1)  .
VARIABLE LABELS ueges4 'Gesamtübereinstimmungen Codierer 4' .
EXECUTE .
compute holsti4=ueges4/3.

* Wenn nun der Wert für einzelne Codierer 0 annimmt, bedeutet das eine
abweichende Codierung.
* Bei größeren Werten gibt es mindestens eine Übereinstimmung.
```

```
* Daraus folgt, dass als abweichende Codierungen die Nullen gezählt werden
müssen.
COUNT
  abwcod = ueges1 ueges2 ueges3 ueges4 (0)  .
VARIABLE LABELS abwcod 'Abweichende Codierungen pro Fall'.
EXECUTE .

*Breakvariable erzeugen, die gleichzeitig die Zahl der Codierer enthält.
compute codierer=4.
exe.

* Alternativ kann die maximale Übereinstimmungszahl bestimmt werden.
compute uesum=ueges1+ueges2+ueges3+ueges4.
compute uemax=4-abwcod.
exe.

* Aus diesen beiden lässt sich die empirische Übereinstimmungshäufigkeit als
Koeffizient berechnen.
compute uecodant=(uesum/uemax)/(codierer-1).

* Dabei ist zu beachten, dass der Quotient im Zähler nicht bestimmbar ist,
wenn die maximale Übereinstimmungszahl 0 ist, wenn also alle Codierer
abweichend voneinander codiert haben.
If missing(uecodant) uecodant=0.
for uecodant (F8.6).
exe.

*Anzahl der möglichen Ausprägungen der Variable.
compute varauspr=5.
exe.

*Fallzahl.
compute fallzahl=50.
exe.
```

Nach diesen Berechnungen und den Krippendorff-Berechnungen kann aggregiert werden:

```
* Aggregieren ACHTUNG Dateinamen ändern!.
AGGREGATE
  /OUTFILE='F:\Eigene'+
 ' Dateien\Veröffentlichungen\Reliabilität\spss\aggrgewkolbneu2.sav'
  /BREAK=codierer varauspr fallzahl
   /ue11_1 = SUM(ue11) /ue22_1 = SUM(ue22) /ue33_1 = SUM(ue33) /ue44_1 =
  SUM(ue44) /ue55_1 = SUM(ue55) /abw12_1 = SUM(abw12) /abw13_1 = SUM(abw13)
 /abw14_1 = SUM(abw14) /abw15_1 = SUM(abw15) /abw23_1 = SUM(abw23) /abw24_1 =
  SUM(abw24) /abw25_1 = SUM(abw25) /abw34_1 = SUM(abw34) /abw35_1 = SUM(abw35)
  /abw45_1 = SUM(abw45) /abwcod = SUM(abwcod) /uecodant = MEAN(uecodant)
  /fallanz=N.
```

Jetzt kann das aggregierte File geöffnet und danach die Berechnungen von den alternativen Koeffizienten erfolgen.

```
GET FILE='F:\Eigene'+
 ' Dateien\Veröffentlichungen\Reliabilität\spss\aggrgewkolbneu2.sav'.

*Übereinstimmungskoeffizient.
compute kolb=uecodant.
for kolb (F8.4).
exe.

* Einfach gewichteter Übereinstimmungskoeffizient bei 5-stufiger Variable.
compute gewkolb=(kolb-(1/varauspr))*(varauspr/(varauspr-1)).
for gewkolb (F8.4).
exe.

*kolb alpha.
compute kolbalph=(1-(1-uecodant)/(prodrand*2/(randges*(randges-3)))).
for kolbalph (f8.4).
exe.
```

Für Rückfragen und Hilfen: kolb@uni-leipzig.de