User-written Stata Program: agrm

Computing Agreement on Ordered Rating Scales

Alejandro Ecker

University of Mannheim

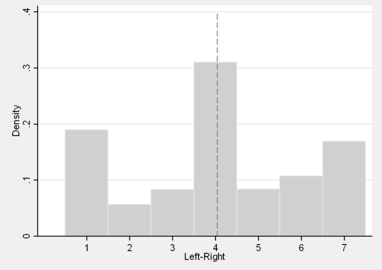
German Stata Users Group meeting
Berlin, June 25, 2010

Outline

1. The problem: Calculating agreement on ordered rating scales
2. The solution: Coefficient of agreement
3. The application: User-written program agrm

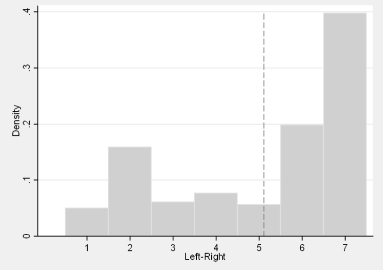
Calculating agreement on ordered rating scales

Figure: Left-right position PDS-CC



$$\text{mean} = 4.04$$
$$\text{sd} = 2.01$$
$$\text{skewness} = -0.08$$

Figure: Left-right position FPÖ



$$\text{mean} = 5.12$$
$$\text{sd} = 2.12$$
$$\text{skewness} = -0.69$$

Coefficient of agreement (van der Eijk, 2001)

Table: Disaggregation of frequency distribution into layers

category frequencies	1 50	2 159	3 61	4 77	5 57	6 198	7 399	weights
	50	50	50	50	50	50	50	0.350
	0	7	7	7	7	7	7	0.042
	0	4	4	4	0	4	4	0.020
	0	16	0	16	0	16	16	0.064
	0	82	0	0	0	82	82	0.246
	0	0	0	0	0	39	39	0.078
	0	0	0	0	0	0	201	0.201
$\sum$	50	159	61	77	57	198	399	1.000

Coefficient of agreement (van der Eijk, 2001)

Table: Disaggregation of frequency distribution into layers

category frequencies	1 50	2 159	3 61	4 77	5 57	6 198	7 399	weights
	50	50	50	50	50	50	50	0.350
	0	7	7	7	7	7	7	0.042
	0	4	4	4	0	4	4	0.020
	0	16	0	16	0	16	16	0.064
	0	82	0	0	0	82	82	0.246
	0	0	0	0	0	39	39	0.078
	0	0	0	0	0	0	201	0.201
$\sum$	50	159	61	77	57	198	399	1.000

Coefficient of agreement (van der Eijk, 2001)

Table: Disaggregation of frequency distribution into layers

category	1	2	3	4	5	6	7	weights
frequencies	50	159	61	77	57	198	399	
	50	50	50	50	50	50	50	0.350
	0	7	7	7	7	7	7	0.042
	0	4	4	4	0	4	4	0.020
	0	16	0	16	0	16	16	0.064
	0	82	0	0	0	82	82	0.246
	0	0	0	0	0	39	39	0.078
	0	0	0	0	0	0	201	0.201
$\sum$	50	159	61	77	57	198	399	1.000

Coefficient of agreement (van der Eijk, 2001)

Table: Disaggregation of frequency distribution into layers

category	1	2	3	4	5	6	7	weights
frequencies	50	159	61	77	57	198	399	
	50	50	50	50	50	50	50	0.350
	0	7	7	7	7	7	7	0.042
	0	4	4	4	0	4	4	0.020
	0	16	0	16	0	16	16	0.064
	0	82	0	0	0	82	82	0.246
	0	0	0	0	0	39	39	0.078
	0	0	0	0	0	0	201	0.201
$\sum$	50	159	61	77	57	198	399	1.000

Coefficient of agreement (van der Eijk, 2001)

Table: Disaggregation of frequency distribution into layers

category	1	2	3	4	5	6	7	weights
frequencies	50	159	61	77	57	198	399	
	1	1	1	1	1	1	1	0.350
	0	1	1	1	1	1	1	0.042
	0	1	1	1	0	1	1	0.020
	0	1	0	1	0	1	1	0.064
	0	1	0	0	0	1	1	0.246
	0	0	0	0	0	1	1	0.078
	0	0	0	0	0	0	1	0.201
$\sum$	-	-	-	-	-	-	-	1.000

Coefficient of agreement (van der Eijk, 2001)

Table: Disaggregation of frequency distribution into layers

category frequencies	1 50	2 159	3 61	4 77	5 57	6 198	7 399	agreement
no agreement	1	1	1	1	1	1	1	0
	0	1	1	1	1	1	1	-
	0	1	1	1	0	1	1	-
	0	1	0	1	0	1	1	-
	0	1	0	0	0	1	1	-
	0	0	0	0	0	1	1	-
	0	0	0	0	0	0	1	-
$\sum$	-	-	-	-	-	-	-	-

Coefficient of agreement (van der Eijk, 2001)

Table: Disaggregation of frequency distribution into layers

category frequencies	1 50	2 159	3 61	4 77	5 57	6 198	7 399	agreement
no agreement	1	1	1	1	1	1	1	0
	0	1	1	1	1	1	1	-
	0	1	1	1	0	1	1	-
	0	1	0	1	0	1	1	-
	0	1	0	0	0	1	1	-
	0	0	0	0	0	1	1	-
perfect agreement	0	0	0	0	0	0	1	1
$\sum$	-	-	-	-	-	-	-	-

Coefficient of agreement (van der Eijk, 2001)

Agreement for unimodal distributions

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	1	1	1	1	1

$$A = 1 - \frac{S-1}{K-1} \approx 0.17$$

S: non-empty categories

K: total number of categories

Measure of unimodality ‘U’

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	0	0	0	1	1

$$U = \frac{(K-2)*TU-(K-1)*TDU}{(K-2)*(TU+TDU)}$$

TU: triples conforming to unimodality

TDU: triples deviating from unimodality

Coefficient of agreement (van der Eijk, 2001)

Agreement for unimodal distributions

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	1	1	1	1	1

$$A = 1 - \frac{S-1}{K-1} \approx 0.17$$

S: non-empty categories
K: total number of categories

Measure of unimodality ‘U’

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	0	0	0	1	1

$$U = \frac{(K-2) * TU - (K-1) * TDU}{(K-2) * (TU + TDU)}$$

TU: triples conforming to unimodality
TDU: triples deviating from unimodality

Coefficient of agreement (van der Eijk, 2001)

Agreement for unimodal distributions

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	1	1	1	1	1

$$A = 1 - \frac{S-1}{K-1} \approx 0.17$$

S: non-empty categories
K: total number of categories

Measure of unimodality ‘U’

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	0	0	0	1	1

$$U = \frac{(K-2) * TU - (K-1) * TDU}{(K-2) * (TU + TDU)}$$

TU: triples conforming to unimodality
TDU: triples deviating from unimodality

Coefficient of agreement (van der Eijk, 2001)

Agreement for unimodal distributions

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	1	1	1	1	1

$$A = 1 - \frac{S-1}{K-1} \approx 0.17$$

S: non-empty categories
K: total number of categories

Measure of unimodality 'U'

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	0	0	0	1	1

$$U = \frac{(K-2) * TU - (K-1) * TDU}{(K-2) * (TU + TDU)}$$

TU: triples conforming to unimodality
TDU: triples deviating from unimodality

Coefficient of agreement (van der Eijk, 2001)

Agreement for unimodal distributions

label category	A 1	B 2	C 3	D 4	E 5	F 6	G 7
	0	1	1	1	1	1	1

$$A = 1 - \frac{S-1}{K-1} \approx 0.17$$

S: non-empty categories
K: total number of categories

Coefficient of agreement ‘A’

label category	A 1	B 2	C 3	D 4	E 5	F 6	G 7
	0	1	0	0	0	1	1

$$A = U * \left(1 - \frac{S-1}{K-1}\right) \approx -0.07$$

User-written program agrm

Agrm command

1. disaggregates frequency distribution into K layers
2. calculates S, TU, and TDU
3. computes U and A

Syntax

```
[by:varlist] agrm varlist [if] [in] [weight] [, options]
```

Options

- `generate(`*newvar*`)`: creates variable newvar with values of A
- `bounds(`*numlist*`)`: customizes lower and upper bounds
- `detail`: displays additional statistics
- `noprint`: suppresses output

Disaggregating frequency distribution

```
tab ‘‘‘i’’_‘touse’’..., matcell(‘freq’)...
                ⋮
mata: disaggr("varfreq",...)
                ⋮
mata:
void function disaggr(matrix varfreq,...)
{
   layer0 = st_matrix("varfreq")
                ⋮
   layer0 = editvalue(layer0,0,.)
   layer1 = J(rows(1), cols(layer0), ///
   rowmin(layer0))
   layer2 = layer0-layer1
   st_matrix("layer1_st", layer1)
}
end
```

```
varfreq[1,7]
   c1  c2 c3 c4 c5  c6  c7
r1 50 159 61 77 57 198 399
```

```
    1  2  3  4  5  6   7
  +-----------------------+
1 | 50 50 50 50 50 50 50  |
  +-----------------------+
    1  2  3  4  5  6   7
  +-----------------------+
1 | 0 109 11 27 7 148 349 |
  +-----------------------+
```

Disaggregating frequency distribution

```
tab ‘‘‘i’’_‘touse’’..., matcell(‘freq’)...
                  ⋮
mata: disaggr("varfreq",...)
                  ⋮
mata:
void function disaggr(matrix varfreq,...)
{
   layer0 = st_matrix("varfreq")
                  ⋮
   layer0 = editvalue(layer0,0,.)
   layer1 = J(rows(1), cols(layer0), ///
   rowmin(layer0))
   layer2 = layer0-layer1
   st_matrix("layer1_st", layer1)
}
end
```

```
varfreq[1,7]
    c1   c2  c3  c4  c5   c6   c7
r1  50  159  61  77  57  198  399
```

```
      1   2   3   4   5   6    7
   +------------------------+
1  | 50  50  50  50  50  50  50  |
   +------------------------+
      1   2   3   4   5   6    7
   +------------------------+
1  | 0  109  11  27  7  148  349 |
   +------------------------+
```

Disaggregating frequency distribution

```
tab ‘‘‘i’’_‘touse’’..., matcell(‘freq’)...
                  ⋮
mata: disaggr("varfreq",...)
                  ⋮
mata:
void function disaggr(matrix varfreq,...)
{
   layer0 = st_matrix("varfreq")
                  ⋮
   layer0 = editvalue(layer0,0,.)
   layer1 = J(rows(1), cols(layer0), ///
   rowmin(layer0))
   layer2 = layer0-layer1
   st_matrix("layer1_st", layer1)
}
end
```

```
varfreq[1,7]
   c1  c2  c3  c4  c5  c6  c7
r1 50 159  61  77  57 198 399
```

```
       1    2    3    4    5    6     7
    +------------------------------------+
1   |  50   50   50   50   50   50   50  |
    +------------------------------------+
       1    2    3    4    5    6     7
    +------------------------------------+
1   |  0   109   11   27   7   148  349  |
    +------------------------------------+
```

Disaggregating frequency distribution

```
tab ‘‘‘i’’_‘touse’’..., matcell(‘freq’)...
                  ⋮
mata: disaggr("varfreq",...)
                  ⋮
mata:
void function disaggr(matrix varfreq,...)
{
   layer0 = st_matrix("varfreq")
                  ⋮
   layer0 = editvalue(layer0,0,.)
   layer1 = J(rows(1), cols(layer0), ///
   rowmin(layer0))
   layer2 = layer0-layer1
   st_matrix("layer1_st", layer1)
}
end
```

```
varfreq[1,7]
   c1  c2  c3  c4  c5  c6  c7
r1 50 159  61  77  57 198 399
```

```
     1   2   3   4   5   6    7
  +-------------------------+
1 | 50  50  50  50  50  50  50  |
  +-------------------------+
     1   2   3   4   5   6    7
  +-------------------------+
1 | 0 109  11  27   7 148 349 |
  +-------------------------+
```

Disaggregating frequency distribution

```
tab ‘‘‘i’’_‘touse’’..., matcell(‘freq’)...
                 ⋮
mata: disaggr("varfreq",...)
                 ⋮
mata:
void function disaggr(matrix varfreq,...)
{
   layer0 = st_matrix("varfreq")
                 ⋮
   layer0 = editvalue(layer0,0,.)
   layer1 = J(rows(1), cols(layer0), ///
   rowmin(layer0))
   layer2 = layer0-layer1
   st_matrix("layer1_st", layer1)
}
end
```

```
varfreq[1,7]
   c1  c2  c3 c4 c5  c6  c7
r1 50 159 61 77 57 198 399
```

```
    1  2  3  4  5  6   7
  +------------------------+
1 | 50 50 50 50 50 50 50   |
  +------------------------+
    1  2  3  4  5  6   7
  +------------------------+
1 | 0 109 11 27 7 148 349  |
  +------------------------+
```

Disaggregating frequency distribution

```
tab ‘‘‘i’’_‘touse’’..., matcell(‘freq’)...
                 ⋮
mata: disaggr("varfreq",...)
                 ⋮
mata:
void function disaggr(matrix varfreq,...)
{
   layer0 = st_matrix("varfreq")
                 ⋮
   layer0 = editvalue(layer0,0,.)
   layer1 = J(rows(1), cols(layer0), ///
   rowmin(layer0))
   layer2 = layer0-layer1
   st_matrix("layer1_st", layer1)
}
end
```

```
varfreq[1,7]
   c1  c2  c3  c4  c5  c6  c7
r1 50 159  61  77  57 198 399
```

```
     1   2   3   4   5   6    7
  +-------------------------+
1 | 50  50  50  50  50  50  50  |
  +-------------------------+
     1   2   3   4   5   6    7
  +-------------------------+
1 | 0  109  11  27  7  148 349 |
  +-------------------------+
```

Calculating TU and TDU

```
forvalues a = 1/‘cat’ {
               ⋮
   forvalues b = 2/‘cat’ {
      if ‘b’==2 {
      local k = 1
      }
      forvalues c = 3/‘cat’ {
         if ‘c’==3 {
         local l=1
         }
         if ‘c’==‘b’ {
            continue
         }
               ⋮
         if ‘l’<‘k’ {
            continue
         }
```

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	1	1	0	1	1

→ ignore triples like BBE, EEE, . . .

→ ignore triples like GAB, EBG, . . .

Calculating TU and TDU

```
forvalues a = 1/‘cat’ {
                 ⋮
   forvalues b = 2/‘cat’ {
      if ‘b’==2 {
      local k = 1
      }
      forvalues c = 3/‘cat’ {
         if ‘c’==3 {
         local l=1
         }
         if ‘c’==‘b’ {
            continue
         }
                 ⋮
         if ‘l’<‘k’ {
            continue
         }
```

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	1	1	0	1	1

→ ignore triples like BBE, EEE, . . .

→ ignore triples like GAB, EBG, . . .

Calculating TU and TDU

```
forvalues a = 1/‘cat’ {
                  ⋮
   forvalues b = 2/‘cat’ {
      if ‘b’==2 {
      local k = 1
      }
      forvalues c = 3/‘cat’ {
         if ‘c’==3 {
         local l=1
         }
         if ‘c’==‘b’ {
            continue
         }
                  ⋮
         if ‘l’<‘k’ {
            continue
         }
```

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
	0	1	1	1	0	1	1

→ ignore triples like BBE, EEE, . . .

→ ignore triples like GAB, EBG, . . .

Calculating TU and TDU

```
matrix triple`x'_`a'`b'`c' = J(1,3,0)
matrix triple`x'_`a'`b'`b'[1,1] = layer`x'_st[1,`a']
matrix triple`x'_`a'`b'`b'[1,2] = layer`x'_st[1,`b']
matrix triple`x'_`a'`b'`b'[1,3] = layer`x'_st[1,`c']
```

```
if rowmiss`x'_`a'`b'`c' != 1 {
   continue
}
if triple`x'_`a'`b'`c'[1,2] ==. {
   local ++tdu_`x'
}
else {
   local ++tu_`x'
}
```

```
triple3_456[1,3]
    c1  c2  c3
r1   4   .   4
```

- center category missing → TDU
- first category missing → TU
- last category missing → TU

Calculating TU and TDU

```
matrix triple‘x’_‘a’‘b’‘c’ = J(1,3,0)
matrix triple‘x’_‘a’‘b’‘b’[1,1] = layer‘x’_st[1,‘a’]
matrix triple‘x’_‘a’‘b’‘b’[1,2] = layer‘x’_st[1,‘b’]
matrix triple‘x’_‘a’‘b’‘b’[1,3] = layer‘x’_st[1,‘c’]
```

```
if rowmiss‘x’_‘a’‘b’‘c’ != 1 {
   continue
}
if triple‘x’_‘a’‘b’‘c’[1,2] ==. {
   local ++tdu_‘x’
}
else {
   local ++tu_‘x’
}
```

```
triple3_456[1,3]
    c1  c2  c3
r1   4   .   4
```

- center category missing → TDU
- first category missing → TU
- last category missing → TU

Calculating TU and TDU

```
matrix triple‘x’_‘a’‘b’‘c’ = J(1,3,0)
matrix triple‘x’_‘a’‘b’‘b’[1,1] = layer‘x’_st[1,‘a’]
matrix triple‘x’_‘a’‘b’‘b’[1,2] = layer‘x’_st[1,‘b’]
matrix triple‘x’_‘a’‘b’‘b’[1,3] = layer‘x’_st[1,‘c’]
```

```
if rowmiss‘x’_‘a’‘b’‘c’ != 1 {
   continue
}
if triple‘x’_‘a’‘b’‘c’[1,2] ==. {
   local ++tdu_‘x’
}
else {
   local ++tu_‘x’
}
```

```
triple3_456[1,3]
    c1  c2  c3
r1   4   .   4
```

- center category missing → TDU
- first category missing → TU
- last category missing → TU

Calculating TU and TDU

```
matrix triple‘x’_‘a’‘b’‘c’ = J(1,3,0)
matrix triple‘x’_‘a’‘b’‘b’[1,1] = layer‘x’_st[1,‘a’]
matrix triple‘x’_‘a’‘b’‘b’[1,2] = layer‘x’_st[1,‘b’]
matrix triple‘x’_‘a’‘b’‘b’[1,3] = layer‘x’_st[1,‘c’]
```

```
if rowmiss‘x’_‘a’‘b’‘c’ != 1 {
   continue
}
if triple‘x’_‘a’‘b’‘c’[1,2] ==. {
   local ++tdu_‘x’
}
else {
   local ++tu_‘x’
}
```

```
triple3_456[1,3]
     c1  c2  c3
r1    4   .   4
```

- center category missing → TDU
- first category missing → TU
- last category missing → TU

Calculating TU and TDU

```
matrix triple`x'_`a'`b'`c' = J(1,3,0)
matrix triple`x'_`a'`b'`b'[1,1] = layer`x'_st[1,`a']
matrix triple`x'_`a'`b'`b'[1,2] = layer`x'_st[1,`b']
matrix triple`x'_`a'`b'`b'[1,3] = layer`x'_st[1,`c']
```

```
if rowmiss`x'_`a'`b'`c' != 1 {
   continue
}
if triple`x'_`a'`b'`c'[1,2] ==. {
   local ++tdu_`x'
}
else {
   local ++tu_`x'
}
```

```
triple3_456[1,3]
       c1   c2   c3
r1      4    .    4
```

- center category missing → TDU
- first category missing → TU
- last category missing → TU

Numerical missing values

label	A	B	C	D	E	F	G	...	DK	NA
category	1	2	3	4	5	6	7	...	98	99
frequencies	50	159	61	77	57	198	399	...	87	102

Option missing

```
[by: varlist] agrm varlist [if] [in] [weight] , missing(numlist)
```

```
if "‘missing’" != "" {
     local misscat: byword count ‘missing’
     forvalues a=1/‘misscat’ {
          local misscat‘a’: word ‘a’ of ‘missing’
          mvdecode ‘‘‘i’’_‘touse’’, mv(‘misscat‘a’’)
     }
}
```

Numerical missing values

label	A	B	C	D	E	F	G	...	DK	NA
category	1	2	3	4	5	6	7	...	98	99
frequencies	50	159	61	77	57	198	399	...	87	102

Option missing

[by: *varlist*] agrm *varlist* [if] [in] [*weight*] , missing(*numlist*)

```
if "‘missing’" != "" {
     local misscat: byword count ‘missing’
     forvalues a=1/‘misscat’ {
          local misscat‘a’: word ‘a’ of ‘missing’
          mvdecode ‘‘‘i’’_‘touse’’, mv(‘misscat‘a’’)
     }
}
```

Numerical missing values

label	A	B	C	D	E	F	G	...	DK	NA
category	1	2	3	4	5	6	7	...	98	99
frequencies	50	159	61	77	57	198	399	...	87	102

Option missing

[by: *varlist*] agrm *varlist* [if] [in] [*weight*] , missing(*numlist*)

```
if "‘missing’" != "" {
     local misscat: byword count ‘missing’
     forvalues a=1/‘misscat’ {
          local misscat‘a’: word ‘a’ of ‘missing’
          mvdecode ‘‘‘i’’_‘touse’’, mv(‘misscat‘a’’)
     }
}
```

Numerical missing values

label	A	B	C	D	E	F	G	...	DK	NA
category	1	2	3	4	5	6	7	...	98	99
frequencies	50	159	61	77	57	198	399	...	87	102

Option missing

[by:*varlist*] agrm *varlist* [if] [in] [*weight*] , missing(*numlist*)

```
if "‘missing’" != "" {
     local misscat: byword count ‘missing’
     forvalues a=1/‘misscat’ {
          local misscat‘a’: word ‘a’ of ‘missing’
          mvdecode ‘‘‘i’’_‘touse’’, mv(‘misscat‘a’’)
     }
}
```

Empty categories

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
frequencies	50	159	0	77	57	198	399

Option categories

[by:*varlist*] agrm *varlist* [if] [in] [*weight*] , categories(*integer*)

```
if ‘pos_‘x’’>‘cat’ {
     di "{err} specify number of categories"
     exit 0
}
if "‘categories’" != "" {
     local cat = ‘categories’
}
```

Empty categories

label	A	B	C	D	E	F	G
category	1	2	3	4	5	6	7
frequencies	50	159	0	77	57	198	399

Option categories

[by:*varlist*] agrm *varlist* [if] [in] [*weight*] , categories(*integer*)

```
if ‘pos_‘x’’>‘cat’ {
     di "{err} specify number of categories"
     exit 0
}
if "‘categories’" != "" {
     local cat = ‘categories’
}
```

Empty categories

label	A	B	C	D	E	F	G	H	I	J
category	1	2	3	4	5	6	7	8	9	10
frequencies	50	159	0	77	57	198	399	0	0	0

Option categories

[by: *varlist*] agrm *varlist* [if] [in] [*weight*] , categories(*integer*)

```
if ‘pos_‘x’’>‘cat’ {
     di "{err} specify number of categories"
     exit 0
}
if "‘categories’" != "" {
     local cat = ‘categories’
}
```

Empty categories

label	A	B	C	D	E	F	G	H	I	J
category	1	2	3	4	5	6	7	8	9	10
frequencies	50	159	61	77	57	198	399	0	0	0

Option categories

[by:*varlist*] agrm *varlist* [if] [in] [*weight*] , categories(*integer*)

```
if ‘pos_‘x’’>‘cat’ {
     di "{err} specify number of categories"
     exit 0
}
if "‘categories’" != "" {
     local cat = ‘categories’
}
```

Concluding Remarks

Coefficient of Agreement 'A'

- not based on standard deviation
- fixed upper and lower bounds
- comparability across rating scales
- easily interpretable values

User-written program agrm

- computation of coefficient of agreement 'A'
- byable, fweights, variable containing values
- handling of numerical missing values
- handling of empty categories

Thank you for your attention!