

Krippendorff's alpha for inter-rater reliability

We calculate Krippendorff's alpha for three dummy examples:

1. Perfect Agreement
2. Systematic Disagreement
3. Wikipedia Example (https://en.wikipedia.org/wiki/Krippendorff%27s_alpha)

and the following student rating datasets:

1. Outcomes rated by all students
2. Outcomes rated by UG students
3. Outcomes rated by PG students
4. Paths rated by all students
5. Paths rated by UG students
6. Paths rated by PG students

For each dataset above, we report:

- Krippendorff's alpha for the whole dataset (entire sample)

Additionally, for the student rating datasets, we report:

- The mean Krippendorff's alpha of 1000 bootstrap samples (sampling with replacement), where each bootstrap sample has the same dimensions as the entire sample
- The 95% Confidence Interval calculated using the 1000 bootstrap samples

Prior to each report, a preview of the dataset is provided (top 5 rows), the unique ratings/outcomes are inspected, and, where applicable, the "dirty" ratings (inconsistent with expected possible ratings) are replaced by either a NaN value (no rating) or a value inferred from the "dirty" rating. The replaced values are reported in the format { 'X' : 'x' , 'Y' : 'y' }, where 'X' and 'Y' are the original ratings replaced by 'x' and 'y' respectively. Also, we check that the condition for reliability data is satisfied.

A summary of the results for the six datasets (4. to 9. above) is provided at the end of the document.

Note: For the two PG datasets, the units are extremely sparsely rated; bootstrapping will have the effect of introducing artificial agreement. So, don't take the confidence intervals too seriously there; I include them nevertheless.

Preliminaries

- Some helper functions for calculation of statistics and data treatment
- Creation of dummy examples
- Reading of datasets from Excel

In [1]:

```
# Some helper functions
import pandas as pd
import numpy as np
```

```

def find_unique_values(df):
    uniq=list(np.unique([item for sublist in [list(df[col].dropna().unique()) for col in df.columns]))
    return uniq
def map_inconsistent_values(df,expected_values,mapping):
    others=[]
    for col in df.columns:
        for row in range(len(df)):
            value=df[col].iloc[row]
            if value not in expected_values+[np.nan]:
                others.append([row,col,value])
    for entry in range(len(others)):
        fix_list=others[entry]
        df[fix_list[1]].iloc[fix_list[0]]=mapping[fix_list[2]]
    print('Here are the inconsistent values and their replacements:')
    print(mapping)
    return df
def canonical_reliability_check(data):
    if data.notnull().sum().min()>=2:
        print('Reliability data condition satisfied')
    else:
        print("The following units don't have pairable values and will be omitted: ")
        print(list(data.notnull().sum().to_frame()[data.notnull().sum()<2].index))
from sklearn.utils import resample
import krippendorff
def krippendorff_summary(data,confidence_interval=0.95,bootstrap_iterations=1000,bootstrap_resample=True):
    dictionary={}
    uniques=find_unique_values(data)
    for i in range(len(uniques)):
        dictionary[uniques[i]]=i+1
    data_mapped=data.replace(dictionary)
    print('Number of units rated (excluding units without pairable values):',data.dropna().shape[0])
    print('Number of raters:',data.shape[0])
    print('Total ratings made: ',data.notnull().sum().sum())
    print('Total ratings made (excluding unpaired values): ',data.dropna(thresh=2,axis=1).shape[0])
    print('Number of possible outcomes/ratings:',len(uniques))
    print('The possible outcomes/ratings are:')
    print(uniques)
    kripp=krippendorff.alpha(reliability_data=data_mapped.to_numpy(),level_of_measurement='nominal')
    print("Krippendorff's alpha of entire sample =",round(kripp,3))
    stats = list()
    if bootstrap:
        for i in range(bootstrap_iterations):
            stats.append(krippendorff.alpha(reliability_data=resample(data_mapped,replace=True,random_state=i),level_of_measurement='nominal'))
        print("Mean Krippendorff's alpha of",bootstrap_iterations,"bootstrapped samples")
        p = ((1.0-confidence_interval)/2.0) * 100
        lower = max(0.0, np.percentile(stats, p))
        p = (confidence_interval+((1.0-confidence_interval)/2.0)) * 100
        upper = min(1.0, np.percentile(stats, p))
        CI=[round(lower,3),round(upper,3)]
        print(int(confidence_interval*100),'% Confidence Interval:',CI)

```

In [2]:

```

# Creation of dummy examples
perfect_agreement=pd.DataFrame({'1':['X','X',np.nan], '2':['X',np.nan,'X'], '3':[np.nan,'X',np.nan], '4':['Y',np.nan,'Y']},index=['A','B','C'])
systematic_disagreement=pd.DataFrame({'1':['X','Y',np.nan], '2':['X',np.nan,'Y'], '3':[np.nan,'Y',np.nan], '4':['Y',np.nan,'X']},index=['A','B','C'])
wikipedia_example=pd.DataFrame({'1':[np.nan,'1',np.nan], '2':[np.nan,np.nan,np.nan], '3':[np.nan,'2',np.nan], '4':[np.nan,'3',np.nan], '5':[np.nan,'3','3'], '6':['3','3','4'], '7':['4','4','5'], '8':['5','5','6'], '9':['2',np.nan,'2'], '10':['1',np.nan,'1'], '11':['1','2','3'], '12':['2','3','4'], '13':['3',np.nan,'3'], '14':[np.nan,np.nan,np.nan]},index=['A','B','C'])

```

In [3]:

```
# Reading excel datasets
outcomes_all=pd.read_csv('Outcome_data.csv',delimiter=';')
outcomes_all=outcomes_all.set_index('Participant')

outcomes_ug=pd.read_csv('Outcome_data_UG.csv',delimiter=';')
outcomes_ug=outcomes_ug.set_index('Participant')

outcomes_pg=pd.read_csv('Outcome_data_PG.csv',delimiter=';')
outcomes_pg=outcomes_pg.set_index('Participant')

paths_all=pd.read_csv('Path_data.csv',delimiter=';')
paths_all=paths_all.set_index('Participant')

paths_ug=pd.read_csv('Path_data_UG.csv',delimiter=';')
paths_ug=paths_ug.set_index('Participant')

paths_pg=pd.read_csv('Path_data_PG.csv',delimiter=';')
paths_pg=paths_pg.set_index('Participant')
```

Dummy Examples

1. Perfect Agreement

This example is constructed to show how perfect agreement among raters (rather absence of disagreement in Krippendorff's formulation) corresponds to an alpha value of 1.

The preview below shows the ratings of three raters 'A','B', and 'C' that have rated four units 1,2,3,4, where the possible nominal ratings are 'X' and 'Y'. The 'NaN' entries indicate where no rating was made.

In [4]: `perfect_agreement.head()`

```
Out[4]:
```

	1	2	3	4
A	X	X	NaN	Y
B	X	NaN	Y	NaN
C	NaN	X	Y	Y

In [5]: `find_unique_values(perfect_agreement)`

Out[5]: ['X', 'Y']

In [6]: `canonical_reliability_check(perfect_agreement)`

Reliability data condition satisfied

In [7]: `krippendorff_summary(perfect_agreement,bootstrap=False)`

```
Number of units rated (excluding units without pairable values): 4
Number of raters: 3
Total ratings made: 8
Total ratings made (excluding unpaired values): 8
Number of possible outcomes/ratings: 2
The possible outcomes/ratings are:
```

```
['X', 'Y']
Krippendorff's alpha of entire sample = 1.0
```

2. Systematic Disagreement

This example is constructed to show how systematic disagreement among raters corresponds to a negative alpha value.

```
In [8]: systematic_disagreement.head()
```

Out[8]:

	1	2	3	4
A	X	X	NaN	Y
B	Y	NaN	Y	NaN
C	NaN	Y	X	X

```
In [9]: find_unique_values(systematic_disagreement)
```

Out[9]: ['X', 'Y']

```
In [10]: canonical_reliability_check(systematic_disagreement)
```

Reliability data condition satisfied

```
In [11]: krippendorff_summary(systematic_disagreement,bootstrap=False)
```

Number of units rated (excluding units without pairable values): 4
Number of raters: 3
Total ratings made: 8
Total ratings made (excluding unpaired values): 8
Number of possible outcomes/ratings: 2
The possible outcomes/ratings are:
['X', 'Y']
Krippendorff's alpha of entire sample = -0.75

3. Wikipedia Example

Note that the ratings 1,2,3,4 are treated as nominal ratings in this example. Alos, the units that do not satisfy the reliability data condition are omitted from the dataset before calculation of the alpha value.

```
In [12]: wikipedia_example.head()
```

Out[12]:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	NaN	NaN	NaN	NaN	NaN	3	4	1	2	1	1	3	3	NaN	3
B	1	NaN	2	1	3	3	4	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN
C	NaN	NaN	2	1	3	4	4	NaN	2	1	1	3	3	NaN	4

```
In [13]: find_unique_values(wikipedia_example)
```

Out[13]: ['1', '2', '3', '4']

```
In [14]: canonical_reliability_check(wikipedia_example)
```

The following units don't have pairable values and will be omitted:
['1', '2', '14']

```
In [15]: krippendorff_summary(wikipedia_example,bootstrap=False)
```

Number of units rated (excluding units without pairable values): 12
Number of raters: 3
Total ratings made: 27
Total ratings made (excluding unpaired values): 26
Number of possible outcomes/ratings: 4
The possible outcomes/ratings are:
['1', '2', '3', '4']
Krippendorff's alpha of entire sample = 0.691

Student Rating Datasets

4. Outcomes rated by all students

```
In [16]: outcomes_all.head()
```

Out[16]:

	BaCe	BaNe	BaSe	BeCi	BeHg	BeHj	Beno	BeSo	BhCe	BhGe	...	SaKu	SaLe	SaN
Participant														
12167348	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
12123294	V	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
10572482	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	V	...	NaN	NaN	Na
16059222	V	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
16154313	NaN	NaN	V	NaN	NaN	NaN	NaN	V	NaN	NaN	...	NaN	NaN	Na

5 rows × 135 columns

```
In [17]: find_unique_values(outcomes_all)
```

Out[17]: ['- ', 'A', 'D', 'I', 'I ', 'I411', 'S', 'V', 'i']

```
In [18]: outcomes_all=map_inconsistent_values(outcomes_all,expected_values=['V','A','D','I'],mapping={'I ':'I','I411':'I','i':'I','- ':nan})
```

Here are the inconsistent values and their replacements:
{'I ': 'I', 'I411': 'I', 'i': 'I', '- ': nan, 'S': nan}

```
In [19]: find_unique_values(outcomes_all)
```

Out[19]: ['A', 'D', 'I', 'V']

```
In [20]: canonical_reliability_check(outcomes_all)
```

Reliability data condition satisfied

```
In [21]: krippendorf_summary(outcomes_all,confidence_interval=0.95,bootstrap_iterations=1000,

Number of units rated (excluding units without pairable values): 135
Number of raters: 292
Total ratings made: 4076
Total ratings made (excluding unpaired values): 4076
Number of possible outcomes/ratings: 4
The possible outcomes/ratings are:
['A', 'D', 'I', 'V']
Krippendorff's alpha of entire sample = 0.65
Mean Krippendorff's alpha of 1000 bootstrapped samples = 0.663
95 % Confidence Interval: [0.634, 0.693]
```

5. Outcomes rated by UG students

```
In [22]: outcomes_ug.head()
```

Out[22]:

	BaCe	BaNe	BaSe	BeCi	BeHg	BeHj	Beno	BeSo	BhCe	BhGe	...	SaKu	SaLe	SaN
Participant														
12167348	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
12123294	V	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
10572482	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	V	...	NaN	NaN	Na
16059222	V	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
16154313	NaN	NaN	V	NaN	NaN	NaN	NaN	V20	NaN	NaN	...	NaN	NaN	Na

5 rows × 135 columns

```
In [23]: find_unique_values(outcomes_ug)
```

Out[23]: ['- ', '1', 'A', 'D', 'I', 'I ', 'I411', 'S', 'V', 'V20', 'V25']

```
In [24]: outcomes_ug=map_inconsistent_values(outcomes_ug,expected_values=['V','A','D','I'],
                                             ,mapping={'I ':'I','I411':'I','-':np.nan,'S':np
```

Here are the inconsistent values and their replacements:
{'I ':'I', 'I411': 'I', '-': nan, 'S': nan, '1': nan, 'V20': 'V', 'V25': 'V'}

```
In [25]: find_unique_values(outcomes_ug)
```

Out[25]: ['A', 'D', 'I', 'V']

```
In [26]: canonical_reliability_check(outcomes_ug)
```

Reliability data condition satisfied

```
In [27]: krippendorf_summary(outcomes_ug,confidence_interval=0.95,bootstrap_iterations=1000,b
```

Number of units rated (excluding units without pairable values): 135
Number of raters: 277
Total ratings made: 3865

Total ratings made (excluding unpaired values): 3865
 Number of possible outcomes/ratings: 4
 The possible outcomes/ratings are:
 ['A', 'D', 'I', 'V']
 Krippendorff's alpha of entire sample = 0.653
 Mean Krippendorff's alpha of 1000 bootstrapped samples = 0.666
 95 % Confidence Interval: [0.637, 0.694]

6. Outcomes rated by PG students

In [28]: `outcomes_pg.head()`

Out[28]:

	BaCe	BaNe	BaSe	BeCi	BeHg	BeHj	Beno	BeSo	BhCe	BhGe	...	SaKu	SaLe	SaM
Participant														
U16403160	NaN	NaN	NaN	NaN	NaN	NaN	I	NaN	NaN	D	...	NaN	NaN	NaN
U15415547	NaN	NaN	NaN	V	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
U19328932	NaN	V	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
U27040756	V	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
u22018499	NaN	V	A	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN

5 rows × 135 columns

In [29]: `find_unique_values(outcomes_pg)`

Out[29]: ['A', 'D', 'I', 'V', 'i']

In [30]: `outcomes_pg=map_inconsistent_values(outcomes_pg,expected_values=['V','A','D','I'],ma`

Here are the inconsistent values and their replacements:
 {'i': 'I'}

In [31]: `find_unique_values(outcomes_pg)`

Out[31]: ['A', 'D', 'I', 'V']

In [32]: `canonical_reliability_check(outcomes_pg)`

The following units don't have pairable values and will be omitted:
 ['BeHg', 'Beno', 'BeSo', 'BhGe', 'BhJe', 'BhKu', 'BhMe', 'BhMo', 'BiNu', 'BoMh', 'BoNi', 'BoSe', 'CiRu', 'CuGi', 'CuNa', 'FaRi', 'FeMu', 'FeRh', 'FeVi', 'FiKu', 'FoKu', 'FoRh', 'FoSe', 'FuRa', 'FuSo', 'GaTo', 'GeWi', 'HaMe', 'HeCa', 'HeKa', 'HeLa', 'HeSo', 'HuBe', 'HuCe', 'HuKu', 'JaLo', 'JiLa', 'KaSo', 'KeBh', 'KeFa', 'KeSo', 'KiLa', 'KoSe', 'KuBh', 'KuLe', 'KuSo', 'LaCu', 'LaNu', 'LuCe', 'LuJe', 'LuKo', 'MaFe', 'MuFa', 'NaPo', 'NeLu', 'NeRo', 'NoRi', 'NuRa', 'PaGo', 'PaNu', 'PeNi', 'ReBh', 'RoHu', 'SaCe', 'SaKe', 'SaKu', 'SaLe', 'SeFu', 'SeKa', 'SuKo', 'VeFa', 'WiGo', 'WoPe']

In [33]: `krippendorf_summary(outcomes_pg,confidence_interval=0.95,bootstrap_iterations=1000,b`

Number of units rated (excluding units without pairable values): 62
 Number of raters: 15
 Total ratings made: 211
 Total ratings made (excluding unpaired values): 138

Number of possible outcomes/ratings: 4
 The possible outcomes/ratings are:
 ['A', 'D', 'I', 'V']
 Krippendorff's alpha of entire sample = 0.644
 Mean Krippendorff's alpha of 1000 bootstrapped samples = 0.855
 95 % Confidence Interval: [0.754, 0.952]

7. Paths rated by all students

In [34]: `paths_all.head()`

Out[34]:

	BaCe	BaNe	BaSe	BeCi	BeHg	BeHj	Beno	BeSo	BhCe	BhGe	...	SaKu	SaLe	SaNa
Participant														
12167348	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
12123294	V20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
10572482	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	V20	...	NaN	NaN	NaN
16059222	V20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
16154313	NaN	NaN	V20	NaN	NaN	NaN	NaN	V20	NaN	NaN	...	NaN	NaN	NaN

5 rows × 135 columns

In [35]: `print(find_unique_values(paths_all))`

```
['-', '30', 'A', 'A30', 'A31', 'A32', 'A33', 'A34', 'A35', 'A36', 'A37', 'A38', 'D',
'D10', 'D11', 'D110', 'D111', 'D12', 'D12 ', 'D123', 'D124', 'D125', 'D126', 'D127',
'D13', 'D14', 'D15', 'D16', 'D17', 'D18', 'D19', 'I', 'I401', 'I402', 'I403', 'I40
6', 'I411', 'I412', 'I413', 'I421', 'I422', 'I423', 'I431', 'I432', 'I433', 'I441',
'I442', 'I443', 'I444', 'I451', 'I452', 'I453', 'I454', 'V', 'V11', 'V20', 'V21', 'V
210', 'V211', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'V29', 'V403', 'i43
3']
```

In [36]: `paths_all=map_inconsistent_values(paths_all,expected_values=['V20', 'A31', 'V25', 'D15',
'D13', 'D126', 'V28',
'I421', 'I411', 'I413',
'I443', 'I402', 'D17',
'A38', 'I441', 'I431',
'V22', 'V211', 'A34',
'V24', 'I454', 'V403',
,mapping={'D12 ': 'D12', 'I': np.nan, '
'V': np.nan, 'D': np.nan, 'i4`

Here are the inconsistent values and their replacements:

```
{'D12 ': 'D12', 'I': nan, '30': nan, 'A': nan, 'V': nan, 'D': nan, 'i433': 'I433',
'-': nan}
```

In [37]: `print(find_unique_values(paths_all))`

```
['A30', 'A31', 'A32', 'A33', 'A34', 'A35', 'A36', 'A37', 'A38', 'D10', 'D11', 'D11
0', 'D111', 'D12', 'D123', 'D124', 'D125', 'D126', 'D127', 'D13', 'D14', 'D15', 'D1
6', 'D17', 'D18', 'D19', 'I401', 'I402', 'I403', 'I406', 'I411', 'I412', 'I413', 'I4
21', 'I422', 'I423', 'I431', 'I432', 'I433', 'I441', 'I442', 'I443', 'I444', 'I451',
'I452', 'I453', 'I454', 'V11', 'V20', 'V21', 'V210', 'V211', 'V22', 'V23', 'V24', 'V
25', 'V26', 'V27', 'V28', 'V29', 'V403']
```

In [38]: `canonical_reliability_check(paths_all)`

Reliability data condition satisfied

```
In [39]: krippendorf_summary(paths_all,confidence_interval=0.95,bootstrap_iterations=1000,boo

Number of units rated (excluding units without pairable values): 135
Number of raters: 292
Total ratings made: 4026
Total ratings made (excluding unpaired values): 4026
Number of possible outcomes/ratings: 61
The possible outcomes/ratings are:
['A30', 'A31', 'A32', 'A33', 'A34', 'A35', 'A36', 'A37', 'A38', 'D10', 'D11', 'D11
0', 'D111', 'D12', 'D123', 'D124', 'D125', 'D126', 'D127', 'D13', 'D14', 'D15', 'D1
6', 'D17', 'D18', 'D19', 'I401', 'I402', 'I403', 'I406', 'I411', 'I412', 'I413', 'I4
21', 'I422', 'I423', 'I431', 'I432', 'I433', 'I441', 'I442', 'I443', 'I444', 'I451',
'I452', 'I453', 'I454', 'V11', 'V20', 'V21', 'V210', 'V211', 'V22', 'V23', 'V24', 'V
25', 'V26', 'V27', 'V28', 'V29', 'V403']
Krippendorff's alpha of entire sample = 0.426
Mean Krippendorff's alpha of 1000 bootstrapped samples = 0.447
95 % Confidence Interval: [0.424, 0.471]
```

8. Paths rated by UG students

```
In [40]: paths_ug.head()
```

Out[40]:

	BaCe	BaNe	BaSe	BeCi	BeHg	BeHj	Beno	BeSo	BhCe	BhGe	...	SaKu	Sale	SaN
Participant														
12167348	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
12123294	V20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
10572482	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	V20	...	NaN	NaN	Na
16059222	V20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
16154313	NaN	NaN	V20	NaN	NaN	NaN	NaN	V20	NaN	NaN	...	NaN	NaN	Na

5 rows × 135 columns

```
In [41]: print(find_unique_values(paths_ug))

['-', 'A', 'A30', 'A31', 'A32', 'A33', 'A34', 'A35', 'A36', 'A37', 'A38', 'D', 'D1
0', 'D11', 'D110', 'D111', 'D12', 'D12 ', 'D123', 'D124', 'D125', 'D126', 'D127', 'D
13', 'D14', 'D15', 'D16', 'D17', 'D18', 'D19', 'I', 'I401', 'I402', 'I403', 'I406',
'I411', 'I412', 'I413', 'I421', 'I422', 'I423', 'I431', 'I432', 'I433', 'I441', 'I44
2', 'I443', 'I444', 'I451', 'I452', 'I453', 'I454', 'V', 'V11', 'V20', 'V21', 'V21
0', 'V211', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'V29', 'V403']
```

```
In [42]: paths_ug=map_inconsistent_values(paths_ug,expected_values=['V20', 'A31', 'V25', 'D15', '
D13', 'D126', 'V28',
I421', 'I411', 'I413
I443', 'I402', 'D17'
A38', 'I441', 'I431'
V22', 'V211', 'A34',
V24', 'I454', 'V403'
,mapping={'D12 ': 'D12', 'I': np.nan, '
V': np.nan, 'D': np.nan, '-'
```

Here are the inconsistent values and their replacements:

```
{'D12 ': 'D12', 'I': nan, 'A': nan, 'V': nan, 'D': nan, '-': nan}
```

In [43]:

```
print(find_unique_values(paths_ug))
```

```
['A30', 'A31', 'A32', 'A33', 'A34', 'A35', 'A36', 'A37', 'A38', 'D10', 'D11', 'D11
0', 'D111', 'D12', 'D123', 'D124', 'D125', 'D126', 'D127', 'D13', 'D14', 'D15', 'D1
6', 'D17', 'D18', 'D19', 'I401', 'I402', 'I403', 'I406', 'I411', 'I412', 'I413', 'I4
21', 'I422', 'I423', 'I431', 'I432', 'I433', 'I441', 'I442', 'I443', 'I444', 'I451',
'I452', 'I453', 'I454', 'V11', 'V20', 'V21', 'V210', 'V211', 'V22', 'V23', 'V24', 'V
25', 'V26', 'V27', 'V28', 'V29', 'V403']
```

In [44]:

```
canonical_reliability_check(paths_ug)
```

Reliability data condition satisfied

In [45]:

```
krippendorf_summary(paths_ug,confidence_interval=0.95,bootstrap_iterations=1000,boot
```

Number of units rated (excluding units without pairable values): 135
Number of raters: 277
Total ratings made: 3816
Total ratings made (excluding unpaired values): 3816
Number of possible outcomes/ratings: 61
The possible outcomes/ratings are:
['A30', 'A31', 'A32', 'A33', 'A34', 'A35', 'A36', 'A37', 'A38', 'D10', 'D11', 'D11
0', 'D111', 'D12', 'D123', 'D124', 'D125', 'D126', 'D127', 'D13', 'D14', 'D15', 'D1
6', 'D17', 'D18', 'D19', 'I401', 'I402', 'I403', 'I406', 'I411', 'I412', 'I413', 'I4
21', 'I422', 'I423', 'I431', 'I432', 'I433', 'I441', 'I442', 'I443', 'I444', 'I451',
'I452', 'I453', 'I454', 'V11', 'V20', 'V21', 'V210', 'V211', 'V22', 'V23', 'V24', 'V
25', 'V26', 'V27', 'V28', 'V29', 'V403']
Krippendorff's alpha of entire sample = 0.43
Mean Krippendorff's alpha of 1000 bootstrapped samples = 0.451
95 % Confidence Interval: [0.426, 0.477]

9. Paths rated by PG students

In [46]:

```
paths_pg.head()
```

Out[46]:

	BaCe	BaNe	BaSe	BeCi	BeHg	BeHj	Beno	BeSo	BhCe	BhGe	...	SaKu	SaLe	SaM
Participant														
U16403160	NaN	NaN	NaN	NaN	NaN	NaN	I401	NaN	NaN	D10	...	NaN	NaN	NaN
U15415547	NaN	NaN	NaN	V20	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
U19328932	NaN	V20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	I40
U27040756	V20	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN
u22018499	NaN	V25	A32	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	I40

5 rows × 135 columns

In [47]:

```
print(find_unique_values(paths_pg))
```

```
['30', 'A30', 'A31', 'A32', 'A33', 'D10', 'D11', 'D12', 'D125', 'D14', 'D18', 'I40
1', 'I402', 'I403', 'I411', 'I412', 'I421', 'I441', 'I442', 'I452', 'V20', 'V210',
'V23', 'V25', 'V28', 'i433']
```

In [48]:

```
paths_pg=map_inconsistent_values(paths_pg,expected_values=['A30', 'A31', 'A32', 'A33', 'I
```

```
'I401','I402','I403','I41
'V20','V210','V23','V25',
,mapping={'30':np.nan,'i433':'I433'}
```

Here are the inconsistent values and their replacements:
{'30': nan, 'i433': 'I433'}

In [49]: `print(find_unique_values(paths_pg))`

```
['A30', 'A31', 'A32', 'A33', 'D10', 'D11', 'D12', 'D125', 'D14', 'D18', 'I401', 'I40
2', 'I403', 'I411', 'I412', 'I421', 'I433', 'I441', 'I442', 'I452', 'V20', 'V210',
'V23', 'V25', 'V28']
```

In [50]: `canonical_reliability_check(paths_pg)`

The following units don't have pairable values and will be omitted:
['BeHg', 'Beno', 'BeSo', 'BhGe', 'BhJe', 'BhKu', 'BhMe', 'BhMo', 'BiNu', 'BoMh', 'Bo
Ni', 'BoSe', 'CiKe', 'CiRu', 'CuGi', 'CuNa', 'FaRi', 'FeMu', 'FeRh', 'FeVi', 'FiKu',
'FoKu', 'FoRh', 'FoSe', 'FuRa', 'FuSo', 'GaTo', 'GeWi', 'HaMe', 'HeCa', 'HeKa', 'HeL
a', 'HeSo', 'HuBe', 'HuCe', 'HuKu', 'JaLo', 'JiLa', 'KaSo', 'KeBh', 'KeFa', 'KeSo',
'Kila', 'KoSe', 'KuBh', 'KuLe', 'KuSo', 'LaCu', 'LaNu', 'LuCe', 'LuJe', 'LuKo', 'MaF
e', 'MuFa', 'NaPo', 'NeLu', 'NeRo', 'NoRi', 'NuRa', 'PaGo', 'PaNu', 'PeNi', 'ReBh',
'RoHu', 'SaCe', 'SaKe', 'SaKu', 'SaLe', 'SeFu', 'SeKa', 'SuKo', 'VeFa', 'WiGo', 'WoP
e']

In [51]: `krippendorff_summary(paths_pg,confidence_interval=0.95,bootstrap_iterations=1000,boot`

Number of units rated (excluding units without pairable values): 61
Number of raters: 15
Total ratings made: 210
Total ratings made (excluding unpaired values): 136
Number of possible outcomes/ratings: 25
The possible outcomes/ratings are:
['A30', 'A31', 'A32', 'A33', 'D10', 'D11', 'D12', 'D125', 'D14', 'D18', 'I401', 'I40
2', 'I403', 'I411', 'I412', 'I421', 'I433', 'I441', 'I442', 'I452', 'V20', 'V210',
'V23', 'V25', 'V28']
Krippendorff's alpha of entire sample = 0.417
Mean Krippendorff's alpha of 1000 bootstrapped samples = 0.764
95 % Confidence Interval: [0.623, 0.899]

Summary

In [52]: `pd.DataFrame({'Dataset':['Outcomes All','Outcomes UG','Outcomes PG'],'Paths All','Pat`

Out[52]:

	Dataset	Alpha entire sample	Bootstrap mean alpha	95% Confidence Interval
0	Outcomes All	0.650	0.663	[0.634 , 0.693]
1	Outcomes UG	0.653	0.666	[0.637 , 0.694]
2	Outcomes PG	0.644	0.855	[0.754 , 0.952]
3	Paths All	0.426	0.447	[0.424 , 0.471]
4	Paths UG	0.430	0.451	[0.426 , 0.477]
5	Paths PG	0.417	0.764	[0.623 , 0.899]

In []: