```
'''
env
Python 3.10.3
scikit-learn==1.0.2
scipy==1.8.0
'''

'''
load data from localized tools and calculate correlated score
'''
import re
import os
import pandas as pd

out_dir = "./output"
os.makedirs(out_dir,exist_ok=True)

localized_data = pd.read_csv("./raw-data/raw-quantitative - localized tool.csv",encoding="utf_8_sig")
localized_data = localized_data\
    .rename(columns={x:re.search(r'.\d+',x).group(0) for x in localized_data.columns[4:]})
l=localized_data.copy()

l["B-2"]=l["L11"]/5*4
l["B-3"]=l["L11"]/5*4
l["B-4"]=(l["L3"]/2*3+l["L4"]/2*3+l["L5"])/7*4
l["B-5"]= l["L1"]*3+1
l["B-6"]= 1
l.loc[l["L2"].str.contains("A",na=False),"B-6"]=l["B-6"]+1.5
l.loc[l["L2"].str.contains("B",na=False),"B-6"]=l["B-6"]+1.5
l["B-9"]= (l["L6"]+l["L7"]/2*3+l["L8"])/8*4

localized_data=l.copy()
localized_data.to_csv(f"{out_dir}/calculated_localized_data.csv",encoding="utf_8_sig")

localized_data
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	time period	learner	assessor	double-assessment	L1	L2	L3	L4	L5	L6	...	L8	L9	L10
0	1	L-A	A-S	NaN	1	A	2	1	1	2	...	1	NaN	blood sampling,establish peripheral vascular a...
1	1	L-T	A-S	NaN	1	A	1	1	0	0	...	1	NaN	establish peripheral vascular access, make adm...
2	2	L-M	A-S	NaN	0	NaN	1	1	1	2	...	3	NaN	establish peripheral vascular access, transfer...
3	2	L-A	A-S	NaN	1	A	2	1	3	2	...	3	NaN	establish peripheral vascular access, make adm...
4	2	L-T	A-S	NaN	1	A	2	1	1	2	...	3	NaN	establish peripheral vascular access, make adm...
5	3	L-H	A-K	NaN	1	A	2	1	2	3	...	3	NaN	establish peripheral vascular access, make adm...
6	3	L-T	A-K	1.0	1	A	2	2	2	2	...	3	NaN	make initial assessment on admission
7	3	L-T	A-O	1.0	1	A	2	1	2	3	...	3	NaN	make a document related to admission
8	5	L-H	A-O	NaN	1	A	2	1	2	2	...	3	NaN	make a document related to admission
9	6	L-H	A-O	NaN	1	A	2	1	2	2	...	3	NaN	make a document related to admission
10	6	L-K	A-O	NaN	1	NaN	2	1	1	1	...	2	NaN	make a document related to admission
11	7	L-K	A-S	NaN	1	A	1	2	1	1	...	1	NaN	establish peripheral vascular access,
12	7	L-H	A-S	NaN	1	A	1	2	2	2	...	3	NaN	establish peripheral vascular access,communica...
13	8	L-K	A-K	NaN	1	A	1	1	1	2	...	3	NaN	history taking, physical examincation, make in...
14	8	L-Y	A-K	NaN	1	A	1	1	2	3	...	3	NaN	establish peripheral vascular access,make a pl...
15	9	L-KR	A-S	NaN	1	A	2	1	1	2	...	2	NaN	establish peripheral root, order blood test, c...
16	10	L-KA	A-O	NaN	1	NaN	2	2	3	3	...	3	NaN	make a document related to admission
17	10	L-W	A-O	NaN	1	NaN	2	1	1	3	...	1	NaN	make a document related to admission
18	11	L-I	A-S	NaN	1	A	2	1	2	2	...	2	NaN	establish peripheral vascular access,communica...
19	11	L-T	A-S	NaN	1	A	2	1	1	2	...	1	NaN	establish peripheral vascular access, make ini...

20 rows × 21 columns

```
'''
load data from generic ruburic
'''

import pandas as pd
import re

generic_data = pd.read_csv("./raw-data/raw-quantitative - generic rubric.csv",encoding="utf_8_sig")
generic_data
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	time period	learner	assessor	double-assessment	B-1	B-2	B-3	B-4	B-5	B-6	B-7	B-8	B-9
0	1	L-A	A-S	NaN	2.0	2.0	2.0	2.5	2.5	2.0	NaN	2.0	2.0
1	1	L-T	A-S	NaN	2.0	1.5	1.5	1.5	1.5	2.0	NaN	1.5	1.5
2	2	L-M	A-S	NaN	2.0	2.0	2.5	2.5	2.5	2.5	2.0	2.5	2.0
3	2	L-A	A-S	NaN	3.0	3.0	3.0	3.0	3.0	3.0	2.5	3.0	3.0
4	2	L-T	A-S	NaN	2.0	2.0	2.5	3.0	2.0	2.5	2.0	2.0	2.0
5	3	L-H	A-K	NaN	3.0	2.5	3.0	3.5	3.5	3.0	NaN	3.0	3.5
6	3	L-T	A-K	1.0	3.0	3.5	2.5	3.0	3.0	NaN	NaN	3.5	3.5
7	3	L-T	A-O	1.0	3.0	3.0	3.0	3.5	3.0	3.5	NaN	3.0	3.0
8	5	L-H	A-O	NaN	3.0	3.0	3.0	3.0	3.0	3.5	NaN	3.0	3.0
9	6	L-H	A-O	NaN	3.5	3.0	3.0	3.0	3.0	3.0	NaN	NaN	3.0
10	6	L-K	A-O	NaN	3.0	3.0	3.0	3.0	3.0	3.0	NaN	NaN	3.0
11	7	L-K	A-S	NaN	2.0	1.5	1.0	1.5	1.5	2.0	1.5	1.5	1.5
12	7	L-H	A-S	NaN	3.0	3.0	3.0	3.5	3.0	3.0	2.5	3.0	3.0
13	8	L-K	A-K	NaN	3.0	3.0	3.0	3.0	3.0	3.5	3.0	3.0	3.0
14	8	L-Y	A-K	NaN	3.0	3.5	3.0	3.0	3.5	3.0	3.0	3.5	3.0
15	9	L-KR	A-S	NaN	3.0	2.0	2.0	3.0	2.5	2.5	2.0	2.0	2.0
16	10	L-KA	A-O	NaN	3.0	3.0	3.0	3.0	3.0	3.0	NaN	NaN	3.0
17	10	L-W	A-O	NaN	3.0	2.5	2.5	2.5	2.5	2.5	NaN	NaN	2.5
18	11	L-I	A-S	NaN	3.0	3.0	3.0	3.0	3.0	3.0	2.5	2.5	3.5
19	11	L-T	A-S	NaN	3.0	2.5	2.5	3.0	2.0	2.5	NaN	NaN	2.5

```
'''
calculate correlation using pearsonr
'''
import pandas as pd
from scipy.stats import pearsonr

correlation_table = pd.DataFrame([],columns=["item","correlation","pvalue"])

for name in ["B-2","B-3","B-4","B-5","B-6","B-9"]:
    corr_data = pd.DataFrame({"localized":localized_data[name],"generic":generic_data[name]})
    corr_data = corr_data.dropna(axis=0,how='any')
    correlation, pvalue = pearsonr(corr_data["localized"], corr_data["generic"])
    row = pd.DataFrame({"item":[name],"correlation":[correlation],"pvalue":[pvalue]})
    correlation_table=pd.concat([correlation_table,row])

correlation_table.to_csv(f"{out_dir}/pearsonr_correlation.csv",encoding="utf_8_sig",index=False)
```

```
correlation_table
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	item	correlation	pvalue
0	B-2	0.703914	0.000533
0	B-3	0.69838	0.000615
0	B-4	0.507262	0.022435
0	B-5	0.082409	0.729793
0	B-6	0.043574	0.85941
0	B-9	0.611463	0.004174

```
'''
calculate correlation using spearmanr
'''
import pandas as pd
from scipy.stats import spearmanr

correlation_table = pd.DataFrame([],columns=["item","correlation","pvalue"])

for name in ["B-2","B-3","B-4","B-5","B-6","B-9"]:
    corr_data = pd.DataFrame({"localized":localized_data[name],"generic":generic_data[name]})
    corr_data = corr_data.dropna(axis=0,how='any')
    correlation, pvalue = spearmanr(corr_data["localized"], corr_data["generic"])
    row = pd.DataFrame({"item":[name],"correlation":[correlation],"pvalue":[pvalue]})
    correlation_table=pd.concat([correlation_table,row])

correlation_table.to_csv(f"{out_dir}/spearmanr_correlation.csv",encoding="utf_8_sig",index=False)
correlation_table
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	item	correlation	pvalue
0	B-2	0.641968	0.002276
0	B-3	0.589217	0.006262
0	B-4	0.543306	0.013296
0	B-5	0.170996	0.471023
0	B-6	0.074463	0.761918
0	B-9	0.536465	0.014746

```
'''
calculate cohen kappa in generic rubric
'''


from sklearn.metrics import cohen_kappa_score
generic_scores = generic_data.loc[~generic_data["double-assessment"].isna(),:]\
    .iloc[:,4:]\
    .dropna(how='any', axis=1)\
    .applymap(lambda x:int(x*10))
generic_kappa=cohen_kappa_score(generic_scores.iloc[0,:], generic_scores.iloc[1,:])
```

```
print(f"generic tools kappa: {generic_kappa}")

generic_scores
```

```
generic tools kappa: -0.25
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	B-1	B-2	B-3	B-4	B-5	B-8	B-9
6	30	35	25	30	30	35	35
7	30	30	30	35	30	30	30

```
'''
calculate cohen kappa in generic rubric
'''
localized_scores = localized_data.loc[~generic_data["double-assessment"].isna(),:]\
    .iloc[:,4:]\
    .dropna(how='any', axis=1)\
    .select_dtypes(include=int)
localized_kappa=cohen_kappa_score(localized_scores.iloc[0,:], localized_scores.iloc[1,:])

kappa_table = pd.DataFrame([],columns=["assessment_tool","kappa"])
kappa_table = pd.concat([kappa_table,pd.DataFrame({"assessment_tool":["generic rubric"],"kappa":[generic_kappa]})])
kappa_table = pd.concat([kappa_table,pd.DataFrame({"assessment_tool":["localized tools"],"kappa":[localized_kappa]})])
kappa_table.to_csv(f"{out_dir}/cohen_kappa.csv",encoding="utf_8_sig",index=False)

print(localized_scores)



kappa_table
```

```
   L1  L3  L4  L5  L6  L7  L8  L11  B-5
6   1   2   2   2   2   2   3    4    4
7   1   2   1   2   3   2   3    4    4
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	assessment_tool	kappa
0	generic rubric	-0.25
0	localized tools	0.689655