

CDP_2026

October 11, 2026

1 Analyse du tableau de résultats de la sélection des projets CDP

1.1 Import des packages statistiques et définitions de constantes

```
[1]: %matplotlib inline

import matplotlib
from matplotlib import pyplot

import numpy
import pandas

import scipy.stats

from hades_stats import *

MEAN_NM = "mean"
RANK_NM = "rank"
```

1.2 Définitions de fonctions pour calculer des notes

- Calcul des poids pour faire une note finale à partir de données qualitative (excellent → poor) ou (A, B, C)
- 4 options testées :
 - soit par rang avec classement avec la plus haute note, puis départage sur la seconde, puis...
 - soit par rang avec classement absence de la pire note, puis sur l'absence de la moins pire, etc.
 - soit en effectant des poids par notes: classique 1,2,3,4
 - soit en effectant des poids par notes: -2,1,2,4 pour defavoriser "poor" et bonus pour "excellent".
 - soit plus "scolaire" pour la compréhension:
 - * 18 = Excellent
 - * 16 = Very Good
 - * 14 = Good
 - * 10 = Poor

```
[2]: def compute_weights(df_with_sorted_columns, ranks = True, weights = None):
    sorted_columns = df_with_sorted_columns.columns # important que les
    ↪ colonnes soit rangees: du meilleurs au plus mauvais

    if ranks is True: # astuce numérique on met 100^(nombre de note) car il n'y
    ↪ a pas 100 note d'une valeurs. Avec cette astuce il faudrait 100 B pour faire
    ↪ l'équivalent d'un A, ce qui n'est jamais le cas dans ce tableau
        weights = {c: 100**p for p, c in enumerate(reversed(sorted_columns))}
    ↪ #classement par nombre de meilleure note
    elif ranks in ('r', 'reversed', -1):
        weights = {c: -100**p for p, c in enumerate(sorted_columns)}
    ↪ #classement par absence de la pire note
    else:
        if weights is None:
            weights = numpy.arange(sorted_columns.size)

        weights = {c: w for c, w in zip(reversed(sorted_columns), weights)}

    weights = {i : weights for i in df_with_sorted_columns.index}

    res = pandas.DataFrame(weights, index=sorted_columns).transpose()
    return res

def compute_global_eval(df_with_sorted_columns, ranks = True, weights = None):
    weights = compute_weights(df_with_sorted_columns=df_with_sorted_columns,
    ↪ ranks = ranks, weights = weights)
    df_weighed = df_with_sorted_columns * weights
    n_eval = df_with_sorted_columns.sum(axis=1)
    n_eval = pandas.Series(n_eval, name="N eval")
    sm = df_weighed.sum(axis=1)
    sm = pandas.Series(sm, name="Sum")
    mn = pandas.Series(sm/n_eval, name=MEAN_NM)
    rk = pandas.Series(mn.rank(), name=RANK_NM)
    res = pandas.concat([df_weighed, n_eval, sm, mn, rk], axis=1)
    return res
```

1.3 Lecture des données

Lecture fichier et separation des type de résultat: experts, comite, selection finale

```
[3]: table = pandas.read_csv("Selection.csv", header=[0,1,2], index_col=0)
```

```
[4]: table
```

[4] :

	EXPERTS				
	Appreciation globale		Appreciation criteres		
	Excellent	Very good	Good	Poor	
La Rochelle	3.0	NaN	NaN	NaN	5.0
Nice	2.0	1.0	NaN	NaN	5.0
Poitiers	2.0	1.0	NaN	NaN	4.0
Gueret	2.0	1.0	NaN	NaN	4.0
Niort	2.0	1.0	NaN	NaN	4.0
Nantes	1.0	2.0	NaN	NaN	5.0
Gap	1.0	2.0	NaN	NaN	4.0
Rouen	1.0	1.0	1.0	NaN	4.0
Besançon	1.0	2.0	NaN	NaN	3.0
Caen	1.0	1.0	1.0	NaN	2.0
Brest	1.0	NaN	2.0	NaN	2.0
Paris	1.0	NaN	2.0	NaN	1.0
Lyon	NaN	3.0	NaN	NaN	2.0
Marseille	NaN	3.0	NaN	NaN	2.0
Le Mans	NaN	2.0	1.0	NaN	2.0
Orange	NaN	3.0	NaN	NaN	1.0
Lille	NaN	2.0	1.0	NaN	1.0
Rennes	NaN	1.0	2.0	NaN	1.0
Limoges	NaN	2.0	1.0	NaN	1.0
Aix	NaN	3.0	NaN	NaN	1.0
Angouleme	NaN	1.0	2.0	NaN	1.0
Strasbourg	NaN	3.0	NaN	NaN	NaN

	COMITE						
	Categorisation indicative						En lien
	Very good	Good	Poor	A	B	C	En lien
La Rochelle	4.0	NaN	NaN	13	3	2	1
Nice	3.0	1.0	NaN	2	6	9	2
Poitiers	5.0	NaN	NaN	10	6	0	3
Gueret	5.0	NaN	NaN	15	2	0	2
Niort	4.0	1.0	NaN	18	0	0	1
Nantes	4.0	NaN	NaN	5	7	4	3
Gap	4.0	1.0	NaN	6	9	2	2
Rouen	NaN	5.0	NaN	5	6	6	2
Besançon	5.0	1.0	NaN	5	8	4	2
Caen	5.0	1.0	1.0	5	11	2	1
Brest	2.0	5.0	NaN	2	8	6	3
Paris	5.0	3.0	NaN	12	3	2	2
Lyon	6.0	1.0	NaN	10	5	1	3
Marseille	6.0	1.0	NaN	5	11	3	0
Le Mans	4.0	2.0	1.0	14	4	0	1
Orange	6.0	2.0	NaN	2	8	9	0
Lille	5.0	3.0	NaN	9	7	0	3
Rennes	4.0	3.0	1.0	8	7	2	2

Limoges	4.0	4.0	NaN	6	9	1	3
Aix	6.0	2.0	NaN	4	11	2	2
Angouleme	4.0	3.0	1.0	1	5	9	4
Strasbourg	8.0	1.0	NaN	2	4	12	1

			FINAL
	Nb votes	PROPOSITION	VOTE SELECTION
	Nb votes	PROPOSITION	VOTE SELECTION
La Rochelle	18	A	Oui
Nice	17	C	non
Poitiers	16	A	oui
Gueret	17	A	oui
Niort	18	A	oui
Nantes	16	B	oui
Gap	17	C	non
Rouen	17	C	non
Besançon	17	C	non
Caen	18	B	oui
Brest	16	C	non
Paris	17	A	oui
Lyon	16	B	oui
Marseille	19	C	non
Le Mans	18	A	oui
Orange	19	C	non
Lille	16	A	oui
Rennes	17	A	oui
Limoges	16	B	oui
Aix	17	C	non
Angouleme	15	C	non
Strasbourg	18	C	non

```
[5]: table_experts = table["EXPERTS"]
table_experts = table_experts.fillna(0).map(int)

table_experts
```

	Appreciation globale				Appreciation criteres \	
	Excellent	Very good	Good	Poor	Excellent	
La Rochelle	3	0	0	0	5	
Nice	2	1	0	0	5	
Poitiers	2	1	0	0	4	
Gueret	2	1	0	0	4	
Niort	2	1	0	0	4	
Nantes	1	2	0	0	5	
Gap	1	2	0	0	4	
Rouen	1	1	1	0	4	
Besançon	1	2	0	0	3	

Caen	1	1	1	0	2
Brest	1	0	2	0	2
Paris	1	0	2	0	1
Lyon	0	3	0	0	2
Marseille	0	3	0	0	2
Le Mans	0	2	1	0	2
Orange	0	3	0	0	1
Lille	0	2	1	0	1
Rennes	0	1	2	0	1
Limoges	0	2	1	0	1
Aix	0	3	0	0	1
Angouleme	0	1	2	0	1
Strasbourg	0	3	0	0	0

	Very good	Good	Poor
La Rochelle	4	0	0
Nice	3	1	0
Poitiers	5	0	0
Gueret	5	0	0
Niort	4	1	0
Nantes	4	0	0
Gap	4	1	0
Rouen	0	5	0
Besançon	5	1	0
Caen	5	1	1
Brest	2	5	0
Paris	5	3	0
Lyon	6	1	0
Marseille	6	1	0
Le Mans	4	2	1
Orange	6	2	0
Lille	5	3	0
Rennes	4	3	1
Limoges	4	4	0
Aix	6	2	0
Angouleme	4	3	1
Strasbourg	8	1	0

```
[6]: expert_evals = sorted(set(table_experts.columns.get_level_values(0)))
      expert_evals
```

```
[6]: ['Appreciation criteres', 'Appreciation globale']
```

1.3.1 Calcul de differentes notes globale

Définition de `ground_truth` comme méthode de classement pour présenter les graphiques. Les graphiques sont présentés avec le tri hierarchique par plus grand nombre de excellents.

Ce choix n'a pas d'influence sur les calculs statistiques qui suivent.

```
[7]: dic_expert_eval = {}

for expert_eval in expert_evals:
    dic_expert_eval[expert_eval] = {}
    df_expert = table_experts[expert_eval]
    dic_expert_eval[expert_eval]["ranked"] = compute_global_eval(df_expert,
↪ranks=True) #voir explication dans la définition
    dic_expert_eval[expert_eval]["mean0"] = compute_global_eval(df_expert,
↪ranks=False, weigths=None) #Poor=1 ... Excellent=4
    dic_expert_eval[expert_eval]["mean1"] = compute_global_eval(df_expert,
↪ranks=False, weigths=[-2,1,2,4]) #Poor=-1 ... Excellent=4
    dic_expert_eval[expert_eval]["reversed"] = compute_global_eval(df_expert,
↪ranks=-1) #voir explication dans la définition
    dic_expert_eval[expert_eval]["scholar"] = compute_global_eval(df_expert,
↪ranks=False, weigths=[10,14,16,18]) # classic notation in scholarship
```

```
[8]: #compute_global_eval(df_expert, ranks=-1)
for expert_eval in expert_evals:
    print(expert_eval)
    #print(dic_expert_eval[expert_eval]["reversed"][RANK_NM].sort_values())
    print(dic_expert_eval[expert_eval]["scholar"][MEAN_NM].sort_values())
```

Appreciation criteres

Rennes	14.888889
Angouleme	14.888889
Brest	15.333333
Limoges	15.333333
Le Mans	15.333333
Lille	15.555556
Paris	15.555556
Caen	15.555556
Aix	15.777778
Orange	15.777778
Strasbourg	15.777778
Rouen	15.777778
Lyon	16.222222
Marseille	16.222222
Besançon	16.444444
Gap	16.666667
Niort	16.666667
Gueret	16.888889
Poitiers	16.888889
Nice	16.888889
Nantes	17.111111
La Rochelle	17.111111

Name: mean, dtype: float64

```

Appreciation globale
Rennes      14.666667
Angouleme   14.666667
Brest       15.333333
Limoges     15.333333
Lille       15.333333
Le Mans     15.333333
Paris       15.333333
Aix         16.000000
Orange      16.000000
Marseille   16.000000
Lyon        16.000000
Strasbourg  16.000000
Rouen       16.000000
Caen        16.000000
Besançon    16.666667
Gap         16.666667
Nantes      16.666667
Niort       17.333333
Gueret      17.333333
Poitiers    17.333333
Nice        17.333333
La Rochelle 18.000000
Name: mean, dtype: float64

```

```

[9]: ground_truth = dic_expert_eval['Appreciation globale']['ranked'][RANK_NM]
     ground_truth = ground_truth.sort_values()

```

```

[10]: dic_ls = {'Appreciation globale': "--", 'Appreciation criteres': "--"}
     dic_lw = {'Appreciation globale': 1.5, 'Appreciation criteres': 3}

     fig = pyplot.figure(figsize=(16,10))
     ax = fig.gca()

     ground_truth.plot(color="k", lw=4)

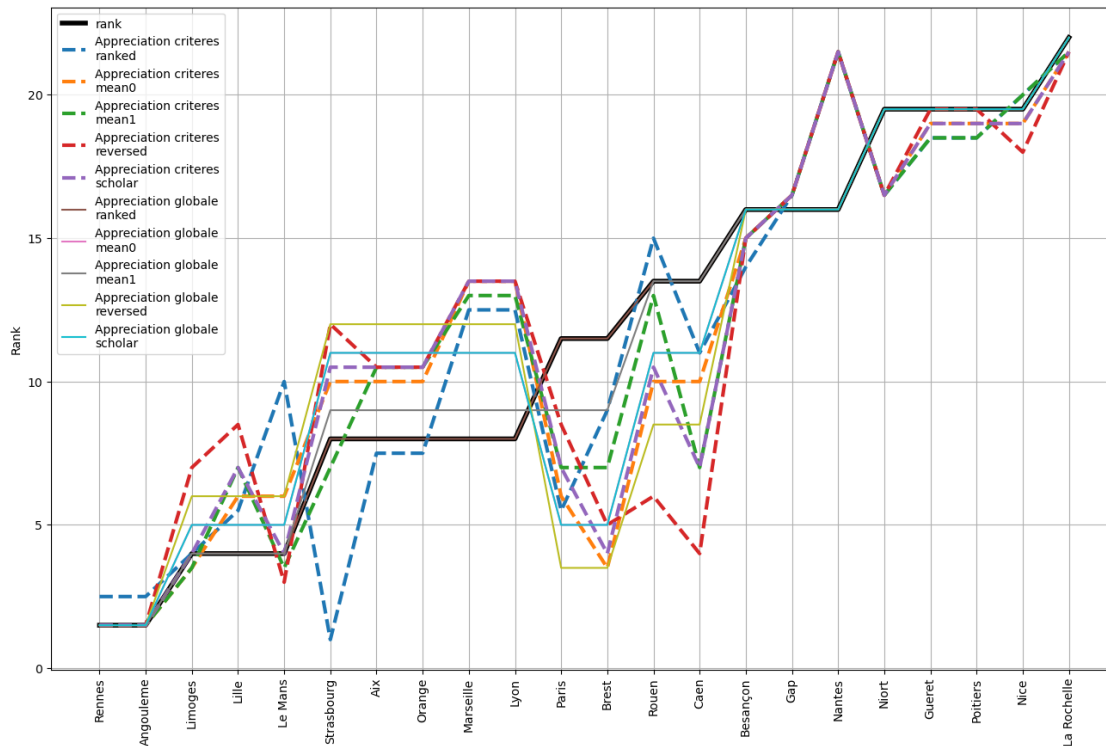
     for crit_name, dic1 in dic_expert_eval.items():
         for mean_name, df in dic1.items():
             rank_series = df[RANK_NM].reindex(ground_truth.index)
             rank_series.plot(label=f"{crit_name}\n{mean_name}",
                               ls=dic_ls[crit_name], lw=dic_lw[crit_name])

     ax.set(xticks=range(ground_truth.size), ylabel="Rank")
     ax.set_xticklabels(ground_truth.index, rotation=90)

     ax.grid(1)
     ax.legend(loc="upper left")

```

[10]: <matplotlib.legend.Legend at 0x7f08a7a63620>



À retenir : La méthode de classement et/ou de notation n'a pas beaucoup d'influence sur les projets bien ou mal évalués, pour les projets du milieu, cela influence certains interclassements.

En particulier, tous les projets au delà de Gap sont mieux évalués que tous les autres, quelle que soit la méthode choisie.

1.4 Comité

```
[11]: comite_table = table["COMITE"]
      comite_evals = comite_table["Categorisation indicative"]

      comite_evals
```

```
[11]:
```

	A	B	C
La Rochelle	13	3	2
Nice	2	6	9
Poitiers	10	6	0
Gueret	15	2	0
Niort	18	0	0
Nantes	5	7	4
Gap	6	9	2
Rouen	5	6	6

Besançon	5	8	4
Caen	5	11	2
Brest	2	8	6
Paris	12	3	2
Lyon	10	5	1
Marseille	5	11	3
Le Mans	14	4	0
Orange	2	8	9
Lille	9	7	0
Rennes	8	7	2
Limoges	6	9	1
Aix	4	11	2
Angouleme	1	5	9
Strasbourg	2	4	12

```
[12]: dic_comite_eval = {}
dic_comite_eval["ranked"] = compute_global_eval(comite_evals, ranks=True)
dic_comite_eval["mean0"] = compute_global_eval(comite_evals, ranks=False,
↪weights=None)
dic_comite_eval["mean1"] = compute_global_eval(comite_evals, ranks=False,
↪weights=[-1,1,2]) # C: -1 B:1 A:2
dic_comite_eval["reversed"] = compute_global_eval(comite_evals, ranks=-1)
dic_comite_eval["scholar"] = compute_global_eval(comite_evals, ranks=False,
↪weights=[12,14,16]) # classic notation in scholarsship
```

```
[13]: #Ici il reste à faire un redressement de chaque note au prorata du nombre de
↪votes (voir colonne correspondantes dans la table "table" en ligne [4]. Ca
↪ne vas rien changer aux résultats... C'est fait dans la colonne Sum
```

```
[14]: dic_comite_eval
```

```
[14]: {'ranked':
      A      B      C      N eval      Sum      mean      rank
La Rochelle 130000 300 2      18 130302 7239.000000 19.0
Nice        20000 600 9      17 20609 1212.294118 4.0
Poitiers    100000 600 0      16 100600 6287.500000 17.0
Gueret      150000 200 0      17 150200 8835.294118 21.0
Niort       180000 0 0      18 180000 10000.000000 22.0
Nantes      50000 700 4      16 50704 3169.000000 11.0
Gap         60000 900 2      17 60902 3582.470588 12.0
Rouen       50000 600 6      17 50606 2976.823529 9.0
Besançon    50000 800 4      17 50804 2988.470588 10.0
Caen        50000 1100 2      18 51102 2839.000000 8.0
Brest       20000 800 6      16 20806 1300.375000 5.0
Paris       120000 300 2      17 120302 7076.588235 18.0
Lyon        100000 500 1      16 100501 6281.312500 16.0
Marseille   50000 1100 3      19 51103 2689.631579 7.0
Le Mans     140000 400 0      18 140400 7800.000000 20.0
```

Orange	20000	800	9	19	20809	1095.210526	2.0		
Lille	90000	700	0	16	90700	5668.750000	15.0		
Rennes	80000	700	2	17	80702	4747.176471	14.0		
Limoges	60000	900	1	16	60901	3806.312500	13.0		
Aix	40000	1100	2	17	41102	2417.764706	6.0		
Angouleme	10000	500	9	15	10509	700.600000	1.0		
Strasbourg	20000	400	12	18	20412	1134.000000	3.0,		
'mean0':		A	B	C	N	eval	Sum	mean	rank
La Rochelle	26	3	0	18	29	1.611111	18.0		
Nice	4	6	0	17	10	0.588235	3.0		
Poitiers	20	6	0	16	26	1.625000	19.0		
Gueret	30	2	0	17	32	1.882353	21.0		
Niort	36	0	0	18	36	2.000000	22.0		
Nantes	10	7	0	16	17	1.062500	8.0		
Gap	12	9	0	17	21	1.235294	12.0		
Rouen	10	6	0	17	16	0.941176	6.0		
Besançon	10	8	0	17	18	1.058824	7.0		
Caen	10	11	0	18	21	1.166667	11.0		
Brest	4	8	0	16	12	0.750000	5.0		
Paris	24	3	0	17	27	1.588235	17.0		
Lyon	20	5	0	16	25	1.562500	15.5		
Marseille	10	11	0	19	21	1.105263	9.0		
Le Mans	28	4	0	18	32	1.777778	20.0		
Orange	4	8	0	19	12	0.631579	4.0		
Lille	18	7	0	16	25	1.562500	15.5		
Rennes	16	7	0	17	23	1.352941	14.0		
Limoges	12	9	0	16	21	1.312500	13.0		
Aix	8	11	0	17	19	1.117647	10.0		
Angouleme	2	5	0	15	7	0.466667	2.0		
Strasbourg	4	4	0	18	8	0.444444	1.0,		
'mean1':		A	B	C	N	eval	Sum	mean	rank
La Rochelle	26	3	-2	18	27	1.500000	16.5		
Nice	4	6	-9	17	1	0.058824	3.0		
Poitiers	20	6	0	16	26	1.625000	19.0		
Gueret	30	2	0	17	32	1.882353	21.0		
Niort	36	0	0	18	36	2.000000	22.0		
Nantes	10	7	-4	16	13	0.812500	7.0		
Gap	12	9	-2	17	19	1.117647	12.0		
Rouen	10	6	-6	17	10	0.588235	6.0		
Besançon	10	8	-4	17	14	0.823529	8.0		
Caen	10	11	-2	18	19	1.055556	11.0		
Brest	4	8	-6	16	6	0.375000	5.0		
Paris	24	3	-2	17	25	1.470588	15.0		
Lyon	20	5	-1	16	24	1.500000	16.5		
Marseille	10	11	-3	19	18	0.947368	9.0		
Le Mans	28	4	0	18	32	1.777778	20.0		
Orange	4	8	-9	19	3	0.157895	4.0		

Lille	18	7	0	16	25	1.562500	18.0		
Rennes	16	7	-2	17	21	1.235294	13.0		
Limoges	12	9	-1	16	20	1.250000	14.0		
Aix	8	11	-2	17	17	1.000000	10.0		
Angouleme	2	5	-9	15	-2	-0.133333	2.0		
Strasbourg	4	4	-12	18	-4	-0.222222	1.0,		
'reversed':									
			A	B	C	N eval	Sum	mean	rank
La Rochelle	-13	-300	-20000		18	-20313	-1128.500000	15.0	
Nice	-2	-600	-90000		17	-90602	-5329.529412	3.0	
Poitiers	-10	-600	0		16	-610	-38.125000	19.0	
Gueret	-15	-200	0		17	-215	-12.647059	21.0	
Niort	-18	0	0		18	-18	-1.000000	22.0	
Nantes	-5	-700	-40000		16	-40705	-2544.062500	7.0	
Gap	-6	-900	-20000		17	-20906	-1229.764706	11.0	
Rouen	-5	-600	-60000		17	-60605	-3565.000000	6.0	
Besançon	-5	-800	-40000		17	-40805	-2400.294118	8.0	
Caen	-5	-1100	-20000		18	-21105	-1172.500000	14.0	
Brest	-2	-800	-60000		16	-60802	-3800.125000	5.0	
Paris	-12	-300	-20000		17	-20312	-1194.823529	13.0	
Lyon	-10	-500	-10000		16	-10510	-656.875000	17.0	
Marseille	-5	-1100	-30000		19	-31105	-1637.105263	9.0	
Le Mans	-14	-400	0		18	-414	-23.000000	20.0	
Orange	-2	-800	-90000		19	-90802	-4779.052632	4.0	
Lille	-9	-700	0		16	-709	-44.312500	18.0	
Rennes	-8	-700	-20000		17	-20708	-1218.117647	12.0	
Limoges	-6	-900	-10000		16	-10906	-681.625000	16.0	
Aix	-4	-1100	-20000		17	-21104	-1241.411765	10.0	
Angouleme	-1	-500	-90000		15	-90501	-6033.400000	2.0	
Strasbourg	-2	-400	-120000		18	-120402	-6689.000000	1.0,	
'scholar':									
			A	B	C	N eval	Sum	mean	rank
La Rochelle	208	42	24	18	274	15.222222	18.0		
Nice	32	84	108	17	224	13.176471	3.0		
Poitiers	160	84	0	16	244	15.250000	19.0		
Gueret	240	28	0	17	268	15.764706	21.0		
Niort	288	0	0	18	288	16.000000	22.0		
Nantes	80	98	48	16	226	14.125000	8.0		
Gap	96	126	24	17	246	14.470588	12.0		
Rouen	80	84	72	17	236	13.882353	6.0		
Besançon	80	112	48	17	240	14.117647	7.0		
Caen	80	154	24	18	258	14.333333	11.0		
Brest	32	112	72	16	216	13.500000	5.0		
Paris	192	42	24	17	258	15.176471	17.0		
Lyon	160	70	12	16	242	15.125000	15.5		
Marseille	80	154	36	19	270	14.210526	9.0		
Le Mans	224	56	0	18	280	15.555556	20.0		
Orange	32	112	108	19	252	13.263158	4.0		
Lille	144	98	0	16	242	15.125000	15.5		

Rennes	128	98	24	17	250	14.705882	14.0
Limoges	96	126	12	16	234	14.625000	13.0
Aix	64	154	24	17	242	14.235294	10.0
Angouleme	16	70	108	15	194	12.933333	2.0
Strasbourg	32	56	144	18	232	12.888889	1.0

```
[15]: comite_global = comite_table['PROPOSITION VOTE'].squeeze()
comite_global = comite_global.replace({"A":"16", "B":"14", "C":"12"}).map(int)

comite_global_df = pandas.DataFrame({MEAN_NM:comite_global, RANK_NM:
    ↪comite_global.rank()})

#dic_comite_eval['global'] = comite_global_df

comite_global_df
```

```
[15]:
```

	mean	rank
La Rochelle	16	18.5
Nice	12	5.5
Poitiers	16	18.5
Gueret	16	18.5
Niort	16	18.5
Nantes	14	12.5
Gap	12	5.5
Rouen	12	5.5
Besançon	12	5.5
Caen	14	12.5
Brest	12	5.5
Paris	16	18.5
Lyon	14	12.5
Marseille	12	5.5
Le Mans	16	18.5
Orange	12	5.5
Lille	16	18.5
Rennes	16	18.5
Limoges	14	12.5
Aix	12	5.5
Angouleme	12	5.5
Strasbourg	12	5.5

```
[16]: comite_truth = comite_global_df[RANK_NM].sort_values()

val_y2 = sorted(set(comite_truth))
lab_y2 = ("C", "B", "A")

fig = pyplot.figure(figsize=(16,10))
ax = fig.gca()
```

```

comite_truth.plot(color="k", lw=4, label="Classement final")

for mean_name, df in dic_comite_eval.items():
    rank_series = df[RANK_NM].reindex(comite_truth.index)
    rank_series.plot(label=f"{crit_name}\n{mean_name}", ls="--", lw=2)

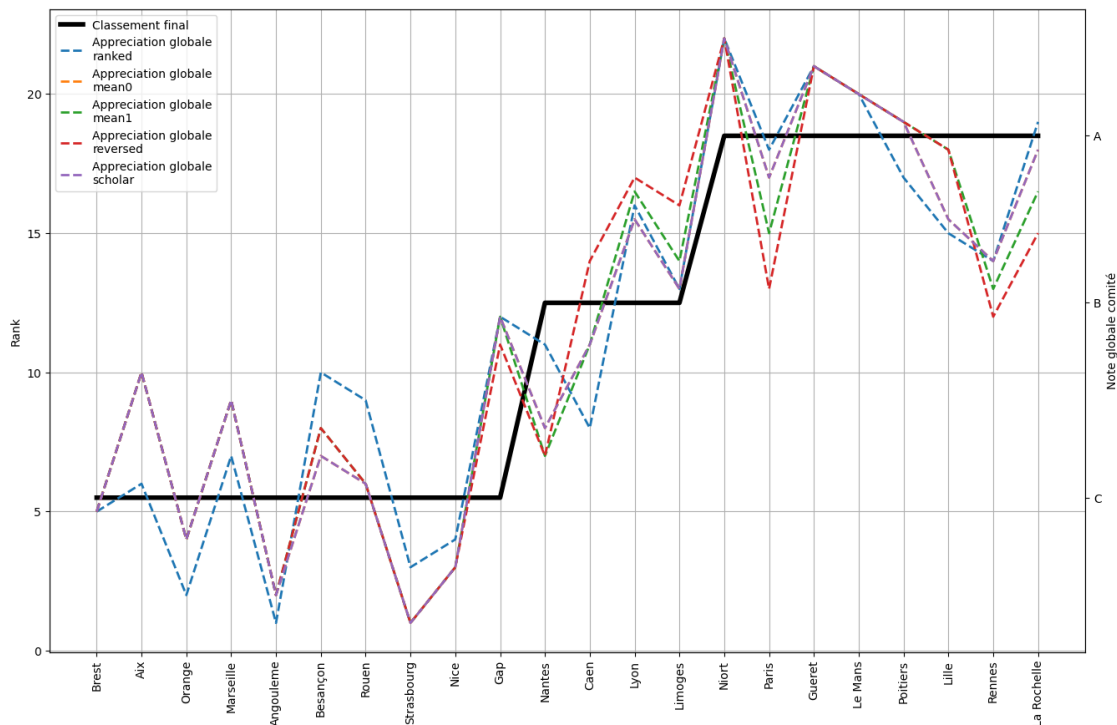
ax.set(xticks=range(comite_truth.size), ylabel="Rank")
ax.set_xticklabels(comite_truth.index, rotation=90)

ax.grid(1)
ax.legend(loc="upper left")

ax2 = ax.twinx()
ax2.sharex(ax)
ax2.set_ylim(ax.get_ylim())
ax2.set_yticks(val_y2)
ax2.set_yticklabels(lab_y2)
ax2.set(ylabel="Note globale comité")

```

[16]: [Text(0, 0.5, 'Note globale comité')]



[]:

À retenir : Relativement cohérent aussi.

- On comprend assez mal le classement de Gap en C.
- On comprend assez mal le classement de Rennes en A.
- Lyon aurait pu être A (surtout comparé à Rennes).
- Besançon et Rouen auraient pu être B.
- Nantes et Caen auraient pu être C.

```
[17]: all_comite_ranks = pandas.DataFrame({k : df[RANK_NM] for k, df in
↳ dic_comite_eval.items()})
```

1.5 Classement final

```
[18]: def final2bool(txt):
    txt = txt.lower()
    if txt=="oui":
        res = True
    elif txt=="non":
        res = False
    else:
        raise ValueError("")

    res = int(res)
    return res

final = table["FINAL"].map(final2bool).squeeze()
final
```

```
[18]: La Rochelle    1
Nice              0
Poitiers          1
Gueret            1
Niort             1
Nantes            1
Gap              0
Rouen             0
Besançon          0
Caen              1
Brest            0
Paris            1
Lyon             1
Marseille         0
Le Mans          1
Orange           0
Lille            1
Rennes           1
```

```

Limoges      1
Aix          0
Angouleme    0
Strasbourg    0
Name: (SELECTION, SELECTION), dtype: int64

```

```
[ ]:
```

2 Tests Statistiques

2.1 Experts Vs Comité Vs Sélection finale

Exisite-t-il une corrélation entre les notes des experts extérieurs et celle du comité ? Exisite-t-il une corrélation entre les notes des experts la sélection finale des projets financés ?

Rappel: tests de corrélation : H_0 = "Il n'y a pas de corrélation entre x et y " $\rightarrow$ si $pvalue < 0.1$ on considère que H_0 est rejetée car très peu probable sous H_0 et on conclut alors que la corrélation entre x et y ne s'est pas produit par hasard.

```

[19]: igrph = 0

for expert_kind, dic1 in dic_expert_eval.items():
    for eval_function, df_expert in dic1.items():
        if eval_function == "scholar":
            glob_kind = MEAN_NM
        else:
            glob_kind = RANK_NM

    df_vals = df_expert[glob_kind]
    comite_vals = dic_comite_eval[eval_function][glob_kind]
    comite_global = comite_global_df[glob_kind]
    assert ((df_vals.index == comite_vals.index) & (comite_vals.index ==
↪final.index)).all()

    fig = pyplot.figure(figsize=(20,10))
    cax = pyplot.axes([0.92,0.1,0.02,0.8])
    ax1 = pyplot.axes([0.03,0.1,0.4,0.8])
    ax2 = pyplot.axes([0.45,0.1,0.4,0.8])
    ax1.sharey(ax2)
    ax1.sharex(ax2)

    bds = [0,0.5,1]
    nbds = len(bds)
    ncol = nbds - 1
    cm = matplotlib.colormaps.get_cmap('jet_r').resampled(ncol)
    nrm = matplotlib.colors.BoundaryNorm(bds, ncol)
    #uncomment these lines
    #extd = "neither"
    #or these lines

```

```

        cb = matplotlib.colorbar.ColorbarBase(cax, cmap=cm, norm=nrm,
↪extend="neither", orientation="vertical")

        fig.sca(ax1)
        pyplot.scatter(comite_vals.values, df_vals.values, c=final.
↪values, cmap=cb.cmap, norm=cb.norm, s=100)
        for x, y, s in zip(comite_vals.values, df_vals.values, final.index):
            pyplot.text(x=x, y=y, s=s)

        kendalltau = scipy.stats.kendalltau(comite_vals.values, df_vals.values)
        flr = FastLinearRegression(comite_vals.values, df_vals.values)
        label = "\n".join([flr.label(fmt=".2f"), f"Tau={kendalltau.statistic:.
↪2f} | pval={kendalltau.pvalue:.2f}"])
        flr.plot(label=label)

        ax1.set(xlabel=f'COMITE -- {eval_function} -- {glob_kind}',
↪ylabel=f'EXPERT -- {expert_kind} -- {eval_function} -- {glob_kind}')

        fig.sca(ax2)

        pyplot.scatter(comite_global.values, df_vals.values, c=final.
↪values, cmap=cb.cmap, norm=cb.norm, s=100)
        ax2.set(xlabel=f'COMITE -- Global eval -- {glob_kind}') #,
↪ylabel=f'EXPERT -- {expert_kind} -- {eval_function}')

        for x, y, s in zip(comite_global.values, df_vals.values, final.index):
            pyplot.text(x=x, y=y, s=s)

        kendalltau = scipy.stats.kendalltau(comite_global.values, df_vals.
↪values)
        flr = FastLinearRegression(comite_global.values, df_vals.values)

        label = "\n".join([flr.label(fmt=".2f"), f"Tau={kendalltau.statistic:.
↪2f} | pval={kendalltau.pvalue:.2f}"])
        flr.plot(label=label)

        all_lims = numpy.concatenate([df_vals, comite_vals]).flatten()
        lim = (math.floor(min(all_lims))-0.5, math.ceil(max(all_lims))+0.5)

        for ax in (ax1, ax2):
            ax.legend(loc="upper left")
            ax.grid(1)
            ax.set(aspect="equal", xlim=lim, ylim=lim)

        cax.set_title("Final selection")

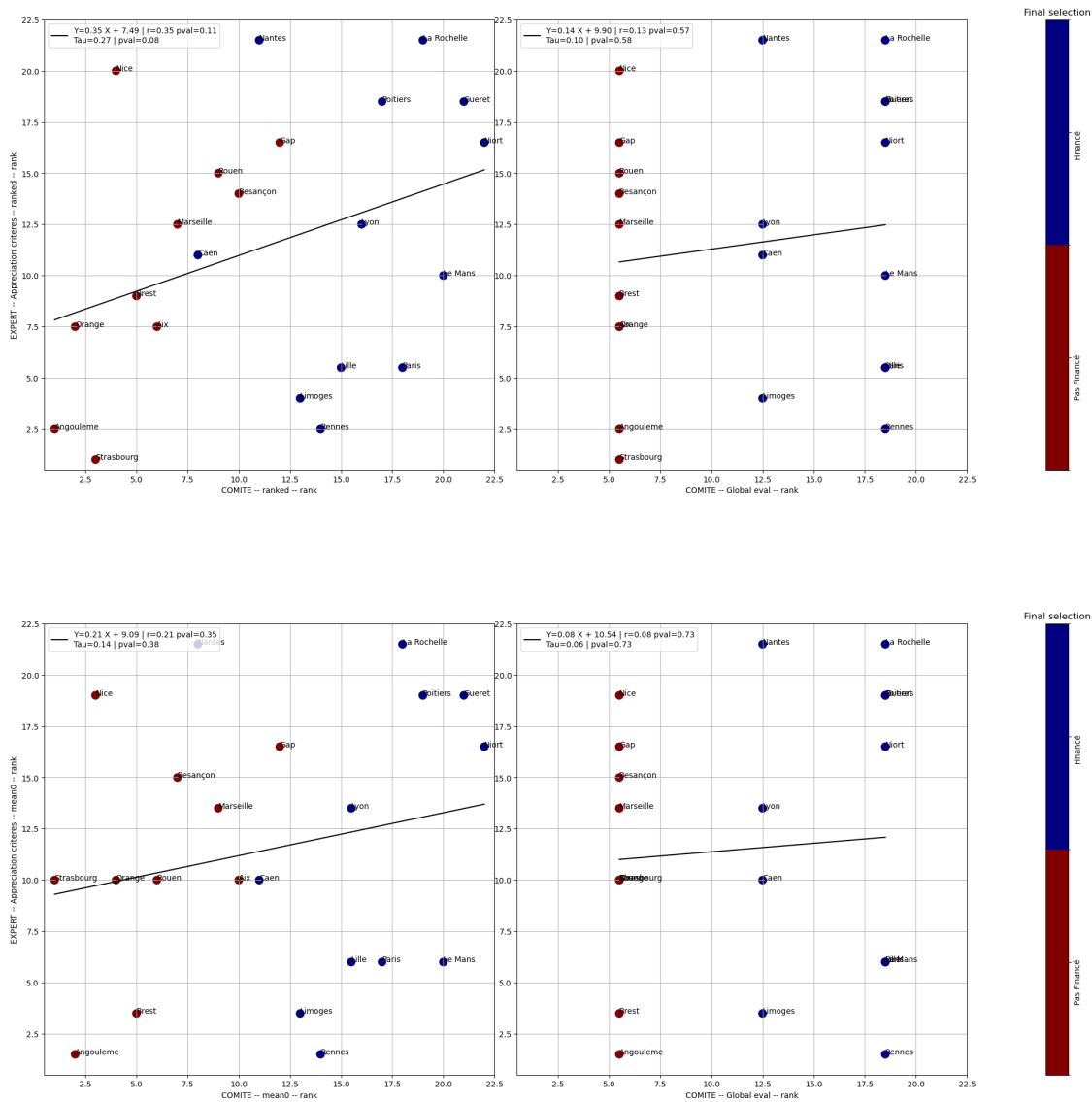
```

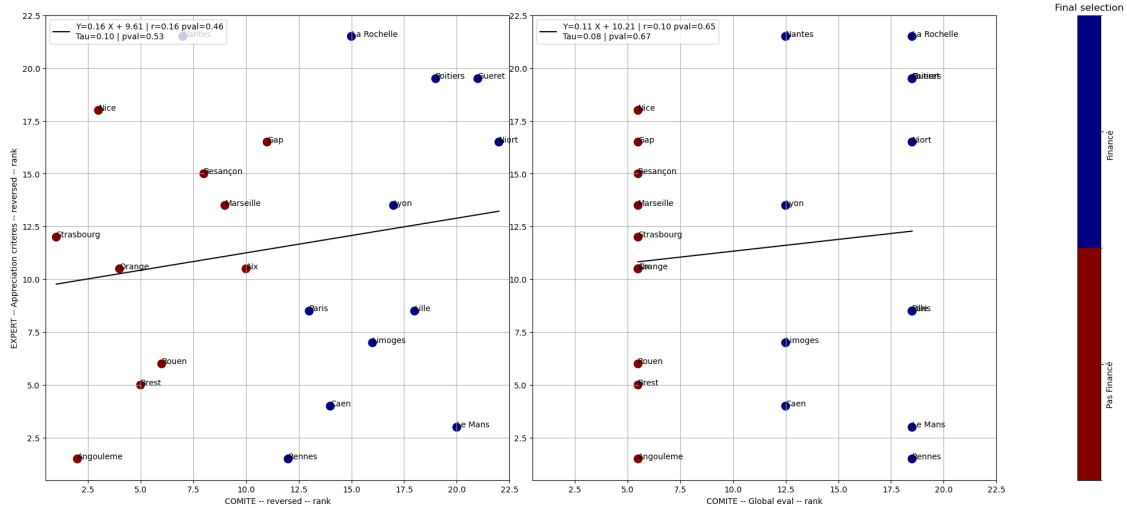
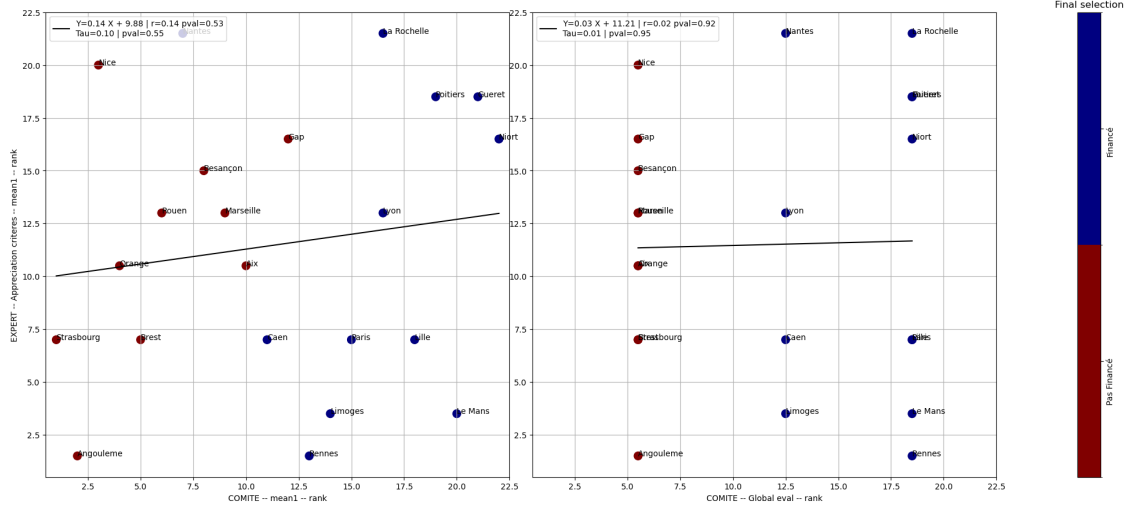
```
cax.set_yticks([0.25,0.75])
cax.set_yticklabels(["Pas Financé", "Financé"], rotation=90)

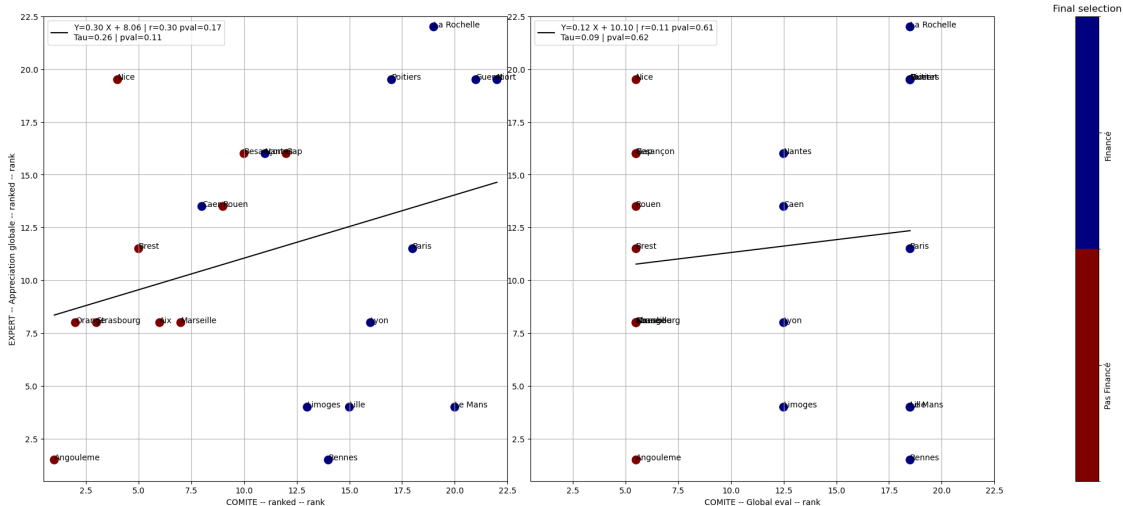
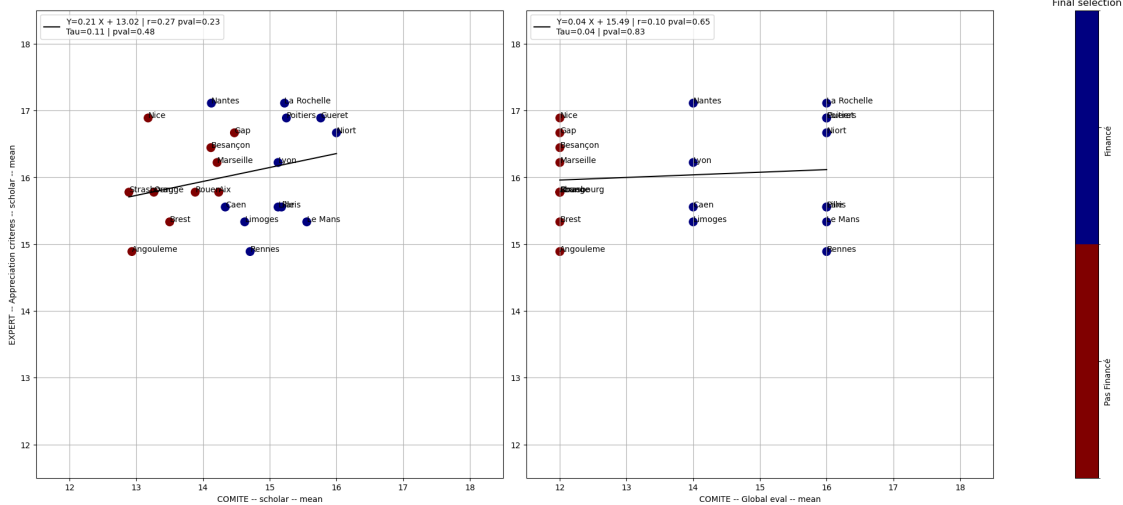
igraph += 1

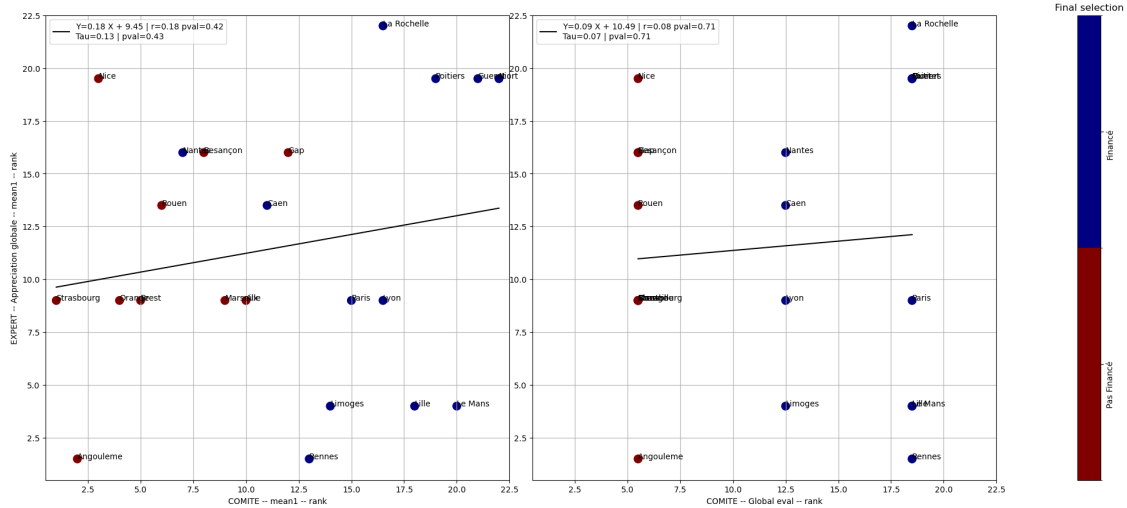
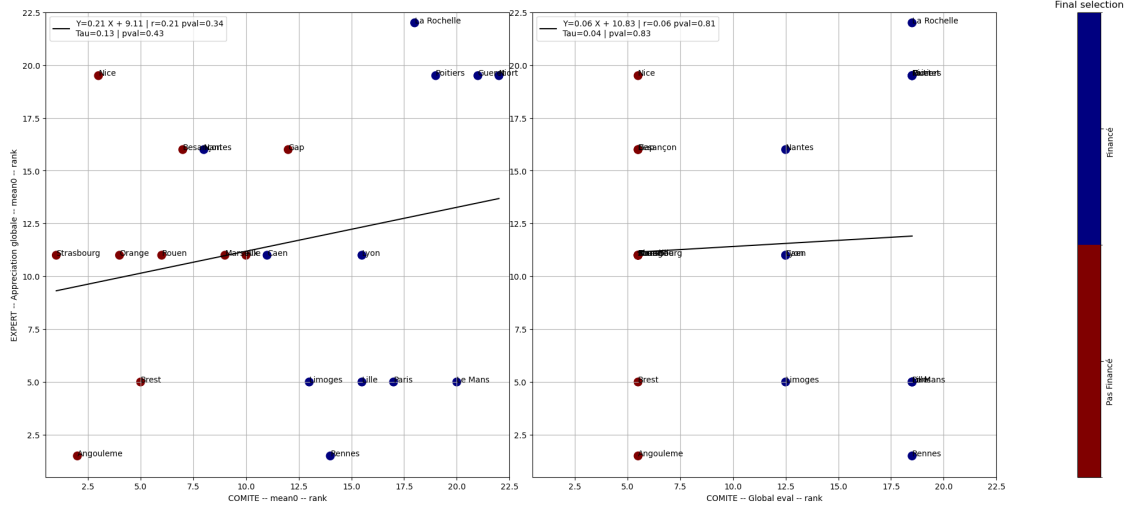
print(igraph)
print("Ntests:", igragh*2)
```

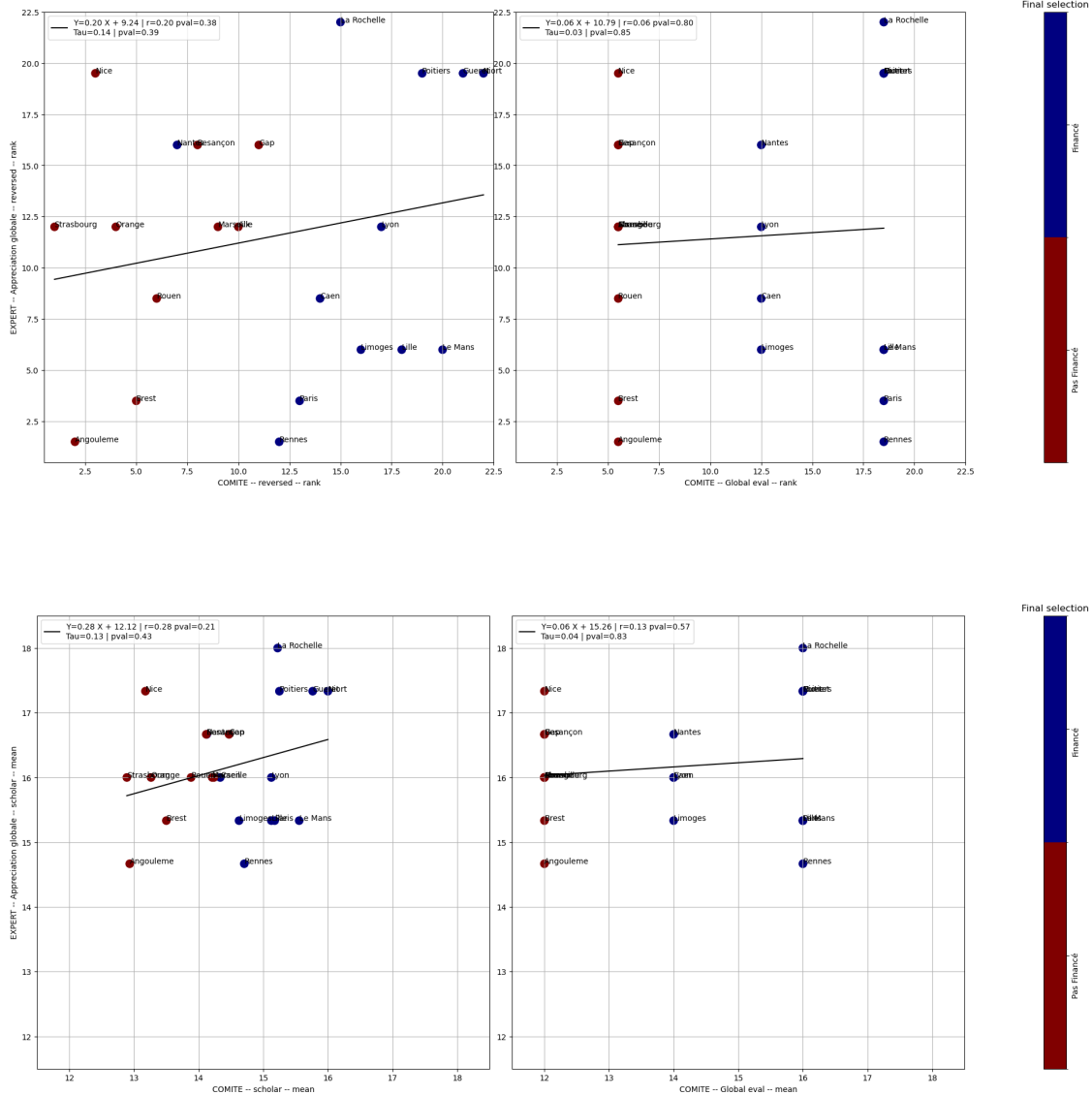
10
Ntests: 20











```
[20]: lim
```

```
[20]: (11.5, 18.5)
```

```
[21]: scipy.stats.pearsonr(comite_global.values, df_vals.values)
```

```
[21]: PearsonRResult(statistic=np.float64(0.12961962542305458),
pvalue=np.float64(0.5653482326777992))
```

```
[22]: scipy.stats.spearmanr(comite_global.values, df_vals.values)
```

```
[22]: SignificanceResult(statistic=np.float64(0.05521437184420979),
pvalue=np.float64(0.8071943886772996))
```

À retenir :

- Sur 10 tests de corrélation (2 par graphique): aucune corrélation **significative** au seuil 5% (et 1 seule au seuil 10%) entre les évaluations des experts et le comité quelle que soit la façon dont on fait la moyenne.
- Aucune corrélation significative au seuil 10% avec la décision finale du comité.

Les meilleures corrélations $r \approx 0.35$ correspondent à des variances expliquées de l'ordre de $r^2 \approx 0.35^2 \approx 10\%$.

2.2 Comparaison projets financés Vs non financés

Est-ce que les échantillons sélectionnés vs non-sélectionnés ont une distribution de notes des experts différente ?

Rappel: tests two samples: H_0 = "Les deux échantillons sont issus de la même distribution" → si $p\text{-value} < 0.1$ on considère que H_0 est rejetée car très peu probable sous H_0 et on conclut alors que la différence de distribution entre x et y ne s'est pas produit par hasard.

https://docs.scipy.org/doc/scipy/tutorial/stats/comparing_two_samples.html

```
[23]: success = final[final==True]
      loss = final[final==False]
```

```
[24]: win_col = cb.mappable.to_rgba(0.75)
      los_col = cb.mappable.to_rgba(0.25)

      for crit, dic1 in dic_expert_eval.items():
          for kind, expert_eval in dic1.items():

              loss_experts = expert_eval.reindex(loss.index)[MEAN_NM]
              loss_experts = Sample(loss_experts)

              success_experts = expert_eval.reindex(success.index)[MEAN_NM]
              success_experts = Sample(success_experts)

              fig = pyplot.figure()
              ax = fig.gca()

              student_test = scipy.stats.ttest_ind(loss_experts.values,
↳success_experts.values)
              ks_test = scipy.stats.ks_2samp(loss_experts.values, success_experts.
↳values)
              ranksum_test = scipy.stats.mannwhitneyu(loss_experts.values,
↳success_experts.values, method="exact")

              tests_txts = [f"{test2sample.__class__.__name__.replace('Result', '')}:
↳pvalue={test2sample.pvalue:.2f}" for test2sample in (student_test, ks_test,
↳ranksum_test)]
```

```

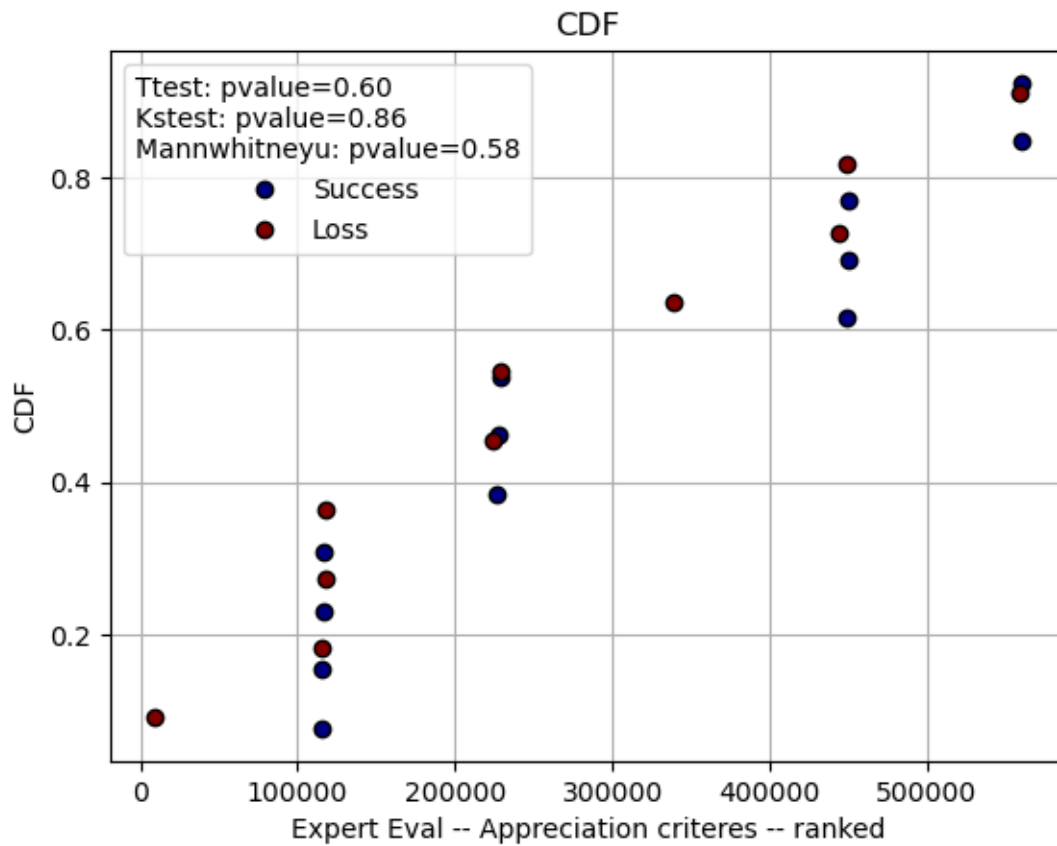
tests_txt = "\n".join(tests_txts)

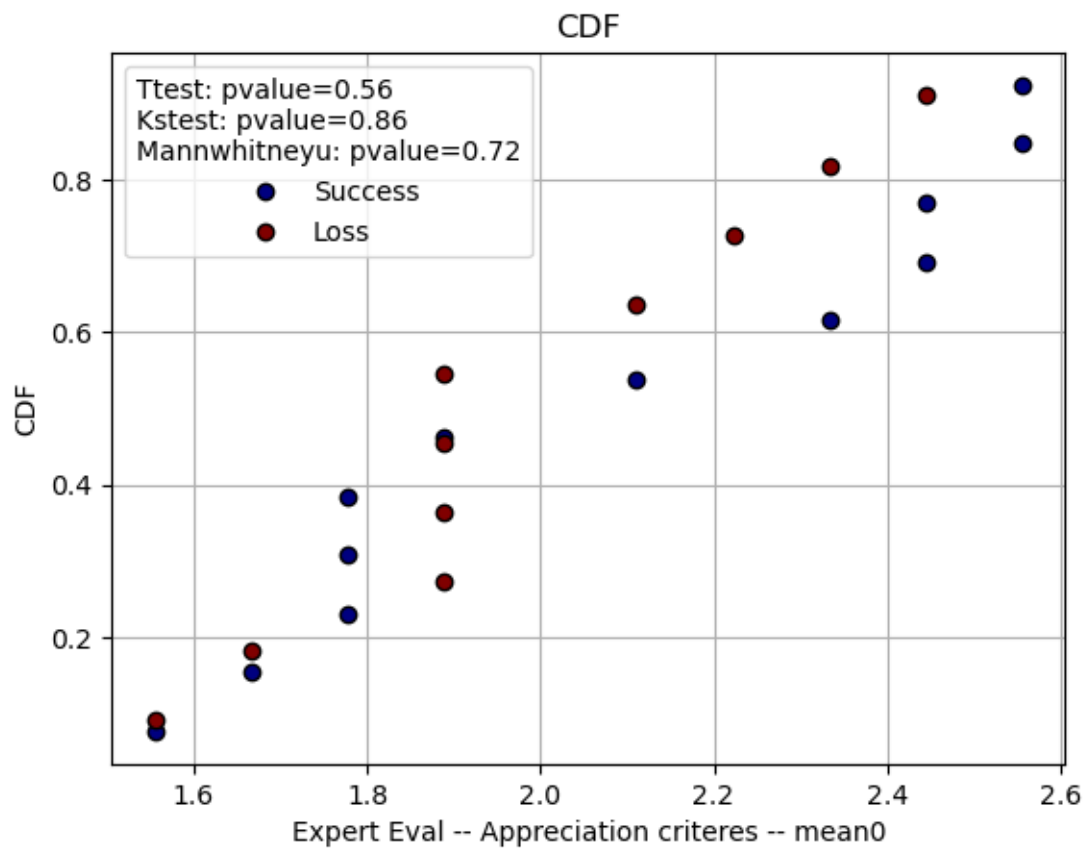
success_experts.cdf_plot(color=win_col, label="Success")
loss_experts.cdf_plot(color=los_col, label="Loss")

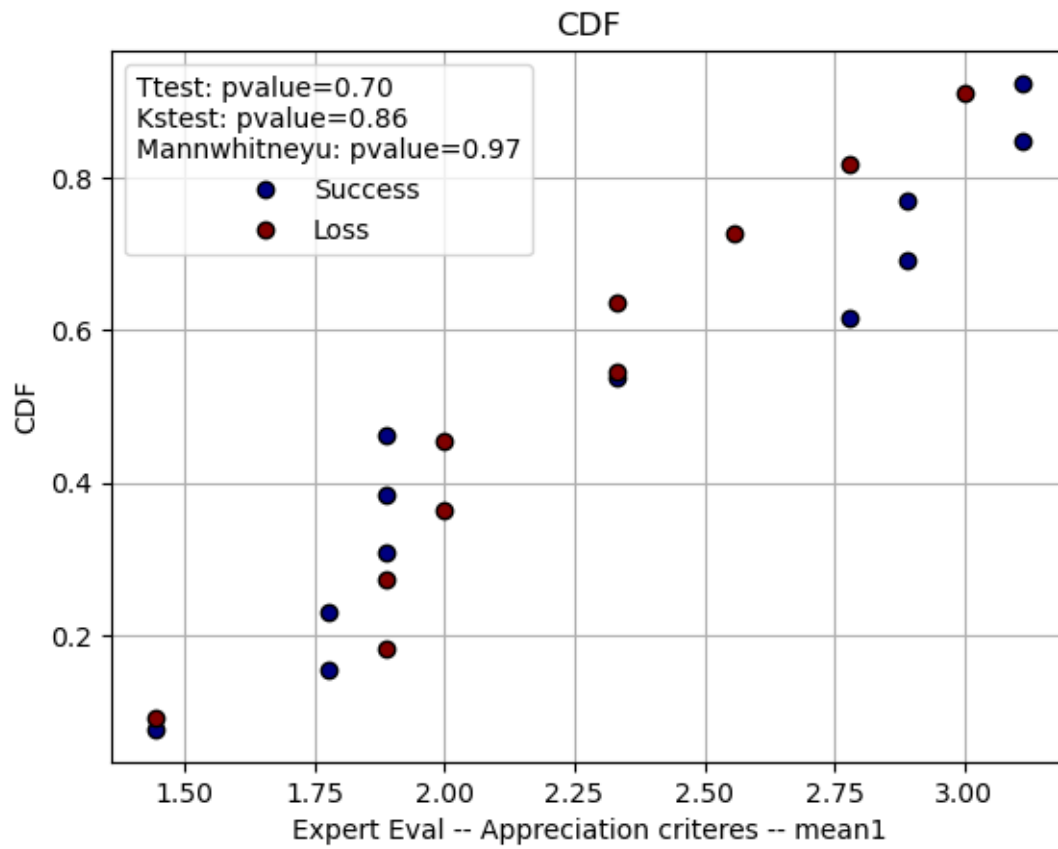
ax.set(xlabel=f"Expert Eval -- {crit} -- {kind}", ylabel="CDF")

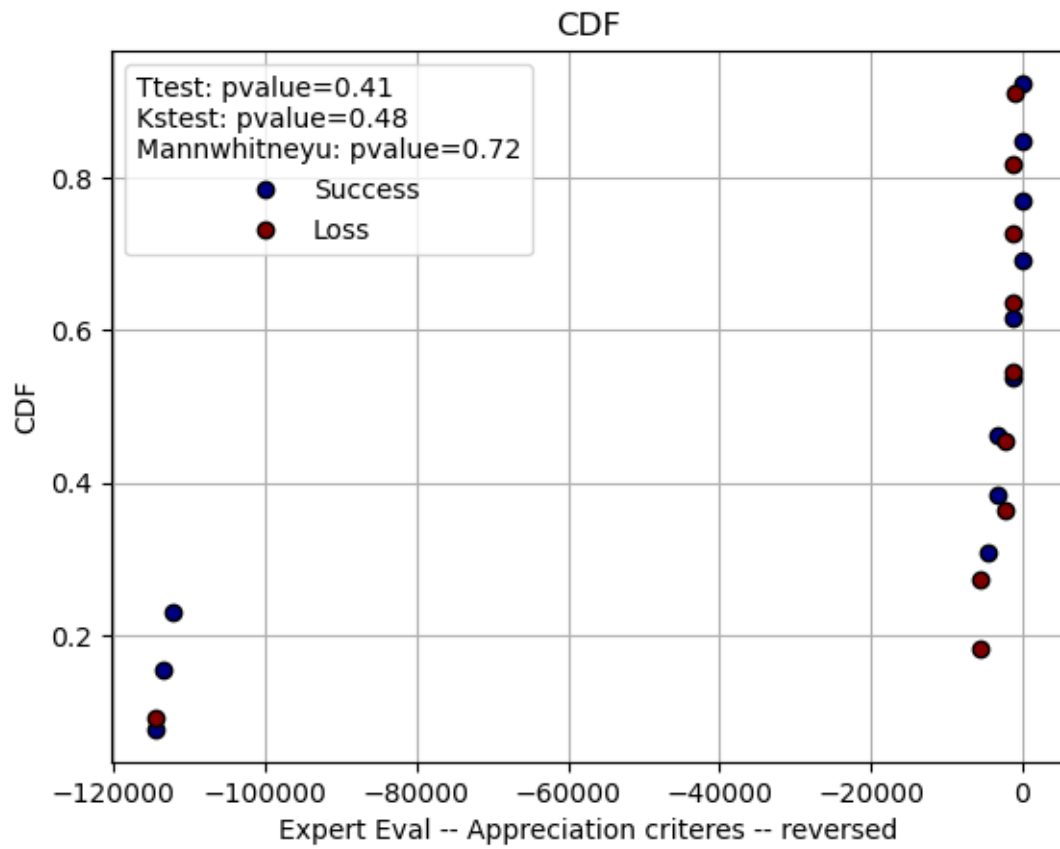
ax.legend(loc="upper left", title=tests_txt)

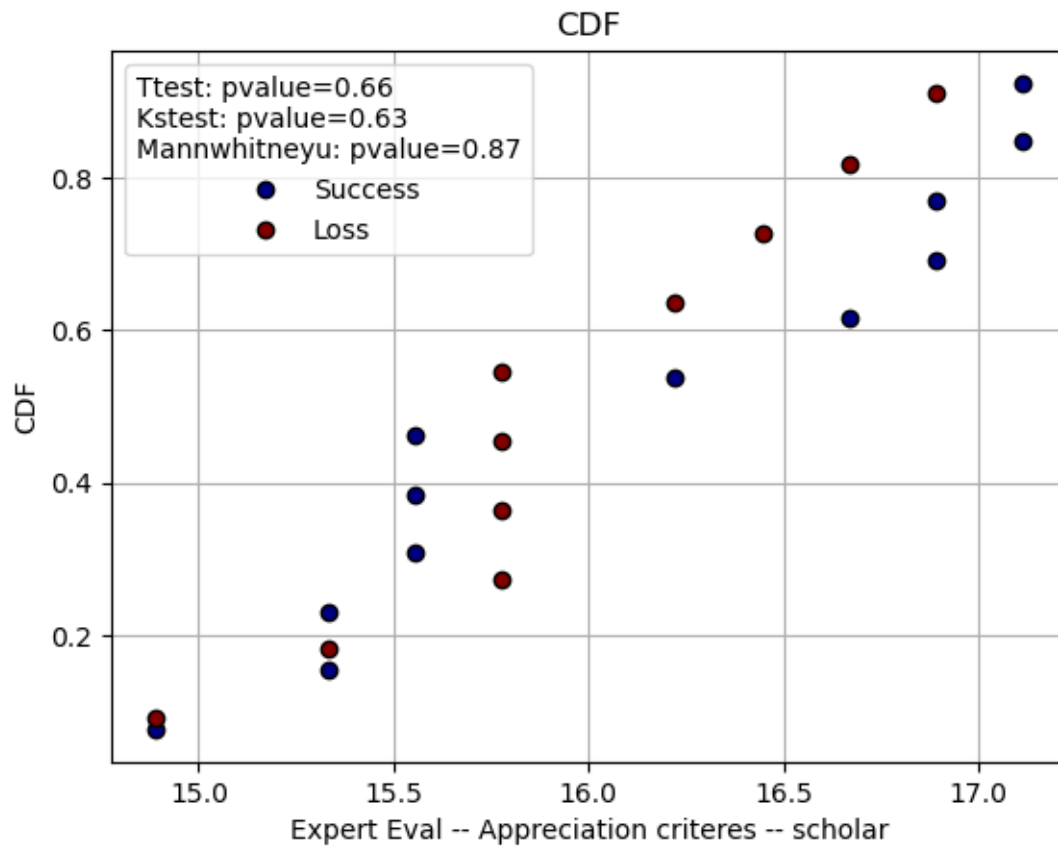
```

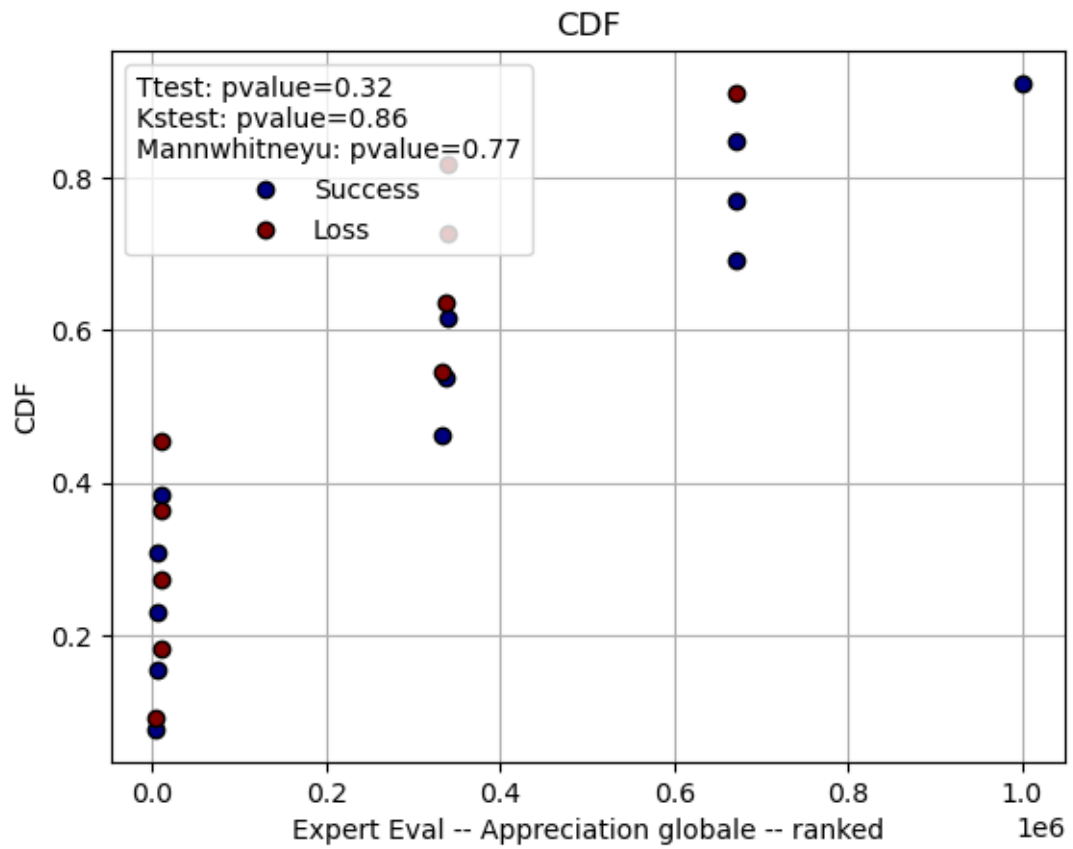


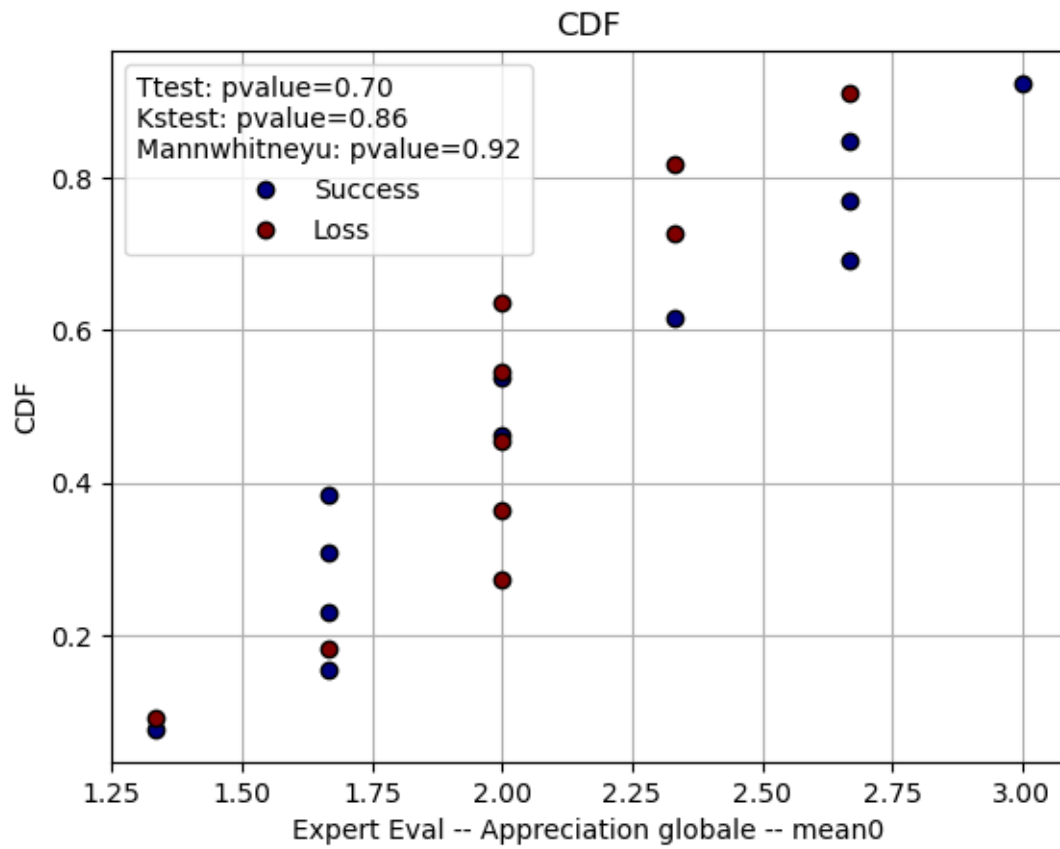


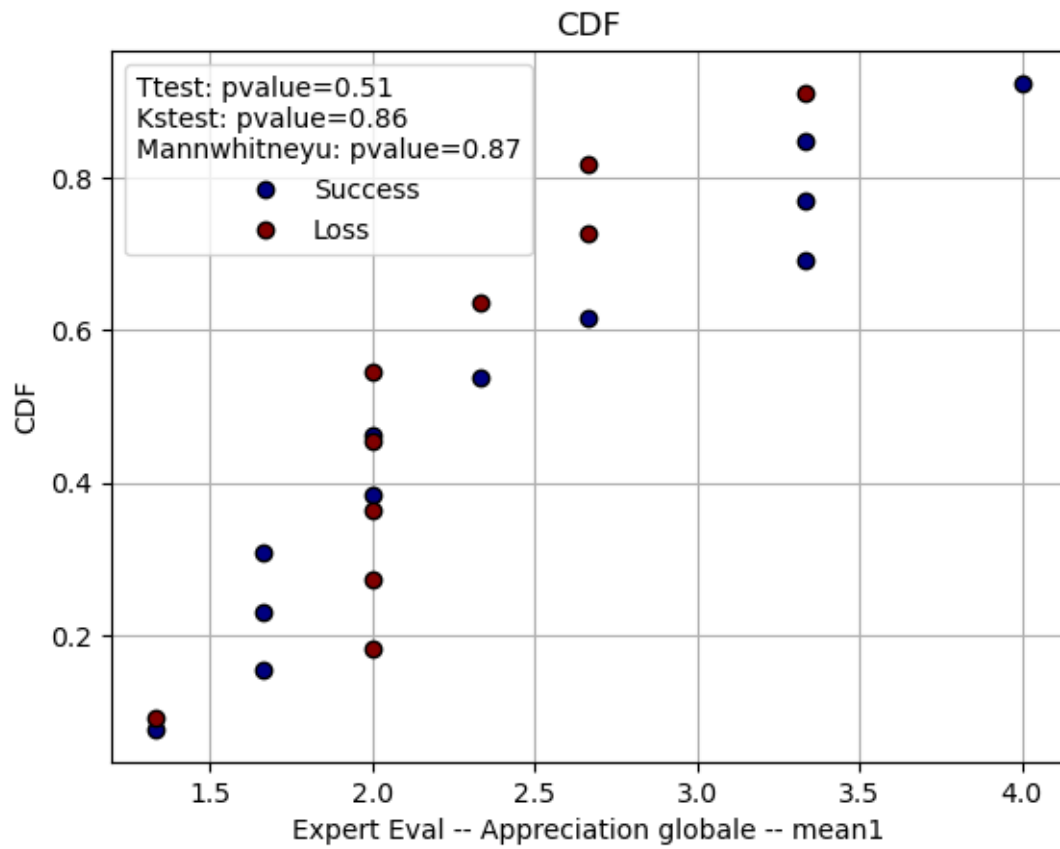


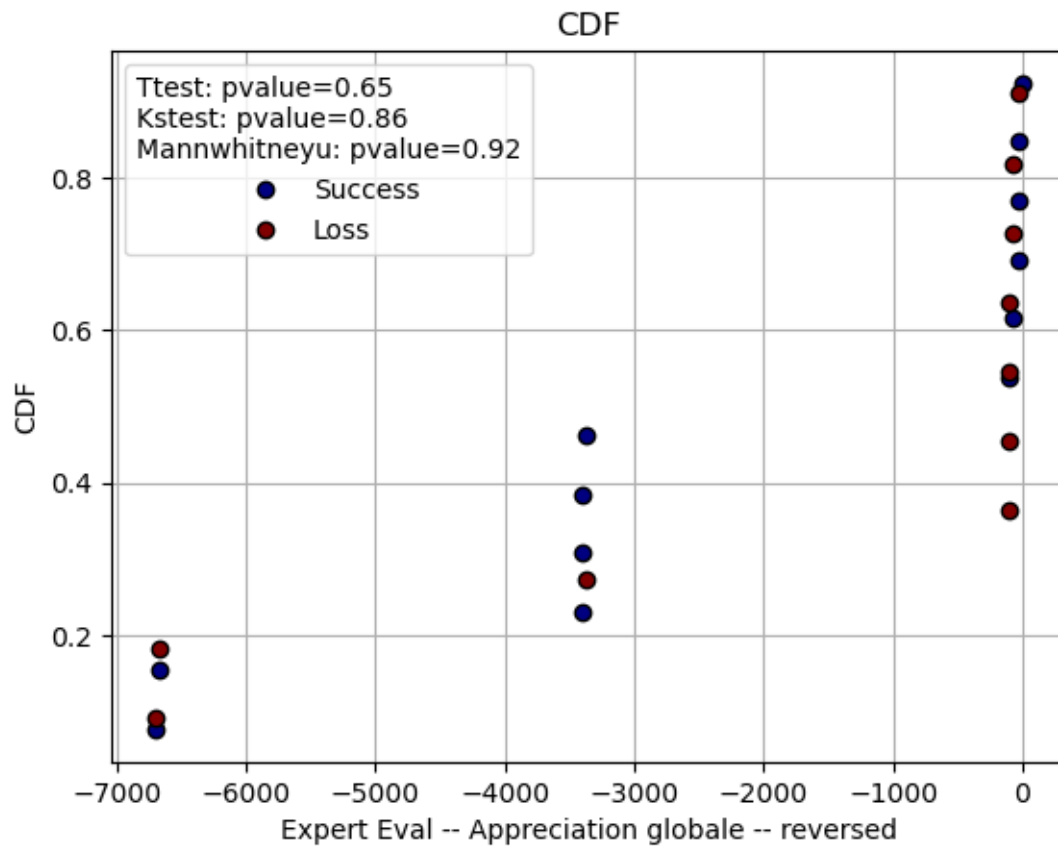


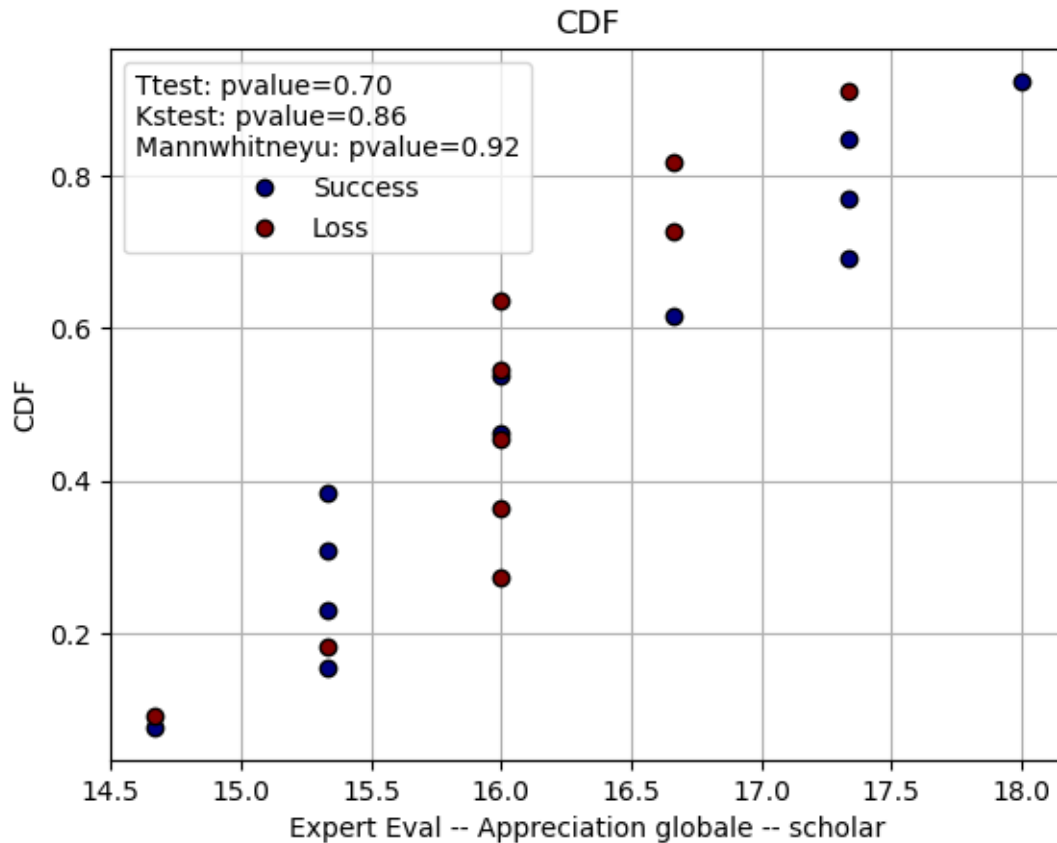












À retenir : Les deux échantillons ne sont pas significativement différents → dit autrement: qu'un projet soit financé ou non, ne dit rien des notes qu'il a obtenu des experts. Dit encore autrement s'il y avait deux d'experts représentant: 1) la distribution des notes obtenues par les projets financés et 2) celle des projets non financés alors ce seraient les mêmes d'experts. Ou dit encore autrement: le comité n'a pas réussi à trouver d'information pertinente dans l'évaluation des experts pour interclasser les projets.

3 Conclusion

Cette analyse ne dit rien des mécanismes et raisons qui se sont produits entre l'évaluation des experts et le classement final. Mais quelque soit ces mécanismes et raisons et **quand bien même cela serait raisonnable** (e.g. poids de l'oral largement supérieur aux notes experts, similarité dans les disciplines des projets déposés, etc) : les experts ont perdu leur temps, leur évaluation n'a servi à rien.

[]: