tcarron_exercise7

May 22, 2022

1 Exercise Set 7

Due: **9:30 23 May 2022**

Discussion: **13:00 27 May 2022**

Online submission at via ILIAS in the directory Exercises / Übungen -> Submission of Exercises / Rückgabe des Übungsblätter

[ ]:
```
# import necessary packages
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy.stats as st
import seaborn as sb
```

2 1. Kendall's tau coefficient [40 Points]

In statistics, the Kendall rank correlation coefficient, commonly referred to as Kendall's tau coefficient (after the Greek letter τ), is a statistic used to measure the ordinal association between two measured quantities. A tau test is a non-parametric hypothesis test for statistical dependence based on the tau coefficient. Spearman's rank correlation is satisfactory for testing a null hypothesis of independence between two variables but it is difficult to interpret when the null hypothesis is rejected. Kendall's rank correlation improves upon this by reflecting the strength of the dependence between the variables being compared.

Consider two samples, x and y, each of size n. The total number of possible pairings of x with y observations is $\frac{n(n-1)}{2}$. Now consider pairs of ordered observations (x_i, y_i) and (x_j, y_j) with $i = 1, \ldots n-1$ and $j = i+1, \ldots, n$. Then $x_1 \leq x_2 \leq \ldots \leq x_n$. Now pair 1 is compared with every other pair $(2, 3, , , n)$, Pair 2 with all following pairs $(3, 4, , , n)$ and so on. In total $\frac{n(n-1)}{2}$ pairs are compared. If for a pair:

- $x_i < x_j$ and $y_i < y_j$, it is concordant
- $x_i < x_j$ and $y_i > y_j$, it is disconcordant
- $x_i \neq x_j$ and $y_i = y_j$, it has a binding in Y
- $x_i = x_j$ and $y_i \neq y_j$, it has a binding in X
- $x_i = x_j$ and $y_i = y_j$, it has a binding in X and Y

The number of pairs that are:

- concordant are C
- disconcordant are D
- having a binding in Y are T_Y
- having a binding in X are T_X
- having a binding in X and Y are T_{XY}

Kendall's tau value compares the number of concordant and disconcordant pairs:

$$\tau = \frac{C - D}{\sqrt{(C + D + T_X)(C + D + T_Y)}}$$

The denominator is the total number of pair combinations, so the coefficient must be in the range $-1 \leq \tau \leq 1$. If the agreement between the two rankings is perfect (when the two rankings are the same) the coefficient has value 1. If the disagreement between the two rankings is perfect (when one ranking is the reverse of the other) the coefficient has value -1.

Use the data file `hubble.dat`, which includes dataset x in the first column and dataset y in the second column.

a. Take the data and reorder it such that $x_1 \leq x_2 \leq \ldots \leq x_n$. **10 Points**

[ ]:
```
df = pd.read_csv("hubble.dat", delimiter="\s+", names=["A", "B"])
# sort "A" into ascending order
a = np.sort(df["A"].to_numpy())
b = df["B"].to_numpy()
print(df)
```

```
       A       B
0   0.04   111.1
1   0.03   -83.3
2   0.19    97.2
3   0.25    27.8
4   0.27   -69.4
5   0.26  -208.3
6   0.42   819.4
7   0.50   819.4
8   0.50   958.3
9   0.63   666.7
10  0.79   777.8
11  0.89   194.4
12  0.89   430.6
13  0.88   888.9
14  0.91  1222.2
15  1.01  1736.1
16  1.10  1472.2
17  1.11  1166.7
18  1.42  1263.9
19  1.70  2111.1
20  2.02  1111.1
```

```
21  2.01  1611.1
22  2.02  1763.9
23  2.02  2250.0
```

b. Compute the number of concordant pairs C, disconcordant pairs D, bound pairs T_X, T_Y, and T_{XY}. **10 Points**

[ ]:
```
# Function that takes a and b as an argument and returns C,D,T_y,T_x and T_xy.
def count_pairs(a, b):
    # numbers of each pair type to be incremented
    c = 0
    d = 0
    ty = 0
    tx = 0
    txy = 0
    # loop over all pairs
    for i in range(len(a) - 1):
        pair1 = [a[i], b[i]]  # set the first pair
        # print("i",i)
        j = i + 1
        # loop over all pairs greater than first pair
        while j < len(a):
            # print("j",j)
            pair2 = [a[j], b[j]]  # set the second pair
            # Check conditions for concordance/binding and increment counters
            if pair1[0] < pair2[0] and pair1[1] < pair2[1]:
                c += 1
            elif pair1[0] < pair2[0] and pair1[1] > pair2[1]:
                d += 1
            elif pair1[0] != pair2[0] and pair1[1] == pair2[1]:
                ty += 1
            elif pair1[0] == pair2[0] and pair1[1] != pair2[1]:
                tx += 1
            elif pair1[0] == pair2[0] and pair1[1] == pair2[1]:
                txy += 1
            else:
                print("oops")
            j = j + 1
    return c, d, ty, tx, txy
```

[ ]:
```
print("c, d, ty, tx, txy", count_pairs(a, b))
```

```
c, d, ty, tx, txy (226, 44, 1, 5, 0)
```

c. Compute τ. **10 Points**

Here I compute $\tau = 0.666$

[ ]:
```
# Function to compute Kendall's tau value given arrays a and b
def kendall_tau(a, b):
    # calculate c, d, ty, tx, txy for a and b
    c, d, ty, tx, txy = count_pairs(a, b)
    # compute numerator
    numer = c - d
    # compute denominator
    denom = np.sqrt((c + d + tx) * (c + d + ty))
    # compute tau
    tau = numer / denom
    return tau


# I have just realised there is also scipy.stats.kendalltau ......
```

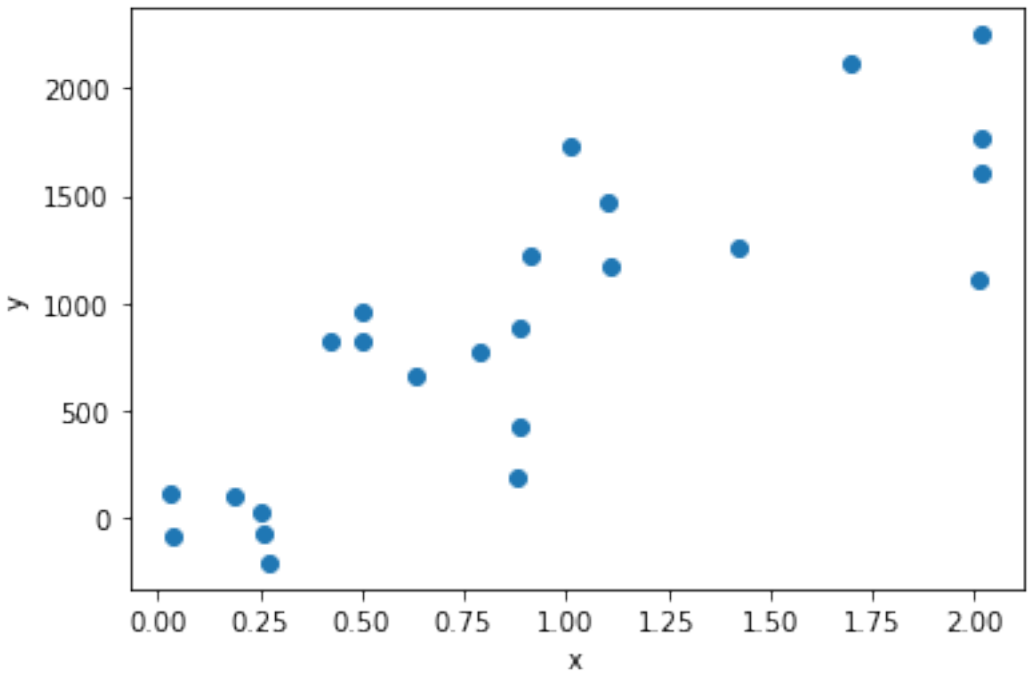
[ ]:
```
#printing both just to check
print("tau=", kendall_tau(a, b))
tau = kendall_tau(a, b)
tau2 = st.kendalltau(a, b)
print("ta2u=", tau2)
```

```
tau= 0.6666845575146066
ta2u= KendalltauResult(correlation=0.6666845575146064,
pvalue=6.077846491041711e-06)
```

d. How should the derived τ be interpreted (what does it tell about the two datasets)? Plot the dataset x vs. y and compare it with τ **10 Points**

The τ value tells us about the correlation between x and y. The plot of x and y shows a moderate positive correlation as one would expect for $\tau = 0.666$

[ ]:
```
plt.figure()
plt.scatter(a, b)
plt.xlabel("x")
plt.ylabel("y")
plt.savefig("plots/xxy_tau.png",dpi=400,bbox_inches="tight")
```



3 2. Permutation tests [40 Points]

This exercise uses the data file `4point2.dat`. The data in the file are 20 uncorrelated (x, y) pairs, followed by 20 correlated pairs.

[ ]:
```
# read in and sort data
df2 = pd.read_csv("4point2.dat", delimiter=",", names=["x", "y"])
# uncorrelated
df2_uncorr = df2.iloc[0:20]
x_uncorr = df2_uncorr["x"].to_numpy()
y_uncorr = df2_uncorr["y"].to_numpy()
# correlated
df2_corr = df2.iloc[20:40]
x_corr = df2_corr["x"].to_numpy()
y_corr = df2_corr["y"].to_numpy()
#print(x_uncorr, y_uncorr)
print(len(x_corr),len(x_uncorr))
```

```
20 20
```

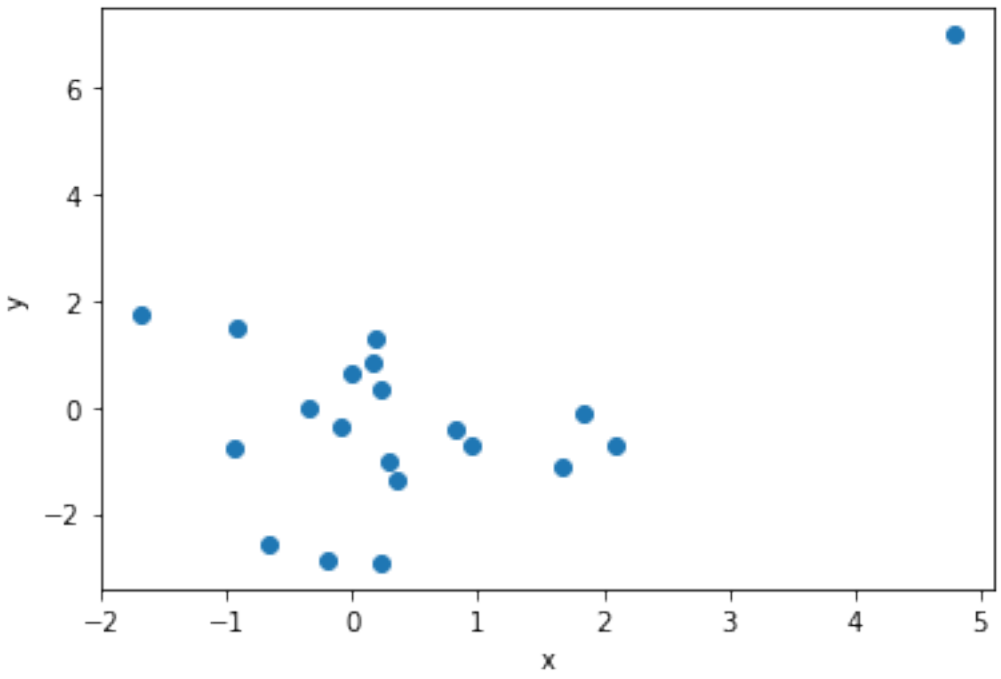
a. Take the uncorrelated pairs (x, y) and compute the values for the Kendall and Spearman rank correlation coefficients and the Peason/Fischer r correlation coefficient. Why are they different? **20 Points**

(This is a good example why we should not blindly rely on the Peason's correlation coefficient)

Below we computed the Kendall and Spearman rank correlation coefficients and the Pearson/Fisher r correlation coefficient. The Kendall and Spearman coefficients show very little correlation, while the Pearson/Fisher r coefficient shows a moderate positive linear correlation. Clearly the three coefficients do not agree. As the plot illustrates, the Pearson/Fisher r coefficient fails when there are outliers in the data.

[ ]:
```
# values for uncorrelated
uncorr_tau = st.kendalltau(x_uncorr, y_uncorr)
uncorr_spearman = st.spearmanr(x_uncorr, y_uncorr)
uncorr_pearson = st.pearsonr(x_uncorr, y_uncorr)
print("-------------------------------------------------")
print("uncorrelated:")
print("tau=", uncorr_tau)
print("spearman=", uncorr_spearman)
print("pearson/Fisher r=", uncorr_pearson)
print("-------------------------------------------------")
plt.figure()
plt.scatter(x_uncorr, y_uncorr)
plt.xlabel("x")
plt.ylabel("y")
plt.savefig("plots/uncorr.png",dpi=400,bbox_inches="tight")
```

```
-------------------------------------------------
uncorrelated:
tau= KendalltauResult(correlation=-0.06315789473684211,
pvalue=0.7246362205272812)
spearman= SpearmanrResult(correlation=-0.08270676691729321,
pvalue=0.7288508763559796)
pearson/Fisher r= (0.4866552193293831, 0.02955758242994905)
-------------------------------------------------
```

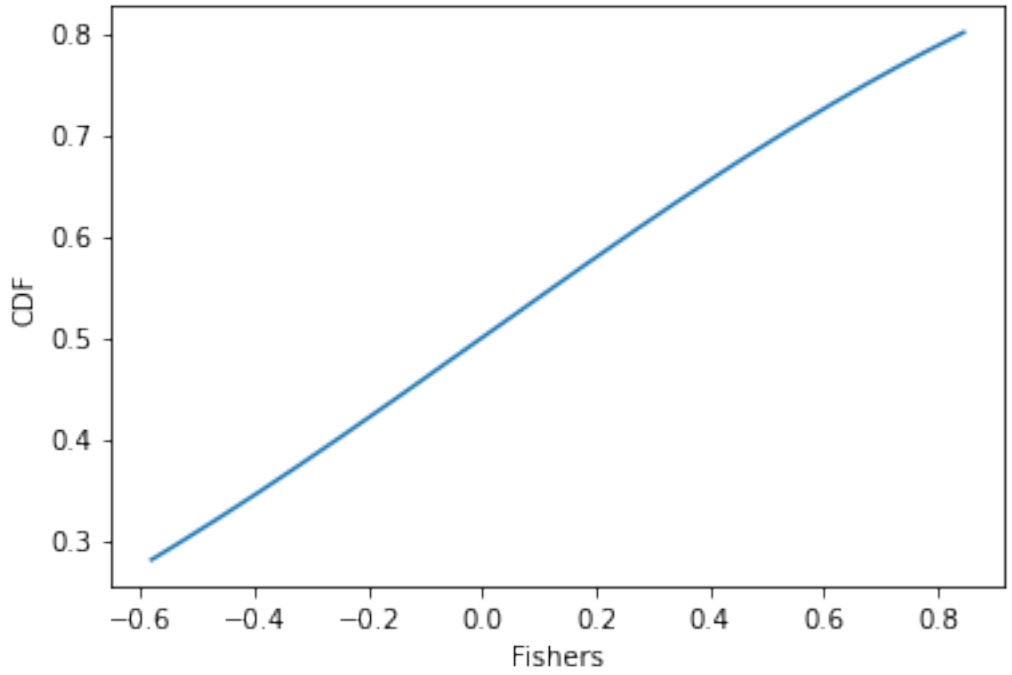


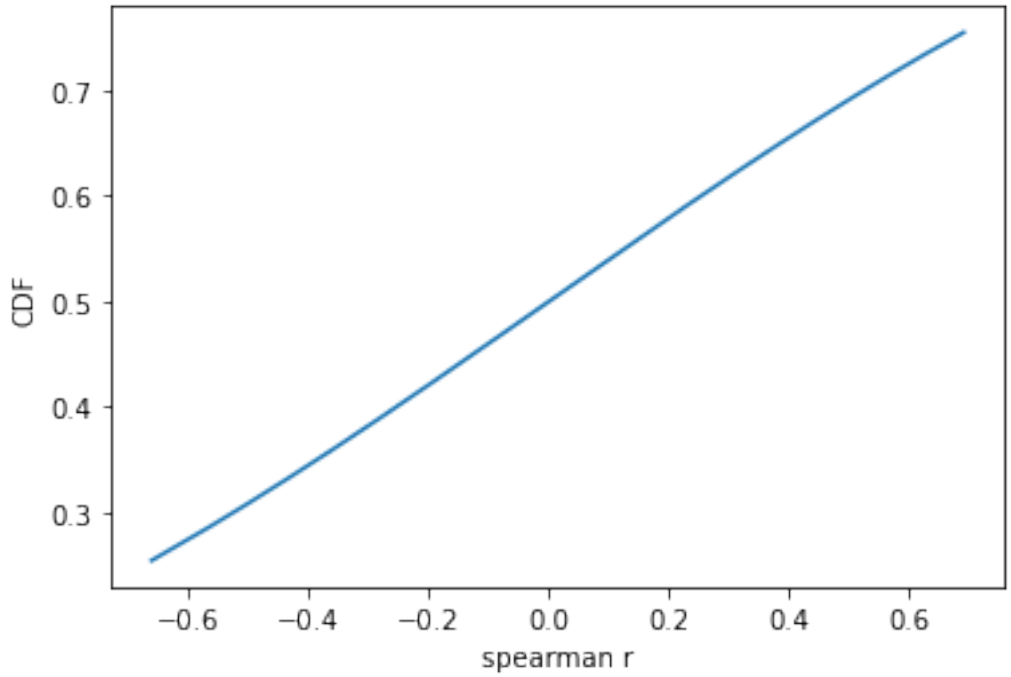
b. By permutation methods we want to derive distributions of Fisher r, Spearman's and Kendall's statistics. To do this, first we take the uncorrelated data set and randomly assign x-values to y-values to make new pairs. This should give the range of values of the test statistic which are consistent with there being no correlation. There are 20! distinct permutations for even this little dataset. Compute your results for 1000 of these at random. (This is achieved by sampling without replacement from the set of 20 X's and assigning each one in order to the set of Y's) For each of the 1000 random samples compute their Fisher r value, Spearman's and Kendall's statistics and plot their cumulative distributions. Explain what the obtained plots tell. **20 Points**

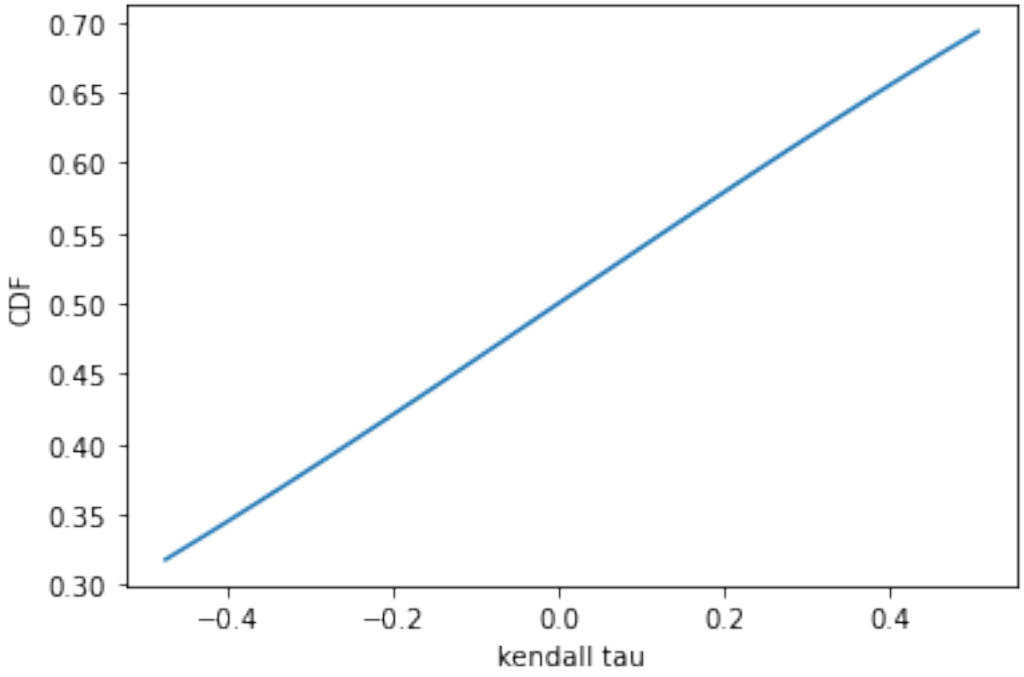
[ ]:
```
x = x_uncorr
y = y_uncorr
fishers = []
spearmans = []
kendalls = []
for i in range(1000):
    temp_x = np.random.choice(x, 20, replace=False)
    fishers.append(st.pearsonr(temp_x, y)[0])
    spearmans.append(st.spearmanr(temp_x, y)[0])
    kendalls.append(st.kendalltau(temp_x, y)[0])


plt.figure(0)
sb.lineplot(x=fishers, y=st.norm.cdf(fishers))
plt.xlabel("Fishers")
plt.ylabel("CDF")
```

```
plt.savefig("plots/fig0.png",dpi=400,bbox_inches="tight")
plt.figure(1)
sb.lineplot(x=spearmans, y=st.norm.cdf(spearmans))
plt.xlabel("spearman r")
plt.ylabel("CDF")
plt.savefig("plots/fig1.png",dpi=400,bbox_inches="tight")
plt.figure(2)
sb.lineplot(x=kendalls, y=st.norm.cdf(kendalls))
plt.xlabel("kendall tau")
plt.ylabel("CDF")
plt.savefig("plots/fig2.png",dpi=400,bbox_inches="tight")
```



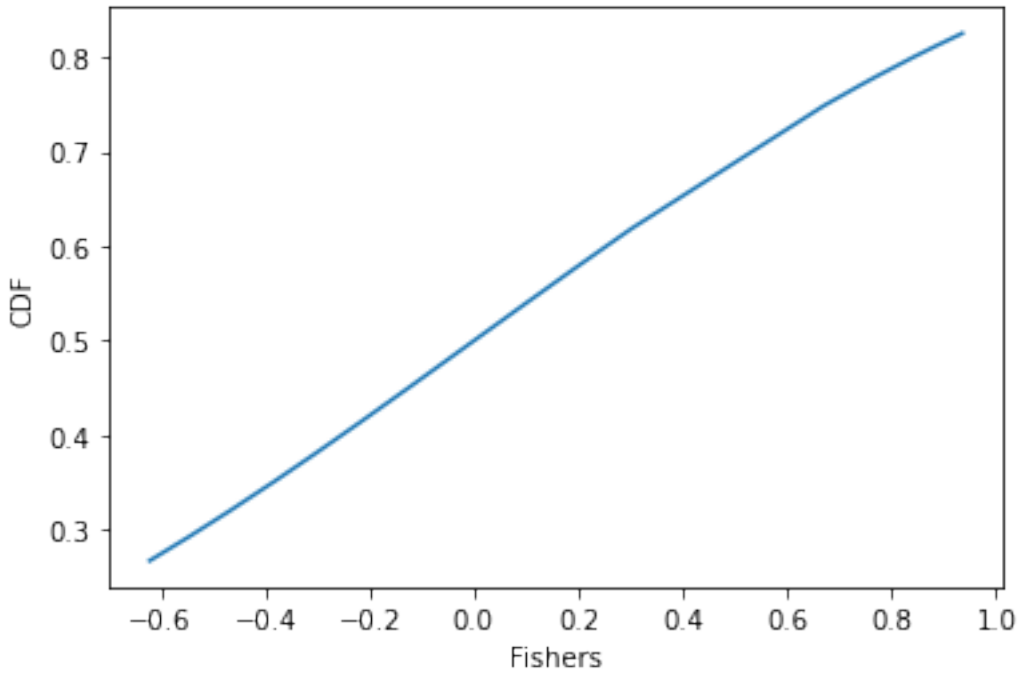


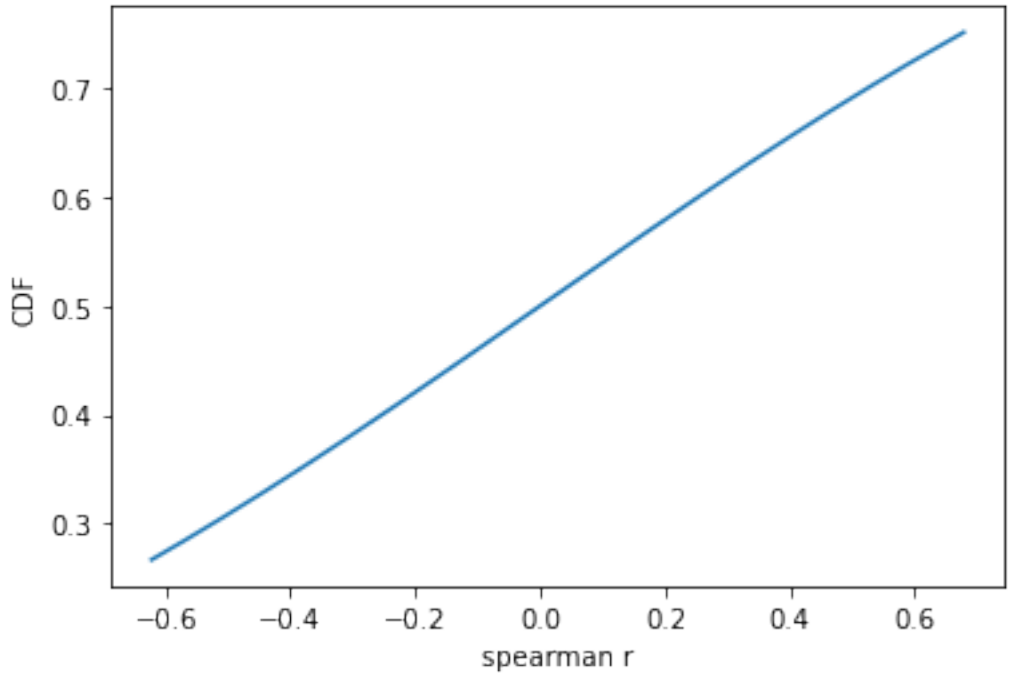


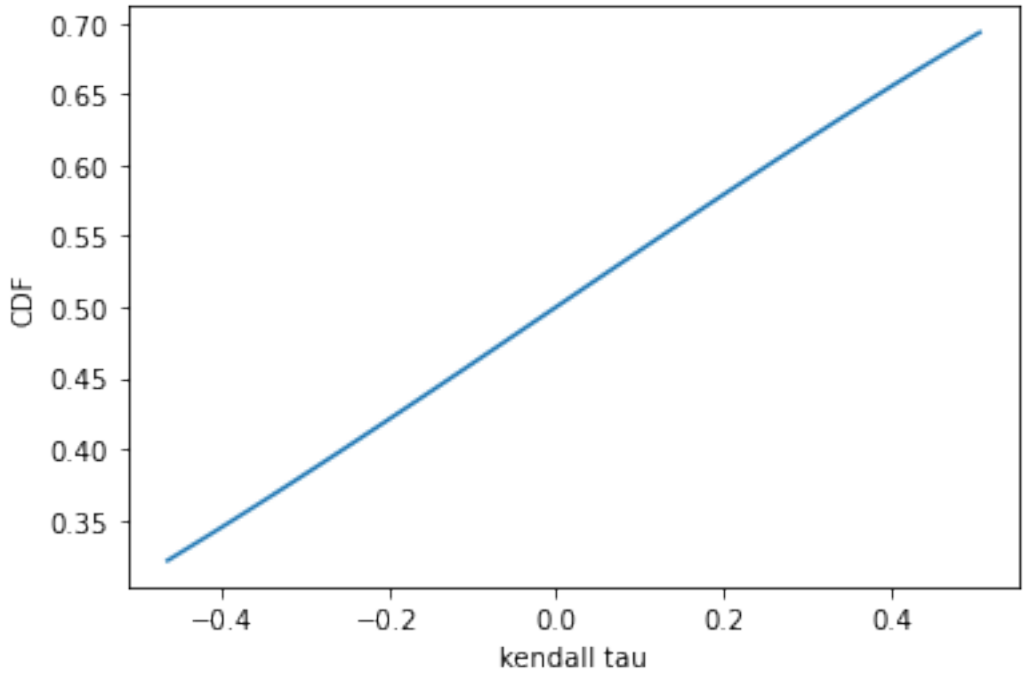
c. Now take the correlated set of data and do the same as **b)**. Interprete the results with comparing the plots of uncorrelated datasets. **20 Points**

[ ]:
```
x = x_corr
y = y_corr
fishers = []
spearmans = []
kendalls = []
for i in range(1000):
    temp_x = np.random.choice(x, 20, replace=False)
    fishers.append(st.pearsonr(temp_x, y)[0])
    spearmans.append(st.spearmanr(temp_x, y)[0])
    kendalls.append(st.kendalltau(temp_x, y)[0])


plt.figure(3)
sb.lineplot(x=fishers, y=st.norm.cdf(fishers))
plt.xlabel("Fishers")
plt.ylabel("CDF")
plt.savefig("plots/fig3.png",dpi=400,bbox_inches="tight")
plt.figure(4)
sb.lineplot(x=spearmans, y=st.norm.cdf(spearmans))
plt.xlabel("spearman r")
plt.ylabel("CDF")
plt.savefig("plots/fig4.png",dpi=400,bbox_inches="tight")
plt.figure(5)
sb.lineplot(x=kendalls, y=st.norm.cdf(kendalls))
plt.xlabel("kendall tau")
plt.ylabel("CDF")
plt.savefig("plots/fig5.png",dpi=400,bbox_inches="tight")
```




0.7
0.6
0.5
0.4
0.3
CDF
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
spearman r


0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
CDF
−0.4
−0.2
0.0
0.2
0.4
kendall tau

Interpretation: The cumulative distribution functions above resemble those of uniform distributions. The only difference between the correlated data and the uncorrelated data is the limits of the given correlation measure, i.e the limits of the x-axis on the CDF plots. Taking the Pearson/Fisher plots as an example, the r value for the correlated data occupies the interval [-0.6,1]. For the uncorrelated data, r lies in the interval [-0.6,0.8]