E6 spec sheet:

totalKS

Mission command preamble: As in general, we won't tell you *how* to do something. That is up to you and your creative problem solving skills. However, we will tell you what we would like this function to do. So here are the specs of the function we would like you to write.

Purpose: The Kolmogorov-Smirnov (KS) test is a classic example of a goodness-of-fit hypothesis test. There are several implementations of this test. In one form, the test determines if a sample plausibly could have come from a population with a known distribution, such as the standard normal distribution. Another implementation tests whether two samples could plausibly have come from the same underlying distribution. How this test works is still somewhat puzzling to students, so we will re-create it here. We will also create a version that uses the full difference under the curve, not just the point of largest separation.

Specific things we would like this function to do:

a) Take in three inputs as arguments: X1 (Sample 1), X2 (Sample2) and a flag, where 1 = classical KS and 2 = total KS.

b) If the flag in a) is set to 1, compute the largest distance between the two cumulative sample distributions. If the flag is set to 2, compute the total difference between the two sample distributions.

c) The function should return the value computed in b)

d) Assumptions: The two samples can be input as 1D numpy arrays, dataframes or even lists. By default, you should assume the inputs to be 1D numpy arrays of arbitrary length.

e) Make sure the function has a clear header as to what inputs the function assumes, what outputs it produces and when it was written.

Hints:

Step 1: Convert each of the two samples (separately) into a cumulative density distribution, from 0 to 100. You can do this manually – by first sorting each sample into ascending values and determining the percentiles or by using an empirical cumulative density function (ecdf). If you do this manually, percentile-resolution is sufficient.

Step 2: If the flag is set to 1, find the x-value (measures) where the two samples have the largest separation in terms of percentiles (y)

Step 3: Find the absolute difference in percentiles at that point. That is the largest distance – return this value if the flag is set to 1. If the flag is set to 2, you can just sweep through all x-values and determine the sum of all the absolute differences (in percentiles) between the two cumulative sample distributions. This absolute distance should be accurate to 2 decimal places (if expressed as a proportion) or the nearest percentile (if expressed as percentiles).

Total separation: You can solve the case for flag 2 with approximate numerical integration. In other words, assess the absolute distance between the two samples (in y-direction) at 1,000 evenly spaced points spanning the entire x-range of the two samples. Return the mean absolute separation value (either as a percentile or a proportion).

Input / output examples:

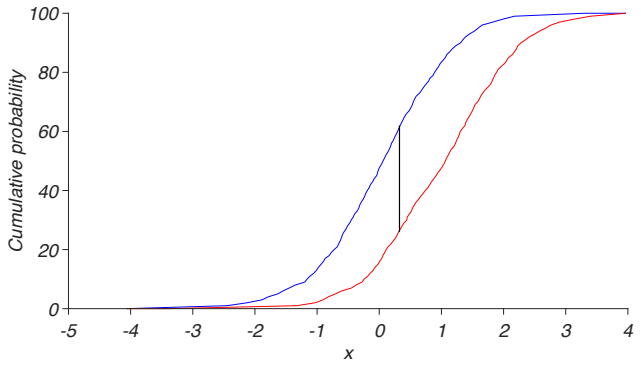
Input: kSinput1.csv

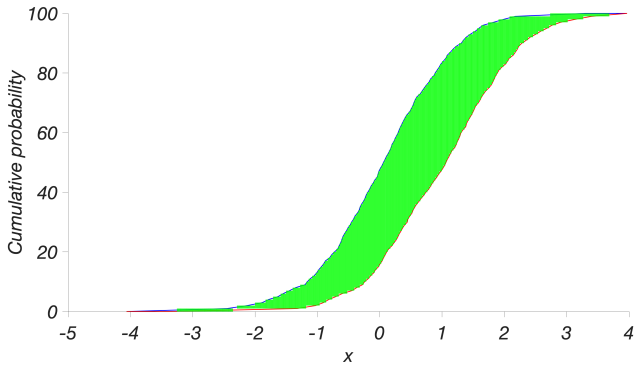
Output:

D = 36 (if flag = 1 and output is percentiles) mean D_{total} = 12.1 (if flag = 2, in percentiles)

D = 0.36 (if flag = 1 and output is cumulative proportion) mean D_{total} = 0.121 (if flag = 2, in cum prop)

This is a visualization of the situation. Note that this is provided just for illustration purposes. Your function does not need to supply such a visualization. Left figure: blue curve = empirical cumulative density function of sample A, red curve = empirical cumulative density function of sample B, black line = largest separation between these two empirical distributions. Right figure: Same as on the left, but green = difference between the two curves





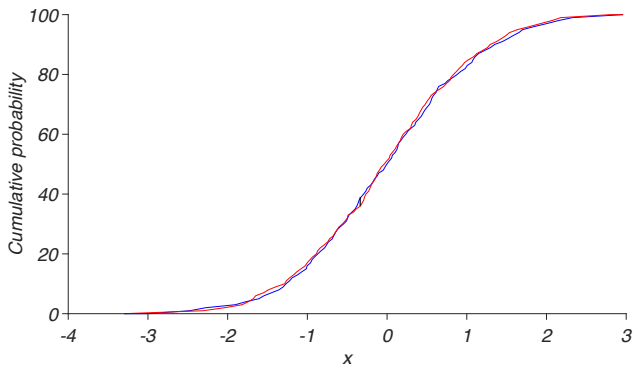
Input: kSinput2.csv

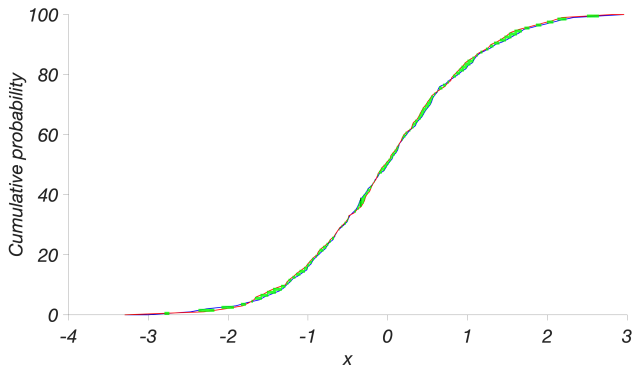
Output:

D = 3 (if flag = 1 and output is percentiles) mean D_{total} = 0.69 (if flag = 2, in percentiles)

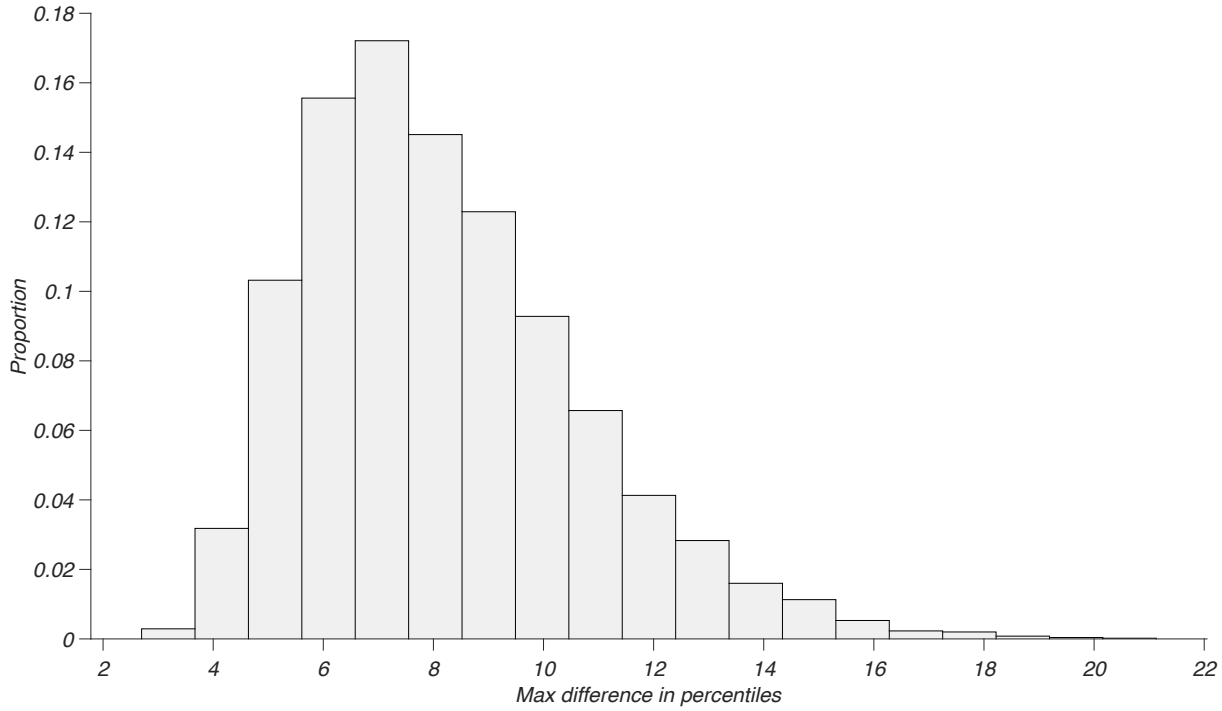
D = 0.03 (if flag = 1 and output is cumulative proportion) mean D_{total} = 0.0069 (if flag = 2, in cum prop)

Below is again a visualization of the situation, for illustration purposes using the same color code as above. Again, this is just an illustration. Your function does not have to produce such a graphic.





Note that your function just needs to yield/return the test statistic, not the p-value as well. If you feel so inclined, you could also return a p-value by calculating a null-distribution of test-statistics for the sample size in question, here for n = 200 and for drawing from a standard normal distribution without a systematic mean difference between samples X1 and X2:



Here is a comparable null distribution (same inputs) for the total mean difference: