E3 spec sheet:

Descriptive statistics in a sliding window

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 a function to do. So here are the specs of the function we would like you to write.

Purpose: Most classical descriptive statistics approaches calculate the parameter over the entire range of a variable. This assumes that the generative process that created the data is stationary – doesn't change over the entire range of the dataset As we will see soon – when exploring time series or data that varies as a function of spatial location – that cannot necessarily be assumed. There are many potential use cases. For instance, let's say that the correlation between the price of gold and stock prices is anticorrelated at -0.3, and the correlation between bonds and housing prices is 0 in the past. If this is the case, you can use these commodities to hedge your portfolio. However, this presumes that this correlation doesn't change. You might want to track that in real time – maybe with a relatively narrow 14 day window - because when all these metrics start to correlate strongly and positively, it is usually a leading sign of a major market crash - like in 2008 – and the right move is to exit the market and to move into cash, quickly.
Correlating over the entire set of available data would give you a false sense of security. How the correlation develops over time is in itself the parameter of interest. There are many other use cases (for example gene expression across a chromosome) where we can't simply assume stationarity. So we need a function that computes these parameters over a subset (window) of the input data, then moves the window. This is what we are asking you to do here.

Specific things we would like this function to do:

a) Take in three arguments/inputs, in this order: 1) An input dataset (make it a 1- or 2D numpy array), 2) a flag for which parameter to calculate, where 1 = mean, 2 = SD and 3 = correlation and 3) the window size (the subset of how many numbers of the dataset to compute the parameter indicated in 2) over).

b) Calculate the parameter described in a2 for the data in a1 over the window size specified in a3. For instance, if we give an input of a 1D numpy array with length 100 numbers, and ask it to calculate the mean, with a window size of 20, the function should calculate the mean from the beginning of the input array (index 0) to location 19, then from location 1 to 20, then from location 2 to 21, and so on, until the last 20 entries in the data array yield the last mean. You can assume an overlap of windowsize-1, so only shift the window by 1 position per calculation.

c) The outputs of this sliding windowed calculation should be assigned to an output array, with one entry, e.g. the mean, sd or correlation per calculation. This output array will be somewhat shorter than the input array, as the sliding should stop once the window hits the end of the input array. Don't zero-pad.

d) The function should return the output array from c), once all calculations are done.

e) The function should be flexible, i.e. output the windowed mean if the input flag in a2 is 1, the windowed SD if it is 2 and the windowed correlation if it is 3.

f) Assumptions: You can assume that the input will be a 2D array (with variables in columns and measures/numbers in rows) if the a2 flag is correlation and a 1D array if the a2 flag is mean or sd.

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

Input / output examples:

Input array A:

1	3	5	7	9

Output of slidWinDescStats(A,1,3):

3	5	7

Output of slidWinDescStats(A,1,4):

4	6

Output of slidWinDescStats(A,2,3):

2	2	2

Output of slidWinDescStats(A,2,4):

2.58	2.58

Note that these results are for the sample standard deviation. If your function calculates the population standard deviation, the results would be a vector of length 3 with 1.633 as entries for window size 3 and a vector of length 2 with 2.236 as entries for window size 2.

Input array B:

1	1
3	2
5	4
7	3
9	5

Output of slidWinDescStats(B,3,3):

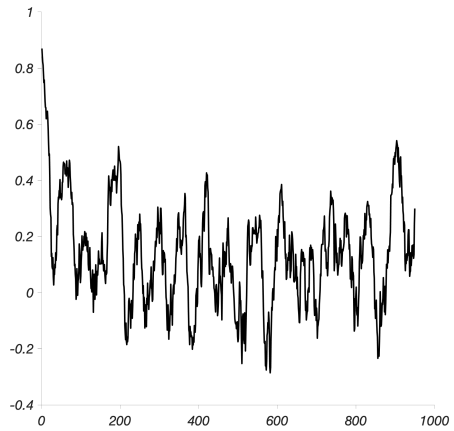
0.982
0.5
0.5

Output of slidWinDescStats(B,3,4):

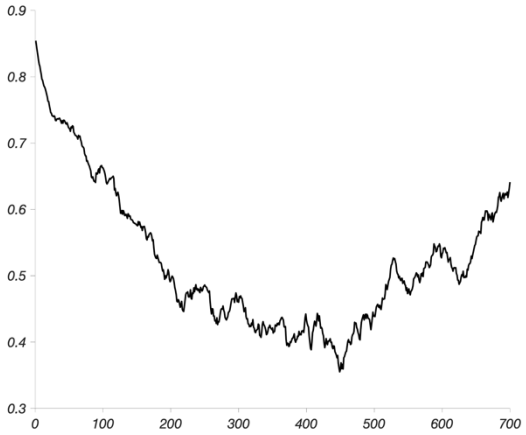
0.8
0.8

If you want to test with a more typical – and more large scale – example load inputArrayExample.csv and outputArrayExample50.csv and test if your function can produce the output array from the input array, asking for a correlation (flag 3) and window size 50. OutputArrayExample300.csv is the same, except for window size 300. This also primes the notion of removing noise by smoothing.

The plot of the windowed correlations should look something like this:



Window size 50



Window size 300