OSTSC

ESPO

1. Calculating the mean vector for each column of the positive data, which is the mean value for each time point

   M

	1	2	3	4	5	6	7	8	9	10
1	0.5336	0.5509	0.5539	0.5843	0.5990	0.6166	0.6228	0.6582	0.7218	0.7820

   R

```
> Me
 [1] 0.5336481 0.5509333 0.5538938 0.5843321 0.5990064 0.6165761 0.6227543 0.6581734 0.7218326 0.7819764
```

2. Calculating the covariance matrix of the positive data (column)

   M

	1	2	3	4	5	6	7	8	9	10
1	0.1349	0.0885	0.0666	0.0450	0.0301	0.0178	0.0119	0.0021	-8.8018e-05	-0.0041
2	0.0885	0.1301	0.0877	0.0577	0.0432	0.0296	0.0220	0.0033	0.0022	-0.0021
3	0.0666	0.0877	0.1370	0.0797	0.0534	0.0399	0.0291	0.0177	0.0110	-0.0019
4	0.0450	0.0577	0.0797	0.1337	0.0931	0.0642	0.0451	0.0305	0.0212	-0.0031
5	0.0301	0.0432	0.0534	0.0931	0.1362	0.0854	0.0578	0.0344	0.0233	0.0010
6	0.0178	0.0296	0.0399	0.0642	0.0854	0.1345	0.0862	0.0506	0.0293	-0.0037
7	0.0119	0.0220	0.0291	0.0451	0.0578	0.0862	0.1399	0.0782	0.0472	-4.0324e-04
8	0.0021	0.0033	0.0177	0.0305	0.0344	0.0506	0.0782	0.1359	0.0663	0.0058
9	-8.8018e-05	0.0022	0.0110	0.0212	0.0233	0.0293	0.0472	0.0663	0.1269	0.0156
10	-0.0041	-0.0021	-0.0019	-0.0031	0.0010	-0.0037	-4.0324e-04	0.0058	0.0156	0.0679

   R

1.348631e-01	0.088459918	0.066609461	0.045016128	0.030148237	0.017843393	0.0118544017	0.002104608	-8.801822e-05	-0.0041297934
8.845992e-02	0.130085605	0.087697139	0.057655225	0.043202252	0.029569199	0.0219744459	0.003333716	2.236872e-03	-0.0021233176
6.660946e-02	0.087697139	0.136970050	0.079659664	0.053411609	0.039911509	0.0290948953	0.017745992	1.101591e-02	-0.0019177316
4.501613e-02	0.057655225	0.079659664	0.133735282	0.093117489	0.064167345	0.0451460214	0.030523237	2.116930e-02	-0.0031046857
3.014824e-02	0.043202252	0.053411609	0.093117489	0.136227948	0.085361608	0.0578129402	0.034435713	2.332335e-02	0.0010167466
1.784339e-02	0.029569199	0.039911509	0.064167345	0.085361608	0.134545603	0.0861548370	0.050641261	2.925992e-02	-0.0037260883
1.185440e-02	0.021974446	0.029094895	0.045146021	0.057812940	0.086154837	0.1398524642	0.078195882	4.724219e-02	-0.0004032377
2.104608e-03	0.003333716	0.017745992	0.030523237	0.034435713	0.050641261	0.0781958817	0.135866161	6.628068e-02	0.0057526227
-8.801822e-05	0.002236872	0.011015912	0.021169301	0.023323347	0.029259920	0.0472421896	0.066280679	1.269116e-01	0.0156174350
-4.129793e-03	-0.002123318	-0.001917732	-0.003104686	0.001016747	-0.003726088	-0.0004032377	0.005752623	1.561744e-02	0.0678758833

3. Calculating eigenvalues and eigenvectors of the covariance matrix PCov

   D: diagonal matrix D of eigenvalues

   V: matrix V whose columns are the relating eigenvectors

$$PCov*V = V*D$$

D:

M

0.4964	0.2599	0.1347	0.0925	0.0682	0.0614	0.0555	0.0409	0.0373	0.0302

R

```
> D
 [1] 0.49636325 0.25985357 0.13466144 0.09252087 0.06820998 0.06144125 0.05551067 0.04088814 0.03731218
[10] 0.03017229
```

V: (Some colums have opposite +/- , I searched this issue online, and it's because the different algorism used by Matlab/R. The source codes of Matlab eig() was in C/C++, so there's no way to find how Matlab calculates the eigenvectors. I don't know how to make the eigenvectors of Matlab and R exactly same.)

M

-0.2615	-0.4298	0.3494	-0.2816	-0.3661	-0.3719	-0.2849	-0.0575	-0.4048	0.1575
-0.3179	-0.4200	0.2442	-0.1462	-0.0952	0.1241	0.1559	0.0810	0.6713	-0.3682
-0.3656	-0.3296	0.1321	0.1609	0.4690	0.4117	0.2535	0.1692	-0.2583	0.4066
-0.4100	-0.1096	-0.2534	0.4043	0.2328	-0.1324	-0.1908	-0.3938	-0.2230	-0.5285
-0.4030	0.0426	-0.4375	0.2673	-0.2364	-0.1995	-0.1994	0.0525	0.3887	0.5326
-0.3808	0.2235	-0.3491	-0.2654	-0.2513	0.0478	0.2658	0.5503	-0.3017	-0.2878
-0.3483	0.3704	0.0169	-0.4887	-0.0248	0.2410	0.1517	-0.6317	0.0266	0.1448
-0.2583	0.4407	0.3734	-0.0966	0.3638	-0.0158	-0.5832	0.3174	0.1177	-0.0555
-0.1827	0.3631	0.5129	0.4623	-0.1967	-0.3025	0.4763	-0.0211	-0.0174	0.0144
0.0013	0.0568	0.1434	0.3259	-0.5405	0.6836	-0.3058	-0.0266	-0.1096	-0.0651

R

-0.261507059	0.42978637	0.34936596	-0.28157959	0.36613636	-0.37192209	0.2849205	-0.05746296	-0.40483146	0.15748660
-0.317875818	0.41997455	0.24424110	-0.14620661	0.09520869	0.12411448	-0.1558977	0.08098279	0.67130860	-0.36818422
-0.365638105	0.32964422	0.13214446	0.16088133	-0.46895140	0.41168024	-0.2534569	0.16920526	-0.25827333	0.40660369
-0.409965114	0.10963756	-0.25339974	0.40429130	-0.23280013	-0.13237739	0.1907754	-0.39377525	-0.22303791	-0.52851257
-0.402959415	-0.04264977	-0.43749067	0.26729884	0.23640917	-0.19948974	0.1993740	0.05248792	0.38870072	0.53261549
-0.380849819	-0.22346112	-0.34912902	-0.26537676	0.25125837	0.04778463	-0.2658154	0.55025136	-0.30172641	-0.28777646
-0.348284214	-0.37037456	0.01685378	-0.48869311	0.02484174	0.24099872	-0.1516519	-0.63169786	0.02655727	0.14481887
-0.258282371	-0.44069087	0.37342230	-0.09657748	-0.36376604	-0.01579662	0.5831573	0.31737148	0.11774244	-0.05545024
-0.182675296	-0.36311444	0.51286172	0.46234022	0.19665053	-0.30247974	-0.4763104	-0.02111650	-0.01735701	0.01441772
0.001260301	-0.05681208	0.14342790	0.32591741	0.54049405	0.68363859	0.3058275	-0.02657336	-0.10958249	-0.06508919

4. Calculating the covariance matrix of the total data (column) TCov.

   Then calculating dT. (dT = V' * TCov * V)

Turning the diagonal of matrix dT to a vector.

M

0.4579	0.1638	0.0867	0.0528	0.0527	0.0413	0.0214	0.0177	0.0185	0.0148

R

```
> dT
 [1] 0.45788128 0.16378062 0.08665568 0.05279657 0.05266202 0.04133698 0.02141124 0.01769672 0.01845588
[10] 0.01477735
```

5. Modifying the eigen spectrum values

M = index of the first number in d which <=0.005 (index starts from 1)

n = the original data column number (time series length)

Alpha = d(1) * d(M) * (M-1) / (d(1) - d(M))

Beta = (M * d(M) - d(1)) / (d(1) - d(M))

dMod = a new created array with length n

```
For i from 1 to n
    if i < M            # the reliable portion of the eigen spectrum
        dMod(i) = d(i)
    else
        dMod(i) = Alpha/(i+Beta)
            if dMod(i) > dT(i)
                dMod(i) = dT(i)
            end
    end
end
```

DD = the square root of dMod

M

Name	Value
Alpha	0.2891
Beta	-0.4175

dMod

0.4964	0.2599	0.1347	0.0925	0.0682	0.0614	0.0555	0.0409	0.0373	0.0148

DD

0.7045	0.5098	0.3670	0.3042	0.2612	0.2479	0.2356	0.2022	0.1932	0.1216

R

Values	
Alpha	0.28912563282355
Beta	-0.417512013265098

dMod

0.4963633	0.2598536	0.1346614	0.09252087	0.06820998	0.06144125	0.05551067	0.04088814	0.03731218	0.01477735

DD

0.7045305	0.5097583	0.3669625	0.3041724	0.2611704	0.2478735	0.235607	0.2022082	0.1931636	0.1215621

6. Starting from here is generating random vectors from the multivariate normal distribution. Even given a same seed number, the vector generated by Matlab and R are different. So for comparison, I used a same vector for both rest test.

   The vector is:

1.5118	-0.3054	0.5758	0.7383	0.4874	-0.8205	0.3295	1.5953	-0.8356	0.1836

   Before Euclidean distance checking, the generated vector need to be modified by

   x = vector*DD*V'+ Me

   DD and Me are the same in R and Matlab. Only V has minor different. I used same V for comparison.

   x:

   M

0.3935	0.0501	0.1194	0.0771	0.1963	0.3338	-0.0592	0.5932	0.8419	0.8568

   R

0.393485	0.05007858	0.1194362	0.07714205	0.196272	0.3337652	-0.05918221	0.5932051	0.8418713	0.8568402

7. The Euclidean distance between x and positive data:

M

First 10 elements from total 600:

1.6736	1.3246	1.9920	1.8604	2.0885	1.2420	1.0674	1.4747	2.1704	1.9786

Sum & Mean:

```
>> temp = sum(PDist);
>> temp

temp =

  955.6397

>> temp = temp/600

temp =

    1.5927
```

R

First 10 elements from total 600:

1.67361	1.324554	1.991967	1.860365	2.088549	1.242046	1.067399	1.474712	2.170398	1.97859

Sum & Mean:

```
> temp = sum(PDist)
> temp
[1] 955.6397
> temp/600
[1] 1.592733
```

8. Index & value of the smallest element in the Euclidean distance between x and negative data

M

```
>> [tmp ind] = min(NDist);
>> tmp

tmp =

    0.3418

>> ind

ind =

       27426
```

R

```
> tmp <- min(NDist)
> tmp
[1] 0.341785
> ind <- which.min(NDist)
> ind
[1] 27426
```

ADASYN

9. Calculating the positive data's number of samples and time series length

M

```
>> NT

NT =

   600

>> NumAtt

NumAtt =

    10
```

R

```
> NT
[1] 600
> NumAtt
[1] 10
```

10. Calculating the number of samples should be generated upon each positive sample. The number of all samples would be generated was set 10,000 in this testing experiment.

M

Samples:

95
15
18
12
18
18
15
19
18
1

Sum:

```
>> sum(No)

ans =

       10000
```

R

Samples:

95
15
18
12
18
18
15
19
18
1

Sum:

```
> sum(No)
[1] 10000
```

11. Finding the k indices corresponding to the closest indices. K default set to 5.

For the first positive data sample:

M

128	243	87	221	408

R

128	243	87	221	408

12. Generating data using a same random generated order list rn.

M (1:10 rows, 1:10 cols)

0.7215	0.7126	0.7342	0.7102	0.6910	0.7024	0.7158	0.7248	0.6664	0.6945
0.7342	0.8027	0.7612	0.7330	0.7579	0.7523	0.7359	0.7631	0.7834	0.8121
0.7196	0.7890	0.7795	0.7484	0.7489	0.7399	0.7493	0.7889	0.7790	0.7878
0.7460	0.7816	0.7890	0.7366	0.7686	0.7372	0.7775	0.7855	0.7865	0.7836
0.7241	0.7831	0.7930	0.7796	0.7811	0.7688	0.7759	0.7786	0.7802	0.7821
0.7325	0.7812	0.7762	0.7652	0.7642	0.7647	0.7700	0.7784	0.7748	0.7783
0.7376	0.7531	0.7599	0.7549	0.7496	0.7567	0.7564	0.7557	0.7573	0.7567
0.7778	0.7136	0.7112	0.7567	0.7657	0.7664	0.7622	0.7413	0.7372	0.7637
0.8009	0.7595	0.7454	0.7576	0.7710	0.7734	0.7711	0.7674	0.7507	0.7378
0.7814	0.7497	0.7761	0.7924	0.8453	0.7675	0.8614	0.7475	0.7667	0.7590

R(1:10 rows, 1:10 cols)

0.7215364	0.7126133	0.7342486	0.7102273	0.6909929	0.7024262	0.7158408	0.7247686	0.6663753	0.6945394
0.7341552	0.8026660	0.7612480	0.7330197	0.7579490	0.7523210	0.7359235	0.7631486	0.7833597	0.8120859
0.7195918	0.7890007	0.7795359	0.7483930	0.7489327	0.7399240	0.7493385	0.7888918	0.7789952	0.7877711
0.7459861	0.7815557	0.7890041	0.7366452	0.7685508	0.7371588	0.7775313	0.7854667	0.7865363	0.7836255
0.7241203	0.7831127	0.7929627	0.7796013	0.7810818	0.7688142	0.7759419	0.7786340	0.7801665	0.7821185
0.7324621	0.7812004	0.7762187	0.7652379	0.7642392	0.7647131	0.7700056	0.7783545	0.7747617	0.7783241
0.7376493	0.7530557	0.7598970	0.7548852	0.7495982	0.7567323	0.7564003	0.7557472	0.7572656	0.7566980
0.7777677	0.7136331	0.7111698	0.7566517	0.7656592	0.7663729	0.7622274	0.7413278	0.7372196	0.7636958
0.8008719	0.7594832	0.7453702	0.7575589	0.7709697	0.7734069	0.7711179	0.7674259	0.7507091	0.7377854
0.7813701	0.7497223	0.7761397	0.7924465	0.8452980	0.7675219	0.8614060	0.7474998	0.7666955	0.7590476