```
// T43mixDesNor.sce
// MIXTURE ESTIMATION (descriptive, normal)
// - static normal componens
// - two-dimensional variable
// Experiments
// - change simulated expectations thS
// - change initial expectations thE (through the statistics S, ka)
// ---------------------------------------------------------------------
exec("ScIntro.sce",-1),
getd(), mode(0)

nd=500;                                                          // 1
// PARAMETERS                                                    // 2
c(1).thS=[1 2];                        // simulated expectations  // 3
c(2).thS=[8 2];                        //   second component      // 4
c(3).thS=[7 9];                        //   third component       // 5
c(4).thS=[2 9];                        //   fourth component      // 6
sd=[1.5 .8 1.2 .9]*1;                  // standard deviations     // 7
nc=length(sd);                         // number of components    // 8
for j=1:nc                                                       // 9
  c(j).sdS=sd(j)*eye(2,2);             // component covariances   // 10
end                                                              // 11
alS=[.1 .2 .4 .3];                     // parameters of pointer model // 12
```

```
                                                                    // 13
// SIMULATION                                                       // 14
for t=1:nd                                                          // 15
  jS=sampCat(alS);                        // pointer value          // 16
  cS(t)=jS;                               // stor pointer value     // 17
  y(t,:)=c(jS).thS+randn(1,2)*c(jS).sdS; // output                 // 18
end                                                                 // 19
                                                                    // 20
// ESTIMATION                                                       // 21
// initialization                                                   // 22
c(1).th=[0 4];                          // initial parameters      // 23
c(2).th=[9 0];                          //  second component       // 24
c(3).th=[9 8];                          //  third component        // 25
c(4).th=[0 7];                          //  fourth component       // 26
ka=[1 1 1 1];                           // initial counter         // 27
for j=1:nc                                                          // 28
  c(j).V=eye(3,3);                      // initial inf. matrix     // 29
  c(j).V(3,1:2)=c(j).th';                                           // 30
  c(j).V(1:2,3)=c(j).th;                                            // 31
  c(j).thE=v2thN(c(j).V/ka(j),2);       //initial point estimates  // 32
end                                                                 // 33
                                                                    // 34
// time loop of estimation                                          // 35
```

```
for t=1:nd                                                              // 36
  for j=1:nc                                                            // 37
    [nill,q(j)]=GaussN(y(t,:),c(j).thE,.1*eye(2,2));// log-proximity    // 38
  end                                                                   // 39
  qn=exp(q-max(q));                     // normalized proximity         // 40
  w=qn/sum(qn);                         // weights                      // 41
  wt(:,t)=w;                            // remember weights             // 42
  for j=1:nc                                                            // 43
    Ps=[y(t,:) 1];                      // extended regression vector   // 44
    c(j).V=c(j).V+w(j)*Ps'*Ps;          // update of inf. matrix        // 45
    ka(j)=ka(j)+w(j);                   // update of counter            // 46
    c(j).thE=v2thN(c(j).V/ka(j),2);     // point estimates              // 47
    c(j).th=[c(j).th; c(j).thE];        // remember point estimates     // 48
  end                                                                   // 49
end                                                                     // 60
                                                                        // 61
// RESULTS                                                              // 62
tx=['b';'r';'g';'k'];                                                   // 63
set(scf(1),'position',[600 100 600 400])   // evolution of par. est.    // 64
for j=1:nc                                                              // 65
  plot(c(j).th(:,1),c(j).th(:,2),'.:'+tx(j))                            // 66
  plot(c(j).th(1,1),c(j).th(1,2),'o'+tx(j),'markersize',8)              // 67
  plot(c(j).th(nd,1),c(j).th(nd,2),'x'+tx(j),'markersize',12)           // 68
```

```
end                                                                    // 69
title 'Evolution of the estimated parameters'                          // 70
mi=min([c(1).th;c(2).th;c(3).th;c(4).th],'r');                          // 71
ma=max([c(1).th;c(2).th;c(3).th;c(4).th],'r');                          // 72
set(gca(),'data_bounds',[mi(1)-1 ma(1)+1 mi(2)-1 ma(2)+1]);             // 73
for j=1:nc                                                             // 74
  plot(c(j).thS(1),c(j).thS(2),'p'+tx(j),'markersize',16);              // 75
end                                                                    // 76
                                                                       // 77
set(scf(2),'position',[1200 100 600 400])  // plot data clusters        // 78
plot(y(:,1),y(:,2),'.')                                                // 79
title 'Data'                                                           // 80
                                                                       // 81
disp 'The final parmeter estimates are'                                // 82
for j=1:nc                                                             // 83
  disp(c(j).th($,:))                                                   // 84
end                                                                    // 85
                                                                       // 86
[nill,cp]=max(wt,'r');                    // accuracy of classification // 87
disp 'Accuracy of classification'                                      // 88
ACC=acc(cS,cp)                                                         // 89
```

Description of the program

This program is based on the previous scalar one. It deals with multivariate data. Th consequence of it is, that the parameters are matrices instead of numbers as it was in the scalar case. This makes the program a bit more complex and unclear.

The structure of the program follows its scalar version `T42mixDesNor1.sce`.

- Rows 1–12 define parameters of the models for simulation.
- Rows 14–19 perform simulation: first the pointer value and then the output.
- Rows 22–33 construct initialization for the estimation.
- Rows 35–60 represent a dynamic loop in which the pointer values and parameters are estimated.
  - Rows 37–39 compute the component proximities.
  - Rows 40–42 construct the component weights
  - Rows 43–49 perform update of the statistics and construct new parameter point estimates.