hDFPmcmc Demo

Scott

2024-02-01

In this demo we show an example of how to use the package to simulate data, perform model fitting and do inference

```
# We first load the package
library(hDFPmcmc)

# The function to simulate data is provided in the package
data = gendata2(myseed = 10,N = 20)
```

By default this function uses the following setting for data simulation $\mu_\eta = 0, N = 50, J = 5, Q = 5$, and we set the number of covariates in modeling main effect and transition to 3.

```
N = 20 # Using 20 to save time
J = 5 # number of periods
dT = 5 # col dimension of spline basis functions
dZ = 3 # col dimension of global effect matrices
dX = 3 # dimension of reallocation coefficients
```

Before we run MCMC, we need to initialize some parameters

```
C_ = matrix(0, nrow = N, ncol = J) # cluster indicator starts at 0
Beta_ = array(0, dim = c(N,dT,J))
Lambda2_ = array(1, dim = c(N,dT,J))
Tau2_ = matrix(1, N, J)
Nulam_ = array(1, dim = c(N,dT,J))
Nutau_ = matrix(1, N, J)
Gamma_ = matrix(0, N, J)
Gamma = data$Gamma
Eta_ = matrix(0, dX, J)
Theta_ = matrix(0, dZ, J)
D_ = C_
muEta = rep(0, dX)
muEta_ = muEta*0
sigEta1 = 0.01*diag(dX)
sigEta2 = 5*diag(dX)
sigEta3 = 10*diag(dX)
sigTheta = diag(dZ)
```

And we are ready to run the MCMC

```
tick = Sys.time()
set.seed(250)
niter = 5000
hDFPoutput = hDFPmcmc(
  iterations = niter,
```

```
  thin = 25,
  data$YList,
  data$TList,
  data$ZList,
  data$X,
  Beta_,
  Lambda2_,
  Nulam_,
  Gamma_,
  C_,
  Tau2_,
  Nutau_,
  Eta_,
  Theta_,
  data$idstart,
  data$idend,
  SigEta = sigEta2,
  SigTheta = sigTheta,
  D_,
  MuEta = muEta_,
  a_glo = 1,
  a_loc = 0.1
)
hdfp_time = Sys.time()-tick # Time difference of 2.401057 mins
```

Here we provide two example inferences for cluster estimation and trajectory estimation within each cluster

```
# Some example inferences
output = hDFPoutput

# get estimated number of cluster (using SALSO package)
library(salso)

dd = output[[10]]
burn = 101:200 # treat
dest = salso(dd[burn,])
dest = matrix(dest,ncol=J) # This is the estimated clusters for all participants over all 5 periods
VI.lb(as.numeric(data$D),as.numeric(dest)) # This measures the different between our estimation and the
```

Cluster estimation

```
## [1] -4.796163e-16
```

```
print(dest) # estimated cluster
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    2    2    2
## [2,]    1    2    2    2    2
## [3,]    2    2    2    2    2
## [4,]    2    2    2    2    2
## [5,]    2    2    2    2    2
## [6,]    1    2    2    2    2
## [7,]    2    2    2    1    2
## [8,]    1    2    2    2    2
```

```
##  [9,]    1    2    2    2    2
## [10,]    2    2    2    2    2
## [11,]    1    2    2    2    2
## [12,]    1    1    2    2    2
## [13,]    2    2    2    2    2
## [14,]    2    2    2    2    2
## [15,]    2    2    2    2    2
## [16,]    1    2    2    2    2
## [17,]    1    2    2    2    1
## [18,]    2    2    2    2    2
## [19,]    2    2    2    2    2
## [20,]    1    2    2    2    2
```

```r
print(data$D) # true cluster
```

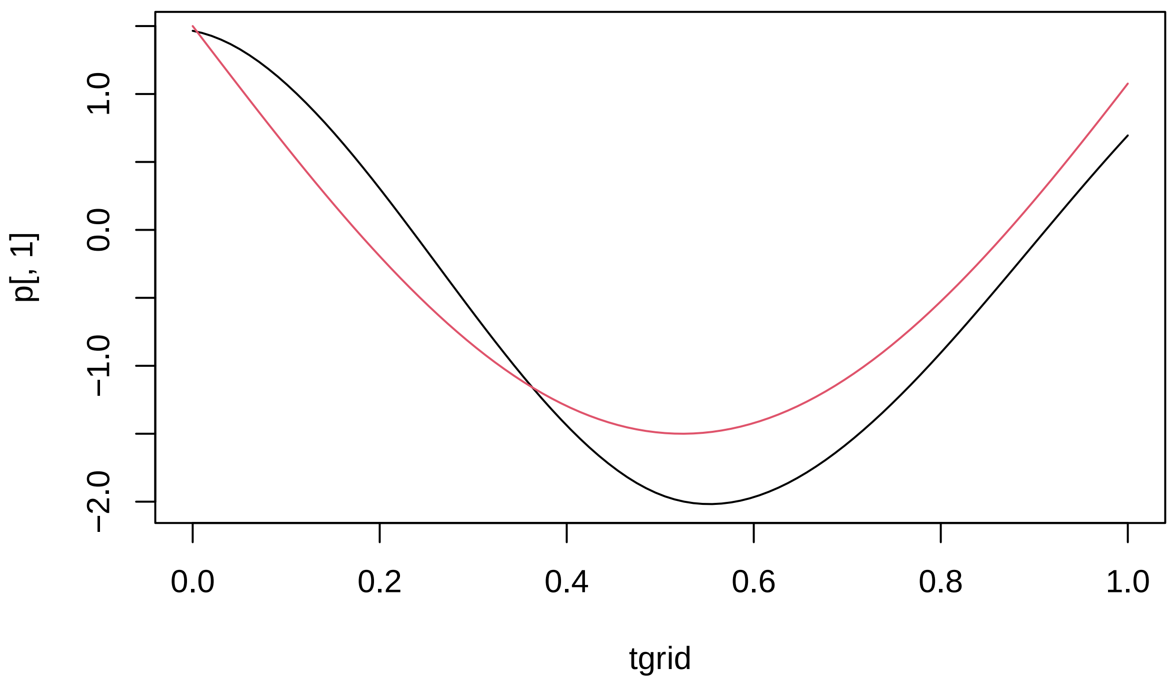
```
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    2    1    1    1    1
##  [2,]    2    1    1    1    1
##  [3,]    1    1    1    1    1
##  [4,]    1    1    1    1    1
##  [5,]    1    1    1    1    1
##  [6,]    2    1    1    1    1
##  [7,]    1    1    1    2    1
##  [8,]    2    1    1    1    1
##  [9,]    2    1    1    1    1
## [10,]    1    1    1    1    1
## [11,]    2    1    1    1    1
## [12,]    2    2    1    1    1
## [13,]    1    1    1    1    1
## [14,]    1    1    1    1    1
## [15,]    1    1    1    1    1
## [16,]    2    1    1    1    1
## [17,]    2    1    1    1    2
## [18,]    1    1    1    1    1
## [19,]    1    1    1    1    1
## [20,]    2    1    1    1    1
```

```r
# get estimated posterior parameter (ignoring main effect since estimated eta is close to 0)
bb = output[[1]]
barray = array(dim=c(length(bb), dim(bb[[1]]))) # convert beta posterior from list to array for faster
for(i in 1:length(bb)){
  barray[i,,,] = bb[[i]]
}
barray = barray[burn,,,]
bmean = apply(barray,c(2,3,4),mean)

duniq = unique(as.numeric(dest)) # get unique clusters
nsubj = dim(bb[[1]])[1]

blist = NULL # for each cluster, get an estimated spline coefficient vector
for(d in duniq){
  bout = data.frame() # get beta estimates
  for(j in 1:5){
```

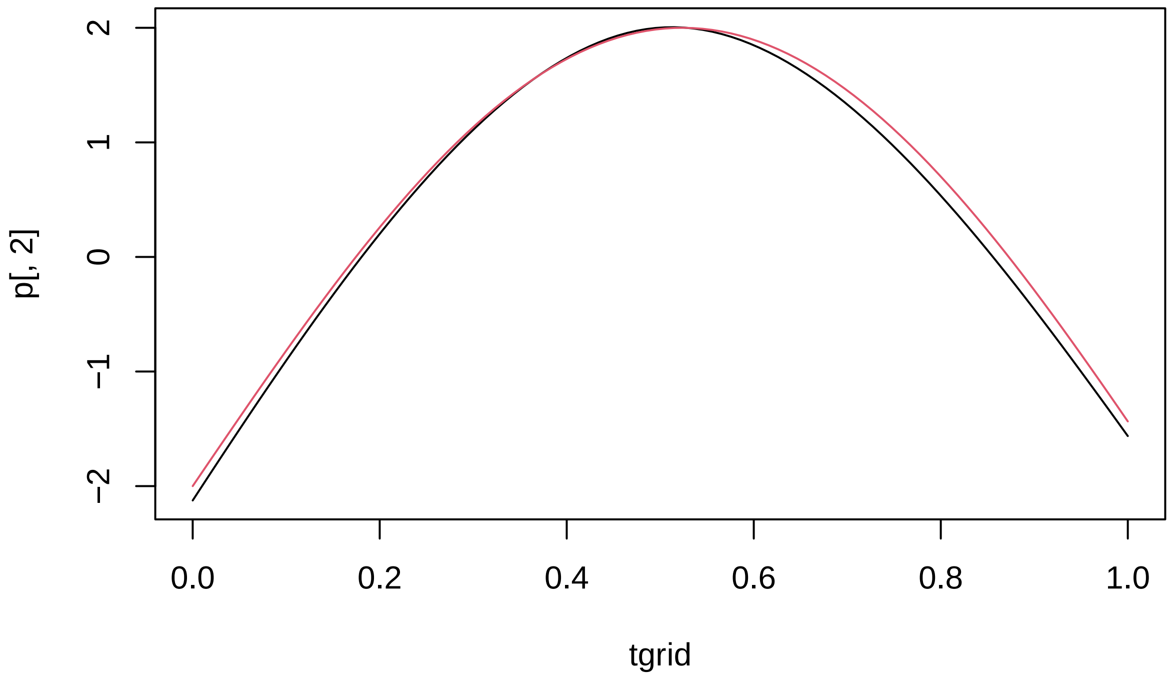
```
    subj = which(dest[,j]==d)
    for(i in subj){
      bout = rbind(bout, bmean[i,,j])
    }
  }
  blist = rbind(blist, as.vector(colMeans(bout)))
}
```

Trajectory estimation With spline coefficients estimated, we are ready to visualize the trajectory for each cluster

```
tgrid = seq(0,1,length.out=100) # simulate a grid of time from 0 to 1

library(splines)
Tj = bs(tgrid, df = dT, intercept = TRUE) # generate cubic spline

p = Tj%*%t(blist) # this would be our estimated log-odds

# for comparison, here are the true log-odds
f11 = function(x){4*sin(3*x)-2}
f12 = function(x){-3*sin(3*x)+1.5}

plot(tgrid, p[,1], type='l') # estimated trajectory for group 1
lines(tgrid, f12(tgrid), type='l', col=2) # truth for group 1
```




```
plot(tgrid, p[,2], type='l') # estimated trajectory for group 2
lines(tgrid, f11(tgrid), type='l', col=2) # truth for group 2
```


p[, 2]
2
1
0
−1
−2
0.0
0.2
0.4
0.6
0.8
1.0
tgrid