```
##ジョイントセグメンテーションモデル2##

##データセット
#顧客数
Y<- read.table("C:/Documents and Settings/Administrator/顧客数.csv",header=TRUE,sep=",")
#購買金額
Y2<- read.table("C:/Documents and Settings/Administrator/購買金額.csv",header=TRUE,sep=",")
#複数店舗顧客数データ
XX<- read.table("C:/Documents and Settings/Administrator/複数店舗.csv",header=F,sep=",")

#渋谷data
Yr1=Y[,13:20]
Yr1=cbind(Yr1[,1:2],Yr1[,5:6])
Yd1=Y[,6:12]

#変数添え字の数
G=length(row.names(Yr1))    #エリア数
D=length(names(Yr1))        #デモグラフィック分類数
I=length(names(Yd))      #商品分類数
C=length(names(XX))      #複数店舗顧客数データ変数の数

X=matrix(0,G,C)
for(g in 1:G){
    for(d in 1:C){
        X[g,d]=XX[g,d]
    }
}
X=X*10/matrix(apply(X,1,sum),G,C)


##ジョイント・セグメンテーション・モデルの推定
oll <- function(x){
    bd1 <- x[[1]]
    bs1 <- x[[2]]
    bd2 <- x[[3]]
    bs2 <- x[[4]]
    br  <- x[[5]]
    r   <- x[[6]]

    lr  <- matrix(0,G,b1)
    lh  <- matrix(0,G,b1)
    ld  <- matrix(0,G,b2)
    zz1 <- matrix(0,G,b1)
    zz2 <- matrix(0,G,b2)
    zz  <- matrix(0,G,b1*b2)
    z0  <- matrix(0,G,b1*b2)

    #第1軸尤度f(x)
    for(k in 1:b1){

        lr[,k]=apply(matrix(1/sqrt(2*pi*bs1[k,]),G,D,byrow=T)*exp(((Yr1-matrix(bd1[k,],G,D,byrow=T
    ))^2)*matrix((-1/(2*bs1[k,])),G,D,byrow=T)),1,prod)
```

```
	}

	for(k in 1:b1){
		lh[,k] <- matrix(factorial(apply(X, 1, sum)), G, 1) * matrix(apply((matrix(br[k,], G, C, byrow = T)^X)/factorial(X), 1, prod), G, 1)
	}

	#第2軸尤度f(y)
	for(j in 1:b2){

		ld[,j]=apply(matrix(1/sqrt(2*pi*bs2[j,]),G,I,byrow=T)*exp(((Yd1-matrix(bd2[j,],G,I,byrow=T))^2)*matrix((-1/(2*bs2[j,])),G,I,byrow=T)),1,prod)
	}

	#尤度計算
	for(i in 1:b1){
		for(j in 1:b2){
			k<-(i-1)*b2+j
			z0[,k]<-r[i,j] * lh[,i] * lr[,i] * ld[,j]   #対数尤度lrは尤度に変換している
		}
	}
	#全体の尤度計算
	LLo<-sum(log(apply(z0,1,sum)))
	#LLoは全体尤度L(x,y)
	return(LLo)
}

calz <-function(x){
	bd1 <- x[[1]]
	bs1 <- x[[2]]
	bd2 <- x[[3]]
	bs2 <- x[[4]]
	br  <- x[[5]]
	r   <- x[[6]]

	lr  <- matrix(0,G,b1)
	lh  <- matrix(0,G,b1)
	ld  <- matrix(0,G,b2)
	zz1 <- matrix(0,G,b1)
	zz2 <- matrix(0,G,b2)
	zz  <- matrix(0,G,b1*b2)
	z0  <- matrix(0,G,b1*b2)

	#第1軸尤度f(x)
	for(k in 1:b1){

		lr[,k]=apply(matrix(1/sqrt(2*pi*bs1[k,]),G,D,byrow=T)*exp(((Yr1-matrix(bd1[k,],G,D,byrow=T))^2)*matrix((-1/(2*bs1[k,])),G,D,byrow=T)),1,prod)
	}

	for(k in 1:b1){
		lh[,k] <- matrix(factorial(apply(X, 1, sum)), G, 1) * matrix(apply((matrix(br[k,], G, C, byrow
```

```
    = T)^X)/factorial(X), 1, prod), G, 1)
    }

    #第2軸尤度f(y)
    for(j in 1:b2){

        ld[,j]=apply(matrix(1/sqrt(2*pi*bs2[j,]),G,l,byrow=T)*exp(((Yd1-matrix(bd2[j,],G,l,byrow=T
    ))^2)*matrix((-1/(2*bs2[j,])),G,l,byrow=T)),1,prod)
    }

    #尤度計算
    for(i in 1:b1){
        for(j in 1:b2){
            k<-(i-1)*b2+j
            z0[,k]<-r[i,j] * lh[,i] * lr[,i] * ld[,j]   #対数尤度lrは尤度に変換している
        }
    }
    #z1はφ(jkg)
    z1 <- z0/matrix(apply(z0,1,sum),G,b1*b2)    #z1:zの期待値計算
    return(z1)
}

likelyhood = function(X,zr,fYr,br,FF){
    #集合の設定
    G <- Set()
    D <- Set()
    B <- Set()
    K <- Set()

    #要素の設定
    g <- Element(set = G)
    d <- Element(set = D)
    b <- Element(set = B)
    k <- Element(set = K)

    #Variable(パラメータ)の設定
    BR <- Variable(index = dprod(b, d))

    #Parameter（変数）の設定
    YR <- Parameter(X, index = dprod(g, d))
    FYR <- Parameter(fYr, index = dprod(g, d))
    Z <- Parameter(zr, index = dprod(g, b))
    Fbr <- Parameter(br, index = dprod(b, d))
    FF<- Parameter(FF, index = dprod(g,k))

    #Expressionの設定
    ZZ <- Expression(index = dprod(g, b))
    LR <- Expression(index = dprod(g, b))

    #式
    BR[b,d] ~ Fbr[b,d]
    LR[g, b] ~ FF[g,1]*Prod((BR[b, d]^YR[g, d])/FYR[g, d], d)
```

```
    ZZ[g, b] ~Z[g, b] * log(LR[g, b])

    #Objectiveの設定
    like = Objective(type = "maximize")

    #尤度関数
    like ~ Sum(Sum(ZZ[g, b], b), g)

    #制約式
    Sum(BR[b, d], d) == 1.
    BR[b, d] <= 1.
    BR[b, d] >= 0.
}


##最小のBICを求めるための変数の用意
miter=0
BIC<-rep(0,miter)
listbd1<-list(0)
listbs1<-list(0)
listbd2<-list(0)
listbs2<-list(0)
listbr<-list(0)
listr<-list(0)
listz<-list(0)
iter <- 1
miter=10
#今回は7×3モデル
b1=7
b2=3
module(nuopt,unload=T)
##推定部分
while(iter<=miter){
    br=matrix(0,b1,C)
    bd1=matrix(0,b1,D)
    bs1=matrix(0,b1,D)
    bd2=matrix(0,b2,I)
    bs2=matrix(0,b2,I)

    #br初期値設定
    BBR=matrix(runif(C*b1),b1,C)    #brの準備
    br<- BBR/apply(BBR,1,sum)   #D×b   #θ

    #bd1・bs1初期値設定
    for(i in 1:D){
        bd1[,i]=matrix(mean(Yr1[,i]),b1,1,byrow=T)
    }
    bs1=matrix(0,b1,D)
    for(j in 1:b1){
        for(d in 1:D){
            bs1[j,d]=var(Yr1[,d])
        }
```

```
}

#bd2・bd2初期値設定
for(i in 1:I){
    bd2[,i]=matrix(mean(Yd1[,i]),b2,1,byrow=T)
}
bs2=matrix(0,b2,I)
for(k in 1:b2){
    for(i in 1:I){
        bs2[k,i]=var(Yd1[,i])
    }
}

#乱数を用いて，各セグメントの重み付け
J=matrix(runif(b1,0.8,1.2),1)
for(j in 1:b1){
    bd1[j,]<-bd1[j,]*J[j]
    bs1[j,]<-bs1[j,]*J[j]
}
K=matrix(runif(b2,0.8,1.2),1)
for(k in 1:b2){
    bd2[k,]<-bd2[k,]*K[k]
    bs2[k,]<-bs2[k,]*K[k]
}

#所属確率φjkの初期値
r<- matrix(1/(b1*b2),b1,b2)
LL1<- oll(list(bd1,bs1,bd2,bs2,br,r))   #完全情報による対数尤度
z<- calz(list(bd1,bs1,bd2,bs2,br,r))    #潜在変数z
diff<- 100  #EMアルゴリズムの尤度の差
print(LL1)  #   尤度を表示

#EMアルゴリズムによる推定
while(diff>0.001){  #尤度の差が十分に小さくなるまで繰り返す
    zr=matrix(0,G,b1)   #第1セグメント基盤のz
    zd=matrix(0,G,b2)   #第2セグメント基盤のz
    zr[,1]=z[,1]+z[,2]+z[,3]
    zr[,2]=z[,4]+z[,5]+z[,6]
    zr[,3]=z[,7]+z[,8]+z[,9]
    zr[,4]=z[,10]+z[,11]+z[,12]
    zr[,5]=z[,13]+z[,14]+z[,15]
    zr[,6]=z[,16]+z[,17]+z[,18]
    zr[,7]=z[,19]+z[,20]+z[,21]
    zd[,1]=z[,1]+z[,4]+z[,7]+z[,10]+z[,13]+z[,16]+z[,19]
    zd[,2]=z[,2]+z[,5]+z[,8]+z[,11]+z[,14]+z[,17]+z[,20]
    zd[,3]=z[,3]+z[,6]+z[,9]+z[,12]+z[,15]+z[,18]+z[,21]

    #bd1・bs1更新
    for(s in 1:b1){
        for(i in 1:D){
            bd1[s,i]=sum(zr[,s]*Yr1[,i])/sum(zr[,s])
            bs1[s,i]=sum(zr[,s]*(Yr1[,i]-bd1[s,i])^2)/sum(zr[,s])
```

```
            }
        }
        #bd2(β推定)
        for(s in 1:b2){
            for(i in 1:I){
                bd2[s,i]=sum(zd[,s]*Yd1[,i])/sum(zd[,s])
            }
        }

        #bs2(σ推定)
        for(s in 1:b2){
            for(i in 1:I){
                bs2[s,i]=sum(zd[,s]*(Yd1[,i]-bd2[s,i])^2)/sum(zd[,s])
            }
        }
        fYr <-factorial(X)#NUOPT用
        FF <-matrix(factorial(apply(X,1,sum)),G,1)
        module(nuopt)
        rslt.problem = System(model=likelyhood,X,zr,fYr,br,FF)
        rslt.solution = solve(rslt.problem)
        br<-matrix(rslt.solution$variables$BR$current,b1,C)
        module(nuopt,unload=T)
        ##rの推定　（φの計算）
        r<-matrix(apply(z,2,sum)/G,b1,b2,byrow=T)  #φ推定
        BICC=-2*LL1+log(G)*((2*(D+C)*b1)+(2*I*b2)+(b1*b2-1))
        print(r)
        print("BIC")
        print(BICC)
        z<-calz(list(bd1,bs1,bd2,bs2,br,r))
        LL<-oll(list(bd1,bs1,bd2,bs2,br,r))
        diff<-LL-LL1
        print("diff")
        print(diff)
        LL1<-LL
        }
    BIC[iter]<- -2*LL1+log(G)*((2*(D+C)*b1)+(2*I*b2)+(b1*b2-1))
    listbd1[[iter]]<-bd1
    listbs1[[iter]]<-bs1
    listbr[[iter]]<-br
    listr[[iter]]<-r
    listz[[iter]]<-z
    iter<-iter+1
    print(iter)
}


##最小のBICを求めパラメータを表示
itern<-which.min(BIC)
BIC[itern]
listbd1[[itern]]
listbs1[[itern]]
listbd2[[itern]]
```

```
listbs2[[itern]]
listbr[[itern]]
listr[[itern]]
listz[[itern]]
```