形を「分布の分布」として捉える

ryamada

2018年10月23日

```
## Warning: package 'rgl' was built under R version 3.4.3
```

```
## Warning: package 'RFOC' was built under R version 3.4.4
```

```
## Warning: package 'igraph' was built under R version 3.4.4
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
## Warning: package 'tagcloud' was built under R version 3.4.4
```

```
## Loading required package: Rcpp
```

```
## Warning: package 'Rcpp' was built under R version 3.4.3
```

```
## Warning: package 'e1071' was built under R version 3.4.4
```

```
## Warning: package 'sets' was built under R version 3.4.4
```

```
##
## Attaching package: 'sets'
```

```
## The following object is masked from 'package:igraph':
##
##     %>%
```

```
## The following object is masked from 'package:rgl':
##
##     %>%
```

```
## Loading required package: polynom
```

```
knit_hooks$set(webgl = hook_webgl)
```

```
my.cell.shape2 <- function(d,k,N){
  # 形の凹凸・複雑さをコントロールするパラメタ、n,k
  n <- d
  #k <- 8
  # メッシュのノード数をコントロールするパラメタ
  n.mesh <- N # 色々試すなら、32くらいにしておくのが無難。送ったhtmlファイルはn.mesh=64
  # 形を球面調和関数係数ベクトルで指定する
  A. <- matrix(runif(n^2), n, n)
  A.[1, 1] <- k
  B <- matrix(rnorm(n^2), n, n)
  # 閉曲面オブジェクトを作る
  xxx <- my.spherical.harm.mesh(A = A., B = B, n = n.mesh)

  #xxx$v <- xxx$v + rnorm(length(xxx$v))*r

  g <- graph.edgelist(xxx$edge,directed=FALSE)
  vname <- paste("",1:length(V(g)),sep="")
  g <- set_vertex_attr(g,"name",value=vname)
  # edge lengths
  w <- sqrt(apply((xxx$v[xxx$edge[,1],]-xxx$v[xxx$edge[,2],])^2,1,sum))

  # thetas,phis
  tmp <- my_sphere_tri_mesh(n.mesh)
  xyz <- tmp$xyz
  tosp <- TOSPHERE(xyz[, 1], xyz[, 2], xyz[, 3])
  tp <- cbind(tosp[[1]]/360 * 2 * pi, tosp[[2]]/360 * 2 * pi)

  return(list(X = xxx$v,X.sp=xyz, E = xxx$edge, tri=xxx$f, angles = cbind(tp[,2],tp[,1]),g=g,w=w ))
}
my.cell.shape3 <- function(A,B,N){
  # 形の凹凸・複雑さをコントロールするパラメタ、n,k
  n <- length(A[,1])
  #k <- 8
  # メッシュのノード数をコントロールするパラメタ
  n.mesh <- N # 色々試すなら、32くらいにしておくのが無難。送ったhtmlファイルはn.mesh=64
  # 形を球面調和関数係数ベクトルで指定する
  #A. <- matrix(runif(n^2), n, n)
  #A.[1, 1] <- k
  A. <- A
  #B <- matrix(rnorm(n^2), n, n)
  # 閉曲面オブジェクトを作る
  xxx <- my.spherical.harm.mesh(A = A., B = B, n = n.mesh)

  #xxx$v <- xxx$v + rnorm(length(xxx$v))*r

  g <- graph.edgelist(xxx$edge,directed=FALSE)
  vname <- paste("",1:length(V(g)),sep="")
  g <- set_vertex_attr(g,"name",value=vname)
  # edge lengths
  w <- sqrt(apply((xxx$v[xxx$edge[,1],]-xxx$v[xxx$edge[,2],])^2,1,sum))

  # thetas,phis
  tmp <- my_sphere_tri_mesh(n.mesh)
  xyz <- tmp$xyz
  tosp <- TOSPHERE(xyz[, 1], xyz[, 2], xyz[, 3])
  tp <- cbind(tosp[[1]]/360 * 2 * pi, tosp[[2]]/360 * 2 * pi)
```

```
  return(list(X = xxx$v,X.sp=xyz, E = xxx$edge, tri=xxx$f, angles = cbind(tp[,2],tp[,1]),g=g,w=w ))
}
my.plot.shape <- function(shape,add=FALSE){
  if(add){
    points3d(shape$X)
  }else{
    plot3d(shape$X)
  }

  segments3d(shape$X[c(t(shape$E)), ])
}



# 還り値
## X : 3D 座標
## X.sp : 球面上の座標
## E : エッジ
## tri : 三角形頂点トリオ
## angles : 球面上の角座標
## g,w : グラフオブジェクトとエッジの重み

my.3Dobj <- function(d=6,k=5,N=15){
  shape1 <- my.cell.shape2(d,k,N)
  # 3D 座標
  X <- shape1$X
  # その球面上座標(このままであと、均等になってしまっている)
  Xsp <- shape1$X.sp
  # 球面上の点配置を少しずらす
  Rot <- Random.Start(3)
  Mat <- diag(rep(1,3))
  Mat <- Mat + rnorm(9)*0.3
  Xsp[,1] <- Xsp[,1]+0.4
  Xsp <- Xsp %*% Mat
  Xsp <- Xsp/sqrt(apply(Xsp^2,1,sum))
  shape1$X.sp <- Xsp
  return(shape1)
}
my.3Dobj2 <- function(A,B,N=15){
  shape1 <- my.cell.shape3(A,B,N)
  # 3D 座標
  X <- shape1$X
  # その球面上座標(このままであと、均等になってしまっている)
  Xsp <- shape1$X.sp
  # 球面上の点配置を少しずらす
  Rot <- Random.Start(3)
  Mat <- diag(rep(1,3))
  Mat <- Mat + rnorm(9)*0.3
  Xsp[,1] <- Xsp[,1]+0.4
  Xsp <- Xsp %*% Mat
  Xsp <- Xsp/sqrt(apply(Xsp^2,1,sum))
  shape1$X.sp <- Xsp
  return(shape1)
}
```

「分布の分布」

閉曲面Sを考える。

ただし、閉曲面の面積は1に標準化されているものとする。

S上の１点vを取ると、S上の点uへの測地線距離$D(u|v)$が定まる。

S上のすべての点$u \in S$について$D(u|v)$を集めると、それはある集合をなす。

これを

$$F_S(v) = \{f_S(u|v) = D(u|v)|v \in S, \forall u \in S\}$$

と表すことにする。

閉曲面の面積が1なので、この集合は確率密度分布になっている。

$F_S(v)$ をS上のすべての点$v \in S$について集めると、それは、「分布の集合」をなす。

それを

$$G_S = \{g_S(v) = F_S(v)|\forall v \in S\} = \{\{D(u|v)|v \in S, \forall u \in S\}|\forall v \in S\}$$

とする。 面積が1の閉曲面全体に対応する集合であるので、これもまた、確率密度分布になっている。

「確率密度分布の確率密度分布」である。

分布を点と見ることが出来る：Okada’s 拡張指数型分布族表現

閉曲面の集合

$$\mathbf{S} = \{S_i\}$$

があるとする。

１つのの閉曲面S_iは、

$$F_{S_i}(v)$$

を持つ。

今、$\mathbf{S}$のすべてのS_iの$F_{S_i}(v)$をすべて合わせた集合

$$\mathbf{F_S} = \{F_{S_i}(v_i)|\forall v_i \in S_i, \forall S_i \in \mathbf{S}\}$$

を考える。

これは、確率密度分布集合なので、この分布集合の下で、拡大指数型分布族表現を与えれば、個々の分布$F_{S_i}(v_i)$は点としての座標を持つ。

これを、$\theta(v_i|S_i)$と書くことにする。

また、その全体$\mathbf{F_S}$はその点の広がりとしての分布

$$\Omega_{\mathbf{S}}$$

を持つ。

したがって、閉曲面S_iが持つ分布集合G_{S_i}は、$\Omega_{\mathbf{S}}$に広がる分布となる。

言い換えると、閉曲面集合の各要素である閉曲面は、$\Omega_{\mathbf{S}}$ 上の分布である。

この分布を紛れがないように

$$K_{S_i}^{\Omega \mathbf{S}}$$

と書くことにする。

閉曲面集合が分布の集合

$$\kappa_{\mathbf{S}} = \{K_{S_i}^{\Omega \mathbf{S}}\}$$

として表されたので、 今度は、$\kappa_{\mathbf{S}}$に対して、拡大指数型分布族表現を与えることで、$K_{S_i}^{\Omega \mathbf{S}}$に座標を与えることが出来る。

これを、

$$\sigma(S_i) \in \Sigma_{\mathbf{S}}$$

と書くことにする。

この結果、閉曲面集合$\mathbf{S}$を点の集合として表すことができる。

実際には、各$F_{S_i}(v_i)$と$F_{S_j}(v_j)$との内積をすべて計算して、それぞれの座標を決め、その後に、G_{S_i}とG_{S_j}との内積を計算しなおすのは、無駄なので、次のようにする。

$F_{S_i}(v_i)$と$F_{S_j}(v_j)$との内積 $q(i,j,v_i,v_j)$は それぞれに付与される座標$\theta_{i,v_i}, \theta_{j,v_j}$を使って

$$\log(q(i,j,v_i,v_j)) = 2 < \theta_{i,v_i}, \theta_{j,v_j} >$$

と表せる。

今、G_{S_1}とG_{S_2}をの内積を考える。

$G_{S_1} = \{F_{S_1}(v) | v \in S_1\}$,$G_{S_2} = \{F_{S_2}(u) | u \in S_2\}$ であるので、この内積は、ガウシアンカーネルを使って推定するとすると

$$Q(G_{S_1}, G_{S_2}) = \frac{1}{|G_{S_1}||G_{S_2}|} \sum_{i=1}^{|G_{S_1}|} \sum_{j=1}^{|G_{S_2}|} e^{-\frac{1}{2s^2}(|\theta_{S_1,v_i} - \theta_{S_2,u_j}|^2)}$$

$$= \frac{1}{|G_{S_1}||G_{S_2}|} \sum_{i=1}^{|G_{S_1}|} \sum_{j=1}^{|G_{S_2}|} e^{-\frac{1}{2s^2}(|\theta_{S_1,v_i}|^2 + |\theta_{S_2,u_j}|^2 - 2(\theta_{S_1,v_i}, \theta_{S_2,u_j}))}$$

ただし $(\theta_{S_1,v_i}, \theta_{S_2,u_j})$は2ベクトルの内積であり、 $|\theta_{S_1,v_i}|^2$も内積であるから、$\theta_{S_i,v}$を計算せずとも、ペアワイズ内積のみを求めればよいことがわかる。

離散版

- 閉曲面は平面グラフである
- 測地線はグラフ最短パスである
- $F_S(v)$には次のように対応を取る。グラフのノードvから、(有限個の)全頂点への最短距離がある。各頂点には、「支配面積」があるので、その支配面積の割合に応じて、「最短距離の重み付き確率質量分布」が$F_S(v)$に対応する。
- ちなみに、各頂点の支配領域は、頂点を含む三角形のそれぞれに対して、頂点と頂点を挟む2辺とそれぞれの垂直二等分線とで囲まれた領域の面積を求め、それをすべての周囲三角形に関して足し合わせたものとすることとする。三角形の外心の定義を思い出すこと。

- G_Sには次のように対応を取る。平面グラフの全領域は、有限個の頂点によって小支配領域に区分されている。頂点ごとに$F_S(v)$が離散的に定まっているので、G_Sには、その頂点支配領域で$F_S(v)$を重みづけした分布を対応させる。

計算の準備

三角形の扱い

三角形の外心的部分面積

```
# triは各行に3頂点座標v1,v2,v3が入っており
# v1,v2,v3の支配領域を返すこととする
my.part.tri <- function(tri){
  L1 <- sqrt(sum((tri[1,]-tri[2,])^2))
  L2 <- sqrt(sum((tri[2,]-tri[3,])^2))
  L3 <- sqrt(sum((tri[3,]-tri[1,])^2))
  L <- c(L1,L2,L3)
  # 外接円半径
  R <- prod(L)/(sqrt(sum(L))*sqrt(sum(L)-2*L[1])*sqrt(sum(L)-2*L[2])*sqrt(sum(L)-2*L[3]))

  h1 <- sqrt(R^2-(1/2*L1)^2)
  h2 <- sqrt(R^2-(1/2*L2)^2)
  h3 <- sqrt(R^2-(1/2*L3)^2)
  a1 <- 1/2 * 1/2*L1*h1 + 1/2*1/2*L3*h3
  a2 <- 1/2 * 1/2*L1*h1 + 1/2*1/2*L2*h2
  a3 <- 1/2 * 1/2*L2*h2 + 1/2*1/2*L3*h3
  return(c(a1,a2,a3))
}
```

平面グラフの扱い

頂点の支配領域面積・割合の算出

```
# tris: 三角形の３頂点IDの行列
# x : verticesの3次元頂点座標
# st : TRUEなら割合に、FALSEなら実面積値を返す
my.vertex.area <- function(tris,x,st=TRUE){
  as <- matrix(0,length(tris[,1]),3)
  for(i in 1:length(as[,1])){
    tmptris <- x[tris[i,],]
    as[i,] <- my.part.tri(tmptris)
  }
  ret <- rep(0,length(x[,1]))
  for(i in 1:length(ret)){
    ret[i] <- sum(as[which(tris==i)])
  }
  if(st){
    ret <- ret/sum(ret)
  }
  return(ret)
}
```

$F_S(v)$の計算とその格納：重み付き離散的確率分布の扱い

グラフ最短距離行列dと 各頂点の支配領域aがあれば、

第i番ノードの$F_S(vi)$は、d[i,]にaの重みベクトルを持たせた確率質量分布となる。

そして、それをすべてのノードについて、ノード重みで考えるG_Sは

dのすべての要素について、その重みを $at(a)$なる、ノード数 x ノード数の正方行列で重みづけしたものとなる。

```
# g はグラフオブジェクト
# w はエッジの長さ
# tri は三角形の頂点ＩＤの３列行列
# x は頂点の座標
# 返り値のdは、頂点間距離行列
# aは頂点の支配領域面積
my.pr.v <- function(g,w,tris,x){
  a <- my.vertex.area(tris,x)
  d <- distances(g,weights=w)
  return(list(d=d,a=a))
}
```

重み付き確率質量分布間の内積

ガウシアンカーネルを用いて、スムージングをした上で、内積をとることにする。

```
# d1,d2は頂点間距離
# a1,a2は、頂点の支配面積
# sはガウシアカーネルの標準偏差
my.IP.weighted <- function(d1,a1,d2,a2,s=1){
  ret <- 0
  tmp <- outer(d1,d2,"-")
  tmp2 <- outer(a1,a2,"*")
  ret <- sum(pnorm(tmp,0,s) * tmp2)
  return(ret)
}
```

閉曲面ペアの関数内積の計算

２つの閉曲面のそれぞれの、距離行列と頂点支配領域ベクトルとから、 閉曲面ペア間の関数内積を計算することができる。

各頂点からのグラフ距離分布は、 ガウシアンカーネルで分布推定して、それに基づいて、グリッド化した確率質量分布にするのがよいので、そのようにする。

```
# dは頂点間グラフ距離
# aは頂点の支配領域面積
# xは確率質量を求めたい、「距離相当値」
# sはガウシアンカーネル密度推定の標準偏差
my.hist <- function(d,a,x,s=1){
  ret <- rep(0,length(x))
  for(i in 1:length(x)){
    ret[i] <- sum(a * dnorm(d-x[i],0,s))
  }
  return(ret/sum(ret))
}
```

離散化した確率質量分布を使って、閉曲面間の関数内積を算出する。

```
# d1,d2は頂点間距離
# a1,a2は、頂点の支配面積
# s2はガウシアカーネルの標準偏差
my.IP.obj <- function(d1,a1,d2,a2,s2){
  ret <- 0
  dd12 <- (d1) %*% t(d2)
  theta12 <- 1/2 * log(dd12)
  theta11 <- 1/2*log(apply(d1^2,1,sum))
  theta22 <- 1/2*log(apply(d2^2,1,sum))

  dsq <- theta11+theta22 - 2 * theta12
  d <- sqrt(dsq)

  ret <- sum(dnorm(d,0,s2) * outer(a1,a2,"*"))

  return(ret)
}
```

```
# da1,da2は２つの閉曲面のdANDaとする
# s1はmy.IP.weightedのためのガウシアンカーネル用標準偏差
# s2は２つの閉曲面の関数内積のためのガウシアンカーネル用標準偏差
my.IP.obj2 <- function(d1,a1,d2,a2,s1,s2){
  n1 <- length(a1)
  n2 <- length(a2)
  A1s <- apply(d1^2,1,sum)
  A2s <- apply(d2^2,1,sum)
  A12 <- d1 %*% t(d2)
  A12. <- t(t(-2*A12 + A1s) + A2s)
  tmp <- 1/sqrt(2*pi*s2^2) * exp(-A12./(2*s2^2))
  tmp.aa <- tmp * outer(a1,a2,"*")
  ret <- sum(tmp.aa)
  #for(i in 1:n1){
  #  A1 <- 1/2*log(A1s[i])
  #  for(j in 1:n2){
  #    A2 <- 1/2*log(A2s[j])
  #    tmp <- 1/2*log(A12[i,j])
  #    lensq <- (A1+A2-2*tmp)
  #    ret <- ret + 1/(n1*n2)*1/sqrt(2*pi*s2^2)*exp(-lensq/(2*s2^2))
  #  }
  #}
  return(ret)
}
```

実験

分布の平均でやってみる

行列Aは、球面調和関数係数を指定する行列になっている。

球面調和関数は、L行M列で指定できる。

L=0, M=0 L=1, M=-1,0,1 L=1, M=-2,-1,0,1,2 … となっている。

この実験では、M>=0の係数のみいじれるように作ってある。

$A[1,1]$はL=0,M=0を、 $A[2,1]$はL=1,M=0を表している。

$A[1,1]$は真球に相当する。この係数は大きめに固定し、それ以外の係数を選んで、0から次第に増やして行くことにする。

どのセルを変えるかを指定し、いくつかのパターンを作る。

```
n.obj0 <- 10

nn <- 8
A0 <- matrix(0,nn,nn)
A0[1,1] <- 5
B0 <- matrix(0,nn,nn)
cnt <- 1
scl <- 3
```

$A[3,1]$を動かす

```
param <- c(6,2)
Objs <- list()

for(i in 1:n.obj0){
  #Objs[[i]] <- my.3Dobj(d=ds[i],k=ks[i],N=15)
  A <- A0
  A[param[1],param[2]] <- i * 1/n.obj0 * scl
  B <- B0

  Objs[[cnt]] <- my.3Dobj2(A,B,N=15)
  cnt <- cnt+1
}
```

$A[3,2]$を動かす

```
param <- c(4,3)



for(i in 1:n.obj0){
  #Objs[[i]] <- my.3Dobj(d=ds[i],k=ks[i],N=15)
  A <- A0
  A[param[1],param[2]] <- i * 1/n.obj0 * scl
  B <- B0

  Objs[[cnt]] <- my.3Dobj2(A,B,N=15)
  cnt <- cnt+1
}
```

作った２系列をx軸方向に2倍する。

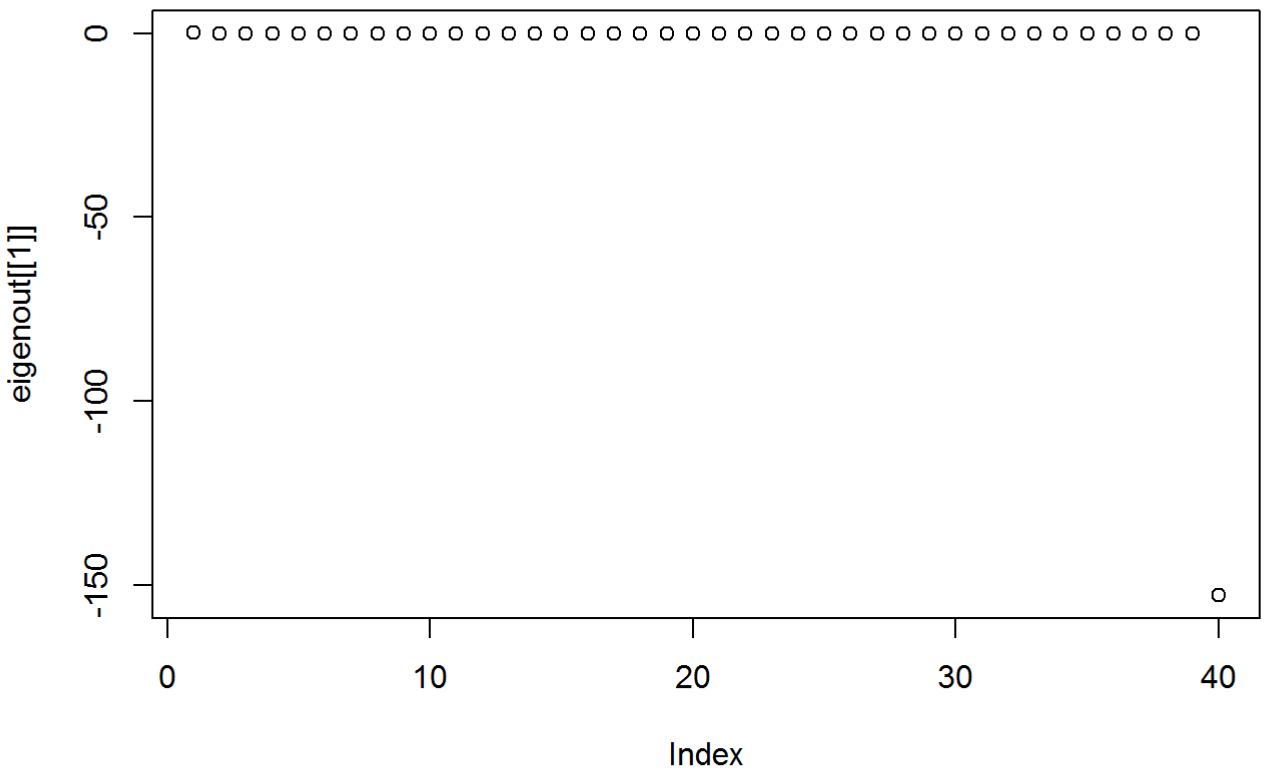
```
for(i in 1:n.obj0){
  #Objs[[cnt]] <- list()
  Objs[[cnt]] <- Objs[[i]]
  Objs[[cnt]]$X[,1] <- Objs[[cnt]]$X[,1]*2
  cnt <- cnt+1
}
```

```
for(i in 1:n.obj0){
  #Objs[[cnt]] <- list()
  Objs[[cnt]] <- Objs[[i+n.obj0]]
  Objs[[cnt]]$X[,1] <- Objs[[cnt]]$X[,1]*2
  cnt <- cnt+1
}
```

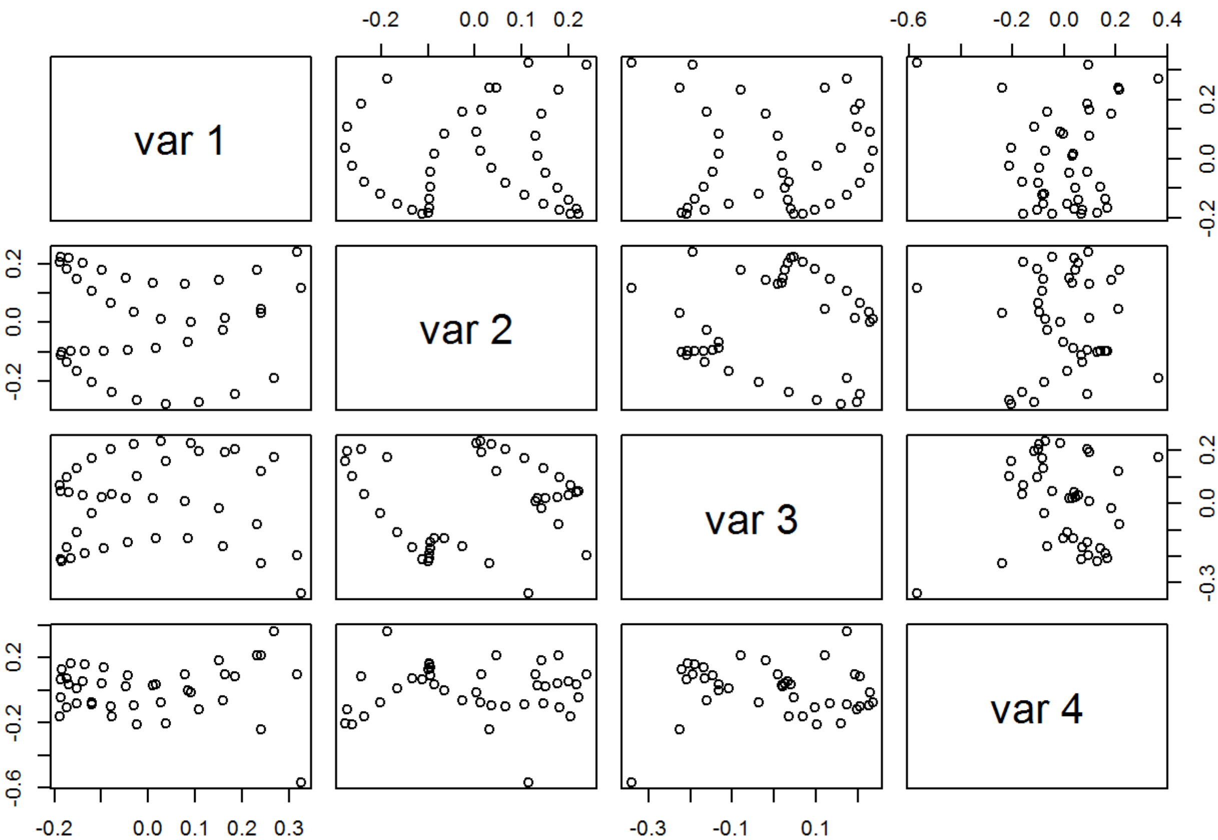
$A[4,3]$を動かす

```
param <- c(4,3)



for(i in 1:n.obj0){
  #Objs[[i]] <- my.3Dobj(d=ds[i],k=ks[i],N=15)
  A <- A0
  A[param[1],param[2]] <- i * 1/n.obj0 * scl
  B <- B0

  Objs[[cnt]] <- my.3Dobj2(A,B,N=15)
  cnt <- cnt+1
}
```

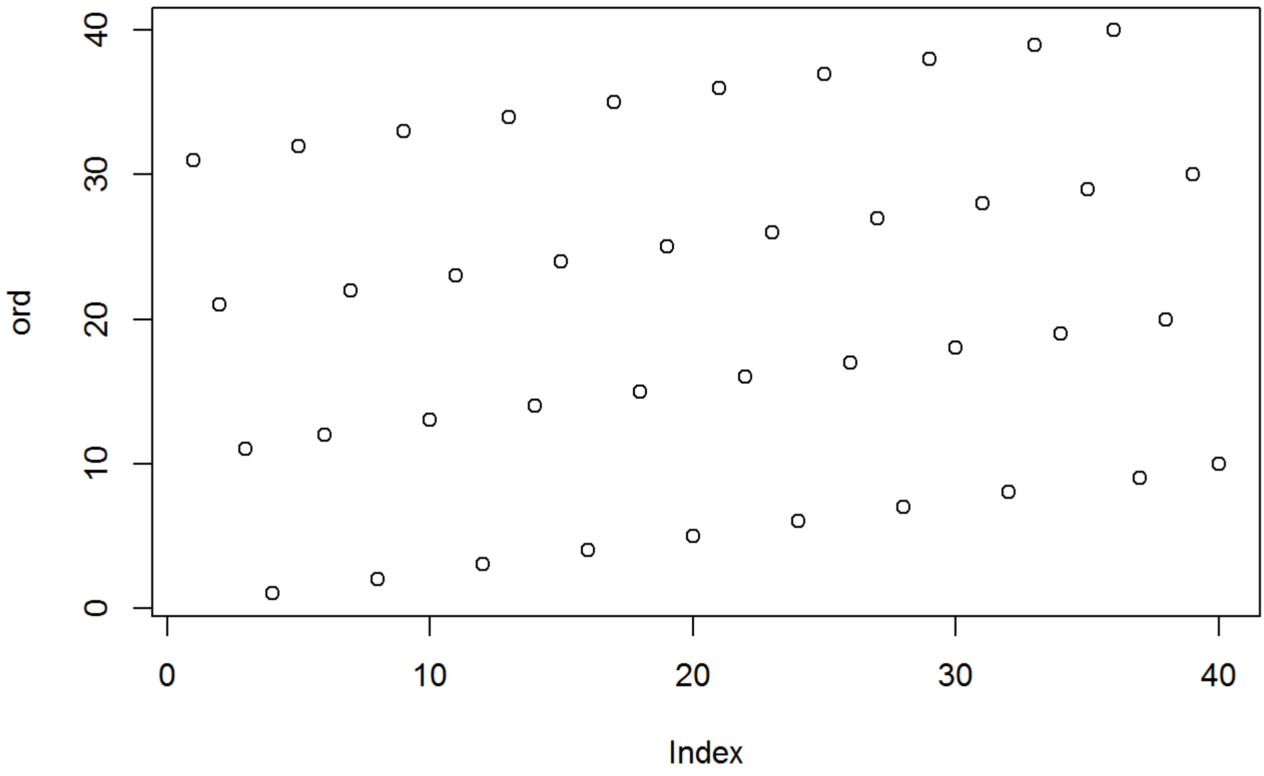
```
n.obj <- length(Objs)
dANDa <- list()
for(i in 1:n.obj){
  dANDa[[i]] <- my.pr.v(Objs[[i]]$g,Objs[[i]]$w,Objs[[i]]$tri,Objs[[i]]$X)
}

tmp <- matrix(0,n.obj,2)
for(i in 1:n.obj){
  tmp[i,] <- range(dANDa[[i]]$d)
}
d.range <- range(tmp)
s1 <- 1
# ガウシアンカーネルのsdの３倍の余裕を前後に持たせる
d.range. <- c(d.range[1]-3*s1,d.range[2]+3*s1)
# グリッド数
Nval <- 100
d.value <- seq(from=d.range.[1],to=d.range.[2],length=Nval)

# オブジェクトごとに、距離分布を算出する

h.list <- list()
for(i in 1:n.obj){
  h.list[[i]] <- matrix(0,length(dANDa[[i]]$a),Nval)
  for(j in 1:length(h.list[[i]][,1])){
    h.list[[i]][j,] <- my.hist(dANDa[[i]]$d[j,],dANDa[[i]]$a,d.value,s1)
  }
}

#matplot(t(h.list[[1]]),type="l")



h.mean <- matrix(0,n.obj,Nval)
for(i in 1:n.obj){
  h.mean[i,] <- apply(h.list[[i]] * dANDa[[i]]$a,2,sum)
}

IPmean <- h.mean %*% t(h.mean)
eigenout <- eigen(log(IPmean))

plot(eigenout[[1]])
```



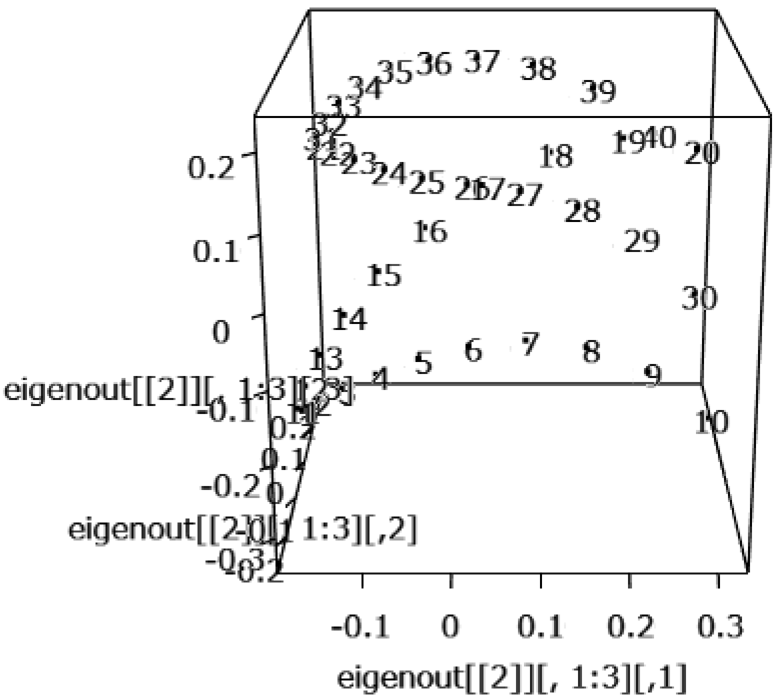

```
pairs(eigenout[[2]][,1:4])
```



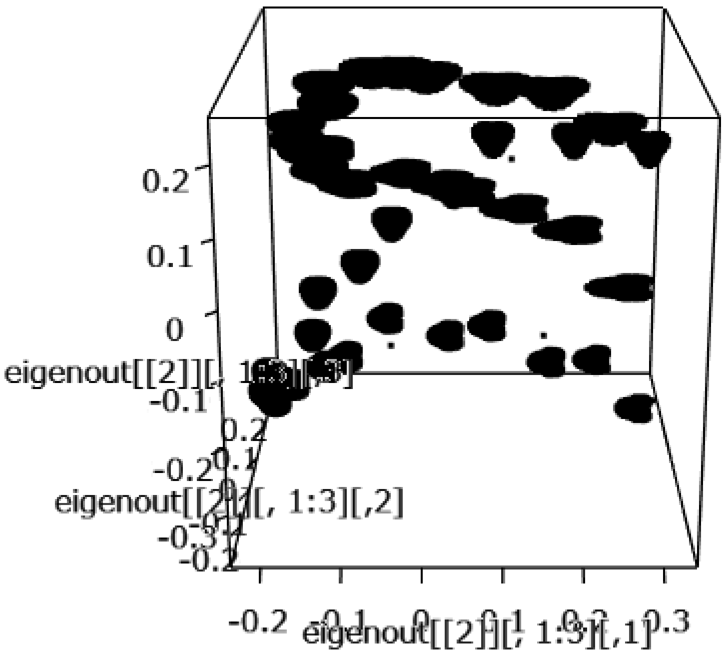
```
ord <- order(eigenout[[2]][,1])
plot(ord)
```



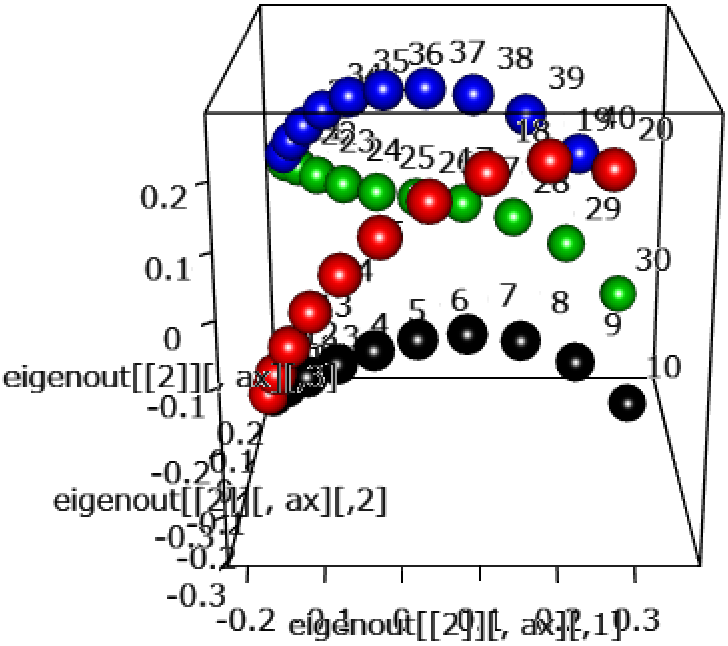

```
rg <- range(eigenout[[2]][,1:3])
rgdif <- rg[2]-rg[1]
plot3d(eigenout[[2]][,1:3])
text3d(eigenout[[2]][,1:3],texts=paste("",1:n.obj))
#for(i in 1:n.obj){
  #tmpobj <- Objs[[i]]
  #tmpobj$X <- tmpobj$X*rgdif/n.obj * 0.8
  #tobeadded <- eigenout[[2]][i,1:3] + sign(i %% 2 -0.5)*rnorm(3)*rgdif*0.02
  #tmpobj$X <- t(t(tmpobj$X) + tobeadded)
  #my.plot.shape(tmpobj,add=TRUE)
#}
```




```
rg <- range(eigenout[[2]][,1:3])
rgdif <- rg[2]-rg[1]
plot3d(eigenout[[2]][,1:3])
#text3d(eigenout[[2]][,1:3],texts=paste("",1:n.obj))
for(i in 1:n.obj){
  tmpobj <- Objs[[i]]
  tmpobj$X <- tmpobj$X*rgdif/n.obj * 0.8
  tobeadded <- eigenout[[2]][i,1:3] + sign(i %% 2 -0.5)*rnorm(3)*rgdif*0.02
  tmpobj$X <- t(t(tmpobj$X) + tobeadded)
  my.plot.shape(tmpobj,add=TRUE)
}
```




```
ax <- 1:3
plot3d(eigenout[[2]][,ax])
spheres3d(eigenout[[2]][,ax],radius=0.03,color=rep(1:4,each=n.obj0))
text3d(eigenout[[2]][,ax]+0.05,texts=paste("",1:n.obj))
```



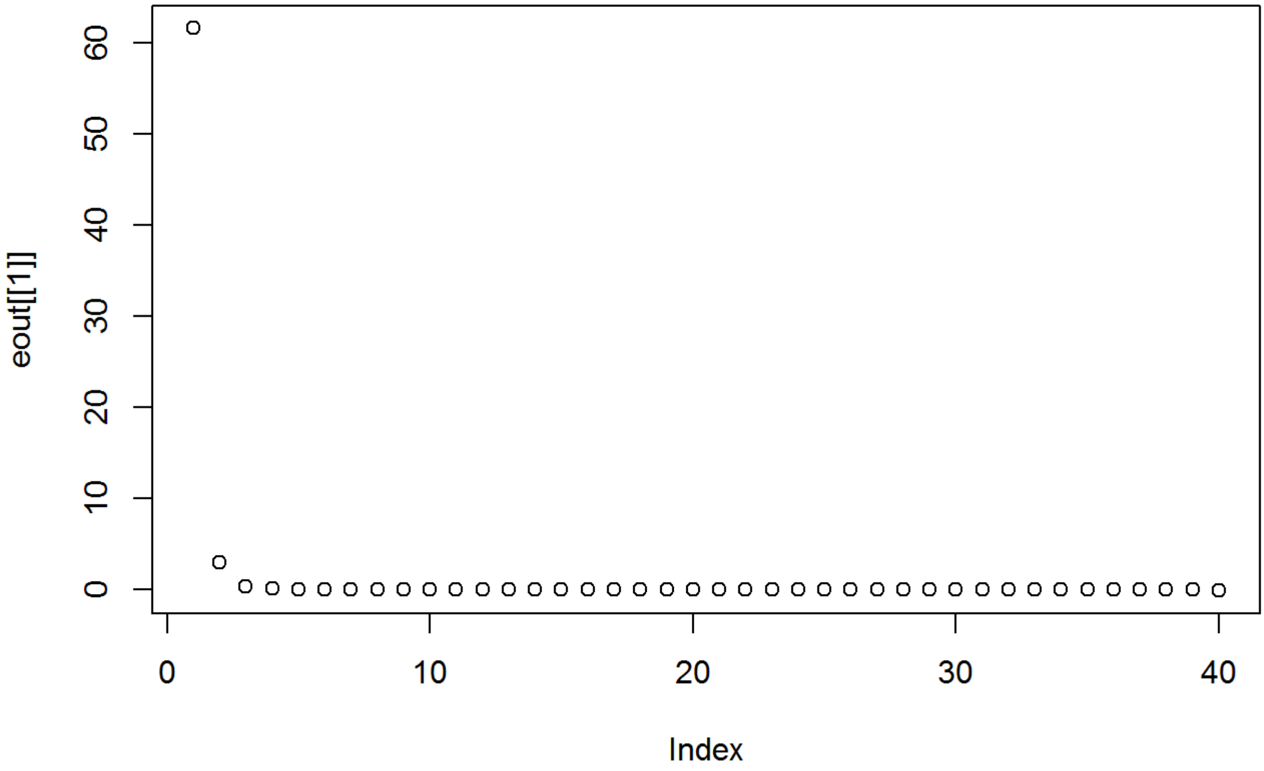
「分布の分布」作戦

分布の平均とそれほど違わない？

```
s1 <- 0.01
s2 <- 0.01
ObjIP <- matrix(0,n.obj,n.obj)
for(i in 1:n.obj){
  for(j in 1:n.obj){
    ObjIP[i,j] <- my.IP.obj2(h.list[[i]],dANDa[[i]]$a,h.list[[j]],dANDa[[j]]$a,s1,s2)
  }
}
```

```
logIP <- 1/2*log(ObjIP)
eout <- eigen(logIP)
ord2 <- order(eout[[2]][,1])
```

```
plot(eout[[1]])
```




```
pairs(eout[[2]][,1:4])
```