Distributional Quantitification of Dissimilarity of Distributions/Shapes 分布・形の異同計量を分布で行うこと

ryamada

2019年9月18日

Shape is Distribution

S2-isomorphic shapes can be mapped on a unit sphere with Willmore energy flow.

The mapping generates density distribution on the unit sphere of “area” of the original shape.

The following notes will describe how to quantitate dissimilarity of two distributions on a unit sphere.

It means dissimilarity of shapes and densities on a unit sphere are quantitate.

Distributional Quantification

When shapes in 3D or spherical distributions are compared, 3D rotation of shapes/distributions may be allowed.

In this case, the optimal rotation for two shapes/distributions has to be found to quantitate their dissimilarity.

Unfortunately it is not easy to estimate the optimal rotation.

This difficulty is originated from the plan to find the optimal rotation.

Therefore the difficulty would not be the issue if we have a method to quantitate their dissimilarity without finding the optimal rotation.

Actually the difficulty to find the optimal rotation is due to the exsitence of many local optima.

The pair of shapes/distributions that have many local optima whose dissimilarity value to the global optima’s, are less specific each other in terms of their patterns than the pair whose local optima dissimilarity values are not close to one of global optima.

This means the values of local optima and number of local optima have some information on the dissimilarity between the two.

More generally, the comparison of two shapes/distributions with many rotational alignments that are not even local optima but are general positions, provides information on their dissimilarity.

This is the idea of “distributional quantitation” of shapes/distributions with multiple alignments.

Procedures to obtain distribution to quantitate dissimilairty of shapes/distributions

Shapes are transformed to deistibutions.

Therefore we will discuss the method to compare two distributions on a unit sphere, hereafter.

- Rotate one of two distributions with adequately many rotations that are reasonably evenly spaced in the 3d-rotation manifold.

- quantitate the dissimilarity between every rotated distribution and the other distribution.
- The measure of dissimilarity between two distributions can be KL divergence-based. It can be inner product of distibution functions. Symmetric measure would be beneficial.

In the following discussion, we assume two distributions are represented by many points.

Using kernel method to estimate distribution, we can calculate likelihood to obtain one point set from the distribution that is estimated from the other point set.

Generation of point data sets

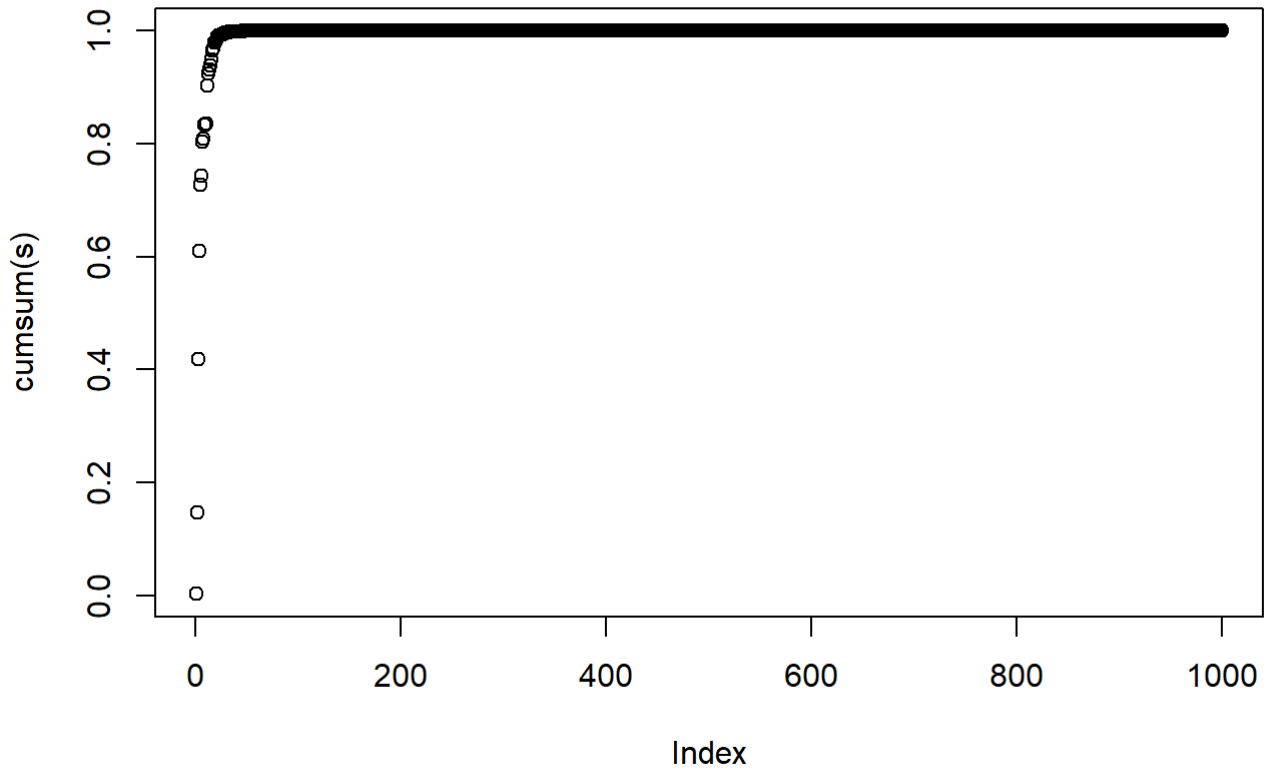
実際の乱点発生プラン

- Dirichlet process categorical distribution generation for mixture of “2d-normal distributions”
- Generate multinomial distribution
- Generate 2d Eucledian normal distributions
- Map normal dists to sphere
- Generate N points in the mixture of normal dists on shpere

Stick-breaking process

```
library(dirichletprocess)

N <- 1000
alpha <- 5
s <- StickBreaking(alpha,N)
plot(cumsum(s))
```

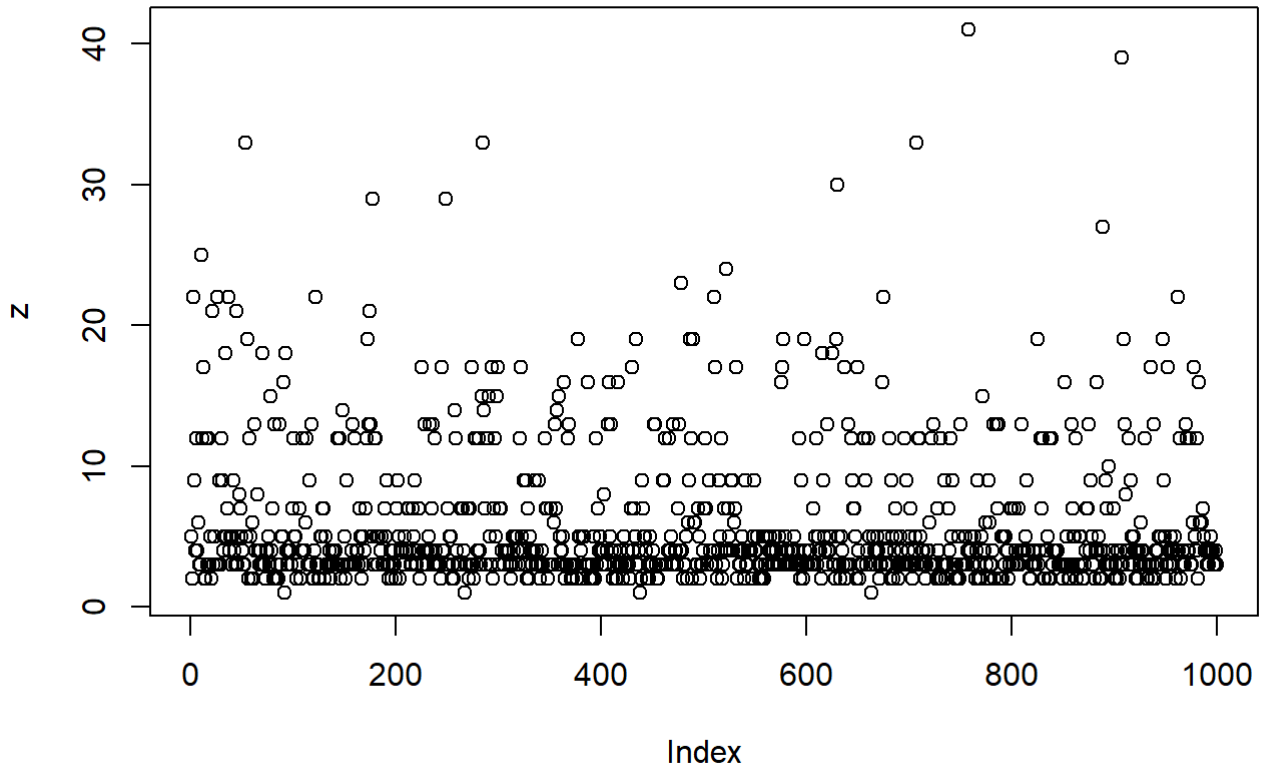


```
sum(s) # 1
```

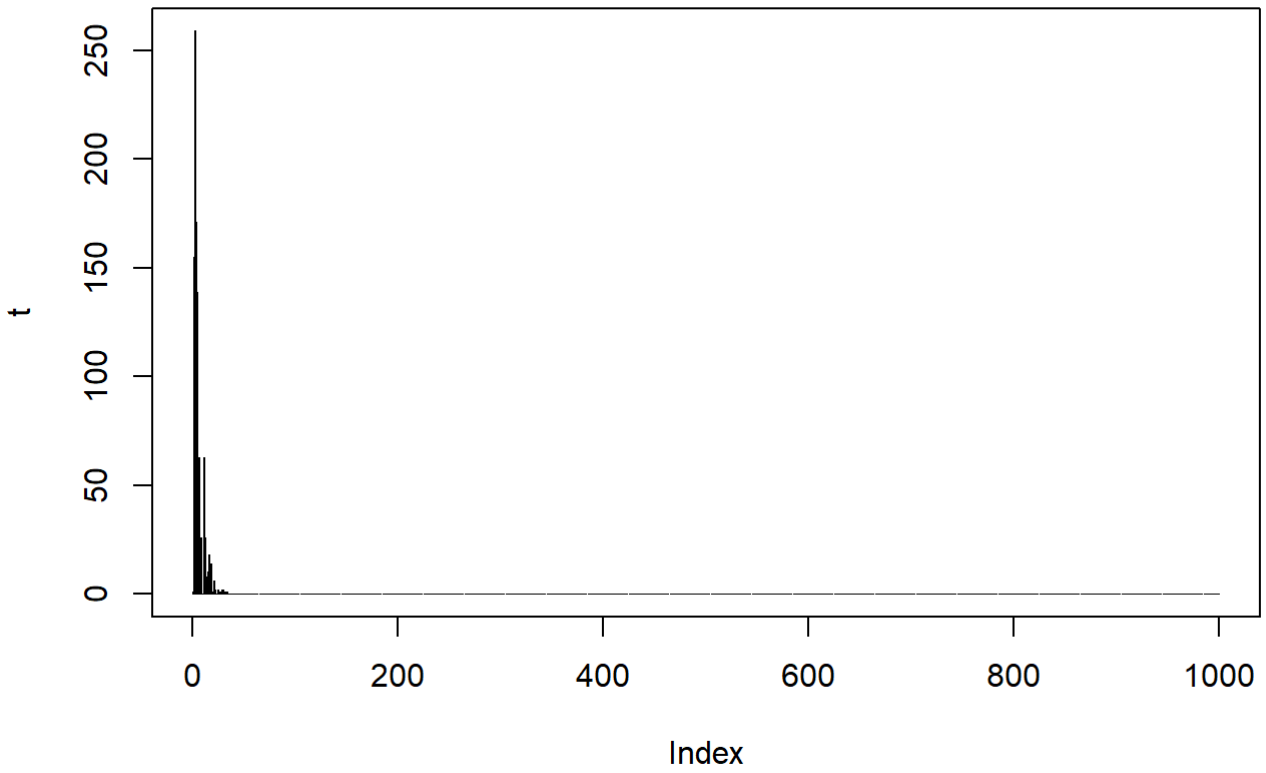
```
## [1] 1
```

Category assinment to N points

```
z <- sample(1:N,N,replace=TRUE,prob=s)
plot(z)
```




```
t <- table(z)
# もしくは
t <- rmultinom(1,N,prob=s)
plot(t,type="h")
```




```
sum(t) # Nに一致
```

```
## [1] 1000
```

Parameters of 2d normal distributions

Center of normal distributions

```
my.runifS2 <- function(n){
    X <- matrix(rnorm(n*3),ncol=3)
    X <- X/sqrt(apply(X^2,1,sum))
    return(X)
}
mus <- my.runifS2(N)
```

Var-covar matrix of 2d normal distributions

```
library(MCMCpack)
```

```
## Loading required package: coda
```

```
## Loading required package: MASS
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
```

```
## ## Copyright (C) 2003-2019 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
```

```
## ##
## ## Support provided by the U.S. National Science Foundation
```

```
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```
sigmas <- list()
df <- 3
m <- matrix(c(1,0,0,1),2,2) * 0.01
for(i in 1:N){
  sigmas[[i]] <- rwish(df,m)
}
```

Random point generation on the 2d Eucledian plane

```
library(mvtnorm)
x <- matrix(0,N,2)
for(i in 1:N){
  x[i,] <- rmvnorm(1,c(0,0),sigmas[[i]])
}
```

Map the points on the plane to the sphere

```
# pが接点
# x が球面上の点
# 関数の帰り値にpを加えた座標が切平面に乗る
my.Riemannian.log <- function(x,p){
  cos.t <- sum(x*p)
  if(abs(cos.t)>1){
    cos.t <- sign(cos.t) * 1
  }
  t <- acos(cos.t)
  return((x-p*cos.t)*t/sin(t))
}
# 上記 log関数の逆関数
my.Riemannian.Exp <- function(x,p){
  r <- sqrt(sum(x^2))
  ret <- p * cos(r) + x/r*sin(r)
  return(ret)
}
```

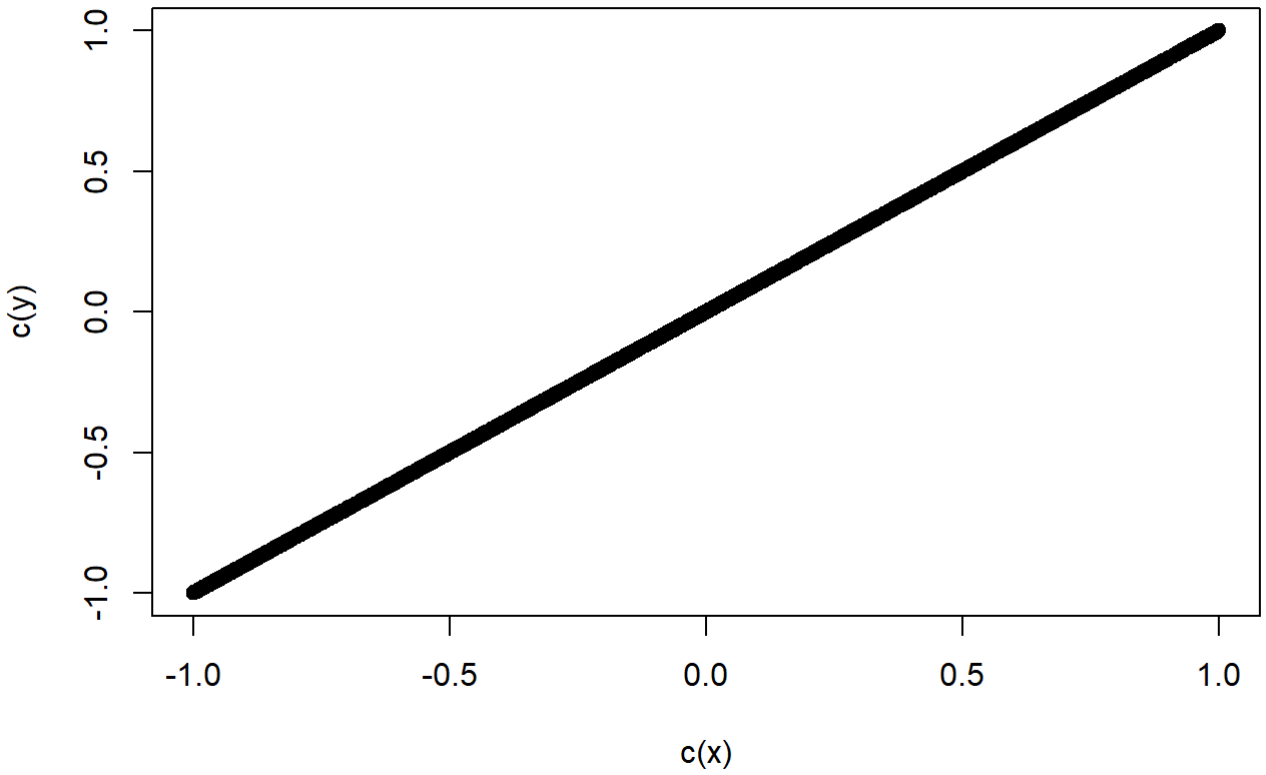
Visualization of the mapping method

```
library(rgl)
x <- my.runifS2(1000)
p <- my.runifS2(1)
## xxは点 pからのベクトル
xx <- t(apply(x,1,my.Riemannian.log,p))
xx_p <- t(t(xx) + c(p))
plot3d(x)
spheres3d(p,color="red",radius=0.1)
spheres3d(xx_p,radius=0.05)

s <- sample(1:length(x[,1]),100)

for(i in 1:length(s)){
  segments3d(rbind(x[s[i],],xx_p[s[i],]))
}
```

```
y <- t(apply(xx,1,my.Riemannian.Exp,p))
plot3d(y)
plot(c(x),c(y)) # 元に戻っている
```



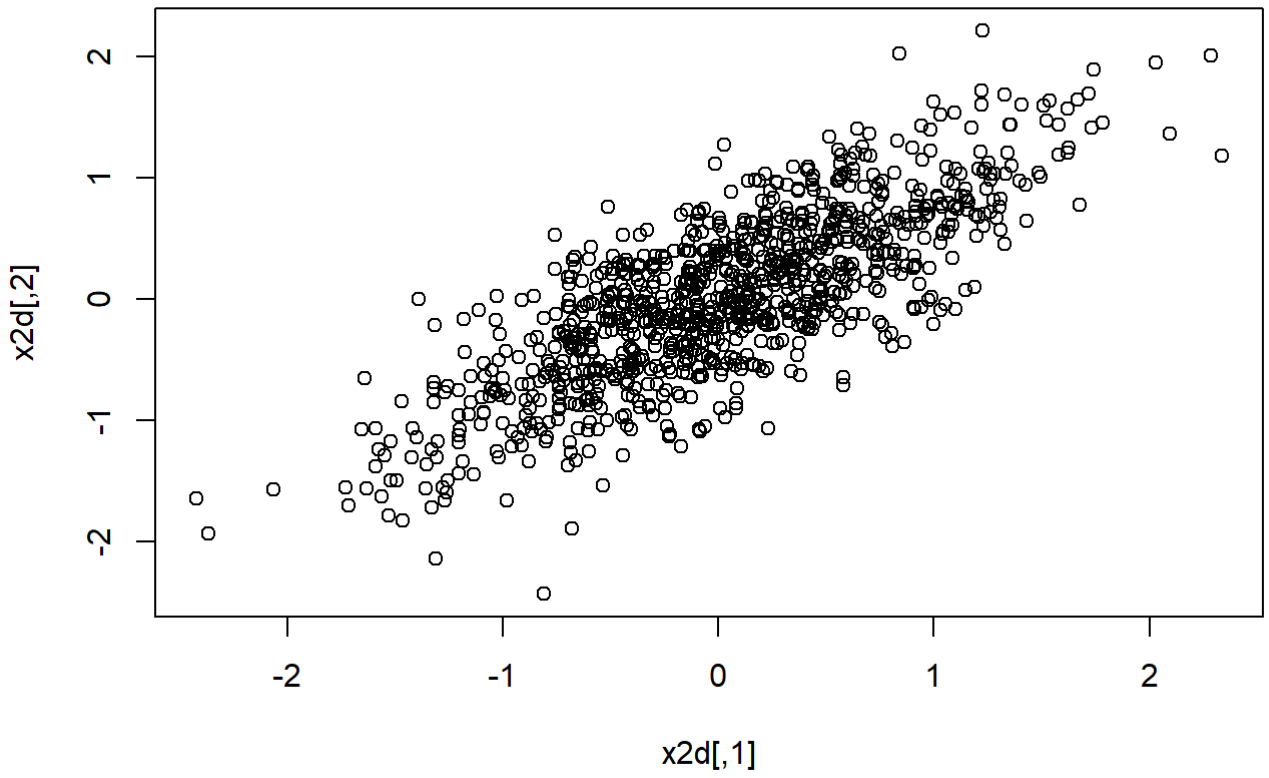
```
# こちらは、接点pが(0,0,1)とみなし、二次元平面上の点xをxy平面上の点とし
# 球面に写像した上で
# (0,0,1)を接点pに移す大円回転でxの対応点も移す
my.Tp2S2 <- function(x,p){
  tmp.p <- c(0,0,1)

  # 回転させたいが、まず、pのz = 1平面上の対応点をとる
  p. <- my.Riemannian.log(p,tmp.p)
  p. <- c(p.[1:2],0)
  # その方向の単位ベクトルにする
  p. <- p./sqrt(sum(p.^2))
  # p.と直交するベクトルを取る
  q. <- c(p.[2],-p.[1],0)

  # z = 1 平面上で (x-tmp.p)ベクトルを、(p.-tmp.p)と(q.-tmp.p)の線形和で表すこととし
  # その係数を求める
  M <- cbind(p.[1:2],q.[1:2])
  coefs <- solve(M) %*% x
  # tmp.pでの接平面をpでの接平面に移す
  # ベクトル p. - tmp. p が p.' - p に移るとき
  # p.' - pは以下に代わる
  # theta はtmp.pとpとの角
  theta <- acos(sum(tmp.p*p))
  p._p <- c(cos(theta)*p.[1:2],-sin(theta))
  # q. - tmp.p = q.' - pとなる
  q._p <- q.
  #M. <- cbind(p._p,q._q)
  x._p <- coefs[1] * p._p + coefs[2] * q._p
  x. <- c(x._p) + c(p)
  #return(x.)
  return(my.Riemannian.Exp(x._p,p))
}
```

Example

```
mu <- my.runifS2(1)
sigma <- matrix(c(0.5,0.4,0.4,0.5),2,2)
n <- 1000
x2d <- rmvnorm(n,c(0,0),sigma)
plot(x2d)
```




```
x3d <- t(apply(x2d,1,my.Tp2S2,mu))
plot3d(x3d)
spheres3d(mu,radius=0.1,color="red")
```

Functions of the procedure above

```
my.DPMMS2 <- function(N,alpha=10,df=3,m=matrix(c(1,0,0,1),2,2)*0.01,sms=NULL){
  if(is.null(sms)){
    s <- StickBreaking(alpha,N)
    mus <- my.runifS2(N)
    sigmas <- list()
    for(i in 1:N){
      sigmas[[i]] <- rwish(df,m)
    }
  }else{
    s <- sms$s
    mus <- sms$mus
    sigmas <- sms$sigmas
  }

  t <- rmultinom(1,N,prob=s)


   x <- matrix(0,N,3)
  cnt <- 1
  for(i in 1:length(t)){
    if(t[i]!=0){
        x2d <- matrix(rmvnorm(t[i],c(0,0),sigmas[[i]]),ncol=2)
      x3d <- t(apply(x2d,1,my.Tp2S2,mus[i,]))
      x[cnt:(cnt+t[i]-1),] <- x3d
      cnt <- cnt + t[i]
    }
  }
  return(list(x=x,s=s,t=t,mus=mus,sigmas=sigmas))
}
my.plot3d.dpmms2 <- function(out,r=0.05){
  x <- out$x
  t <- out$t
  plot3d(x)
  cnt <- 1
  for(i in 1:length(t)){
    if(t[i]!=0){
        spheres3d(x[cnt:(cnt+t[i]-1),],color=i,radius=0.05)
    cnt <- cnt + t[i]
    }
  }
}
```

Distributional quantification of dissimilarity between two point sets

Two point sets on S2

```
N <- 1000
alpha <- 10
s <- StickBreaking(alpha,N)
mus <- my.runifS2(N)
df=3
m=matrix(c(1,0,0,1),2,2)*0.01
sigmas <- list()
for(i in 1:N){
  sigmas[[i]] <- rwish(df,m)
}
sms1 <- list(s=s,mus=mus,sigmas=sigmas)
s2 <- StickBreaking(alpha,N)
sms2 <- list(s=s2,mus=mus,sigmas=sigmas)
dpmms2out1 <- my.DPMMS2(N,sms=sms1)
dpmms2out2 <- my.DPMMS2(N,sms=sms2)
```

```
plot3d(dpmms2out1$x)
```

```
plot3d(dpmms2out2$x)
```

```
my.plot3d.dpmms2(dpmms2out1)
```

```
my.plot3d.dpmms2(dpmms2out2)
```

```
# Nは乱点の数を決める引数
# f1 は黄金比の値をデフォルトとする値。この値を黄金比から変えると格子としての良い性格がなくなる
# k=1をデフォルトとし、フィボナッチ格子を描くためにはk=1。ただし、その背景にあるらせんを描くためにはこの値をいじって、らせん上の点の数を増やす必要がある
fib.lattice.S2 <-function(N,f1=(sqrt(5)+1)/2,k=1){
    f2 <- f1-1 # 黄金比はx^2-x-1=0の１つの根。もう１つの根を取り出す
    #P <- 2*N+1
    i <- seq(from=-N, to=N,by=k)
    theta <- asin(2*i/(2*N+1))
    phi <- 2*pi*i*f2
    x <- cos(theta)*cos(phi)
    y <- cos(theta)*sin(phi)
    z <- sin(theta)
    return(cbind(x,y,z))
}

library(rgl)


# フィボナッチ格子点の座標
N <- 100
fl <- fib.lattice.S2(N)


plot3d(fl)
# らせんの座標
N2 <- N
fl2 <- fib.lattice.S2(N2,k=0.01)
# らせんを描いて
plot3d(rbind(fl2),radius=0.01)
# フィボナッチ格子を重ね描き
points3d(fl,size=10)
#rgl.snapshot("fib.png")

# 回転クオータニオンをフィボナッチ球面上らせんを軸として発生
# Nは単位純虚四元数ベクトル数
# nは回転角の数
library(onion)
my.fib.rot.q <- function(N,n){
    fl <- fib.lattice.S2(N)
    fl.q <- fl[,1] * Hi + fl[,2] * Hj + fl[,3] * Hk
    theta <- seq(from=0,to=1,length = n+1)
    theta <- theta[-n]
    tmp <- expand.grid(theta,1:length(fl.q))
    return(cos(tmp[,1]/2) + sin(tmp[,1]/2) * fl.q[tmp[,2]])
}


rot.q <- my.fib.rot.q(100,100)

my.vonMises.like <- function(X1,X2,k,log=FALSE){
    n1 <- length(X1[,1])
    n2 <- length(X2[,1])
    tmp <- k * sum(X1 %*% t(X2))/(n1*n2)
    if(!log){
        tmp <- exp(tmp)
    }
```

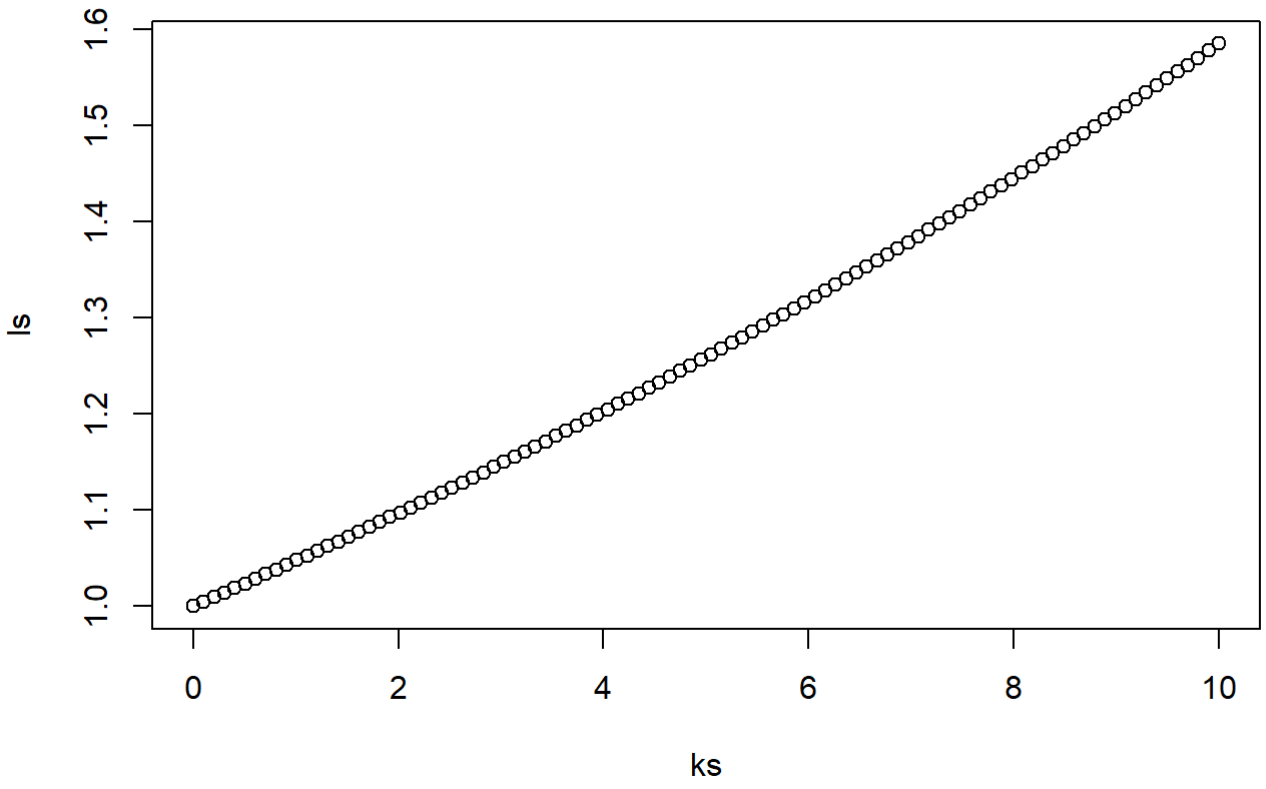
```
    return(tmp)
}

my.rot3d.by.q <- function(x,q){
  tmp.x <- Hi * x[1] + Hj * x[2] + Hk * x[3]
  rot.x <- Conj(q) * tmp.x * q
  return(c(i(rot.x),j(rot.x),k(rot.x)))
}
```

Calculate vonBises-type relative likelihood-like value with various k values

```
x1 <- dpmms2out1$x
x2 <- dpmms2out2$x
ks <- seq(from=0,to=10,length=100)
ls <- rep(0,length(ks))
for(i in 1:length(ks)){
  ls[i] <- my.vonMises.like(x1,x2,k=ks[i])
}
```

```
plot(ks,ls)
```



Rotate one distribution with many evenly-spaced rotations and measure von-Mises-type likelihood

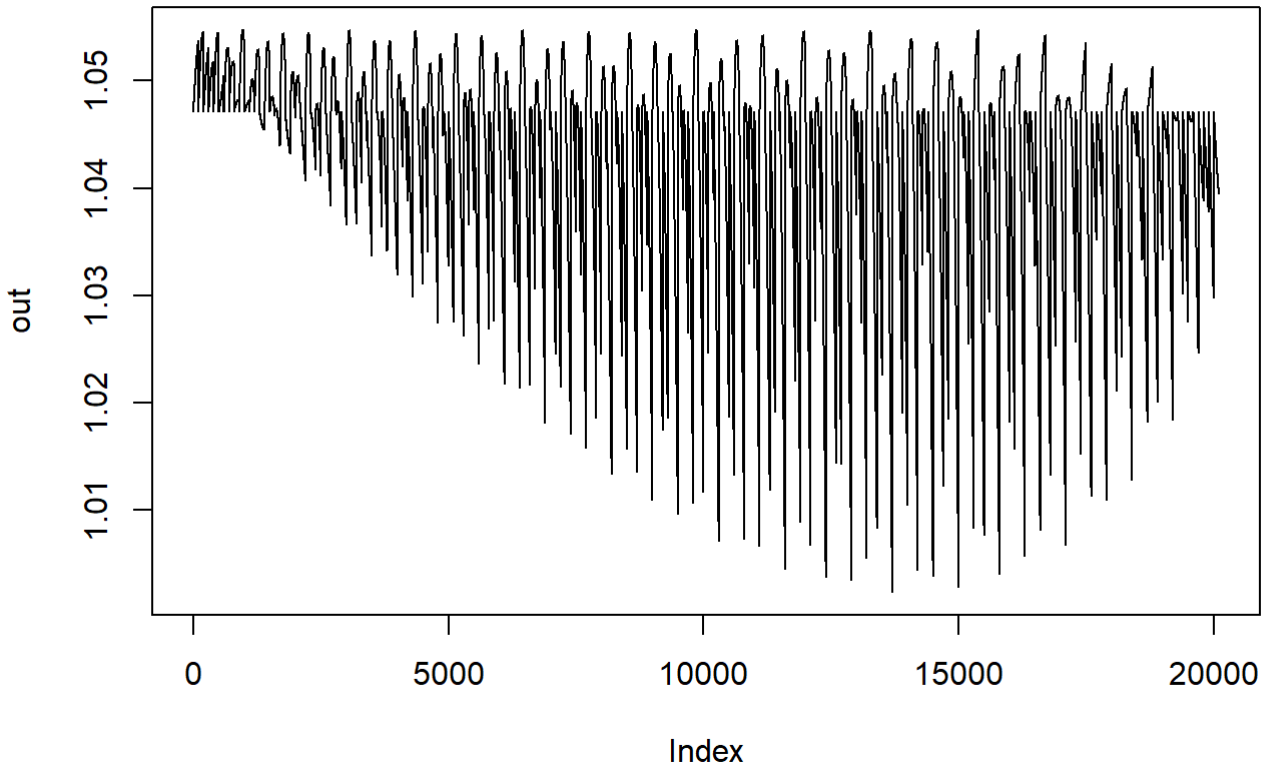
```
my.distributional.quant <- function(x1,x2,k,rot.q=NULL,N=100,n=100,log=FALSE){
  if(is.null(rot.q)){
    rot.q <- my.fib.rot.q(N,n)
  }

  x2.q <- x2[,1] * Hi + x2[,2] * Hj + x2[,3] * Hk
  ret <- rep(0,length(rot.q))
  for(i in 1:length(ret)){
    tmp <- Conj(rot.q[i]) * x2.q * rot.q[i]
    tmp.x2 <- cbind(i(tmp),j(tmp),k(tmp))
    ret[i] <- my.vonMises.like(x1,tmp.x2,k=k,log=log)
  }
  return(ret)
}
```

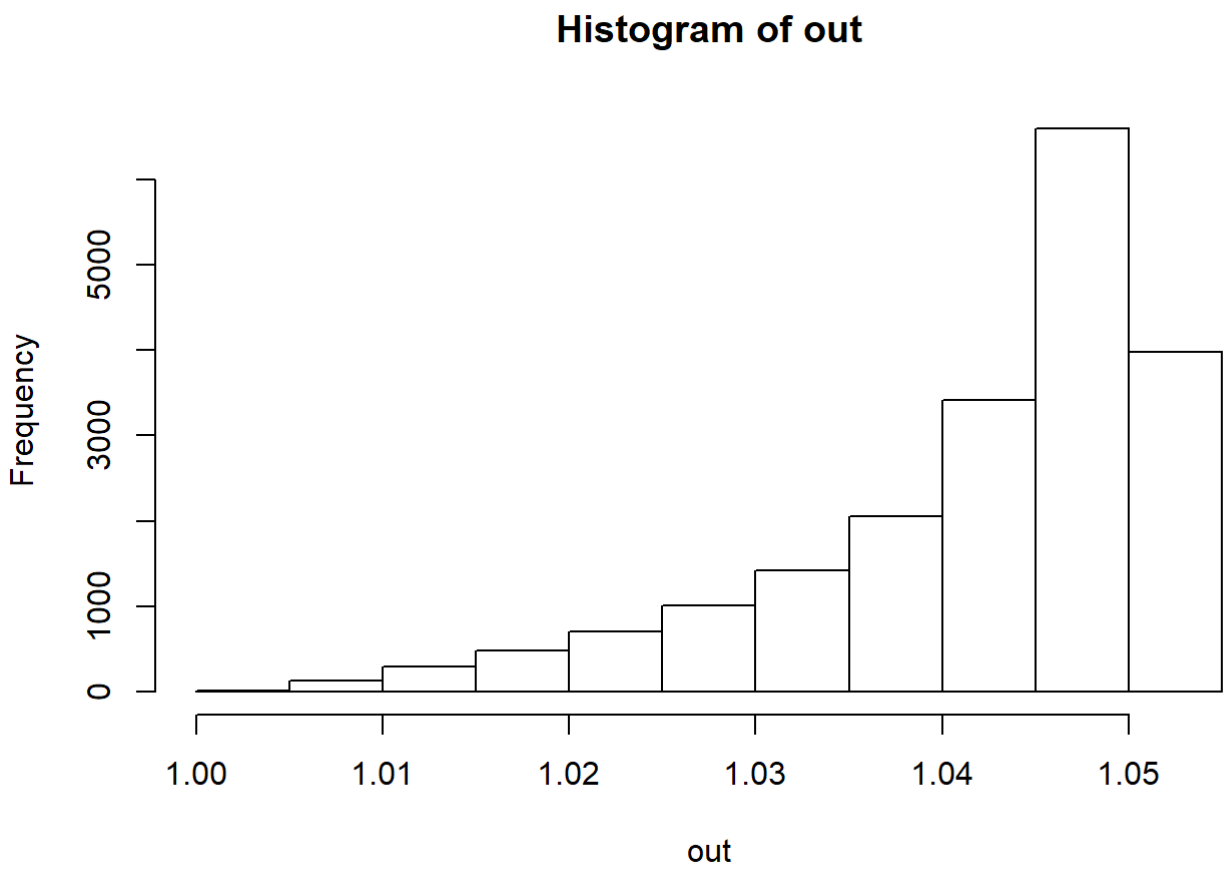
```
N <- 100
n <- 100

k <- 1
out <- my.distributional.quant(x1,x2,k=k,N=N,n=n)
```

```
plot(out,type="l")
```




```
hist(out)
```



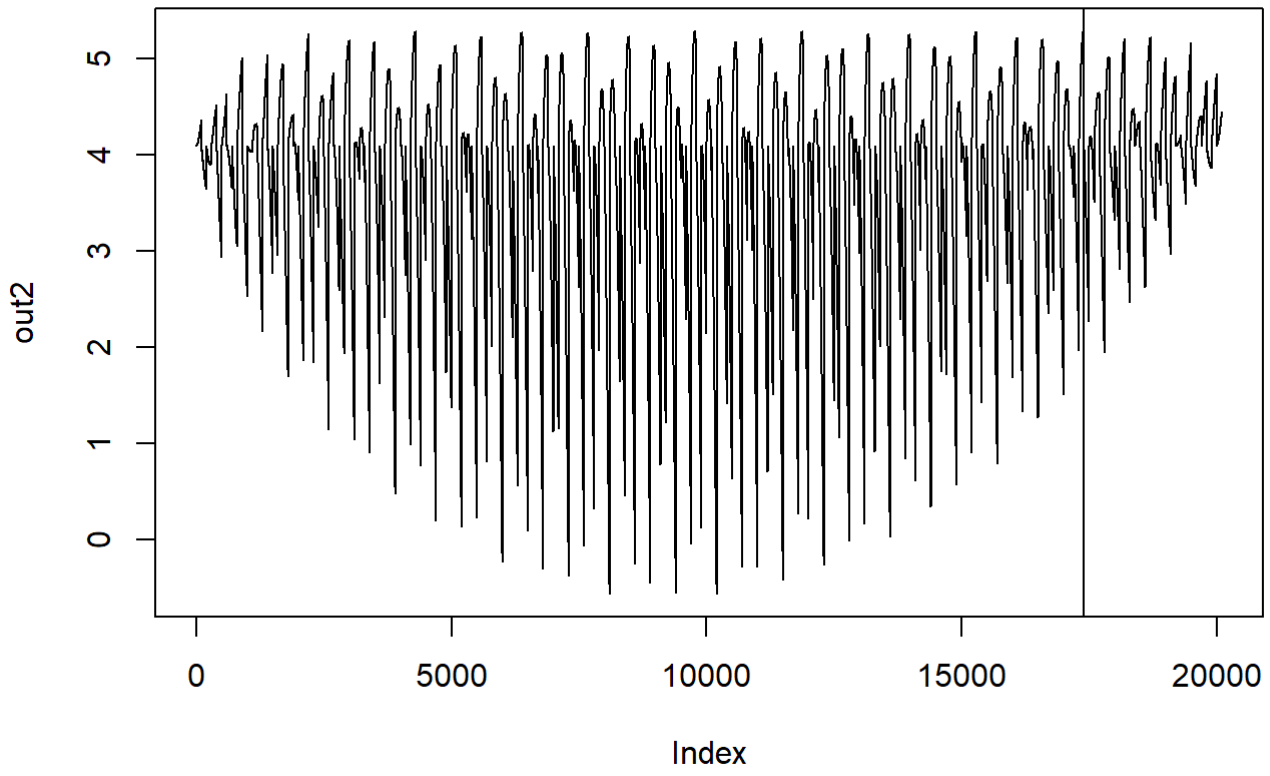

```
rot.q <- my.fib.rot.q(N,n)
max.q <- rot.q[which(out==max(out))]
```

```
rot.x2.max <- t(apply(x2,1,my.rot3d.by.q,max.q))
plot3d(rbind(x1,x2))
spheres3d(x1,radius=0.1,color="black")
spheres3d(x2,radius=0.1,color="red")
spheres3d(rot.x2.max,radius=0.1,color="blue")
```

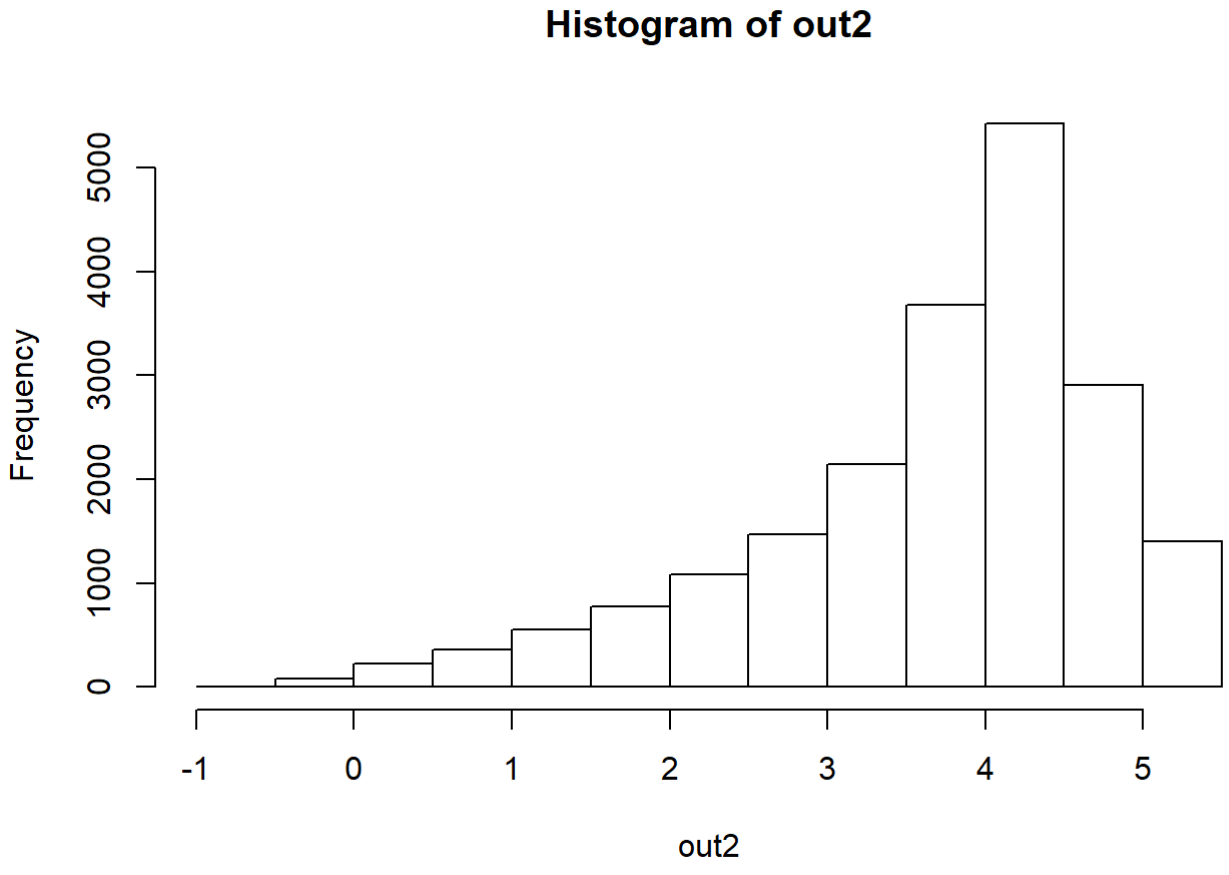
```
rot.q <- my.fib.rot.q(N,n)
qid <- sample(1:length(rot.q),1)
this.rot <- rot.q[qid]
x1.rot <- t(apply(x1,1,my.rot3d.by.q,this.rot))
```

```
out2 <- my.distributional.quant(x1.rot,x1,k=100,N=N,n=n,log=TRUE)
```

```
plot(out2,type="l")
abline(v=qid)
```


out2
0
1
2
3
4
5
0
5000
10000
15000
20000
Index

```
hist(out2)
```


Histogram of out2
Frequency
0
1000
2000
3000
4000
5000
-1
0
1
2
3
4
5
out2