球面調和関数分解方針

ryamada

2018年10月20日

必要とするパッケージなどの準備

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

```
## 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 'knitr' was built under R version 3.4.4
```

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

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

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

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

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

```
##
## 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)
```

球面調和関数分解の方針

方針変更の理由

これまで使っていたSPHARMからの出力係数の正体が不明であるから。

1. 三角メッシュはスムージングする

- 球面調和関数分解は曲面分解差分の線型最小二乗法なので、その方が効率的・効果的だから)

2. 球化後、球面上に均等配置した点のセットに対する、オリジナル三角メッシュの座標を補間して求める。これにより、いくつかのメリットがある

- 補間により情報が増える(線型補間を線形分解に使っても情報は増えないが、線型補間により、増えた点の球面調和関数値は非線形関数なので、情報が増える)
- 固定点を用いて球面調和関数分解をする場合は、固定点用の行列を一度、計算しておけば、球面調和関数係数の計算は単なる行列演算になるから

3. 球面調和関数分解の次数Lは頂点数に合わせて変える必要がある。オーバーフィッティングを起こすからである。均等配置点の数を予め決めておけば、適当なLの値も決めておくことができる

方針に必要な関数など

1. 球面上均等点集合と、それに対応する、球面調和関数分解用行列。それらを計算する関数
2. 形ごとの球化後情報(オリジナル頂点座標、球面座標、三角形トリオ情報)から、均等点のオリジナル座標を算出する関数

関数説明

球面上均等点集合と、それに対応する球面調和関数分解用の逆行列作成関数

```
# N = 40に対して、L=20はおそらく妥当
# Zが係数計算用行列
# SpStが球面上均等配置点の3D座標
# shapeはその他もろもろの情報
my.even.Z <- function(N=40,L=20){
  shape <- my.cell.shape2(2,2,N)
  Z <- my.Z(L,shape$angles[,1],shape$angles[,2])
  Z.inv <- solve(t(Z)%*% Z)%*% t(Z)
  #Z.inv <- ginv(Z)
  return(list(SpSt=shape$X.sp,angles=shape$angles,Z.inv=Z.inv,Z=Z,shape=shape))
}
```

球化オブジェクトの情報から球面均等配置点に対応する座標を求める関数

```
# SpPt: 球面上の(均等)点座標
# オリジナルの形の座標
# 形の球面マップ座標
# 頂点トリオ
my.tri.interploation <- function(SpPt,X,Xsp,tri){
  ret <- matrix(0,length(SpPt[,1]),3)
  for(i in 1:length(tri[,1])){
    inout.multi <- my.inside.triangle.multi(t(SpPt),t(Xsp[tri[i,],]))
    crds <- t(X[tri[i,],]) %*% inout.multi[[2]]
    ret[inout.multi[[1]],] <- t(crds)
  }
  return(ret)
}
```

球面調和関数係数を求める関数、形を復元する関数

係数算出関数

```
# anglesはZ,Z.inv算出のときに用いた球面均等配置座標の角座標
# Xinterpolatedはそのオリジナル3D座標
my.spcoef.est2 <- function(Xinterpolated,Z.inv){
  ret <- Z.inv %*% Xinterpolated
  return(ret)
}
```

球面調和関数係数から、球面均等配置点に対するオリジナル3D座標を復元

```
my.coef2shape2 <- function(coef,Z){
  Z %*% coef
}
```

球面の任意の点セットの角座標から3D座標を復元

```
my.coef2shape <- function(coef,angles){
  L <- sqrt(length(coef[,1]))-1
  Z <- my.Z(L,angles[,1],angles[,2])
  Z %*% coefs
}
```

この方針での係数推定処理フロー

球面上均等配置点とそれに対応する変換逆行列を作る

NとLとは指定する。

```
N=40
L=20
SpStAndZ <- my.even.Z(N,L)
```

複数の形とその球面マップ情報が必要

実データがあればよいが、シミュレーションで作成することにする。

個々のオブジェクトについて、3D座標、三角形リスト、球面上座標をのリストとして作る。

球面上座標が不均一になるように少し工夫しておく。

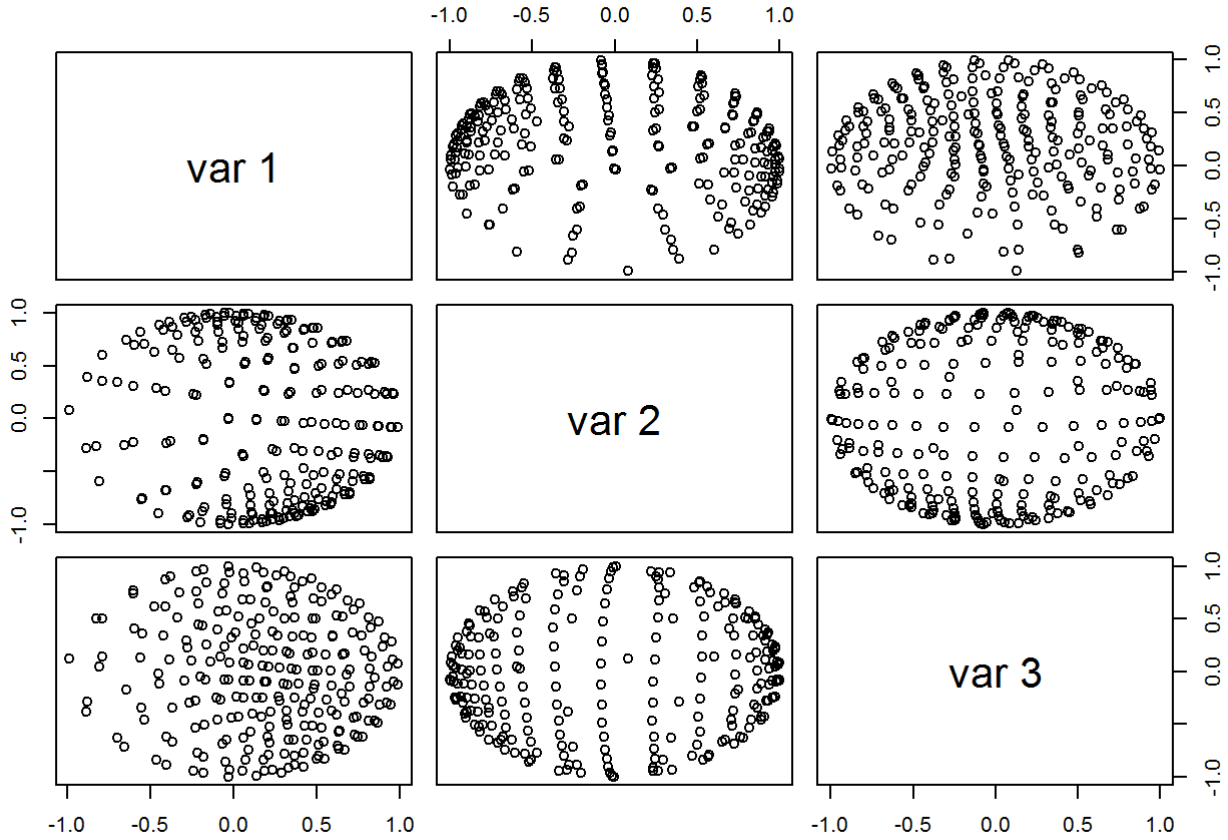
```
# 還り値
## X : 3D 座標
## X.sp : 球面上の座標
## E : エッジ
## tri : 三角形頂点トリオ
## angles : 球面上の角座標
## g,w : グラフオブジェクトとエッジの重み

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)
}
```

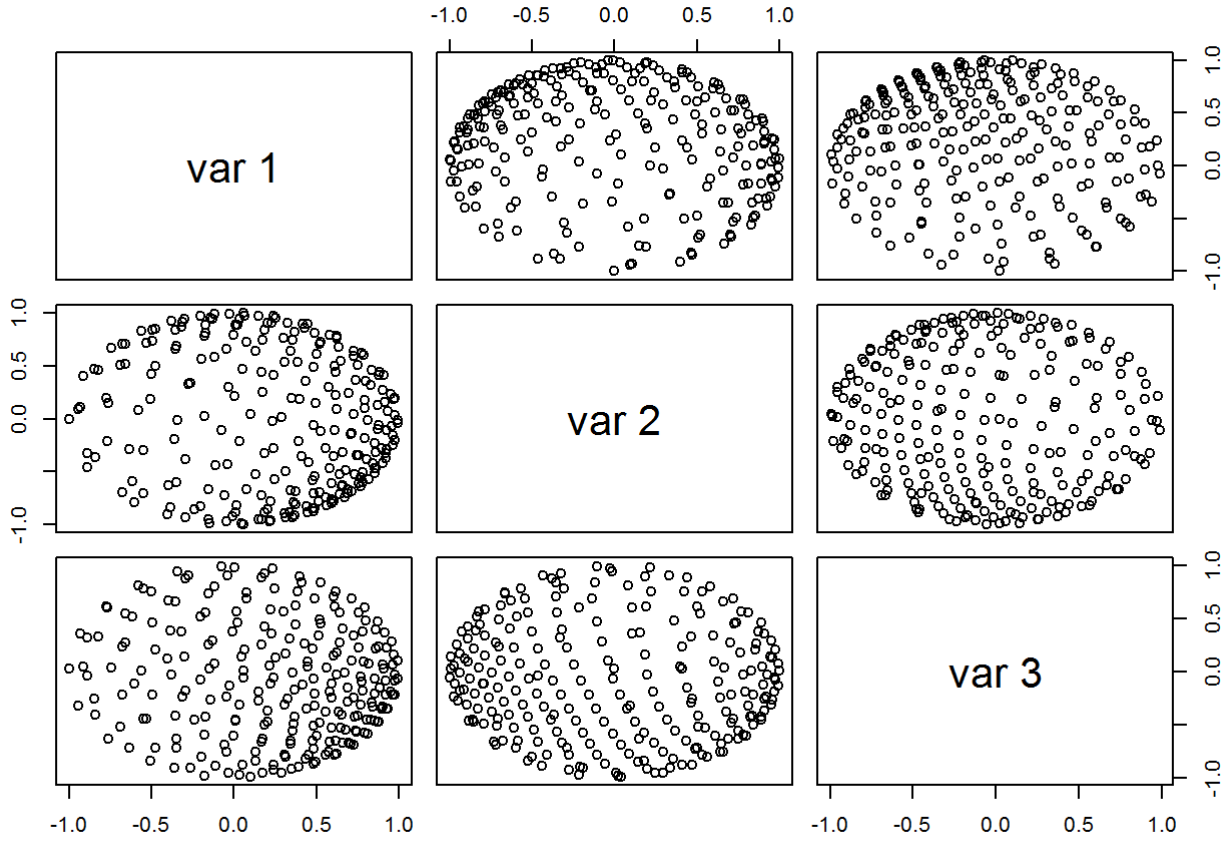
```
n.obj <- 10
Objs <- list()
for(i in 1:n.obj){
  Objs[[i]] <- my.3Dobj()
}
```

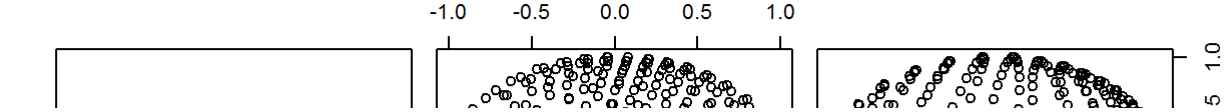
```
my.plot.shape(Objs[[1]])
```

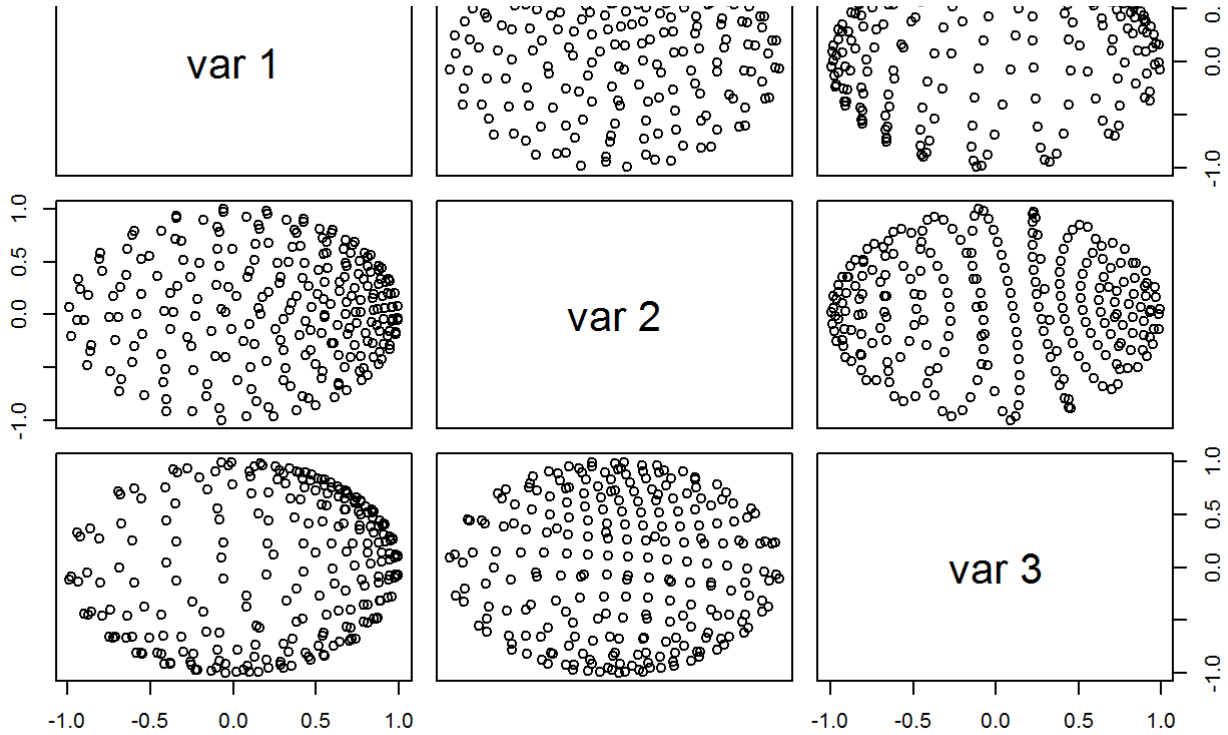
球面の点の不均等具合を確認

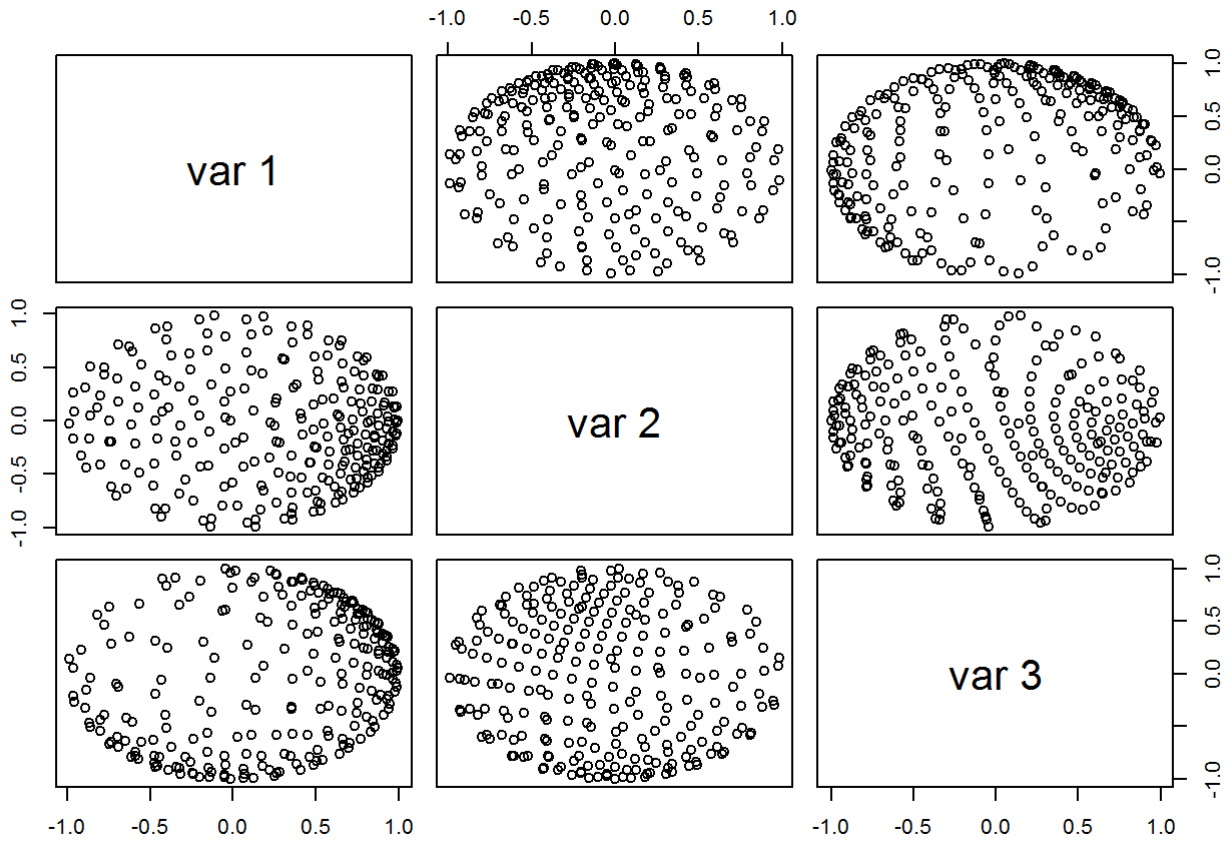
```
for(i in 1:n.obj){
  pairs(Objs[[i]]$X.sp)
}
```

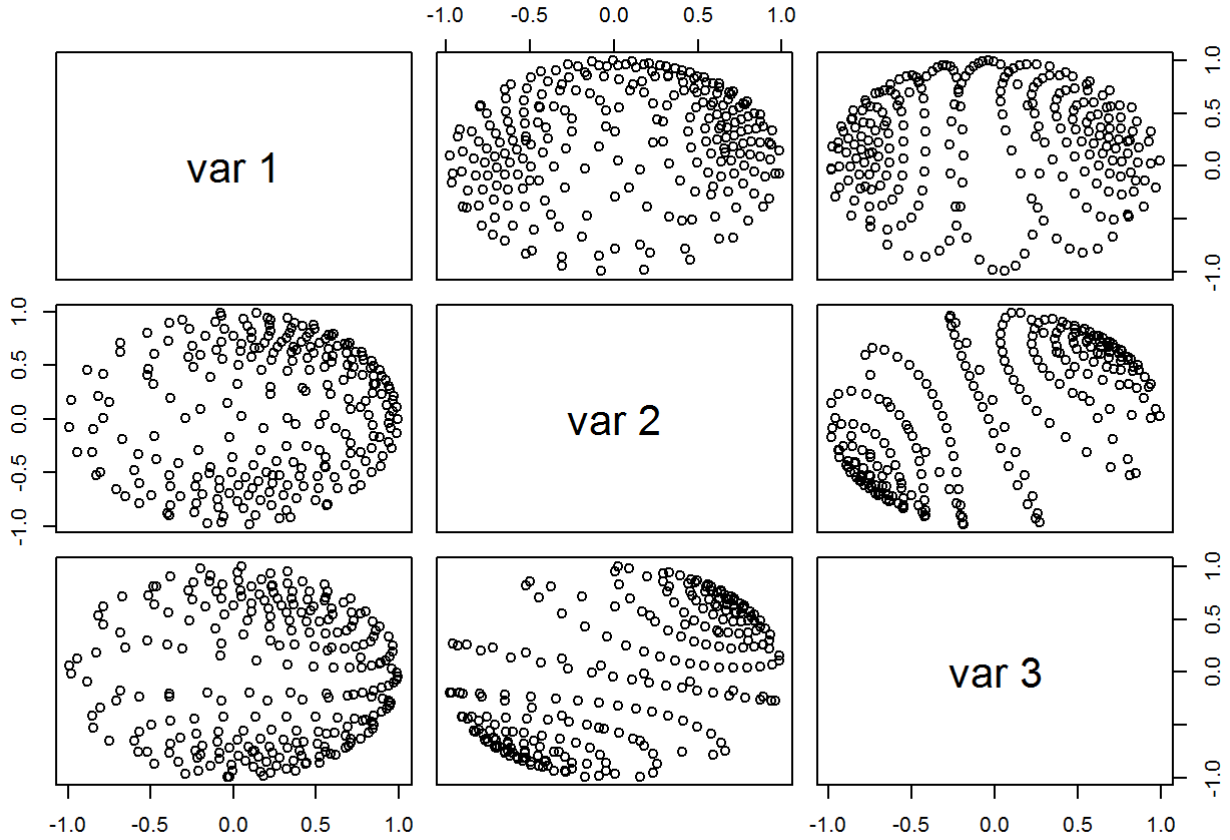

var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0

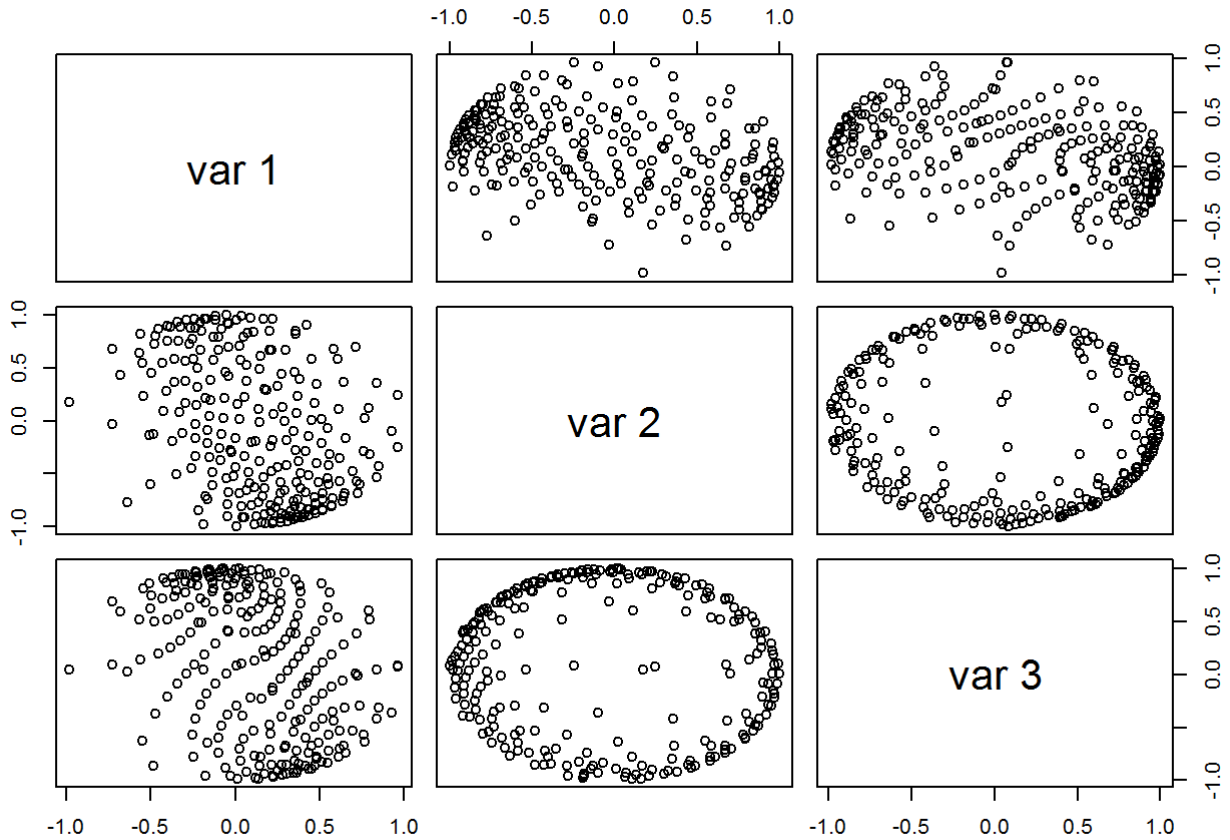

var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0

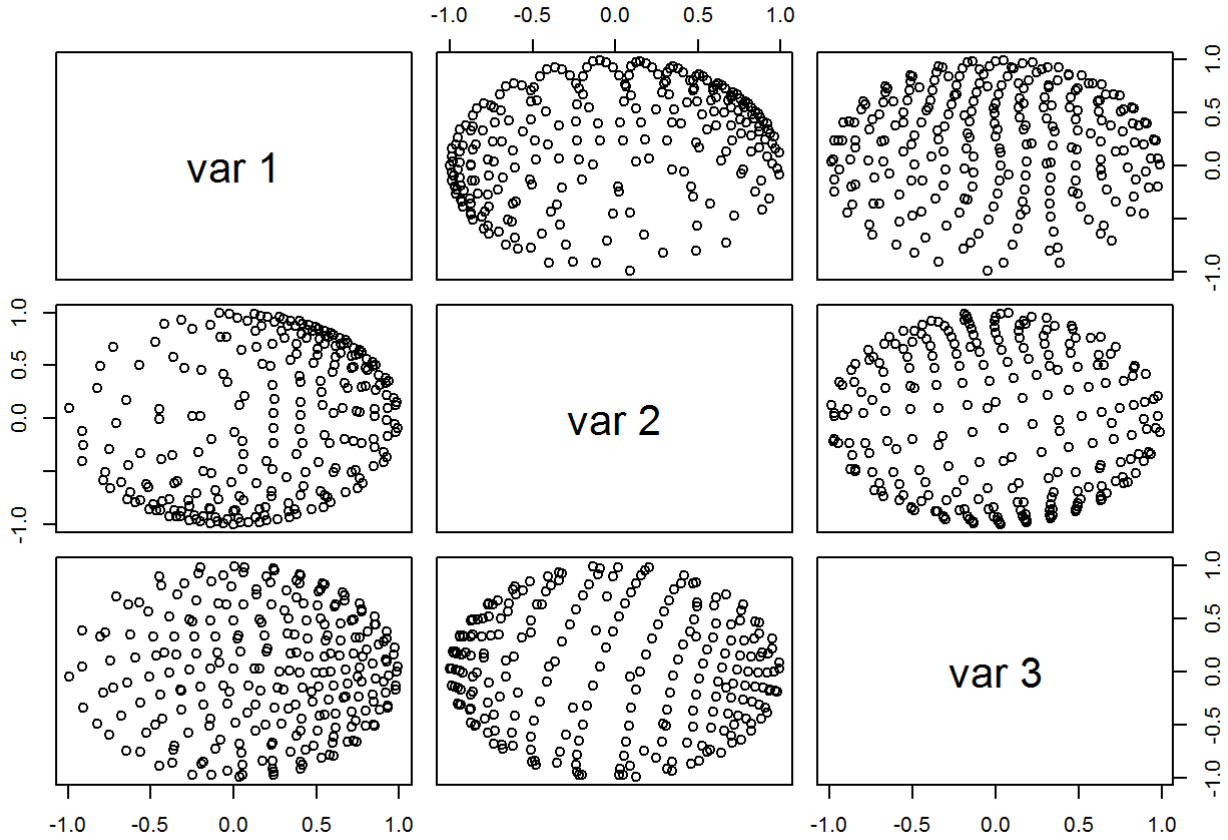


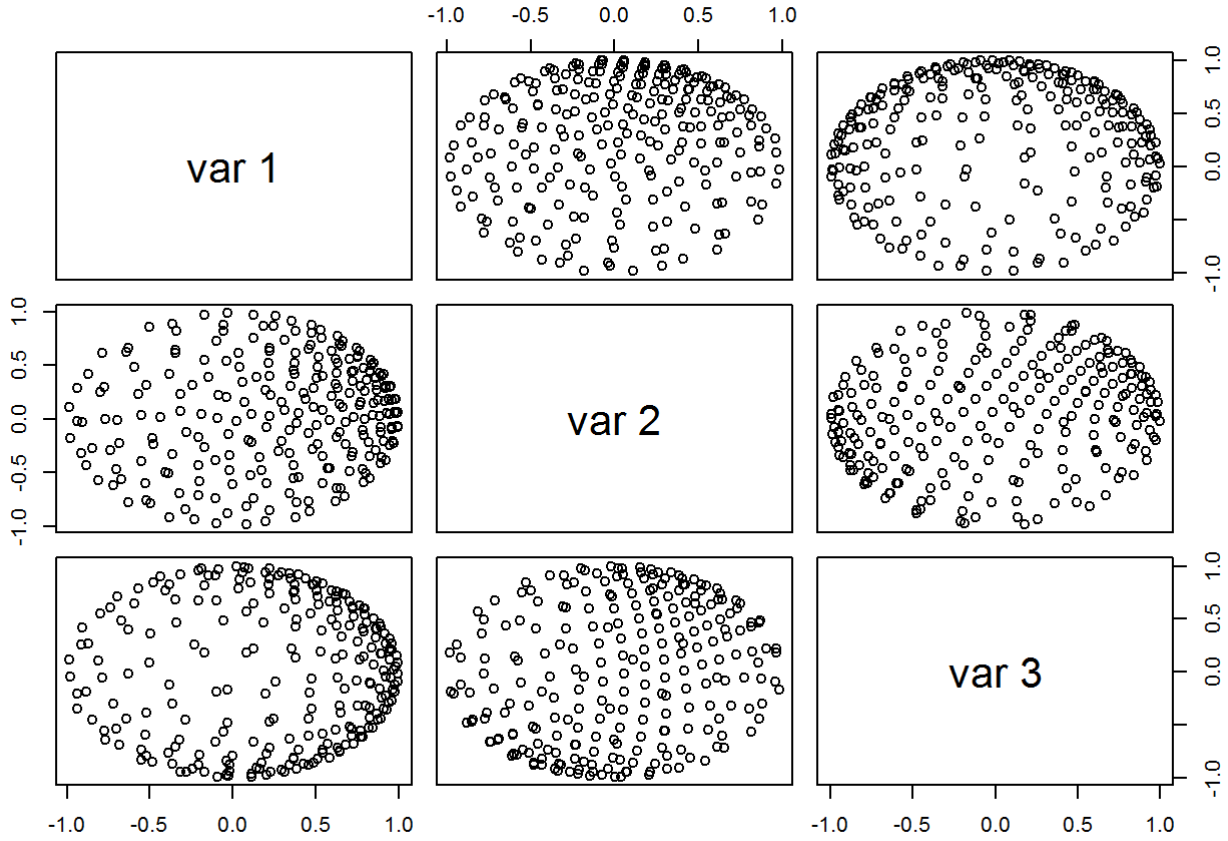

var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0

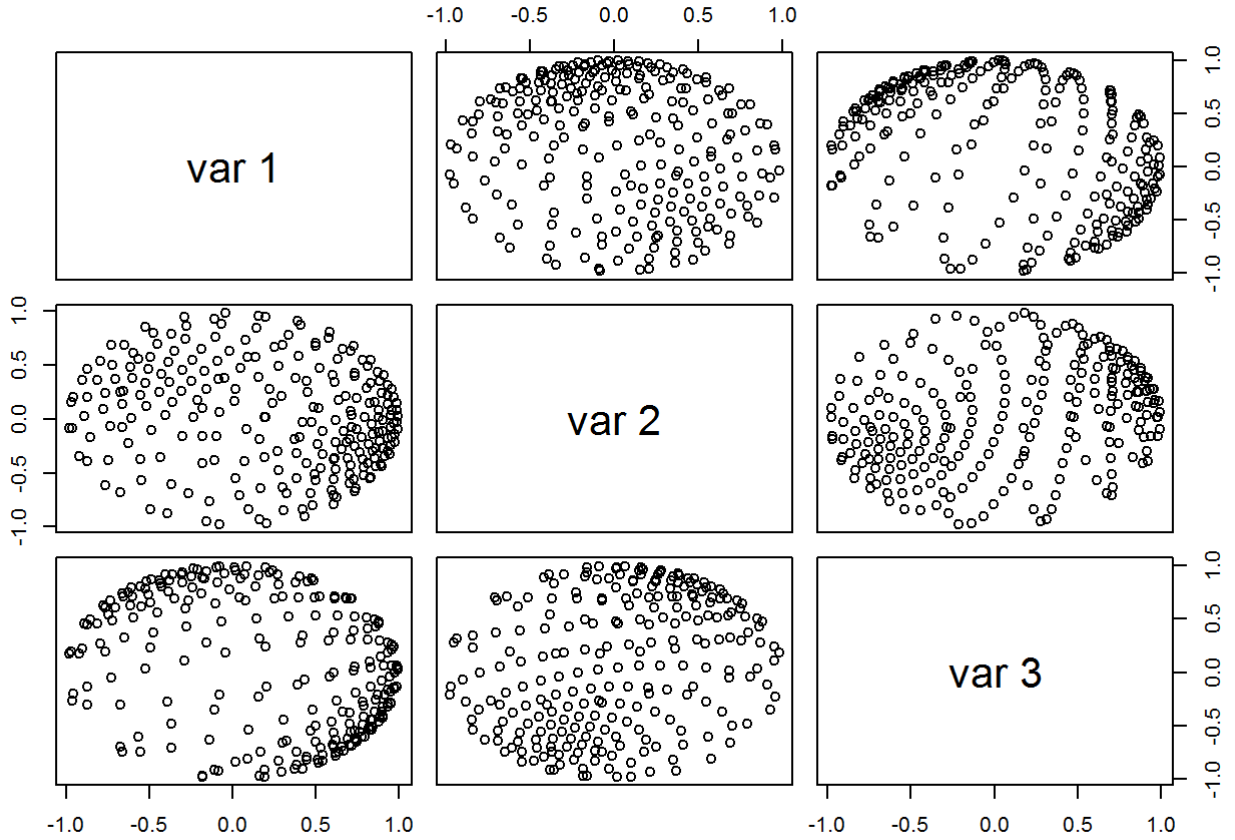

var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0

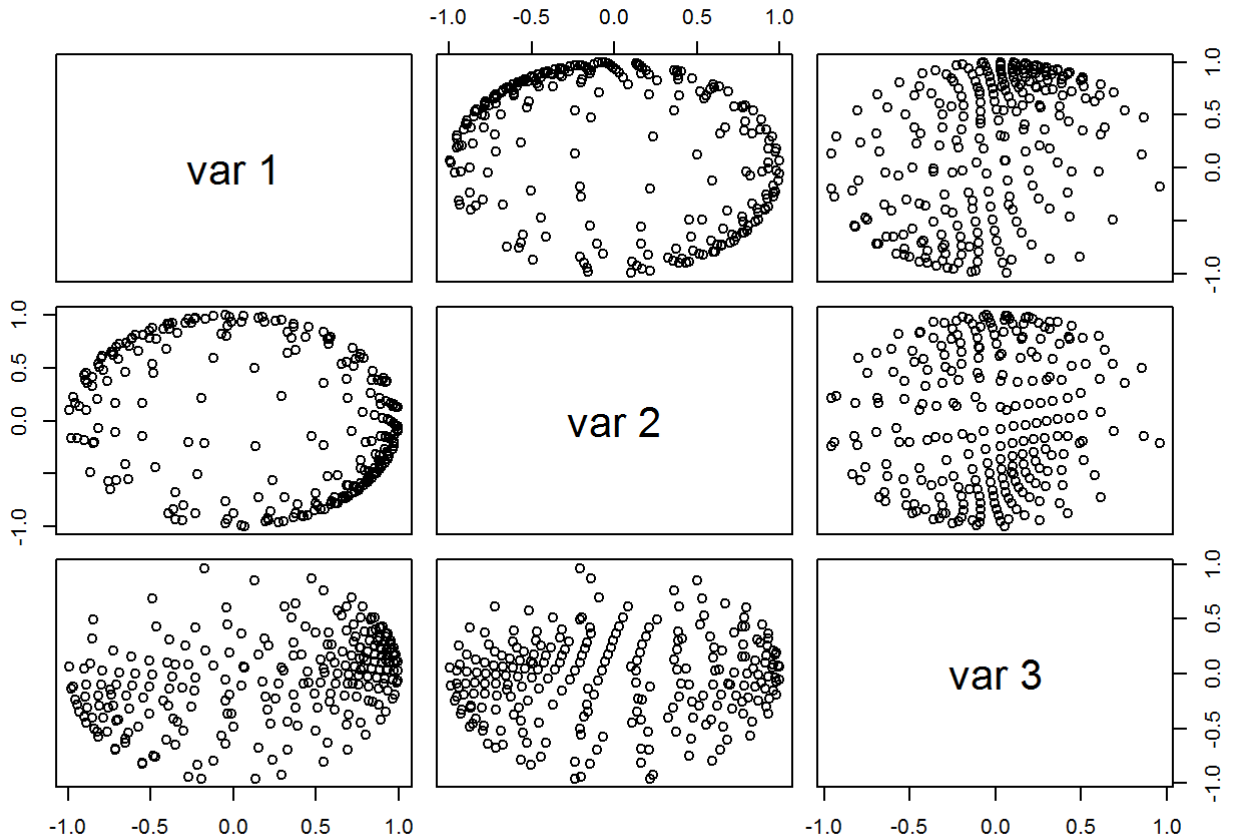

var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0


var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0


var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0


var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0





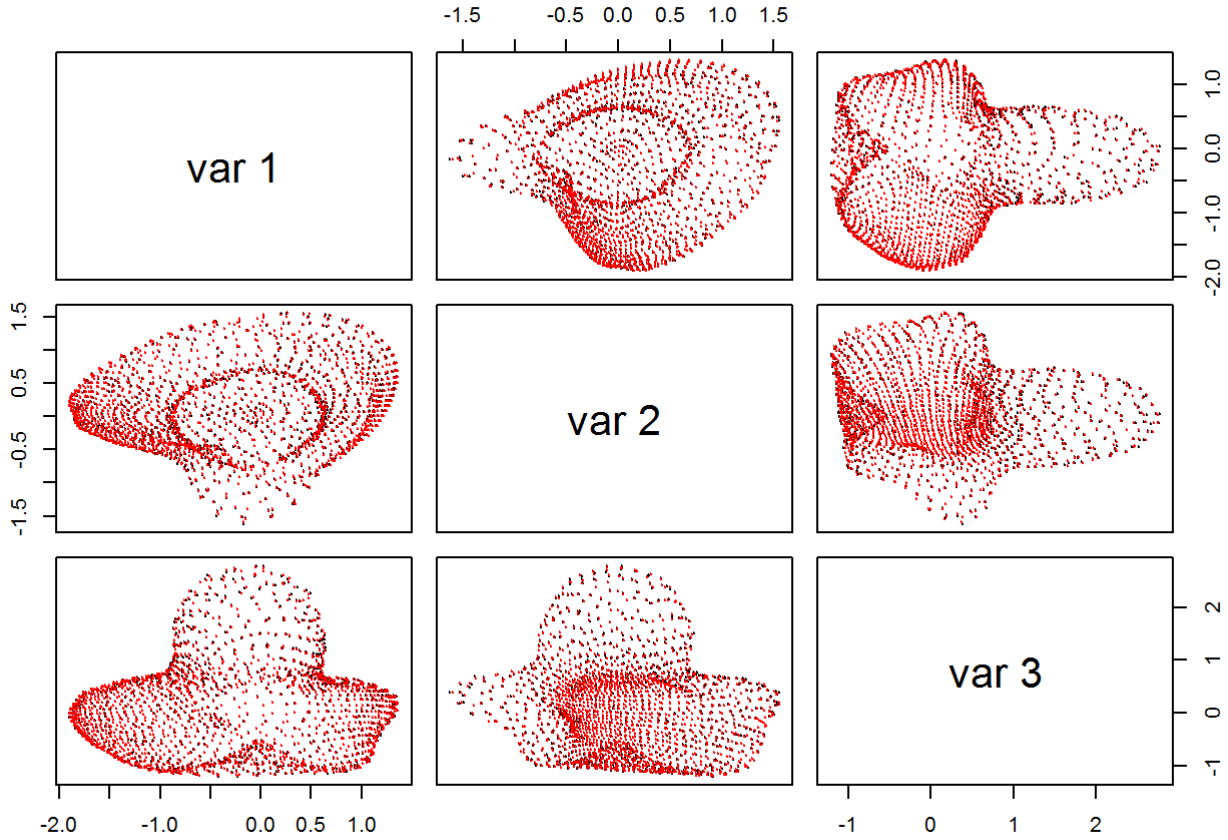
それぞれのオブジェクトについて、球面上均等点に対応する3D座標を計算し、 球面調和関数係数を算出する。

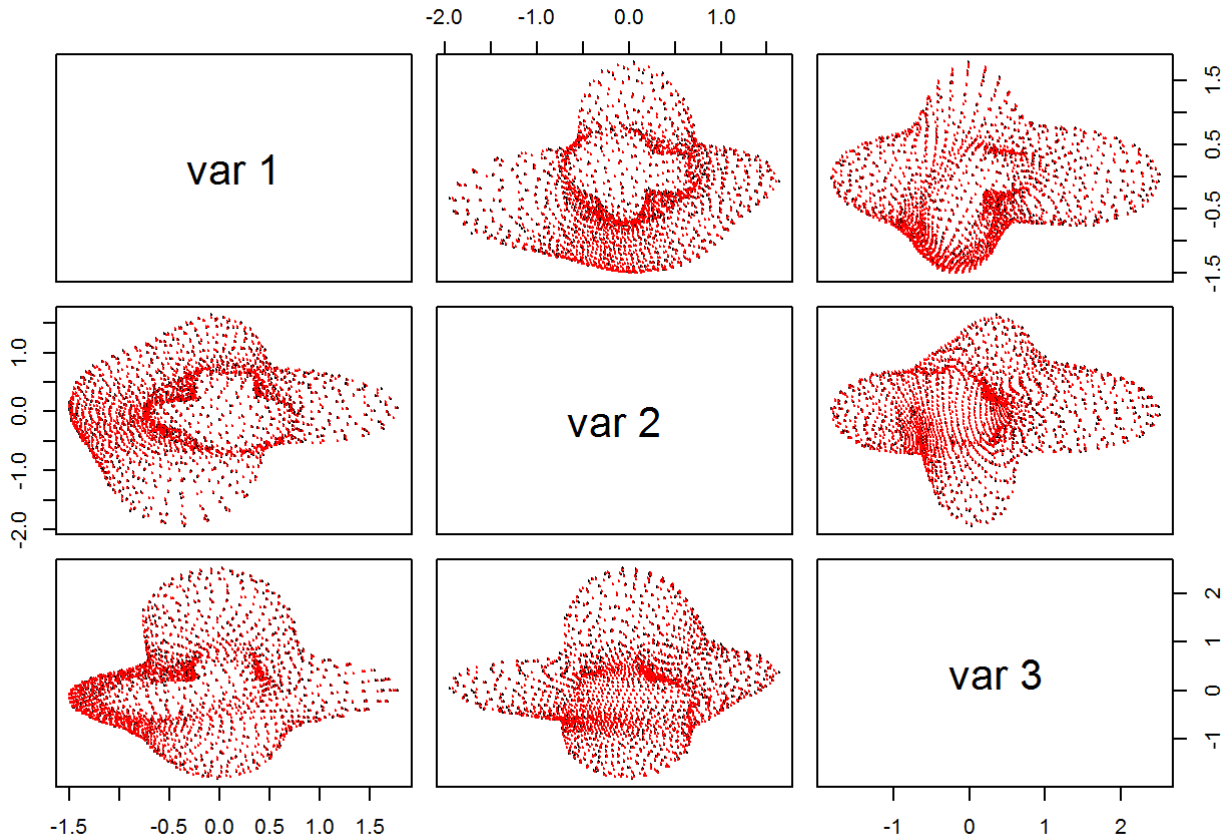
```
# 球面上均等点の3D座標
X.st <- list()
# 球面調和関数係数
coefs <- list()
for(i in 1:n.obj){
  X.st[[i]] <- my.tri.interploation(SpStAndZ$SpSt,Objs[[i]]$X,Objs[[i]]$X.sp,Objs[[i]]$tri)
  coefs[[i]] <- my.spcoef.est2(X.st[[i]],SpStAndZ$Z.inv)
}
```

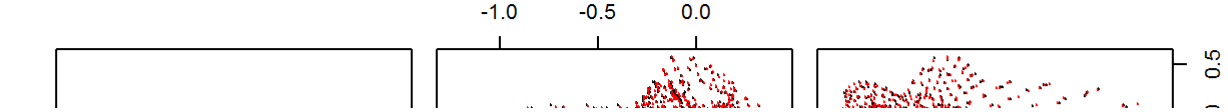
結果の確かめ

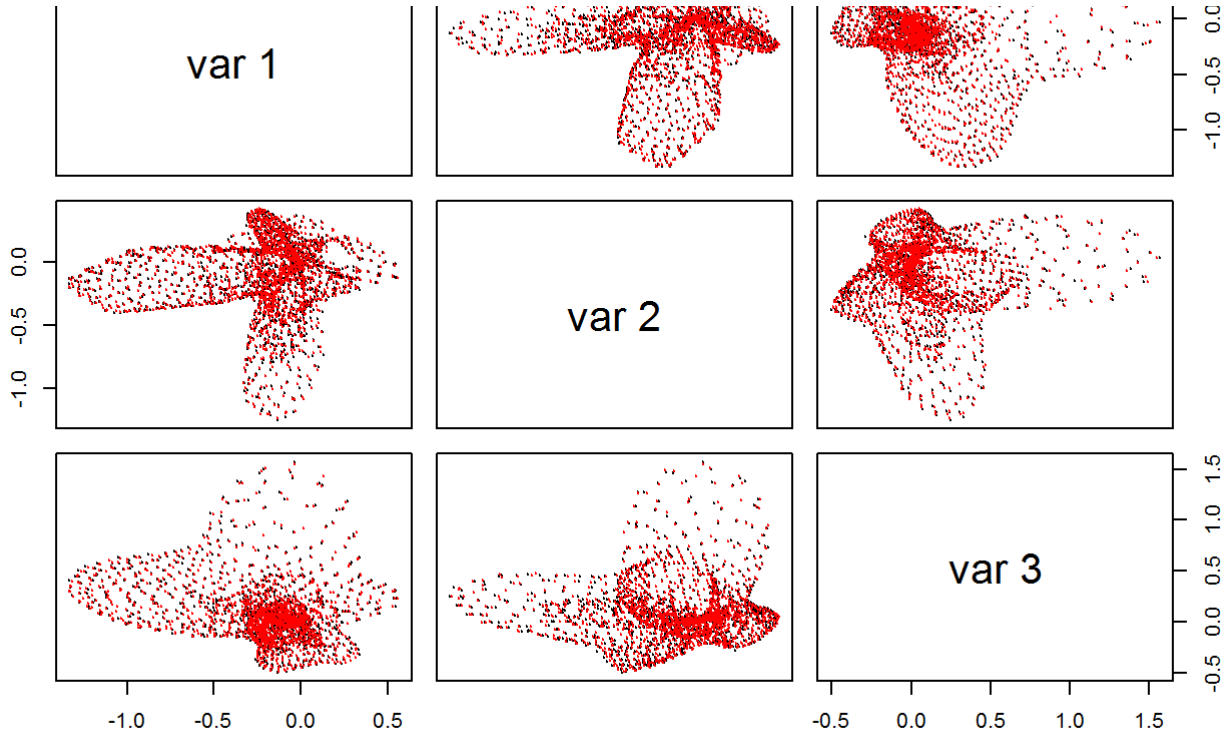
```
X.reproduced <- list()
for(i in 1:n.obj){
  X.reproduced[[i]] <- my.coef2shape2(coefs[[i]],SpStAndZ$Z)
}
```

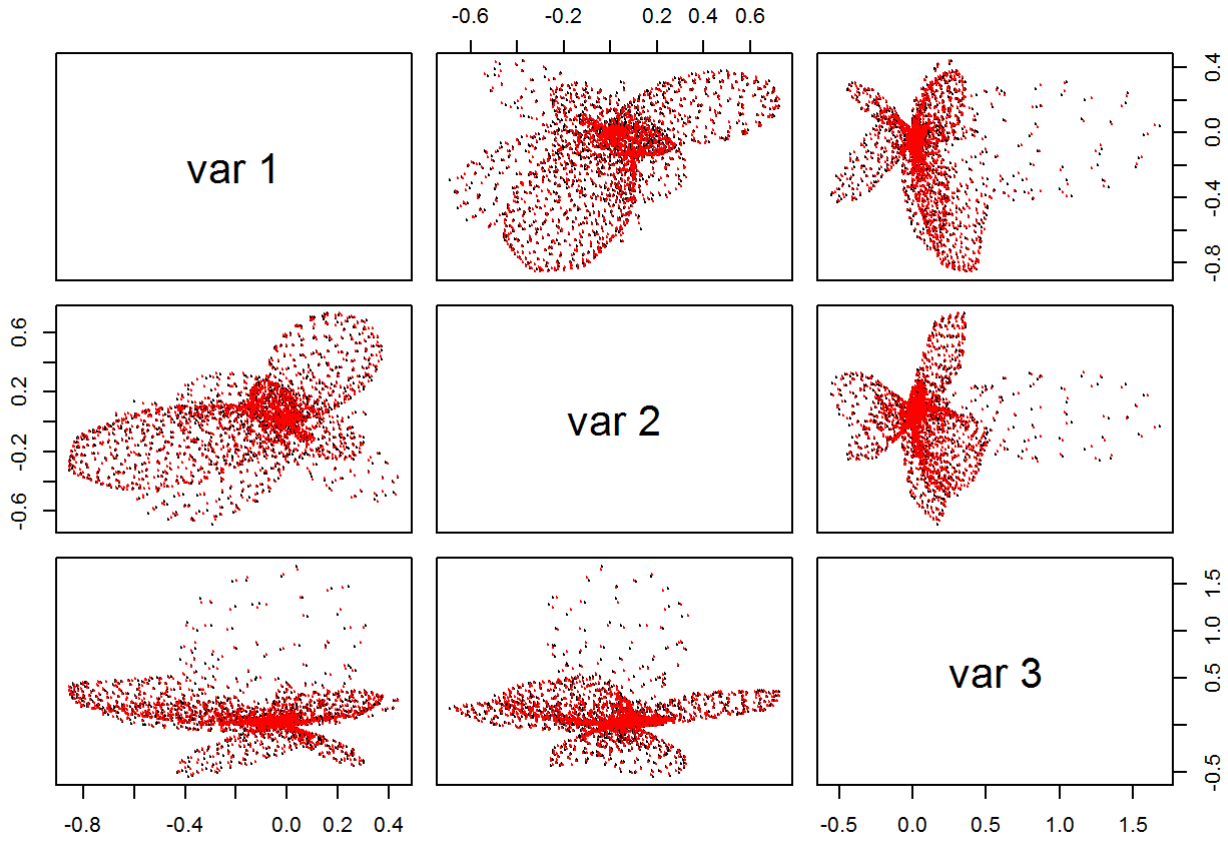
```
for(i in 1:n.obj){
  pairs(rbind(X.st[[i]],Re(X.reproduced[[i]])),col=rep(1:2,each=length(X.st[[i]][,1])),pch=20,cex=0.1)
}
```

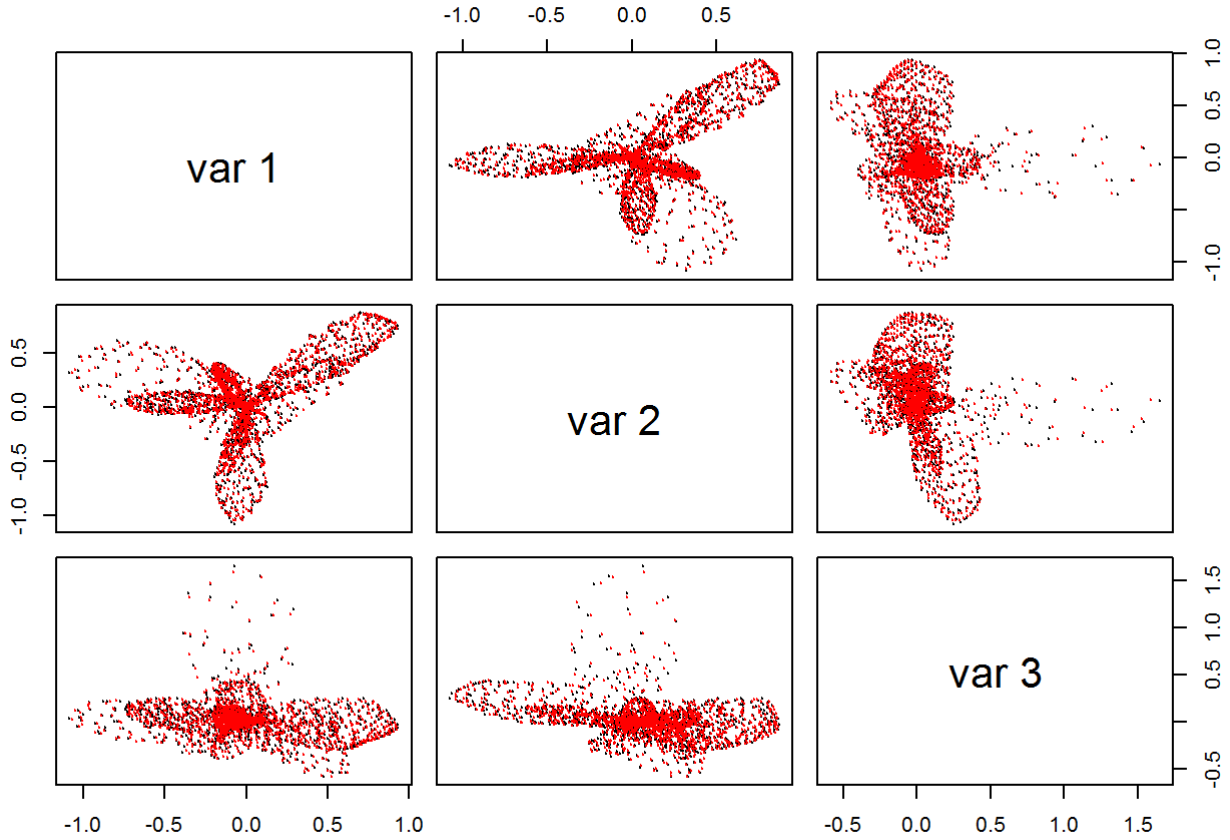

var 1
var 2
var 3
-1.5 -0.5 0.0 0.5 1.0 1.5
-2.0 -1.0 0.0 0.5 1.0
-1.5 -0.5 0.5 1.5
-2.0 -1.0 0.0 1.0
-1 0 1 2

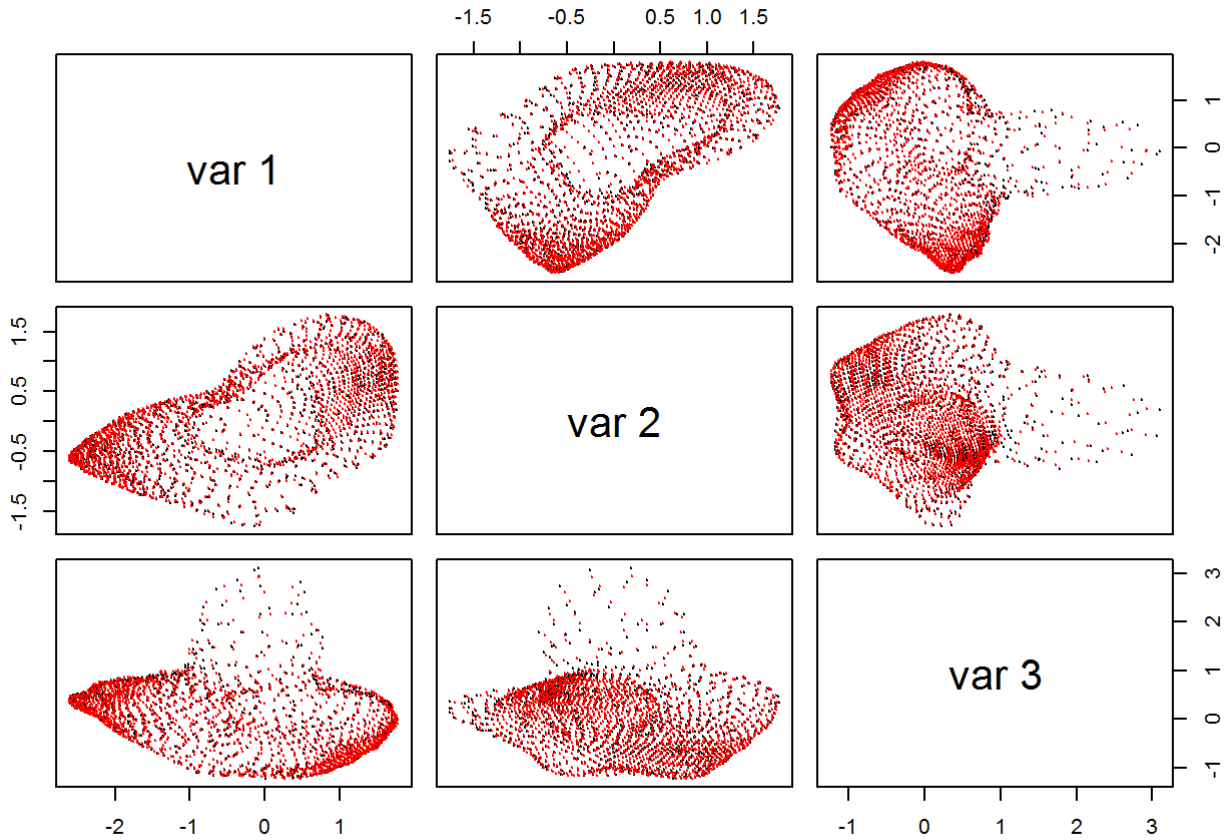

var 1
var 2
var 3
-2.0 -1.0 0.0 1.0
-1.5 -0.5 0.0 0.5 1.0 1.5
-2.0 -1.0 0.0 1.0
-1.5 -0.5 0.5 1.5
-1 0 1 2

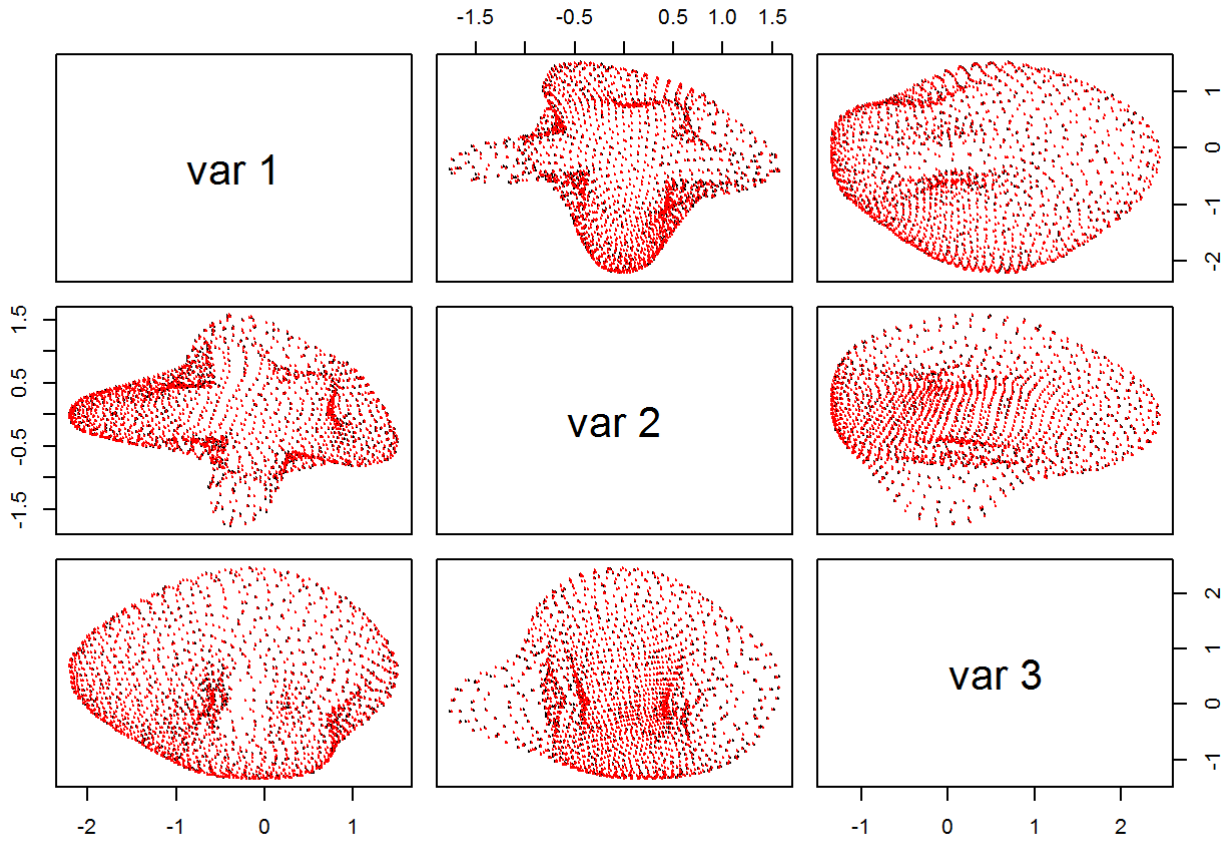

-1.0 -0.5 0.0
0.5

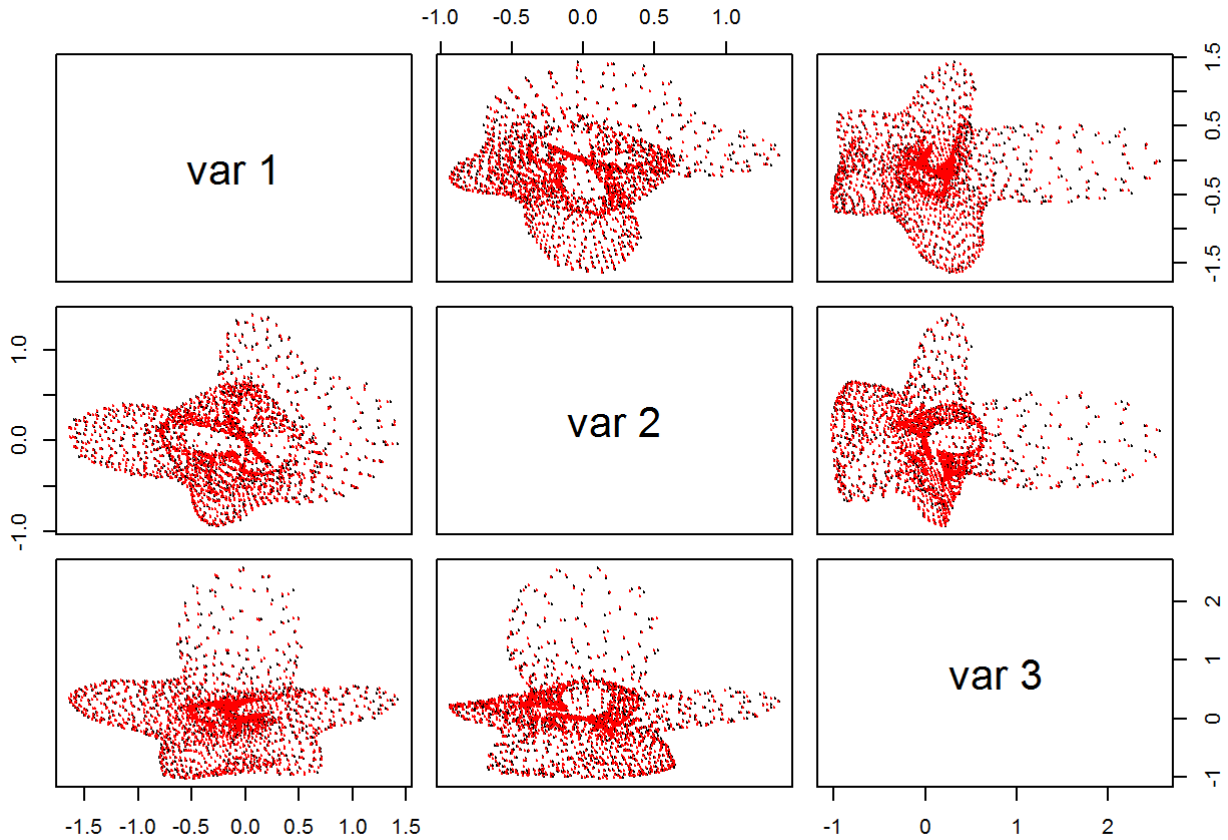

var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0
1.5

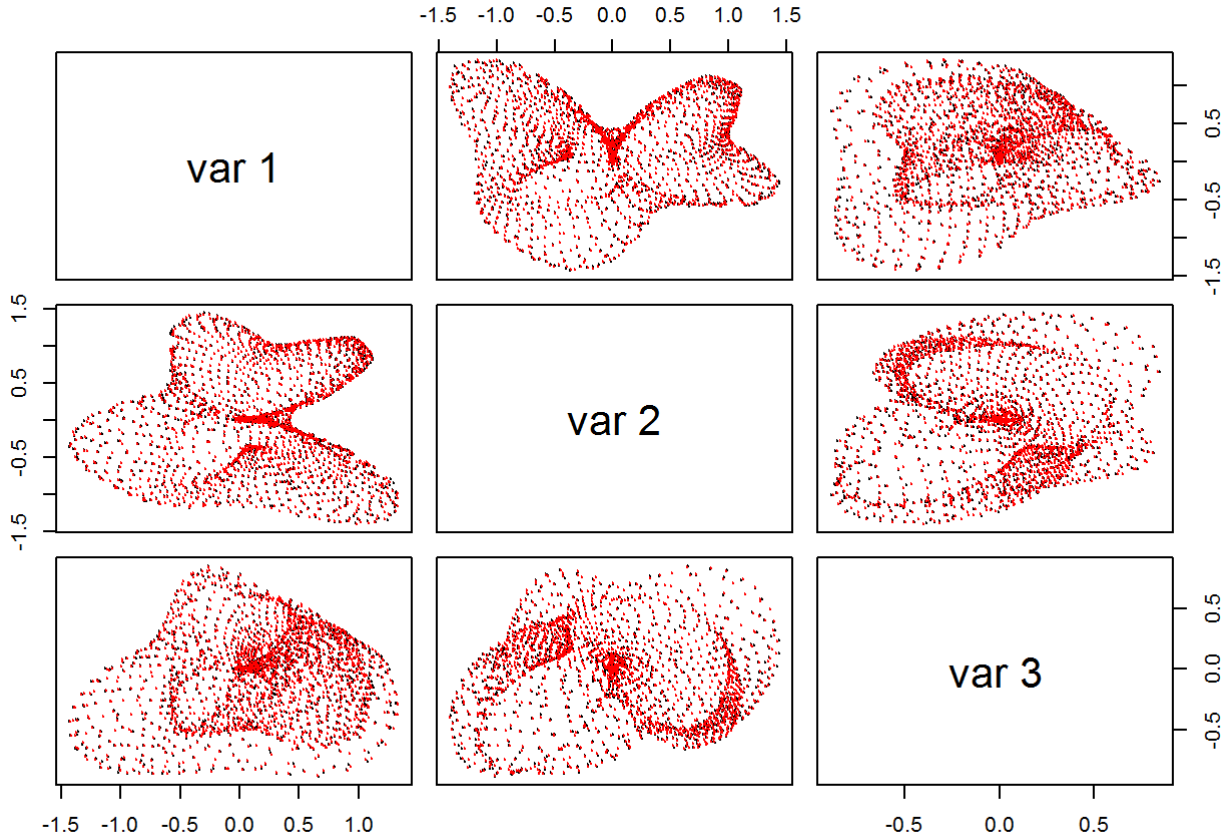

var 1
var 2
var 3
-0.6
-0.2
0.2
0.4
0.6
-0.8
-0.4
0.0
0.2
0.4
0.6
-0.5
0.0
0.5
1.0
1.5

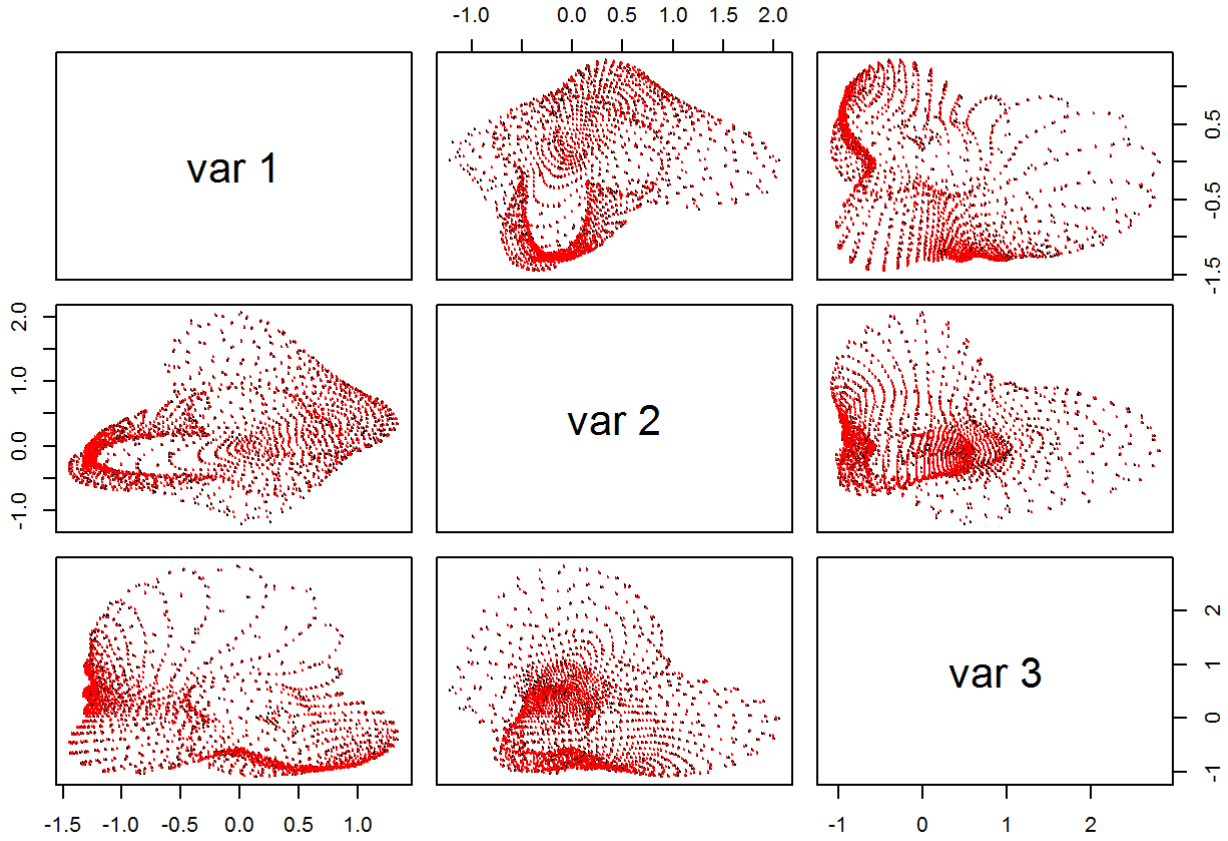

var 1
var 2
var 3
-1.0
-0.5
0.0
0.5
1.0
1.5


var 1
var 2
var 3
-1.5
-0.5
0.5
1.0
1.5
-2
-1
0
1
2
3

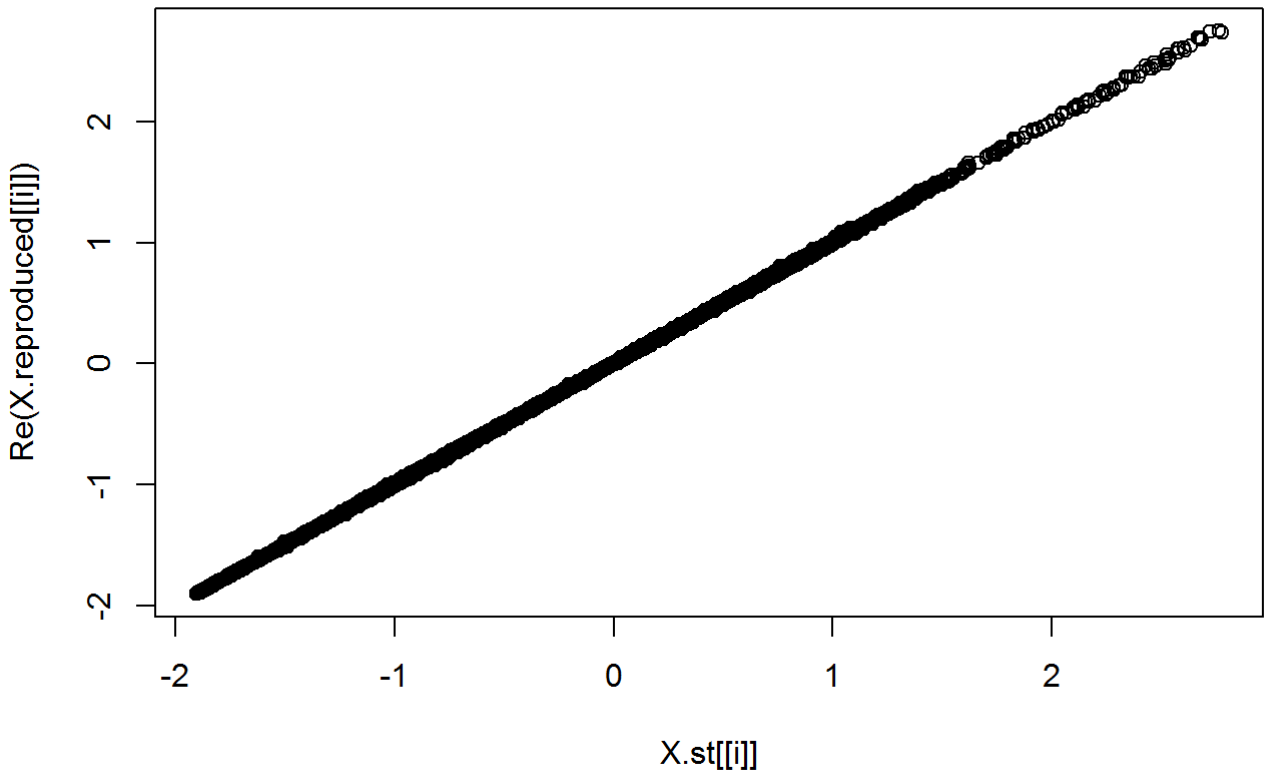

var 1
var 2
var 3
-1.5 -0.5 0.5 1.0 1.5
-2 -1 0 1
-1.5 -0.5 0.5 1.5
-2 -1 0 1
-1 0 1 2
-2 -1 0 1
-1 0 1 2

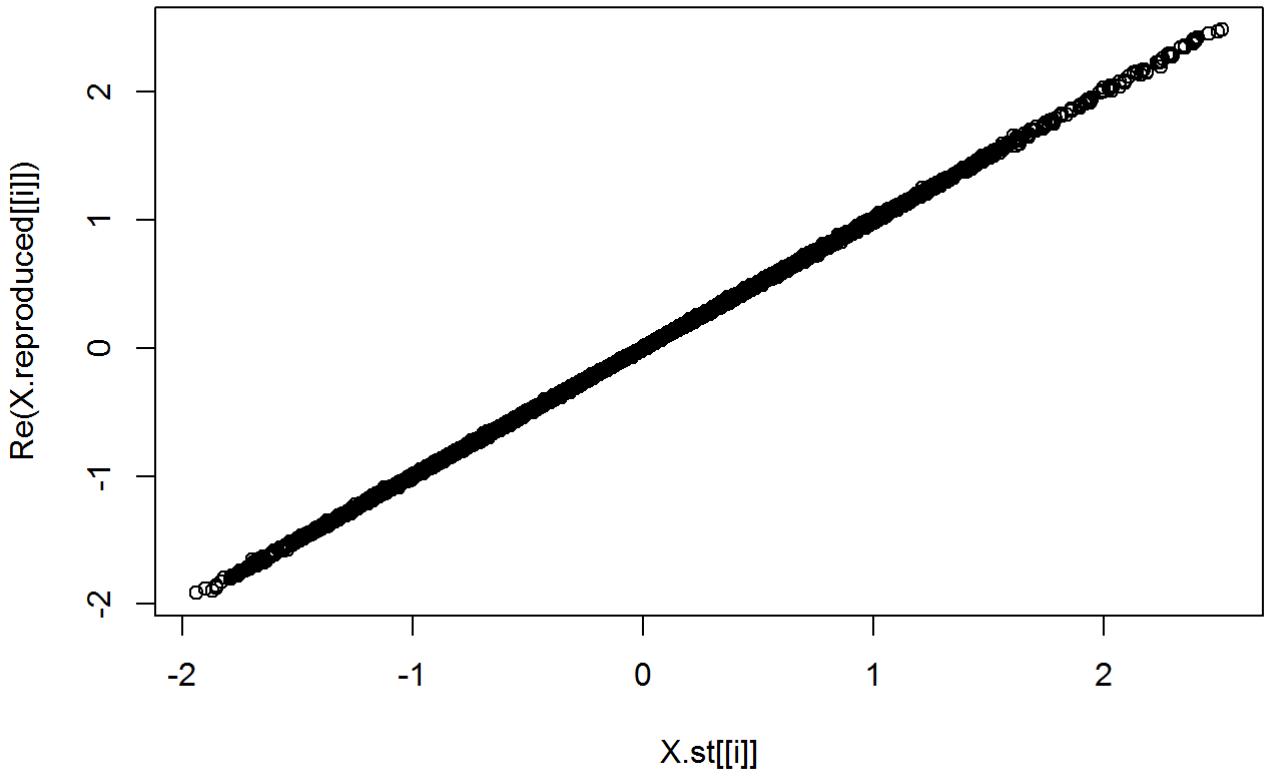

var 1
var 2
var 3
-1.0 -0.5 0.0 0.5 1.0
-1.5 -0.5 0.5 1.5
-1.0 0.0 1.0
-1.5 -1.0 -0.5 0.0 0.5 1.0 1.5
-1 0 1 2
-1 0 1 2

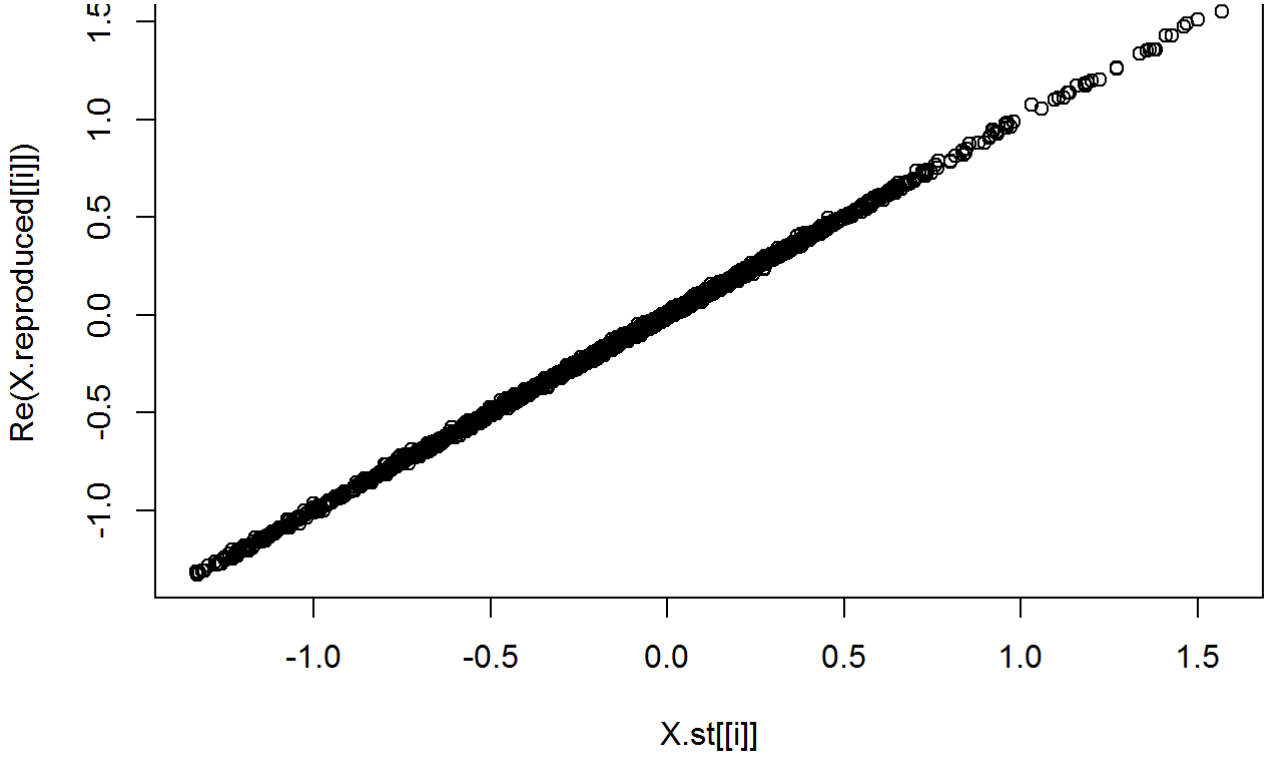


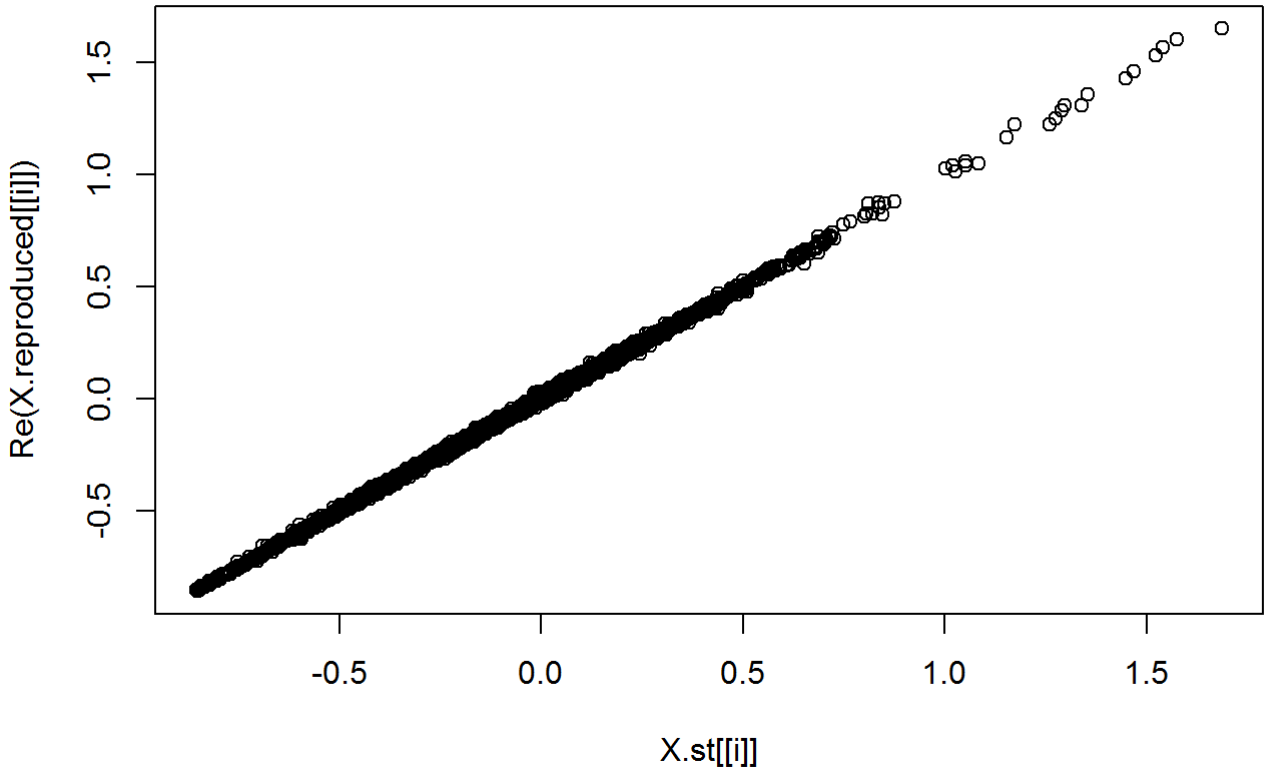


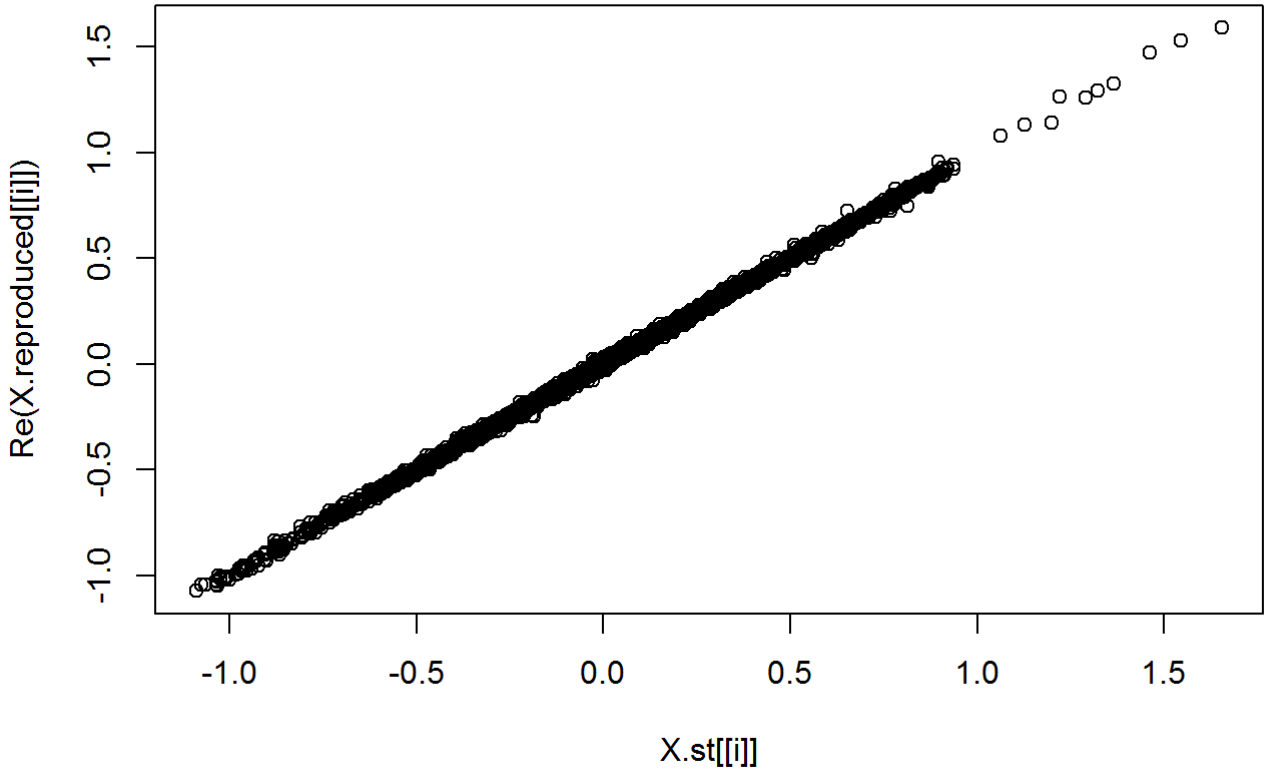

```
for(i in 1:n.obj){
  plot(X.st[[i]],Re(X.reproduced[[i]]))
}
```

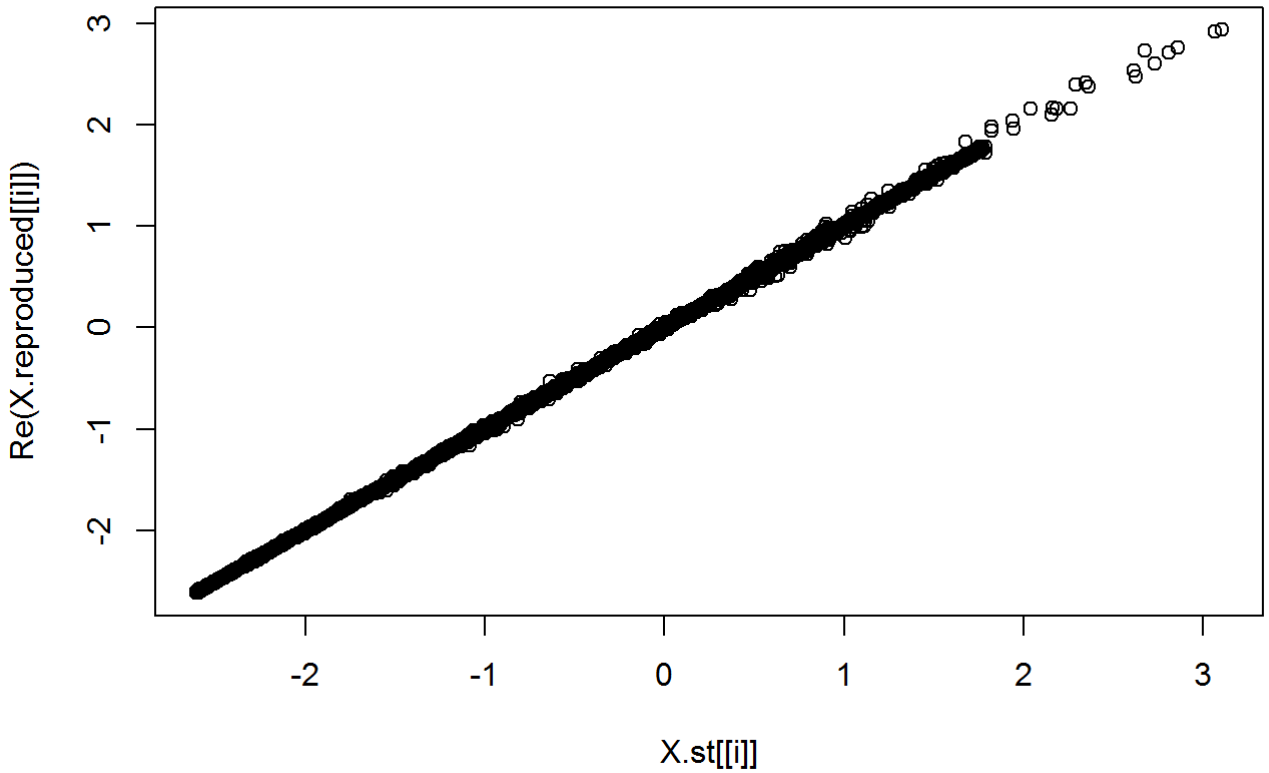

Re(X.reproduced[[i]])
-2
-1
0
1
2
X.st[[i]]

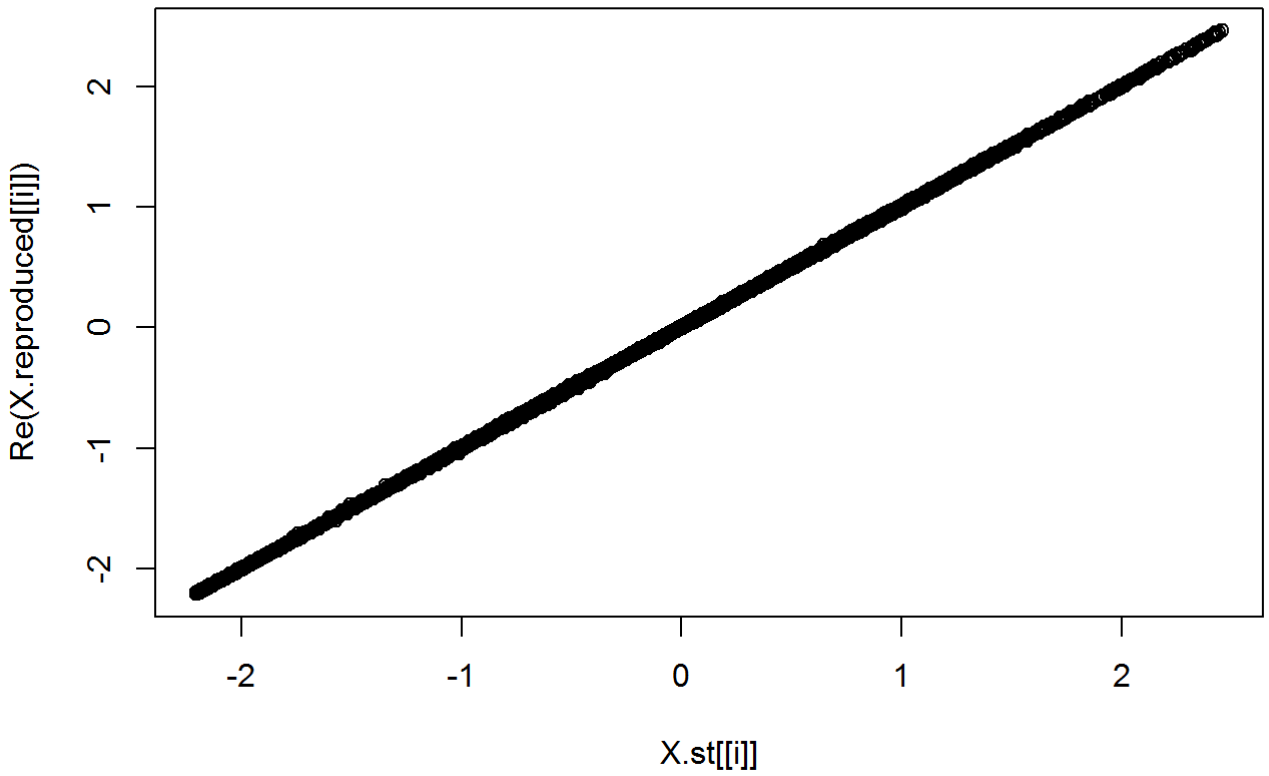

Re(X.reproduced[[i]])
-2
-1
0
1
2
X.st[[i]]

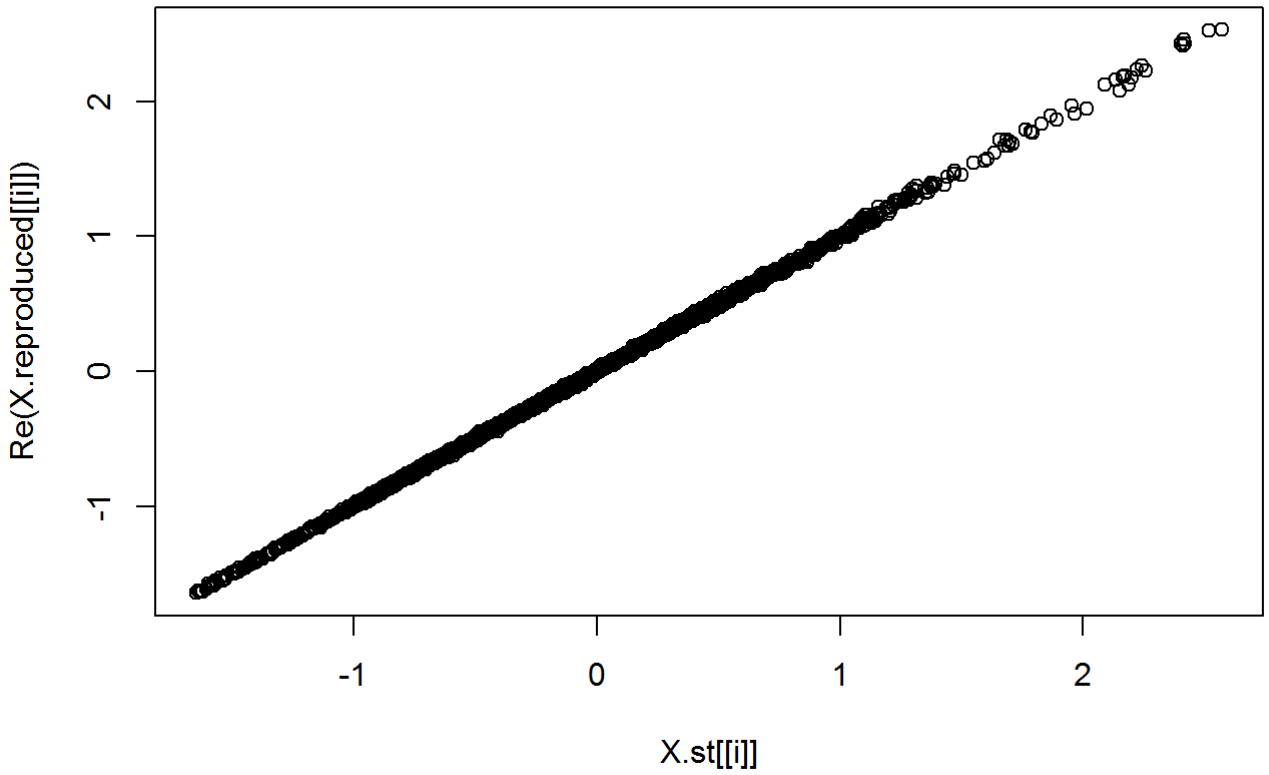

Re(X.reproduced[[i]])
X.st[[i]]
-1.0
-0.5
0.0
0.5
1.0
1.5

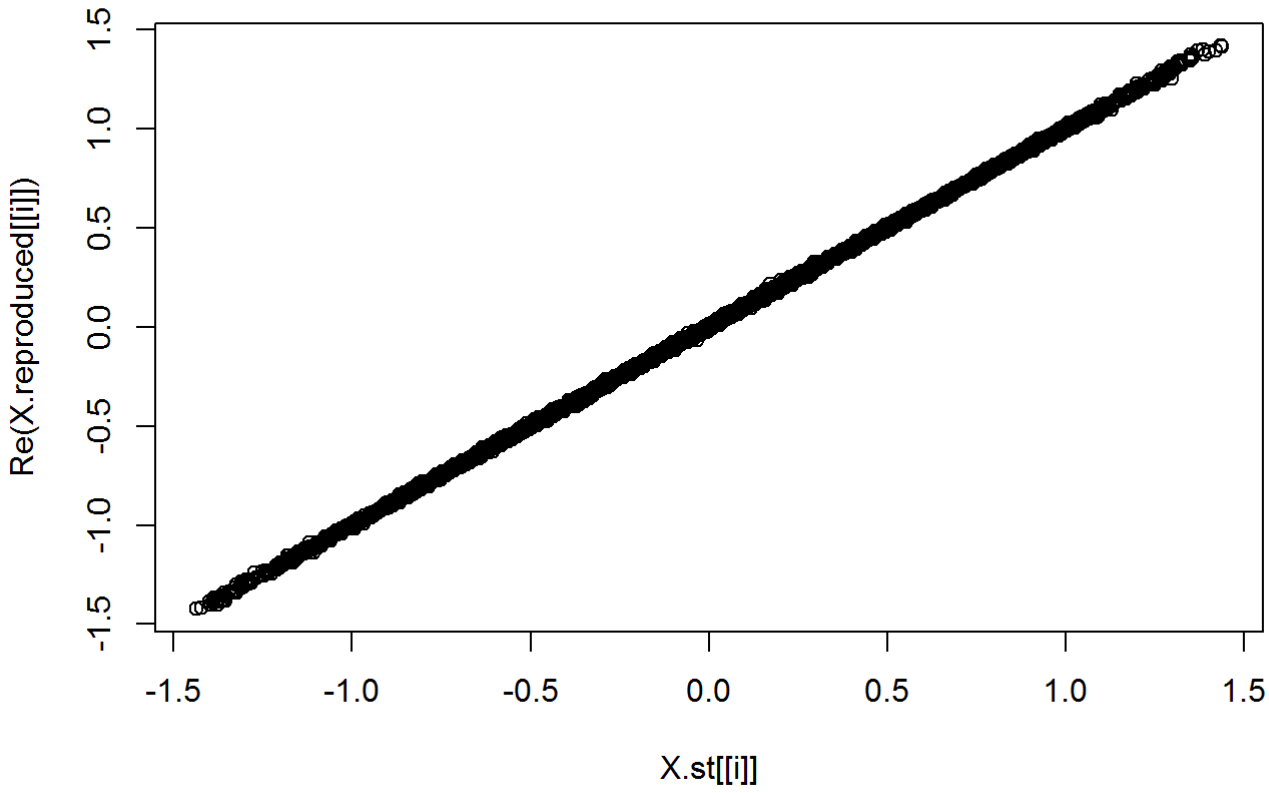

Re(X.reproduced[[i]])
X.st[[i]]
-0.5
0.0
0.5
1.0
1.5

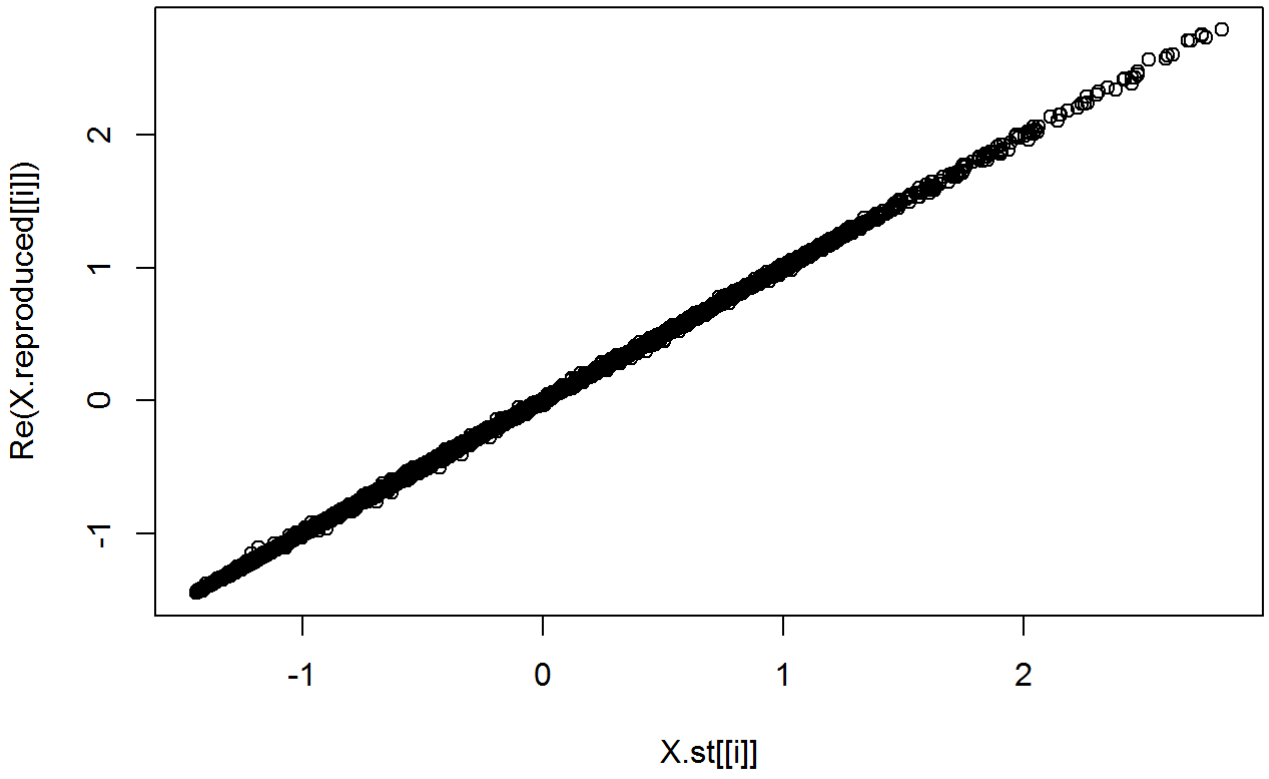

Re(X.reproduced[[i]])
-1.0
-0.5
0.0
0.5
1.0
1.5
X.st[[i]]


Re(X.reproduced[[i]])
-2
-1
0
1
2
3
X.st[[i]]


Re(X.reproduced[[i]])
-2
-1
0
1
2
X.st[[i]]


Re(X.reproduced[[i]])
-1
0
1
2
X.st[[i]]


Re(X.reproduced[[i]])
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
X.st[[i]]


Re(X.reproduced[[i]])
-1
0
1
2
X.st[[i]]