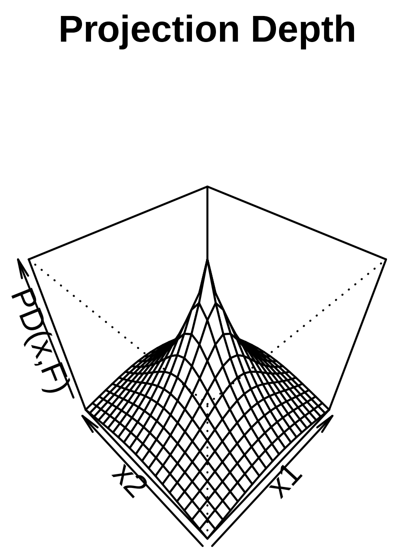
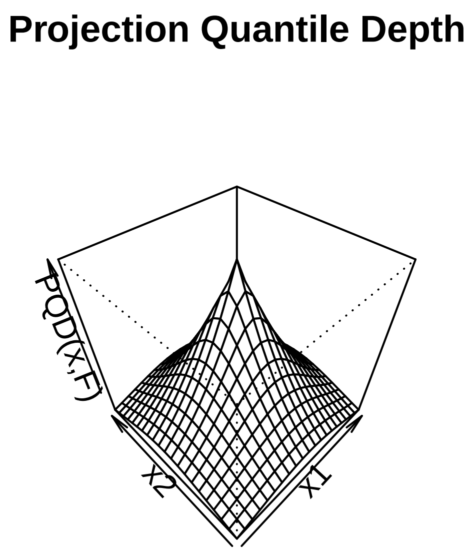
PQDSim.R

Subho

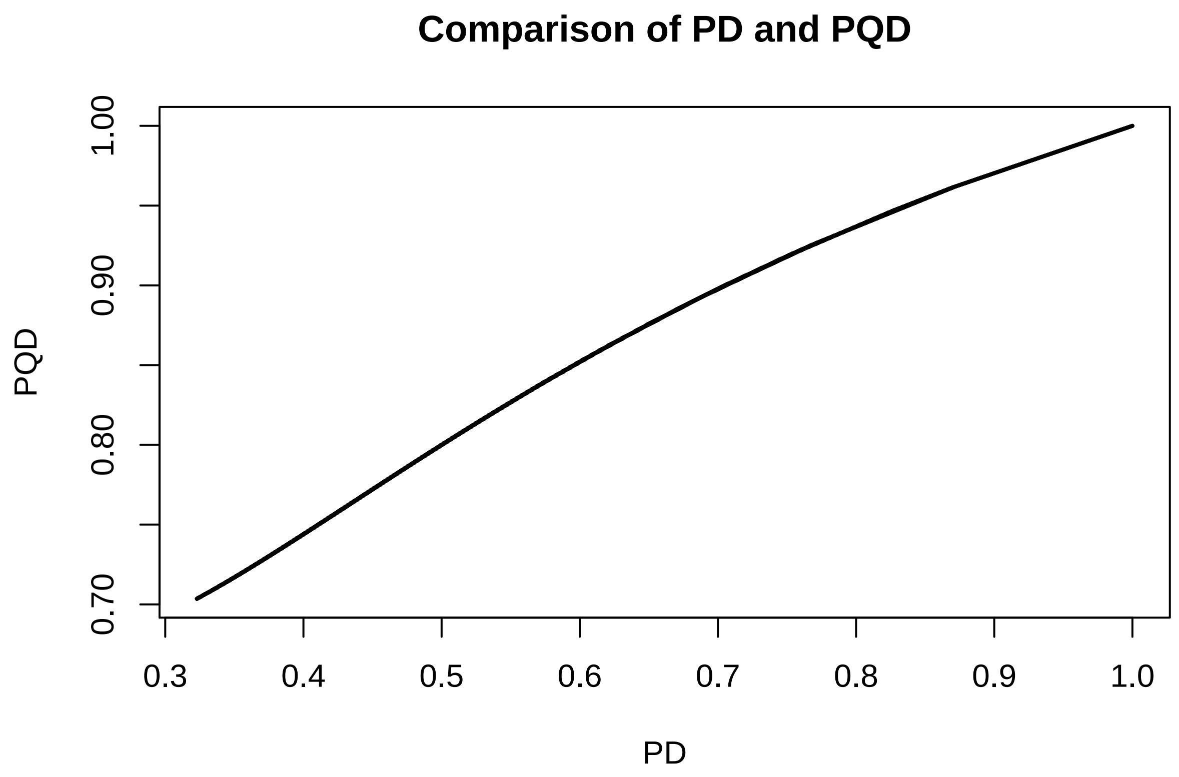
Mon Oct 27 04:22:41 2014

```r
## Initial simulations of Projection quantile depth and comparisions with projection depth

## Functions
## function to generate from multivariate normal
my.mvrnorm = function(n, mu, Sigma){
  p = length(mu)
  # compute square root of covariance matrix
  eo=eigen(Sigma, symmetric=TRUE)
  sigma.sqrt=eo$vec%*%diag(eo$val^0.5)%*%t(eo$vec)

  # generate random normals from runif by box-muller transform
  rnorm.vec = sqrt(-2*log(runif(n*p)))*cos(2*pi*runif(n*p))

  # generate sample matrix
  sample.matrix = matrix(rep(mu, n), nrow=n, byrow=T) +
    matrix(rnorm.vec, nrow=n, ncol=p)%*%sigma.sqrt
  return(sample.matrix)
}

# define grid of points
pts = seq(-1,1,by=.1)
lengrid = length(pts)
xcoord = rep(pts, rep(lengrid,lengrid))
ycoord = rep(pts, lengrid)
xygrid = cbind(xcoord,ycoord)
rm(xcoord,ycoord)

# Bivariate standard normal
# projection depth
norm.vec = sqrt(xygrid[,1]^2+xygrid[,2]^2)
cN = qnorm(.75)
pd.vec = cN/(cN+norm.vec)
# projection quantile depth
pqd.vec = 2/(1+2*pnorm(norm.vec))

par(mfrow=c(1,2))
persp(pts, pts, matrix(pd.vec, nrow=lengrid, byrow=T),
      main="Projection Depth",
      xlab="x1", ylab="x2", zlab="PD(x,F)",
      theta=-45, phi=45)
persp(pts, pts, matrix(pqd.vec, nrow=lengrid, byrow=T),
      main="Projection Quantile Depth",
      xlab="x1", ylab="x2", zlab="PQD(x,F)",
      theta=-45, phi=45)
```



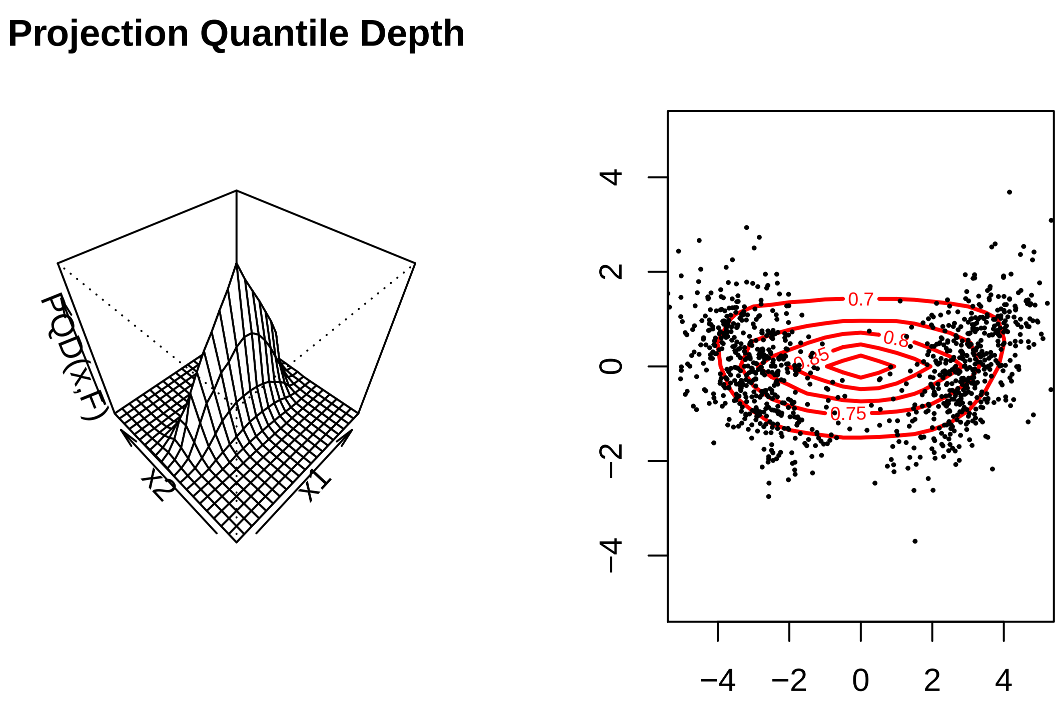


```r
par(mfrow=c(1,1))

# comparing PD and PQD
plot(pd.vec, pqd.vec, type="l", lwd=2,
     main="Comparison of PD and PQD", xlab="PD", ylab="PQD")
```



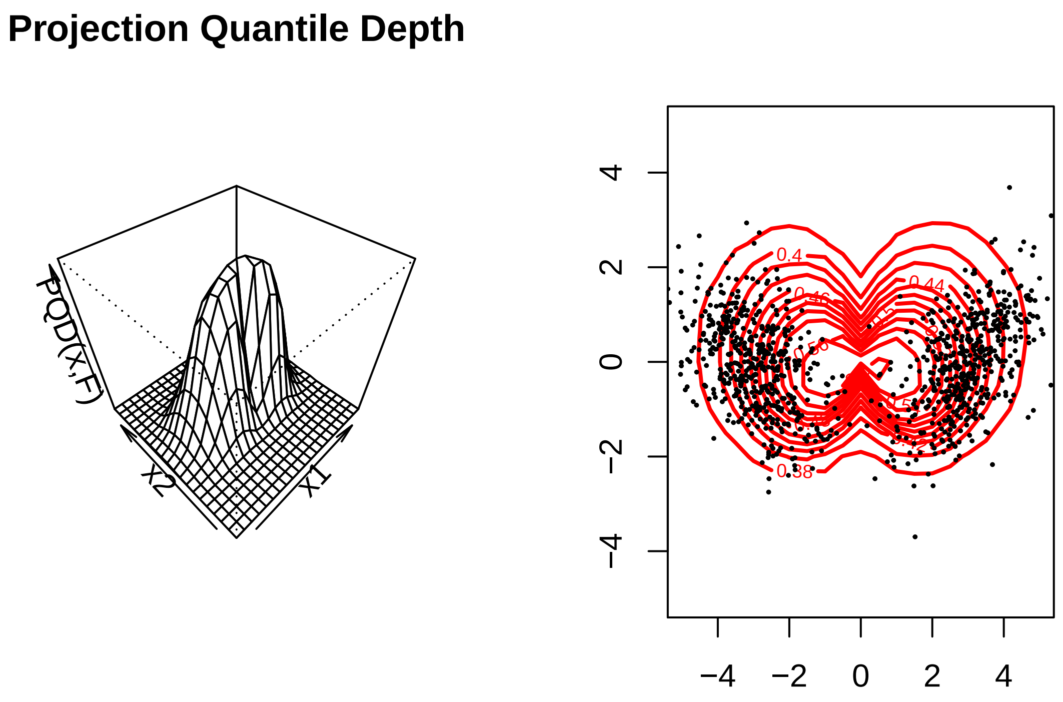
```
## Empirically calculate PQD with grid search
# Bivariate normal mixture
sig = matrix(c(1,.5,.5,1), nrow=2)
sig2 = matrix(c(1,-.5,-.5,1), nrow=2)
X1 = my.mvrnorm(500, mu=c(3,0), Sigma=sig)
X2 = my.mvrnorm(500, mu=c(-3,0), Sigma=sig2)
X = rbind(X1,X2)

# define grid of points
pts = seq(-5,5,by=.5)
lengrid = length(pts)
xcoord = rep(pts, rep(lengrid,lengrid))
ycoord = rep(pts, lengrid)
xygrid = cbind(xcoord,ycoord)
rm(xcoord,ycoord)

# my PQD
npt = dim(xygrid)[1]
Fuxu.mat = matrix(0, nrow=npt, ncol=100)
for(iu in 1:100){
  u = rnorm(2); u = u/sqrt(sum(u^2))
  uecdf = ecdf(X%*%u)
  Fuxu.mat[,iu] = uecdf(xygrid%*%u)
}
EPQD.vec = 1/(1+apply(abs(Fuxu.mat-.5), 1, max))
```

```
par(mfrow=c(1,2))
persp(pts, pts, matrix(EPQD.vec, nrow=lengrid, byrow=T),
      main="Projection Quantile Depth",
      xlab="x1", ylab="x2", zlab="PQD(x,F)",
      theta=-45, phi=45)

# contour plot
z = contour(pts, pts, matrix(EPQD.vec, nrow=lengrid, byrow=T),
        lwd=2, col="red")
points(X, pch=19, cex=.2)
```



```
par(mfrow=c(1,1))

# kodu da's PQD
npt = dim(xygrid)[1]
Fuxu.vec = rep(0,npt)
for(i in 1:npt){
  iu = xygrid[i,]; norm.iu = sqrt(sum(iu^2))
  if(norm.iu>0){
    uecdf = ecdf(X%*%iu)
    Fuxu.vec[i] = uecdf(xygrid[i,]%*%iu)
  }
  else{
    Fuxu.vec[i] = .5
  }
```

```r
}
D.vec = exp(-Fuxu.vec)

par(mfrow=c(1,2))
persp(pts, pts, matrix(D.vec, nrow=lengrid, byrow=T),
      main="Projection Quantile Depth",
      xlab="x1", ylab="x2", zlab="PQD(x,F)",
      theta=-45, phi=45)

# contour plot
z = contour(pts, pts, matrix(D.vec, nrow=lengrid, byrow=T),
            lwd=2, col="red")
points(X, pch=19, cex=.2)
```



```r
par(mfrow=c(1,1))
```