MVM-Assignment2

Mohana Fathollahi, Thanh Binh Viedena Vu

04/Dec/2021

1. PCA and Principal curves for ZIP numbers

Consider the ZIP number data set, from the book of Hastie et al. (2009). Read the training data set (in the file zip.train) and select only the zeros.

Dimension of data set

```
zip.train <- read.table("zip.train")
n <- dim(zip.train)
n
```

```
## [1] 7291  257
```

```
plot.zip <- function(x,use.first=FALSE){
  x<-as.numeric(x)
  if (use.first){
    x.mat <- matrix(x,16,16)
  }else{
    x.mat <- matrix(x[-1],16,16)
  }
  image(1:16,1:16,x.mat[,16:1],
        col=gray(seq(1,0,l=12)))
  if (!use.first) title(x[1])
}
```

Dimension of Zeros in data set

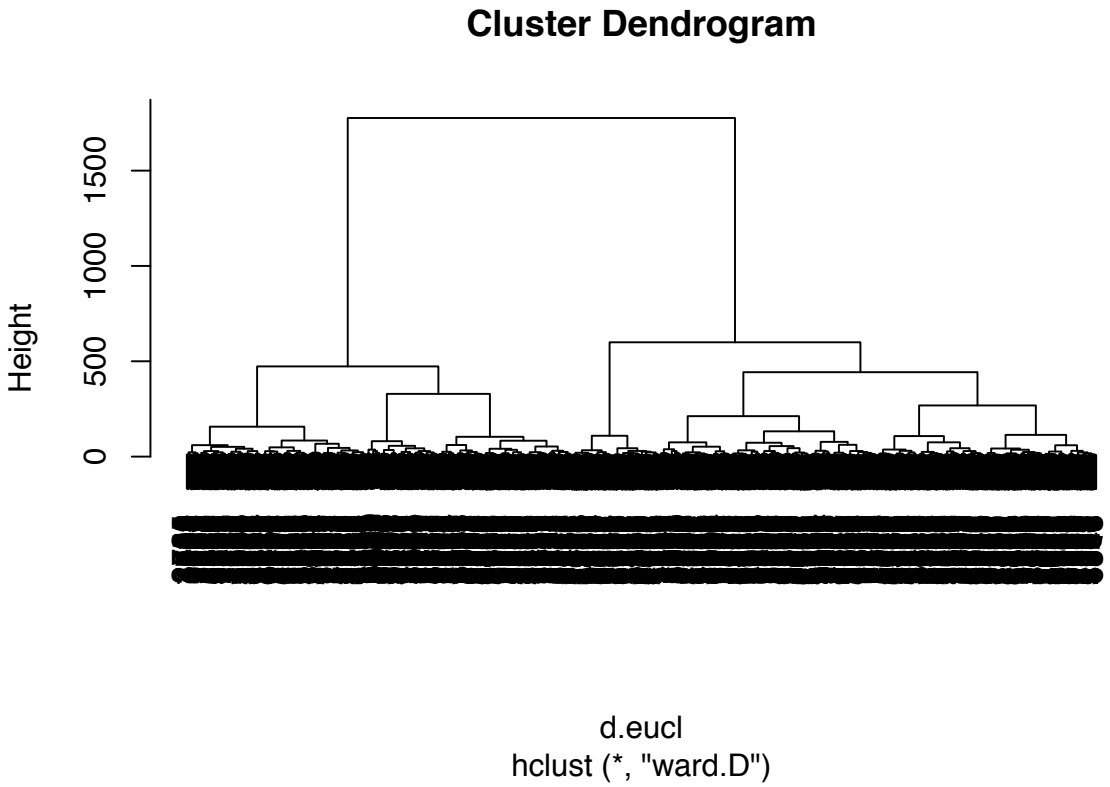
```
zip.train.0 <-zip.train[zip.train[,'V1']==0,]
dim.zip.0 <- dim(zip.train.0)
dim.zip.0
```

```
## [1] 1194  257
```

1a) Using Hierarchical Clustering

Do a hierarchical clustering of these data using the ward.D method, plot the resulting dendogram and cut it into k = 4 clusters.

```
set.seed(123)
d.eucl <-dist(zip.train.0[,-1])
zip.eucl <- hclust(d.eucl, method="ward.D")
plot(zip.eucl)
```



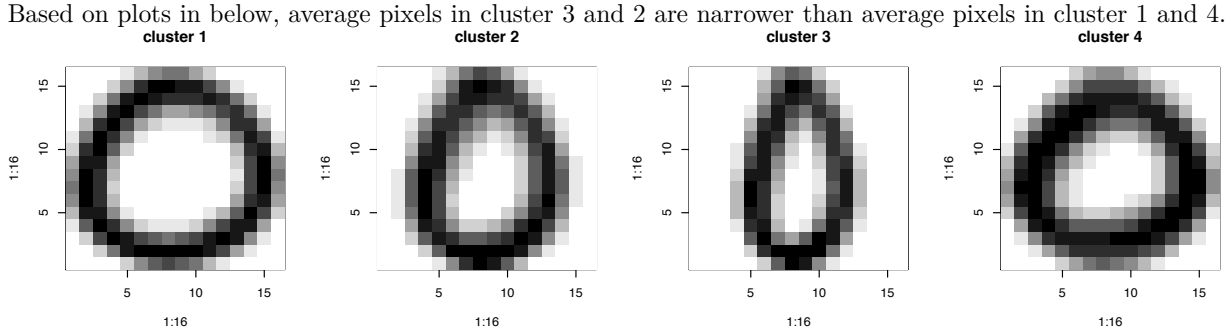

```
cutree.ward <- cutree(zip.eucl,4)
table(cutree.ward)
```

```
## cutree.ward
##   1   2   3   4
## 236 574 115 269
```

As we can see, 48% of zeros are clustered in group 2 and 10% of zeros are clustered in group 3.

1b) Average pixels in each cluster

Based on plots in below, average pixels in cluster 3 and 2 are narrower than average pixels in cluster 1 and 4.



1c) Compute PCs and plot scatterplot of scores

Compute the principal components for this data set. Plot the scatter plot of the scores in the first two PCs, using a different color for points in different clusters.

```
zip.0.PC <- princomp(zip.train.0[,-1])
```

```
eigvals <- zip.0.PC$sdev^2
pctg.VE <- 100*eigvals/sum(eigvals)
cum.pctg.VE <- cumsum(pctg.VE)
cum.pctg.VE[1:16]
```

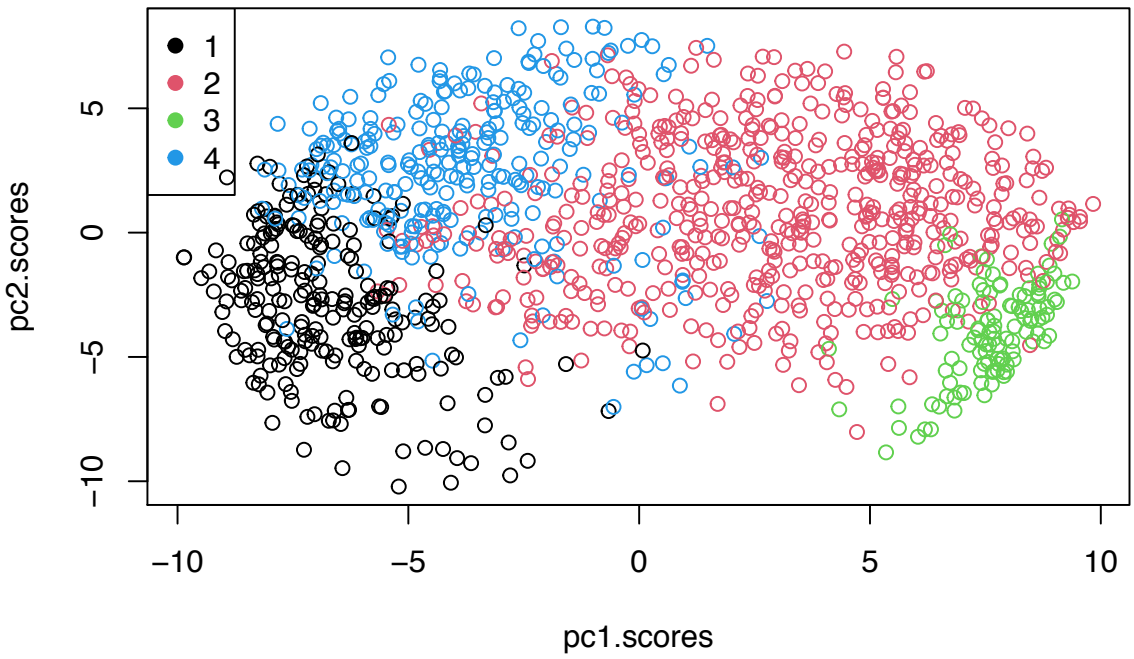
```
##    Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7    Comp.8
## 28.19983 40.92671 51.00227 56.44046 61.05261 64.97223 67.73900 70.10517
##    Comp.9   Comp.10   Comp.11   Comp.12   Comp.13   Comp.14   Comp.15   Comp.16
## 72.12687 73.87254 75.45920 76.67458 77.78173 78.85629 79.86541 80.81684
```

First two components can describe 41% of variation in data and with 16 components we can describe 80% of variation.

Extract PC.scores for first three PCs

```
zip.0.PC <- princomp(zip.train.0[,-1])
x1<-zip.0.PC$scores[,1]
x2<-zip.0.PC$scores[,2]
x3<-zip.0.PC$scores[,3]
cluster<-zip.train.0.label$cutree.ward
scores.cluster<-as.data.frame(cbind(x1,x2,x3,cluster))
```

```
plot(x1,x2,col= factor(cluster),xlab="pc1.scores",ylab = "pc2.scores")
legend("topleft",
       legend = levels(factor(cluster)),
       pch = 19,
       col = factor(levels(factor(cluster))))
```



1d) Bivariate analysis of the k clusters

For each one of the k clusters obtained above, do the following tasks: (A unique scatter plot of the scores in PC1 and PC2 should be done, over which the k densities are represented simultaneously) - Consider the bivariate data set of the scores in PC1 and PC2 of the points in this cluster. - Estimate non-parametrically the joint density of (PC1,PC2), conditional to this cluster. Use the default bandwidth values. - Represent

the estimated bivariate density using the level curve that covers the 75% of the points in this cluster.

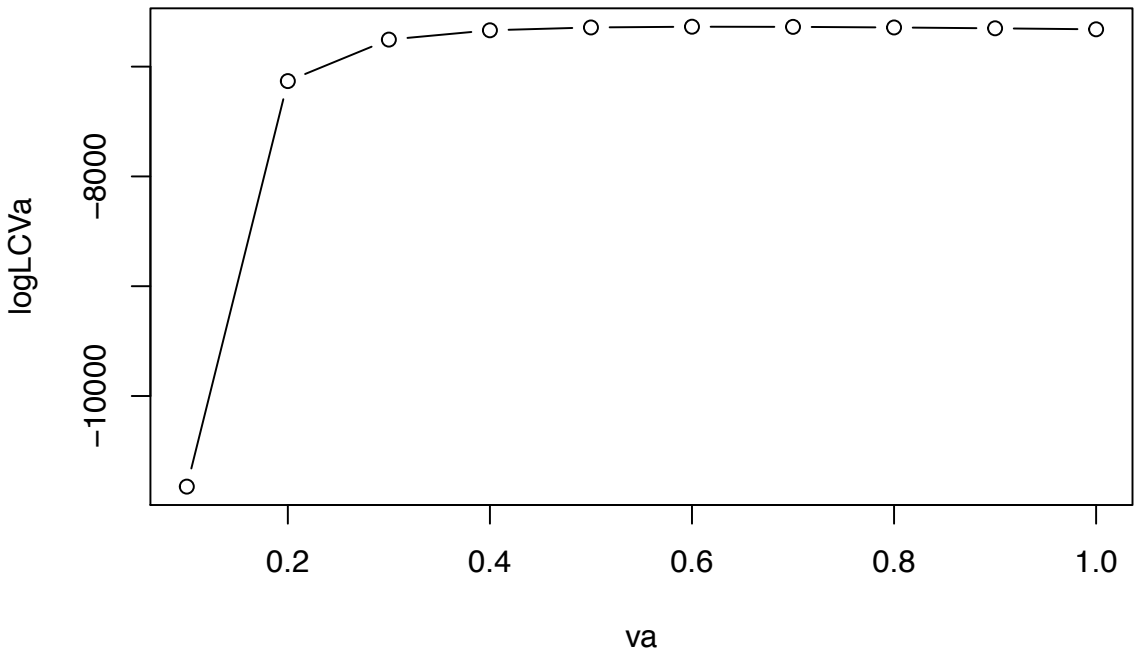
Optimum value for "a" is estimated based on maximum log-likelihood cross-validation method. With respect to below plot, a= 0.6 is an optimum value.

```
zip.cross <- zip.0.PC$scores[,1:2]
va <- seq(0.1,1,by=0.1)
na <- length(va)
logLCVa <- numeric(na)

require(sm)
```

```
## Loading required package: sm
```

```
## Package 'sm', version 2.2-5.7: type help(sm) for summary information
```

```
n <- dim(zip.cross)[1]
for (j in 1:na){
  a <- va[j]
  for (i in 1:n){
    new.point <- matrix(zip.cross[i,],ncol=2)
    f.hat.i <- sm.density(zip.cross[-i,],h=a*c(1,1),display="none",eval.grid=FALSE,
                    eval.points=new.point)$estimate
    logLCVa[j] <- logLCVa[j] + log(f.hat.i)
  }
}
plot(va,logLCVa,type="b")
```

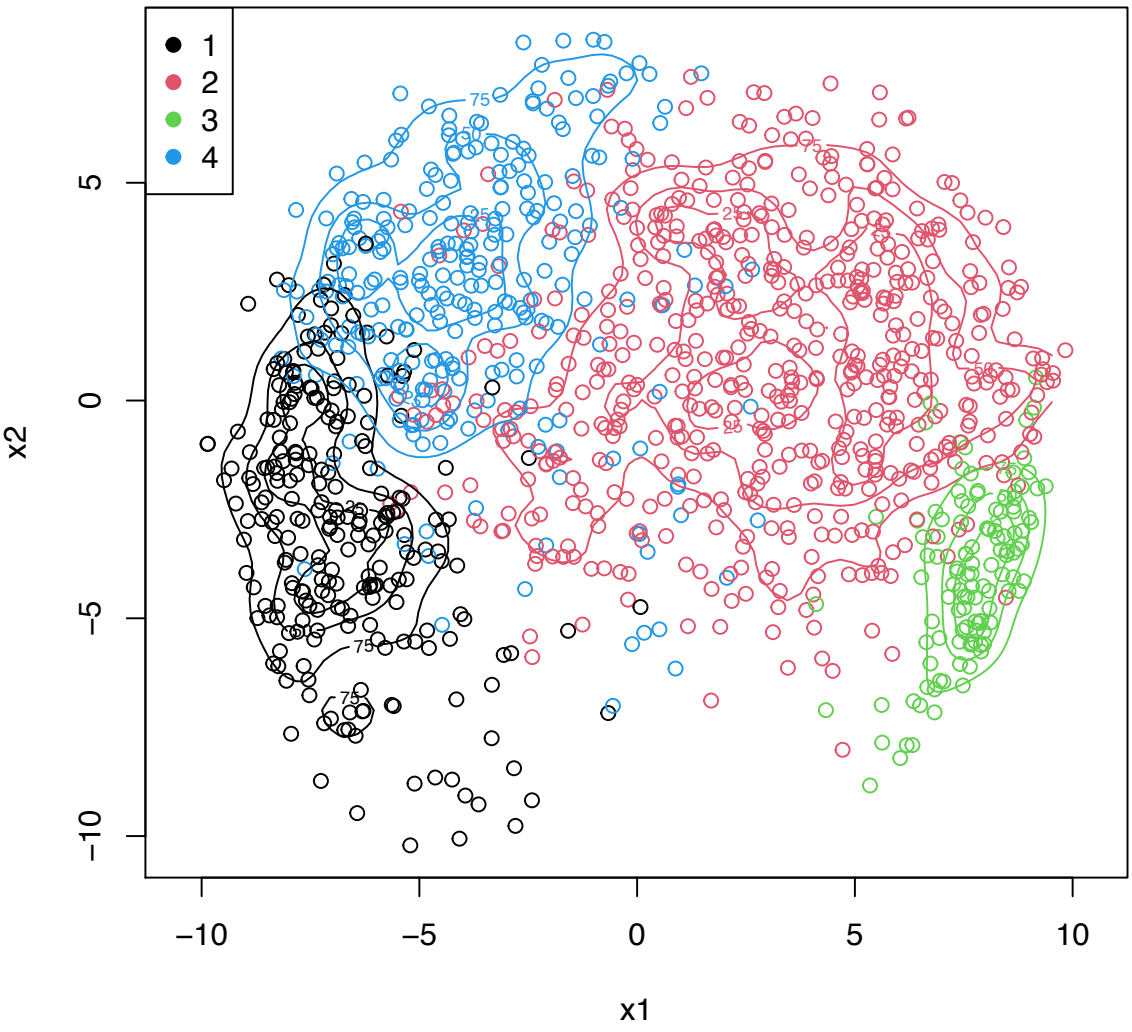



```
a.optimum <- va[which.max(logLCVa)]
a.optimum
```

```
## [1] 0.6
```

Joint density of (PC1,PC2) for c =75.

```
k<-4
plot(scores.cluster[,1:2], col=scores.cluster[,4], as=1)
for (j in 1:k){
  cl.j <- (scores.cluster[,4]==j)
  sm.density(scores.cluster[,1:2][cl.j,],h=a.optimum*c(1,1),
             display="slice",prpose=75,
             col=j, cex=4, add=TRUE)
}
legend("topleft",
       legend = levels(factor(scores.cluster[,4])),
       pch = 19,
       col = factor(levels(factor(scores.cluster[,4]))))
```



1e) Represent PC obtained from 256-dim set of zeros

Over the previous plot, represent the principal curve obtained from the 256-dimensional set of zeros using the package princurve.

```
require(princurve)
```

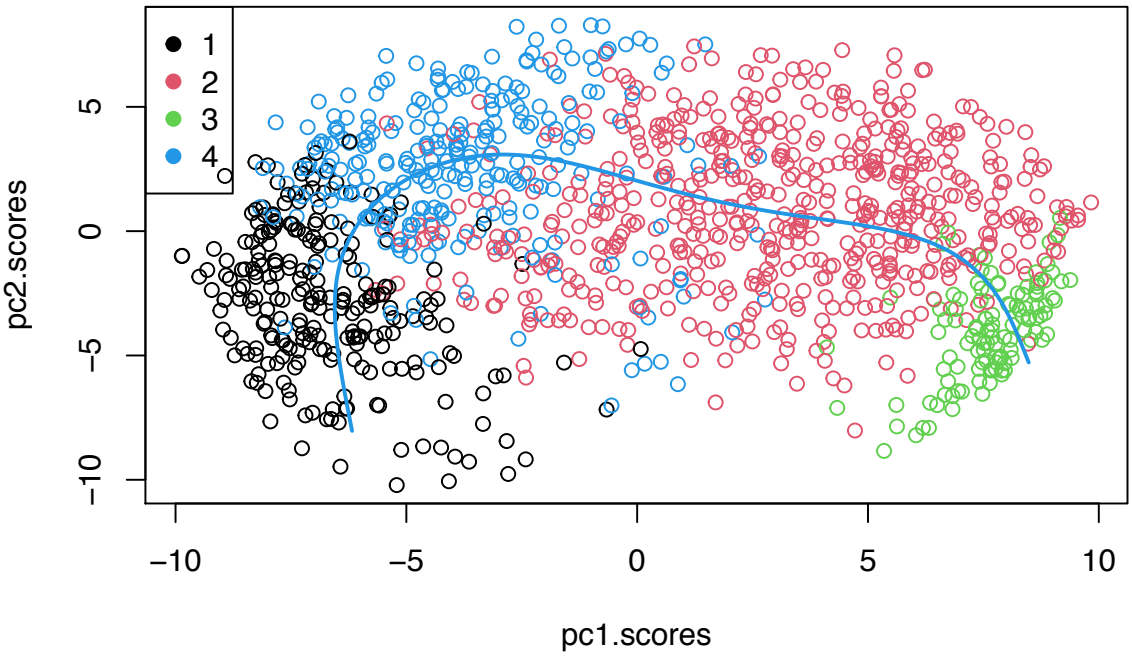
```
## Loading required package: princurve
```

```
zip.prcv.0 <- principal_curve(as.matrix(zip.train.0[,-1]))

Var.pr.cv <- var(zip.prcv.0$lambda)
Var.resid.pr.cv <- zip.prcv.0$dist/(n-1)
(pctg.VE.pr.cv <- 100*Var.pr.cv /(Var.pr.cv + Var.resid.pr.cv))
```

```
## [1] 56.02582
```

Principal curve explained 56% of variability in data. With using first three components, we plotted principal curves for 3 combinations of PCs and ords.

```
proj.pc.0 <- predict(zip.0.PC, zip.prcv.0$s)

plot(x1,x2,col= factor(cluster),xlab="pc1.scores",ylab = "pc2.scores")
legend("topleft",
       legend = levels(factor(cluster)),
       pch = 19,
       col = factor(levels(factor(cluster))))
lines(proj.pc.0[zip.prcv.0$ord,c(1,2)],col=4,lwd=2)
```



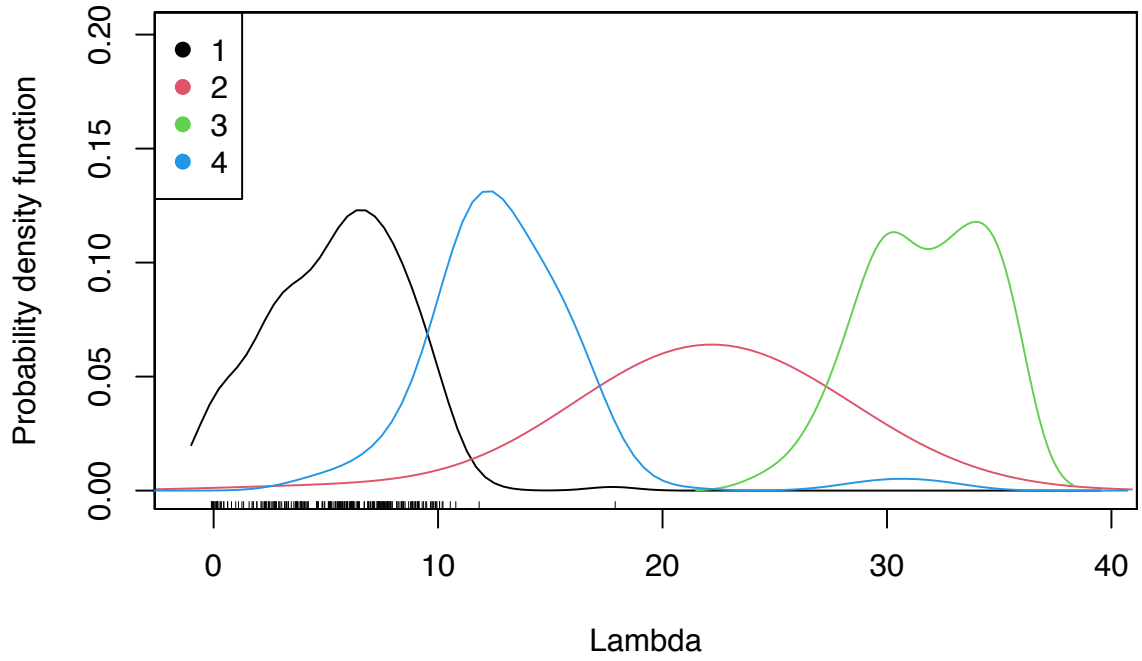
1f) Univariate analysis of k clusters

For each one of the k clusters obtained above, do the following tasks: (A unique plot should be done, at which the k densities are represented simultaneously) - Consider the univariate data set of the lambda scores over the principal curve of the points in this cluster. - Estimate non-parametrically the density function of lambda, conditional to this cluster. Use the default bandwidth value. - Plot the estimated density function.

lambda densities for each cluster have been plotted. Lambda in cluster 2 has normal shape and in cluster 1 and 4 more or less has same shape, as we saw in PC scores these two clusters were near to each other.

```
lambda.prcr<- cbind(zip.prcv.0$lambda, cluster)

x1<- lambda.prcr[lambda.prcr[,2] ==1,]
x2<-lambda.prcr[lambda.prcr[,2] ==2,]
```

```
x3<-lambda.prcr[lambda.prcr[,2] ==3,]
x4<-lambda.prcr[lambda.prcr[,2] ==4,]

sm.density(x1[,1],col=x1[,2], cex=4,xlim=c( min(c(x1[,1],x2[,1],x3[,1],x4[,1]))-1,max(c(x1[,1],x2[,1],x3

sm.density(x2[,1],col=x2[,2], add = TRUE)
sm.density(x3[,1],col=x3[,2], add = TRUE)
sm.density(x4[,1],col=x4[,2], add = TRUE)

legend("topleft",
       legend = levels(factor(lambda.prcr[,2])),
       pch = 19,
       col = factor(levels(factor(lambda.prcr[,2]))))
```



2. Choose smoothing parameter in PCs

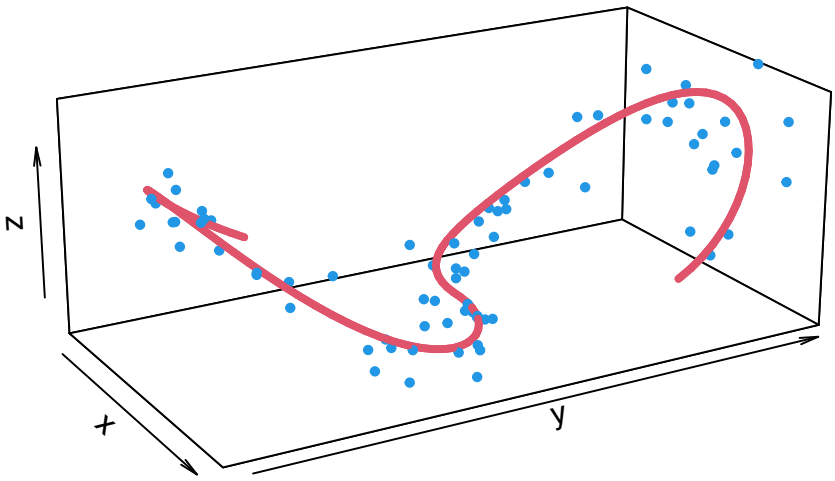
Consider the 3-dimensional data set generated by the following code provided by the exercise sheet:

```
t <- seq(-1.5 * pi, 1.5 * pi, l = 100)
R <- 1
n <- 75
sd.eps <- .15

set.seed(1)
y <- R * sign(t) - R * sign(t) * cos(t / R)
x <- -R * sin(t / R)
z <- (y / (2 * R)) ^ 2
rt <- sort(runif(n) * 3 * pi - 1.5 * pi)
eps <- rnorm(n) * sd.eps
ry <- R * sign(rt) - (R + eps) * sign(rt) * cos(rt / R)
rx <- -(R + eps) * sin(rt / R)
rz <- (ry / (2 * R)) ^ 2 + runif(n, min = -2 * sd.eps, max = 2 * sd.eps)
```

```
XYZ <- cbind(rx, ry, rz)

require(plot3D)
```

```
## Loading required package: plot3D
```

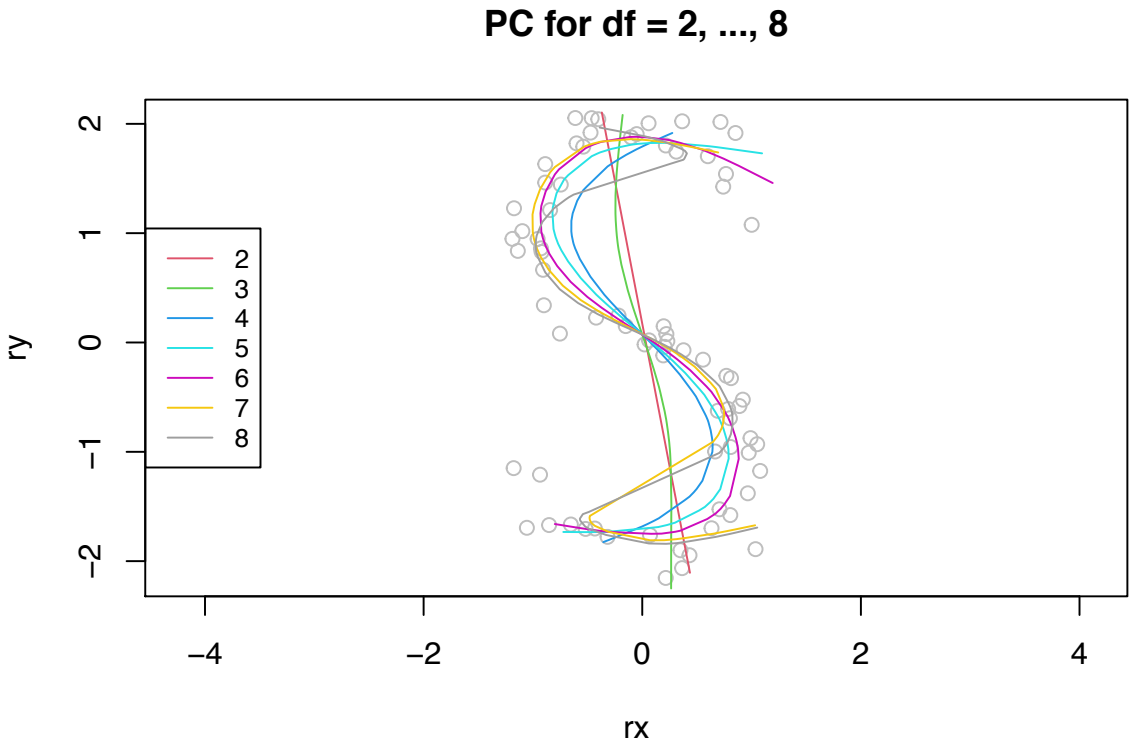
```
lines3D(x, y, z, colvar = NULL, phi = 20, theta = 60, r = sqrt(3), d = 3,
  scale = FALSE, col = 2, lwd = 4, as = 1,
  xlim = range(rx), ylim = range(ry), zlim = range(rz)
)
points3D(rx, ry, rz, col = 4, pch = 19, cex = .6, add = TRUE)
```



2a) Choose df using LOOCV

Choose the value of the degrees of freedom df by leave-one-out cross-validation. Restrict the search of df to seq(2,8,by=1).

```
require(princurve)

df <- seq(2,8,by=1)

plot(XYZ,col="gray",asp=1, main="PC for df = 2, ..., 8")
for (i in df){
  fit <- principal_curve(XYZ,df=i)
  lines(fit,col=i)
}
legend("left",legend=df,col=df,lty=1,cex=0.8)
```



From all these different lines we can already recognize a S-shape. However, after getting a general idea of all the possible degrees of freedom, we compute the optimal solution using LOOCV.

```
dist <- c()
for (i in 1:length(df)){
  dist[i] <- 0
  for (j in 1:length(XYZ[,1])){
    fit.ij <- principal_curve(XYZ[-j,],df=df[i])
    point <- matrix(XYZ[j,],ncol=ncol(fit.ij$s[fit.ij$ord,]),byrow=TRUE)
    curve.ij <- project_to_curve(point, fit.ij$s[fit.ij$ord,])
    dist[i] <- dist[i] + curve.ij$dist
  }
}
names(dist) <- c(seq(2,8,by=1))
print(dist)
```

```
##         2         3         4         5         6         7         8
## 49.220432 38.696680 19.790712 11.134465  8.084836  9.558809  9.878976
```

```
df.optimal <- as.numeric(names(dist)[dist==min(dist)])
df.optimal
```

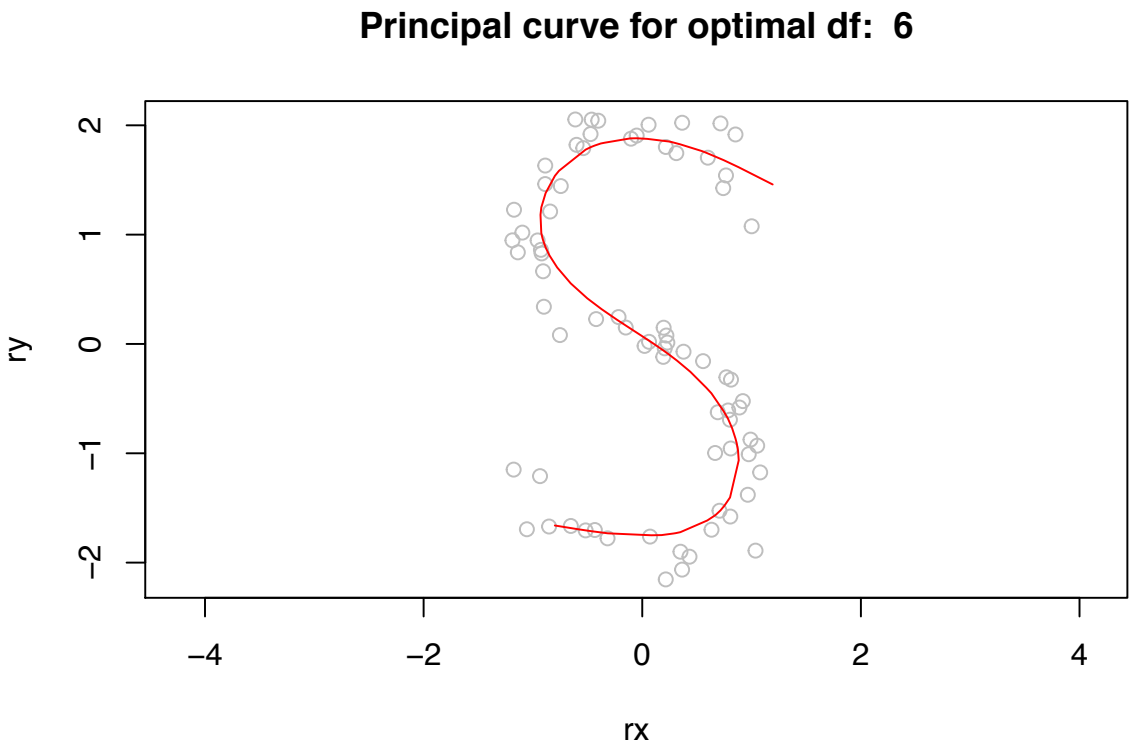
```
## [1] 6
```

Comparing the distances from all degrees, we select the minimum distance, which gave us 6 as the optimal degree of freedom.

2b) Graphical representation of PC with optimal df

Give a graphical representation of the principal curve output for the optimal df and comment on the obtained results.

```
plot(XYZ,col="gray",asp=1,
     main=paste("Principal curve for optimal df: ", df.optimal))
fit <- principal_curve(XYZ,df=df.optimal)
lines(fit, col="red")
```



As predicted earlier, this curve shows us a clear S-shape that goes through the points.

2c) LOOCV using df=50

Compute the leave-one-out cross-validation for df=50 and compare it with the result corresponding to the optimal df value you found before.

```
dist50 <- 0
for (i in 1:length(XYZ[,1])){
  fit50 <- principal_curve(XYZ[-i,],df=50)
  point <- matrix(XYZ[i,],ncol=ncol(fit50$s[fit50$ord,]),byrow=TRUE)
  curve50 <- project_to_curve(point, fit50$s[fit50$ord,])
  dist50 <- dist50 + curve.ij$dist
}
print(dist50)
```

```
## [1] 74.71754
```

```
library(rgl)
plot3d(XYZ,col="black")
lines3d(fit50$s,col="red",lwd=2)
lines3d(principal_curve(XYZ,df=df.optimal)$s,col="blue",lwd=2)
```

As we know, the larger the degree of freedom becomes, the more it might tend to overfit as it has less smoothing. Initially, one might assume that this leads the distance to become smaller. However, based on the distances we calculated using LOOCV previously, we can see that after our optimal df=6 the distance

started to increase again which would lead us to think that the distance for df=50 is quite large. Hence, we would conclude that our current optimal degree of freedom is better.

Through calculations, we can see that our assumption proved to be right as the sum-of-squared distances from the points to their projections of the optimal df is still better, as the distance for df=50 lies at almost 75.

We can see that the curve follows a similar general structure as the optimal curve in a 3-dimensional space. Here we can recognize that the overfitting applies to all axes. Hence, we can conclude that a too high df, or in our case df=50, is not better than our current optimal df. But this could also be seen in leave-one-out cross-validation, where we had quite a large distance.