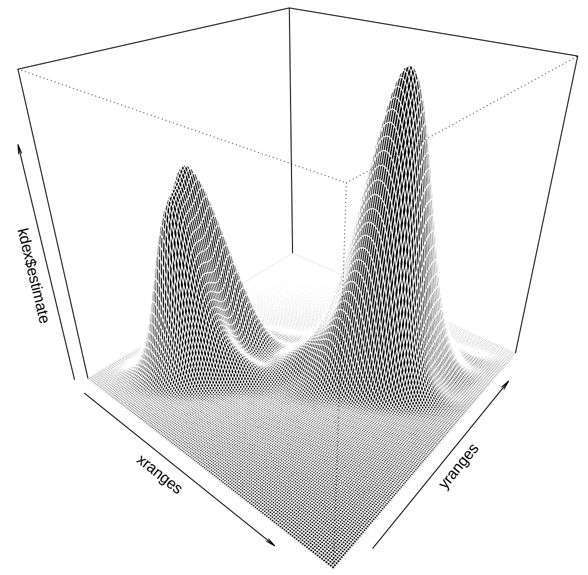
EXERCISE 10

WEIYU LI

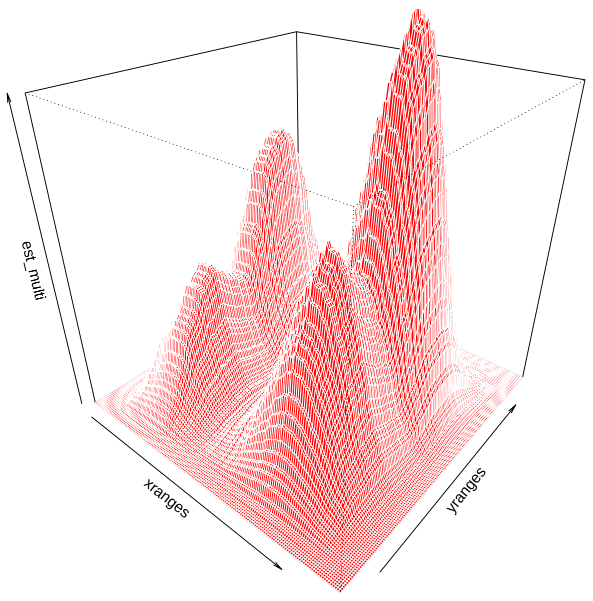
1. For the dataset *faithful* in R, use (i) multiplicative kernel and (ii) spherical-symmetric kernel to estimate the joint pdf of *eruptions* and *waiting*. Consider different bandwidth selection methods.

```
library('ks')
x <- as.matrix(faithful)
n <- nrow(x)
kdex <- kde(x)
xranges <- unlist(kdex$eval.points[1]) # the x ranges
yranges <- unlist(kdex$eval.points[2]) # the y ranges
persp(xranges, yranges, kdex$estimate,
      phi = 30, theta = 40, col = 'black', border = 0)
# this gives an estimation with Gaussian kernel
# which is both multiplicative and symmetric
```

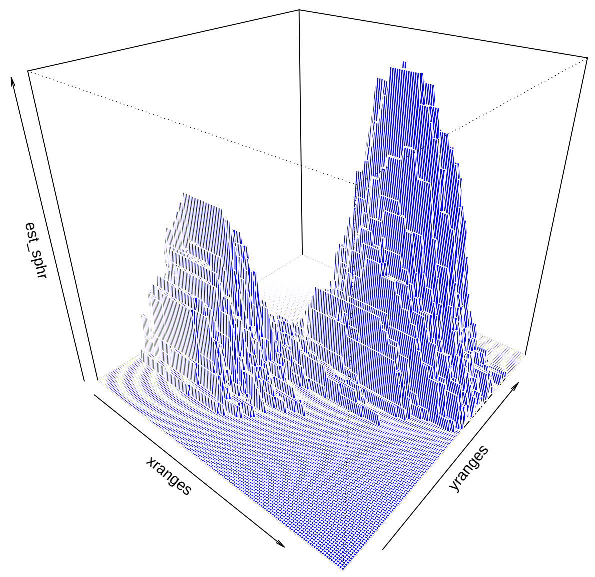


Date: 2019/11/04.
liweiyu@mail.ustc.edu.cn.

```
# Next, we try some hand-written kde
### (1) multiplicative kernel with I(-1 < x <= 1) / 2 coordinate-wise
# initialize
h_multi1 <- .5
h_multi2 <- 10
est_multi1 <- xranges
est_multi2 <- yranges

# calculation
for (i in 1:length(xranges)) {
  est_multi1[i] <- sum(abs(x[,1]-rep(xranges[i],n)) / h_multi1 < 1) / (n * h_multi1)
}
for (i in 1:length(yranges)) {
  est_multi2[i] <- sum(abs(x[,2]-rep(yranges[i],n)) / h_multi2 < 1) / (n * h_multi2)
}
est_multi <- est_multi1 %*% t(est_multi2)
persp(xranges, yranges, est_multi,
      phi = 30, theta = 40, col = 'red', border = 0, zlim = c(0, 0.05))
```



```
### (2) spherical kernel with I(|x| <= 1) / pi
euclidean <- function(x){
  # return the euclidean distance of each column
  y <- x[,1]
  for (i in 1:length(y)) {
    y[i] <- sqrt(sum(x[i,]^2))
  }
  return(y)
}
est_sphr <- kdex$estimate
for (i in 1:length(xranges)) {
  for (j in 1:length(yranges)) {
    tmp_pt <- cbind(rep(xranges[i], n), rep(yranges[j], n))
    est_sphr[i,j] <- sum(euclidean(x - tmp_pt) <= 1) / (pi * n)
  }
}
persp(xranges, yranges, est_sphr,
      phi = 30, theta = 40, col = 'blue', border = 0)
```



□

2. Suppose $X_1, \dots, X_n$ *i.i.d.* $\sim f(x)$, and an independent $U \sim Unif\{1, 2, \dots, n\}$. Let $Y = X_U + hZ$, where Z has density $p(x)$ and is independent with $X_1, \dots, X_n, U$.

(1) Prove that with samples $X_1, \dots, X_n$, the density of Y is exactly the KDE of f with kernel $p(\cdot)$ and bandwidth h, denoted as $\hat{f}(\cdot)$.

Proof. First note that the KDE

$$\hat{f}(y) = \frac{1}{nh}\sum_{i=1}^{n} p(\frac{y - X_i}{h}).$$

On the other hand, since Y given X_i is random with respect to U and Z, the pdf of Y is

$$f_Y(y) = \sum_{i=1}^{n} f_{Y|U}(y|U = i)P(U = i) = \sum_{i=1}^{n} f_{hZ}(y - X_i)\frac{1}{n} = \sum_{i=1}^{n} p(\frac{y - X_i}{h})\frac{1}{nh},$$

which is exactly the same as $\hat{f}(y)$. □

(2) Given $X_1, \dots, X_n$, solve $Var(Y)$. Compare it with the sample variance with $X_1, \dots, X_n$.
Solve. From (1), we know that the pdf of Y is $\hat{f}(y) = \frac{1}{nh}\sum_{i=1}^{n} p(\frac{y-X_i}{h})$, then we have

$$\begin{aligned} EY &= \int y\hat{f}(y)dy = \sum_{i=1}^{n} \frac{1}{n}\int yp\left(\frac{y - X_i}{h}\right)\frac{1}{h}dy \\ &= \sum_{i=1}^{n}\frac{1}{n}\int (X_i + hz)p(z)dz \\ &= \frac{1}{n}\sum_i X_i + hEZ, \\ EY^2 &= \sum_{i=1}^{n}\frac{1}{n}\int (X_i + hz)^2 p(z)dz \\ &= \frac{1}{n}\sum_i X_i^2 + \frac{1}{n}\sum_i 2hX_iEZ + h^2EZ^2. \end{aligned}$$

Therefore, we obtain

$$Var(Y) = \frac{1}{n}\sum_i X_i^2 - \left(\frac{1}{n}\sum_i X_i\right)^2 + h^2VarZ = S_n^2 + h^2VarZ,$$

where $S_n^2 = \frac{1}{n}\sum_i (X_i - \bar{X})^2 = \frac{n-1}{n}S^2$ and S^2 is the sample variance. □

3. For the dataset *wines* (`https://raw.githubusercontent.com/egarpor/handy/master/datasets/wines.txt`), remove variable *vintage* and normalize the variables, then perform PCA. Use clustering methods based on density estimation (refer to *cluster.R*).

(1) With 3 PCs, how many clusters can be found? If min.clust.size is set as 5% of the data size, then what's the result? Is the result sensitive to min.clust.size?

(2) With the same min.clust.size and 6 PCs, how many clusters can be found? Use the scatter plots to identify PCs that helps clustering or not.

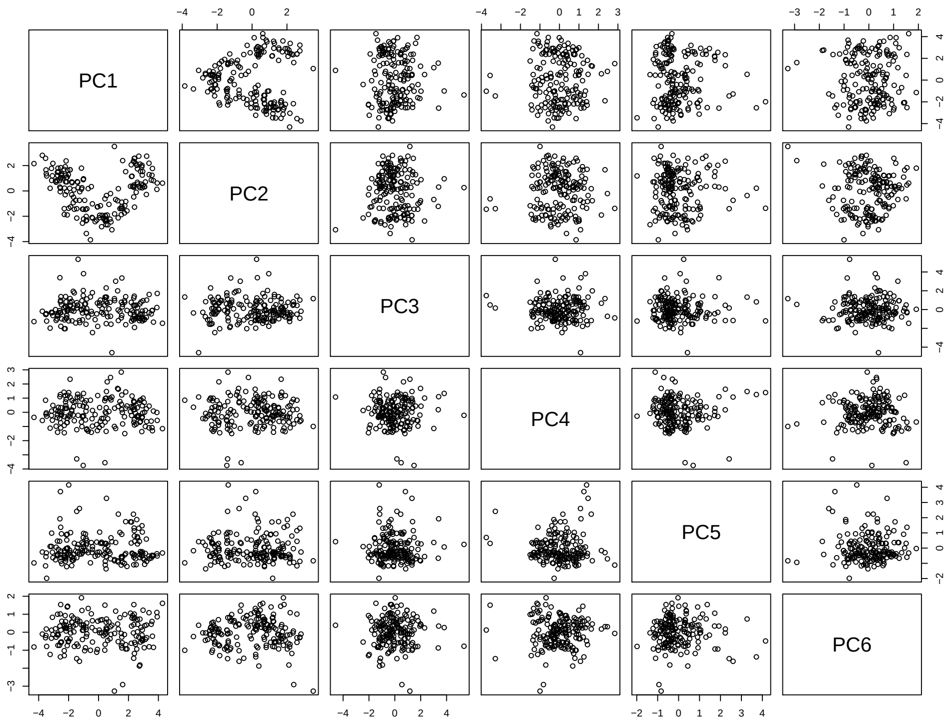
(3) With 2 PCs, how many clusters can be found? Compare the result with that of 3 PCs.

(4) Compare the results with the true clusters.

(5) Compare the results with the results of k-means with $k = 3, 4$.

```
library('mclust')
library('ks')
x <- read.table('wines.txt', header = T)
n <- nrow(x)
p <- ncol(x) -1
true_cl <- unclass(as.factor(x[,'vintage']))
data_cl <- as.matrix(x[, -(p + 1)])
# data normalization: each column is with norm $\sqrt{n}$
# if normalizing the norm to be 1, then each entry will be too small
for (i in 1:p) {
  data_cl[, i] <- data_cl[, i] - mean(data_cl[, i])
  norm <- sqrt(sum(data_cl[, i]^2))
  data_cl[,i] <- data_cl[,i] * sqrt(n) / norm
}
# perform PCA
pca_cl <- prcomp(data_cl, center = F, scale. = F) # PCA informations
data_pca <- data_cl %*% pca_cl$rotation # data after PCA rotation

### (1) cluster using 3 PCs
kms1.1 <- kms(data_pca[, 1:3])
summary(kms1.1) # 4 clusters
kms1.2 <- kms(data_pca[, 1:3], min.clust.size = 0.05 * n)
summary(kms1.2) # 3 clusters
# This is sensitive to min.clust.size

### (2) cluster using 6 PCs
kms2 <- kms(data_pca[, 1:6], min.clust.size = 0.05 * n)
summary(kms2) # 6 clusters
pairs(data_pca[, 1:6])

### (3) cluster using 2 PCs
kms3 <- kms(data_pca[, 1:2])
summary(kms3) # 4 clusters

### (4) comparison with true clusters
classError(kms1.1$label, true_cl)$errorRate # 0.05649718
classError(kms1.2$label, true_cl)$errorRate # 0.05084746
classError(kms2$label, true_cl)$errorRate # 0.2711864
classError(kms3$label, true_cl)$errorRate # 0.1412429
# note that all the true 1st cluster can be clustered in one cluster

### (5) comparison with kmeans
# Here we only show the comparison result between (1) and k-means
kmeans1.1 <- kmeans(data_pca[, 1:3], 4)
```

```
classError(kmeans1.1$cluster, true_cl)$errorRate
# 4-means error: 0.1016949
classError(kms1.1$label, true_cl)$errorRate
# kms error with 4 clust: 0.05649718
# all the first and last true clusters are clustered in one cluster
# to see this, you can use:
# cbind(kmeans1.1$cluster, kms1.1$label, true_cl)

kmeans1.2 <- kmeans(data_pca[, 1:3], 3)
classError(kmeans1.2$cluster, true_cl)$errorRate
# 3-means error: 0.03954802
classError(kms1.2$label, true_cl)$errorRate
# kms error with 3 clust: 0.05084746
# all the first and last true clusters are clustered in one cluster

# note that with the true k, k-means performs better;
# while kms is less sensitive to cluster numbers
```



□