다변량자료분석 및 실습 Lab 6

서울대학교 통계학과 2017-11362 박건도

2021년 11월 23일

1. Interpret the result of Box's M test. Can you assume equal covariance?

```
BoxM(flea[,2:7], group = flea$species)
```

```
## $Chisq
## [1] 49.27496
##
## $df
## [1] 42
##
## $p.value
## [1] 0.2049986
##
## $Test
## [1] "BoxM"
##
## attr(,"class")
## [1] "MVTests" "list"
```

p-value가 0.05보다 크므로 귀무가설을 기각할 수 없다. 따라서 equal covariance라고 가정할 수 있다.

2. Can we assume normality for each population?

```
flea1 <- flea %>% filter(species == "Concinna")
mvnormtest::mshapiro.test(t(flea1[,2:7]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
```

```
## W = 0.8745, p-value = 0.01157
```

```
flea2 <- flea %>% filter(species == "Heikert.")
mvnormtest::mshapiro.test(t(flea2[,2:7]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.88511, p-value = 0.003125
```

```
flea3 <- flea %>% filter(species == "Heptapot.")
mvnormtest::mshapiro.test(t(flea3[,2:7]))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.89801, p-value = 0.02712
```

각각의 경우에 대해 p-value가 0.05보다 모두 작으므로, 각 population의 정규성을 보장할 수 없다.

3. Manually compute the Between-variance matrix B from "mean_vectors".

```
n <- nrow(flea)
p <- ncol(flea) - 1
uniq.id <- unique(flea$species)

mean_vectors <- vector()
W <- matrix(0, nrow = p, ncol = p)
for (gr.id in uniq.id) {
  gData <- as.matrix( flea[flea$species == gr.id,2:7] )
  W <- W + t( scale( gData, scale = FALSE))  %*%  (  scale( gData, scale = FALSE) )
  mean_vectors <- cbind(mean_vectors,
                        colMeans(gData))
}
Tot <- cov(flea[,2:7]) * (n-1)
B <- Tot - W

total_mean_vectors <- colMeans(flea[,2:7])
nis <- table(flea[,1])
```

```r
B1 <- matrix(0, nrow=p, ncol=p)
for (i in 1:3){
  B1 <- B1 + nis[uniq.id[i]] * (mean_vectors[,i] - total_mean_vectors) %*% t(mean_vectors[,i] - total_m
}
norm(B1 - B, "F")
```

```
## [1] 3.107417e-11
```

Frobenius norm of B, B1 is 3.1×10^{-11}, which is small value. We can say that B = B1.

4. How many positive eigenvalues should you see in the above?

```r
out <- manova(as.matrix(flea[,2:7]) ~ species, data = flea)
sout <- summary(out, test = "Pillai")
eigen(solve(W) %*% B)
```

```
## eigen() decomposition
## $values
## [1]  1.777934e+01+0.00000e+00i  3.885151e+00+0.00000e+00i
## [3] -7.801919e-16+0.00000e+00i  5.176521e-16+5.37222e-17i
## [5]  5.176521e-16-5.37222e-17i -2.129950e-16+0.00000e+00i
##
## $vectors
##               [,1]           [,2]           [,3]                      [,4]
## [1,] -0.2675110+0i -0.02154039+0i  0.09212585+0i -0.01365477+0.00032176i
## [2,]  0.1709100+0i -0.06568276+0i  0.05366521+0i -0.14205033-0.21845195i
## [3,]  0.3961152+0i  0.45893180+0i  0.45454313+0i -0.93360120+0.00000000i
## [4,]  0.1878777+0i -0.31941214+0i -0.07455452+0i  0.10658710+0.03631379i
## [5,] -0.8298174+0i -0.82598507+0i -0.87905718+0i  0.04130340+0.19905070i
## [6,]  0.1357403+0i -0.01810914+0i  0.06102063+0i  0.04777141+0.06335562i
##                          [,5]           [,6]
## [1,] -0.01365477-0.00032176i  0.06756976+0i
## [2,] -0.14205033+0.21845195i -0.48840318+0i
## [3,] -0.93360120+0.00000000i  0.18423700+0i
## [4,]  0.10658710-0.03631379i  0.29937518+0i
## [5,]  0.04130340-0.19905070i -0.79510742+0i
## [6,]  0.04777141-0.06335562i -0.03364654+0i
```

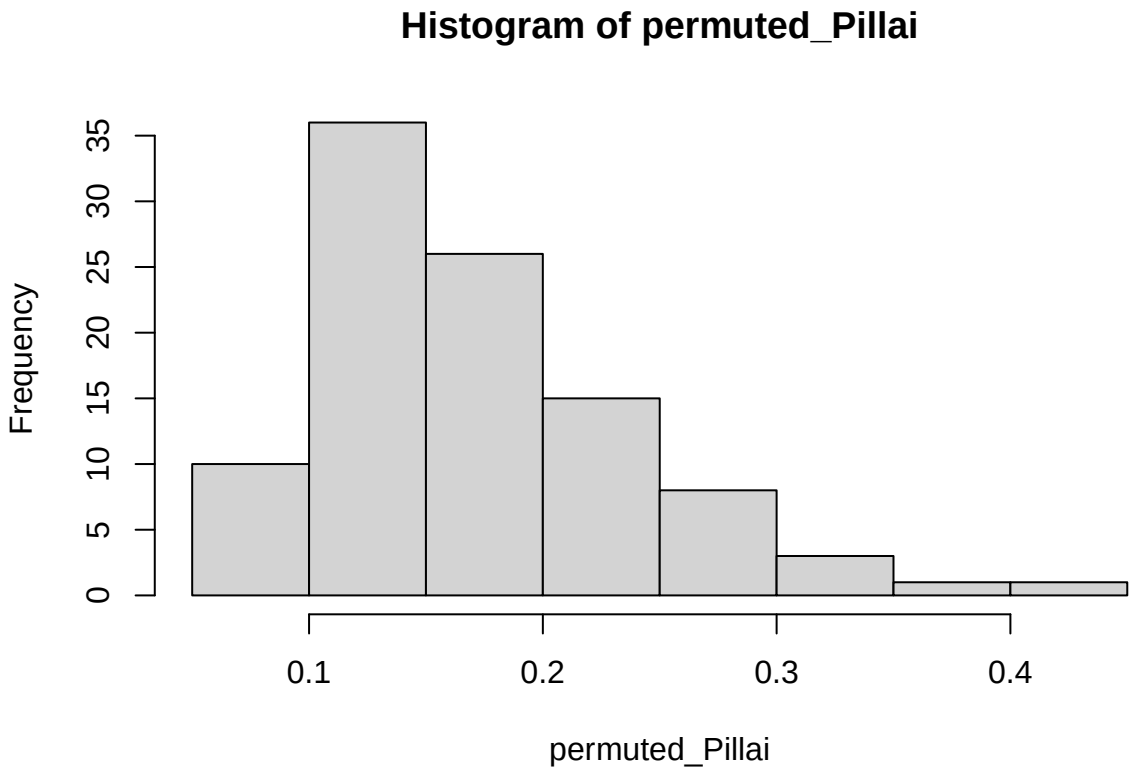
```r
sum(sout$Eigenvalues / (1 + sout$Eigenvalues))
```

```
## [1] 1.742048
```

양수인 eigenvalue가 많으면 eigenvalue의 증가함수의 합으로 표현되는 Pillai나 Hotelling의 statistic이 커지는 경향이 있다. 위의 예시에서는 2개의 eigenvalue를 찾을 수 있고, 나머지 값들 중 음수의 값이 보이긴 하나 거의 0에 가까운 값이므로, 0으로 생각한다면 2개의 positive eigenvalue를 갖고있다고 할 수 있다. 6개 중 2개는 과반을 넘지 않으므로 $H_0 : \mu_1 = \mu_2 = \mu_3$을 기각할 수 없다.

5. MANOVA under permutation-based test.

```
perms <- permute(flea,  100, species)
permuted_Pillai <- map(perms$perm,
                       ~ summary(manova(as.matrix(flea[,2:7]) ~ species, data = .))$stats[3])
permuted_Pillai <- unlist(permuted_Pillai)
hist(permuted_Pillai)
```



Most of value 〉 0.05, so we cannot reject H0. Thus, we can say that $\mu_1 = \mu_2 = \mu_3$.

6. Convert “in.sample.prediction.p” into the posterior probability of “Concinna” given x.

```
flea.b <- flea %>% filter(species != "Heptapot.")
X <- prcomp(flea.b[,2:7])$x %>% as.data.frame %>%
```

```
  dplyr::select(PC1,PC2) %>% mutate(species = factor(flea.b$species))
fit3<- glm(species ~ . , data = X, family  = "binomial")
in.sample.prediction.p <- predict(fit3, X[,1:2])

1 - 1/(1+exp(-in.sample.prediction.p)) # P(X.species == Concinna)
```

```
##            1            2            3            4            5            6
## 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
##            7            8            9           10           11           12
## 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
##           13           14           15           16           17           18
## 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00
##           19           20           21           22           23           24
## 1.000000e+00 1.000000e+00 1.000000e+00 0.000000e+00 2.220446e-15 0.000000e+00
##           25           26           27           28           29           30
## 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##           31           32           33           34           35           36
## 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##           37           38           39           40           41           42
## 0.000000e+00 0.000000e+00 2.220446e-16 0.000000e+00 0.000000e+00 0.000000e+00
##           43           44           45           46           47           48
## 3.774758e-15 0.000000e+00 0.000000e+00 0.000000e+00 5.992762e-11 0.000000e+00
##           49           50           51           52
## 0.000000e+00 0.000000e+00 0.000000e+00 3.852156e-10
```

Since `in.sample.prediction.p` < 0 means that x is “Concinna”, so posterior probability of “Concinna” given x is `1-logistic(in.sample.prediction.p)`.

7. By executing the following lines, compare five different methods of classification.

```
heart$target <- factor(heart$target)

# Split into training and testing data set
set.seed(123)
trainIndex <- createDataPartition(heart$target, p = 0.8, list = FALSE)
train.data <- heart[trainIndex,]
test.data <- heart[-trainIndex,]
```

```r
# Set tuning parameter selection method
control <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

method.list <- c("lda","qda","glm","knn","nb")
out.list <- list()
for (i in 1:5) {
  out.list[[i]] <- train(target ~ . ,
                         data = train.data,
                         method = method.list[i],
                         trControl = control)
}

# Evaluate the each classifier
confusion.out <- list()
for (i in 1:5){
  confusion.out[[i]] <- confusionMatrix(
    predict(out.list[[i]], newdata = test.data),
    reference = test.data$target)
}

# Get each accuracy
sapply(confusion.out, function(x) x$overall['Accuracy'] )
```

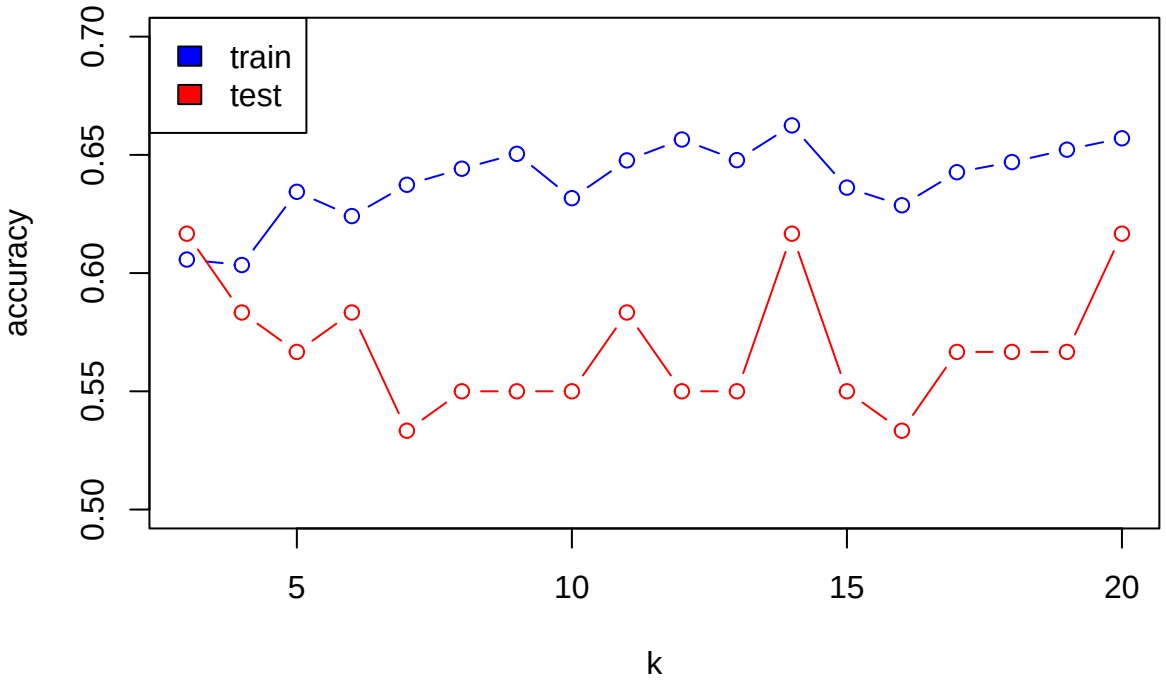
```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.8333333 0.7666667 0.8166667 0.5333333 0.8166667
```

Accuracy: lda 〉 glm = nb 〉 qda 〉 knn.

8. Try each value of k in 3:20 and create a line plot, plotting “k vs training accuracy”, overlaid with “k vs testing accuracy”

```r
train_acc <- vector()
test_acc <- vector()
for (k in 3:20){
  out <- train(target ~ ., data = train.data, method='knn', tuneGrid = expand.grid(k=k))
  train_acc[k] <- out$results[1,2]
  test_acc[k] <- sum(test.data$target == predict(out, newdata = test.data))/nrow(test.data)
}
plot(3:20, train_acc[3:20], type='b', ylim=c(0.5,0.7), col='blue', xlab='k',ylab='accuracy')
```

```
lines(3:20, test_acc[3:20], type='b', col='red')
legend('topleft', legend=c('train', 'test'), fill=c('blue', 'red'))
```



k=7 looks the best choice of k.