hw5

shichenh

11/2/2017

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
cols <- c("class", "alcohol", "malic", "ash", "alcalinity", "magnesium", "phenols", "flavanoids", "nonf
wine <- read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data", col_names =
```

```
## Parsed with column specification:
## cols(
##   class = col_integer(),
##   alcohol = col_double(),
##   malic = col_double(),
##   ash = col_double(),
##   alcalinity = col_double(),
##   magnesium = col_integer(),
##   phenols = col_double(),
##   flavanoids = col_double(),
##   nonflavanoids = col_double(),
##   proanthocyanins = col_double(),
##   color = col_double(),
##   hue = col_double(),
##   dilution = col_double(),
##   proline = col_integer()
## )
```

```
wine <- wine %>% mutate(class = factor(class))

iris <- iris
```

1 Sum-of-Squares Dispersion Functions

```
tss <- function(x){
  # returns the total sum of squares for a vector
  avg <- mean(x)
  return(sum((x- avg)^2))
}
```

```r
bss <- function(x, y) {
  # computes the between group variance
  if (length(x) != length(y)) {
    stop("length of x and y are not the same")
  }
  group.means <- aggregate(x~y, FUN = mean)[,2]
  sum(((group.means - mean(x)))^2 * table(y))
}

wss <- function(x, y) {
  # computes the within group sum of squares
  if (length(x) != length(y)) {
    stop("length of x and y are not the same")
  }
  sum(aggregate(x~y, FUN = tss)[,2])
}
```

```r
tss(iris$Sepal.Length)
```

```
## [1] 102.1683
```

```r
bss(iris$Sepal.Length, iris$Species)
```

```
## [1] 63.21213
```

```r
wss(iris$Sepal.Length, iris$Species)
```

```
## [1] 38.9562
```

2 Sum-of-Square Ratio Functions

```r
cor_ratio <- function(x, y) {
  bss(x, y)/tss(x)
}

F_ratio <- function(x, y) {
  k <- length(levels(y))
  num <- bss(x, y)/(k-1)
  denum <- wss(x, y)/(length(x) - k)
  num/denum
}
```

```r
cor_ratio(iris$Sepal.Length, iris$Species)
```

```
## [1] 0.6187057
```

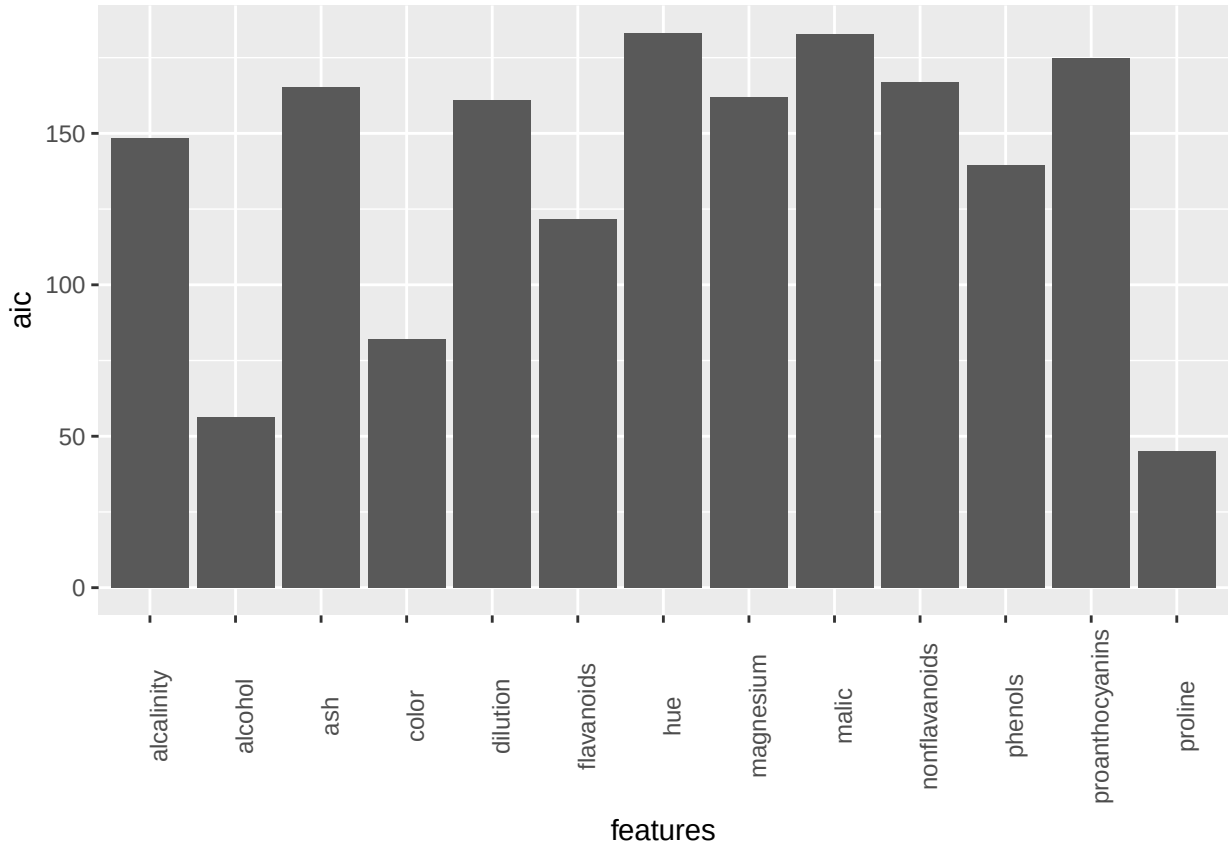
```r
F_ratio(iris$Sepal.Length, iris$Species)
```

```
## [1] 119.2645
```

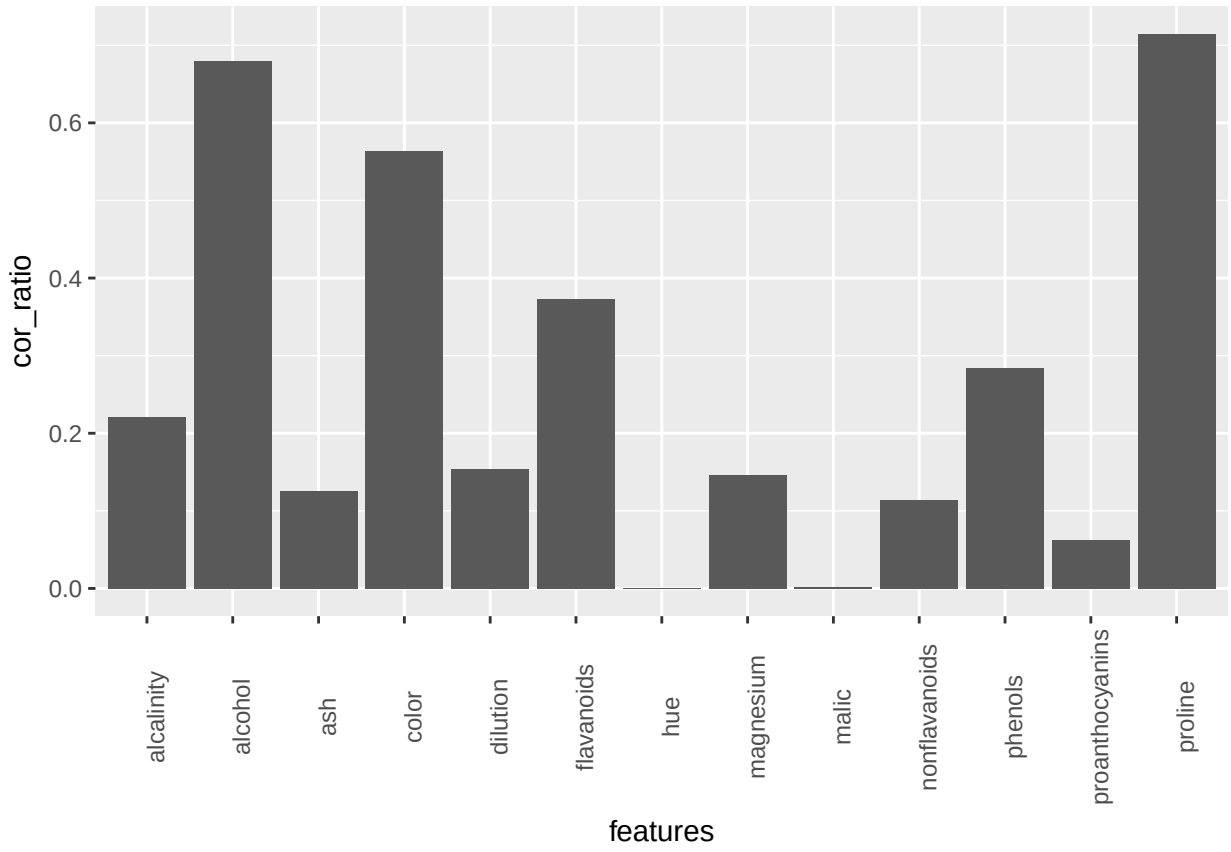
3 Discriminant Power of Predictors

```
# eleminating one class for simple logistic regression
wine12 <- wine %>% filter(class != 3)
```

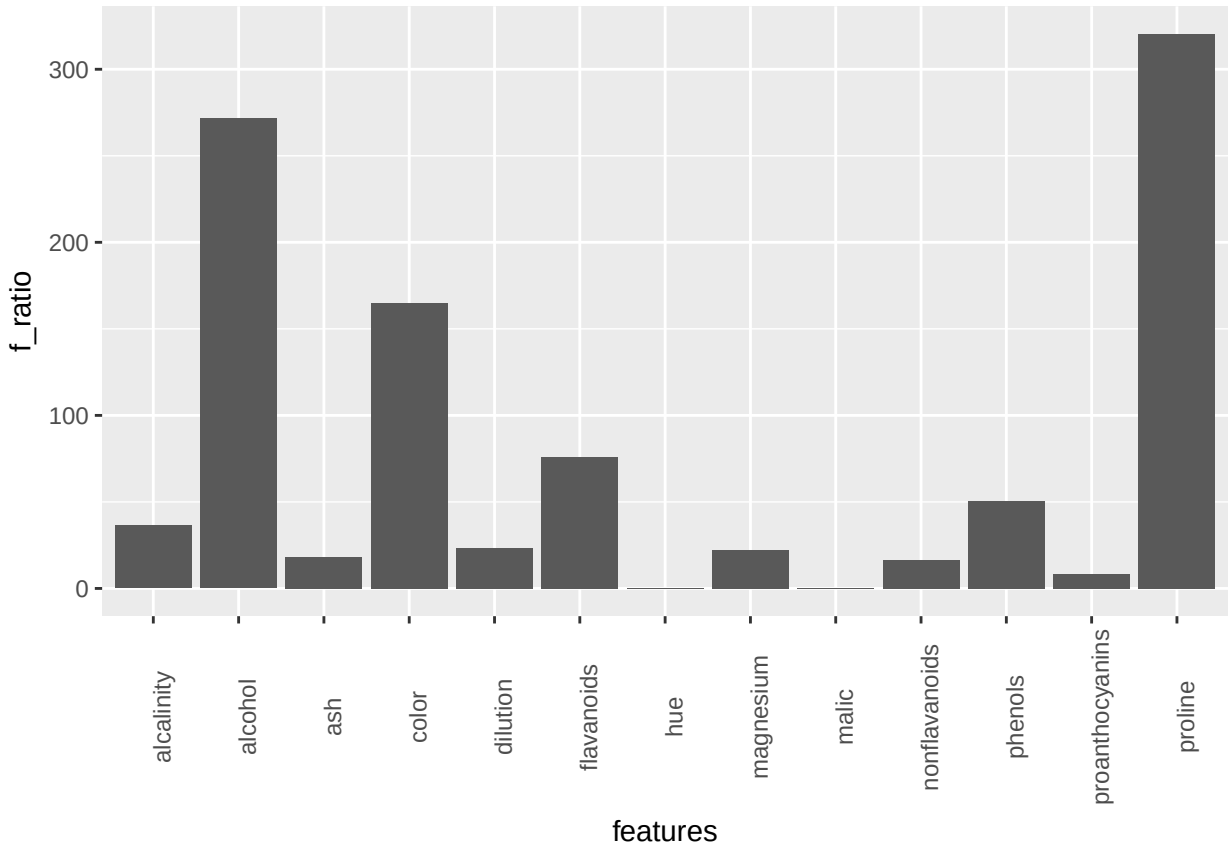
```
features <- colnames(wine12[,-c(1)])

aics <- numeric()
for (i in 1:length(features)) {
  aics[i] <- (glm(class~., family = binomial, data=wine12[,c("class", features[i])]))$aic
}

aic.table <- data.frame(feature=features, aic=aics)

ggplot(aic.table) +
  geom_bar(aes(features, aic), stat = "identity") +
  theme(axis.text.x = element_text(angle=90))
```




```
cor.ratios <- sapply(wine12[,2:ncol(wine12)],
                     FUN = function(x) cor_ratio(x, factor(wine12$class)))
cor.ratios.table <- data.frame(feature=features, cor_ratio=cor.ratios)

ggplot(cor.ratios.table) +
  geom_bar(aes(features, cor_ratio), stat = "identity") +
  theme(axis.text.x = element_text(angle=90))
```




```
f.ratios <- sapply(wine12[,2:ncol(wine12)],
                   FUN = function(x) F_ratio(x, factor(wine12$class)))
f.ratios.table <- data.frame(feature=features, f_ratio=f.ratios)

ggplot(f.ratios.table) +
  geom_bar(aes(features, f_ratio), stat = "identity") +
  theme(axis.text.x = element_text(angle=90))
```



AIC are inversely related to the sum of square ratios.(the smaller aic a predictor has, the stronger it can discrminate the classes, therefore the larger sum-of-square ratio it has.

4. Variance Function

```
total_variance <- function(x) {
  x.mean <- scale(x, scale = F)
  t(x.mean) %*% x.mean/(nrow(x)-1)
}

between_variance <- function(x, y) {
  y.dum <- spatstat::dummify(y)
  x.mean <- scale(x, scale=F)
  t(x.mean) %*% y.dum %*% solve(t(y.dum)%*%y.dum)%*%t(y.dum) %*%x.mean/(nrow(x)-1)
}

within_variance <- function(x, y) {
  y.dum <- spatstat::dummify(y)
  x.mean <- scale(x, scale=F)
  t(x.mean) %*% (diag(rep(1, nrow(y.dum))) - y.dum %*%
                   solve(t(y.dum)%*%y.dum)%*%t(y.dum))%*%(x.mean)/(nrow(x)-1)
}
```

```
total_variance(iris[,1:4])
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
```

```
## Sepal.Length    0.6856935  -0.0424340    1.2743154   0.5162707
## Sepal.Width    -0.0424340   0.1899794   -0.3296564  -0.1216394
## Petal.Length    1.2743154  -0.3296564    3.1162779   1.2956094
## Petal.Width     0.5162707  -0.1216394    1.2956094   0.5810063
```

```
between_variance(iris[,1:4], iris$Species)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length    0.4242425 -0.13391051    1.1090497   0.4783848
## Sepal.Width    -0.1339105  0.07614049   -0.3841584  -0.1539105
## Petal.Length    1.1090497 -0.38415839    2.9335758   1.2535168
## Petal.Width     0.4783848 -0.15391051    1.2535168   0.5396868
```

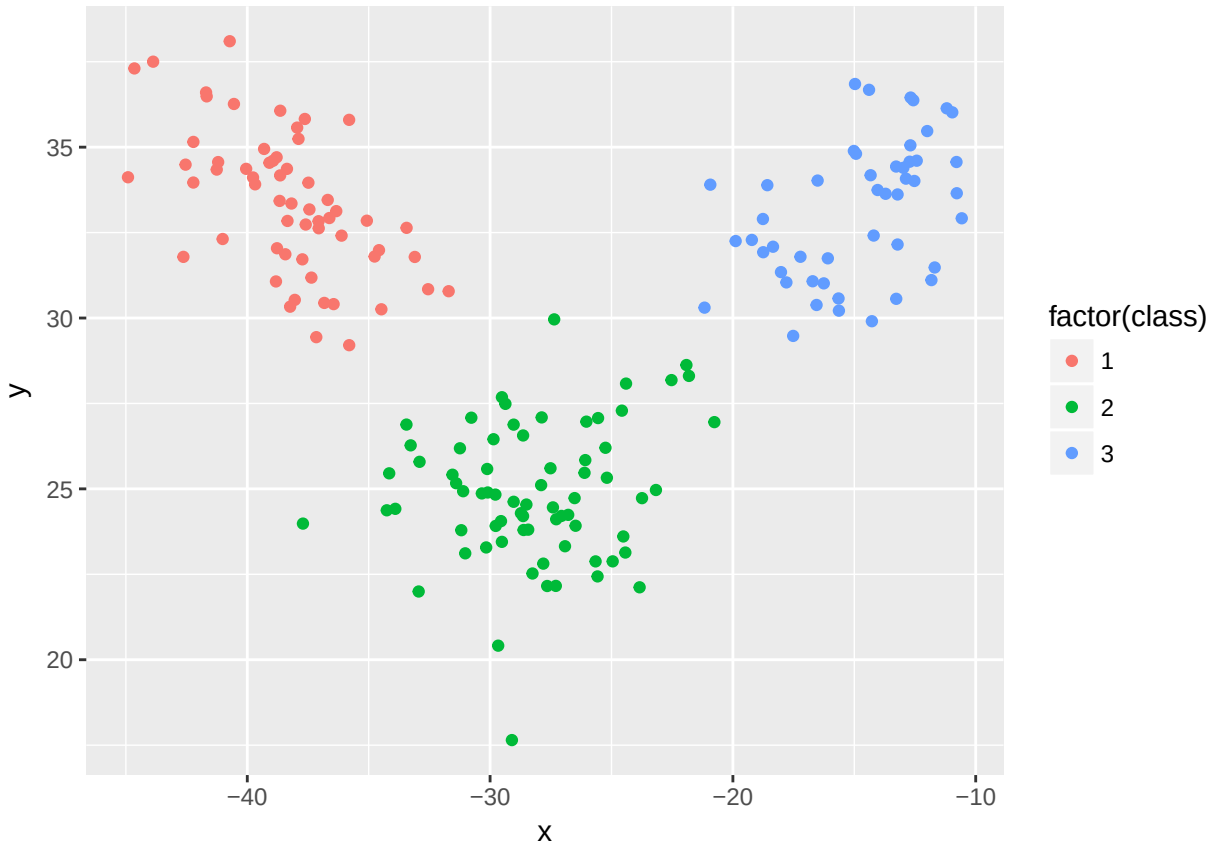
```
within_variance(iris[,1:4], iris$Species)
```

```
##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length   0.26145101  0.09147651   0.16526577  0.03788591
## Sepal.Width    0.09147651  0.11383893   0.05450201  0.03227114
## Petal.Length   0.16526577  0.05450201   0.18270201  0.04209262
## Petal.Width    0.03788591  0.03227114   0.04209262  0.04131946
```

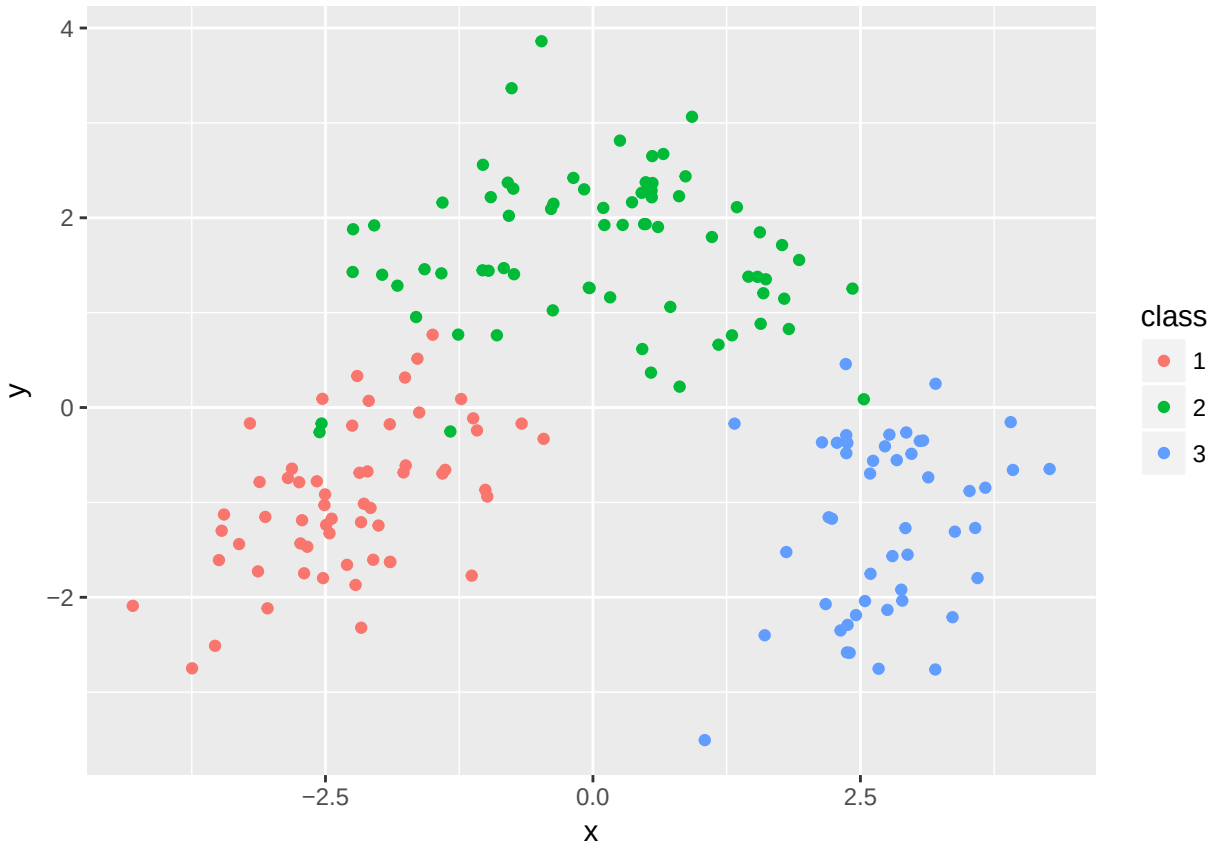
5 Canonical Discriminant Analysis

```
# wine predictor matrix
x <- as.matrix(wine[,-c(1)])
# wine labels
y <- wine$class

mu <- t(aggregate(x~y, FUN=mean)[,-c(1)])
avgs <- apply(x, 2, mean)

c1 <- apply(mu, 2, function(x) x - avgs)
c2 <- apply(c1, 1, function(x) (x * sqrt(as.numeric(table(y))/(length(y)-1))))

cwc <- c2 %*% solve(within_variance(x, y)) %*% t(c2)

wine.svd <- svd(cwc)
wine.uk <- solve(within_variance(x, y)) %*% t(c2) %*% wine.svd$v[,1:2]
```

```
cda <- data.frame(cbind(x %*% wine.uk, y))
colnames(cda) <- c("x", "y", "class")
ggplot(cda) +
  geom_point(aes(x,y, color=factor(class)))
```




```
wine.pca <- data.frame(princomp(scale(x))$scores[,1:2], y)
colnames(wine.pca) <- c("x", "y", "class")

ggplot(wine.pca) +
  geom_point(aes(x, y, color=class))
```



The pca plot is very similar to the Canonical Discriminant Analysis plot with the y inverted into a different direction(which is a non-issue since the -1*eigen vector is still an eigen vector)

```
cor(cda[,1:2], x)
```

```
##      alcohol     malic         ash alcalinity  magnesium     phenols
## x -0.2798969 0.4891760 -0.01918243  0.5299978 -0.1935927 -0.75482118
## y  0.8162180 0.3178155  0.40451247 -0.2148215  0.3355196  0.07008972
##    flavanoids nonflavanoids proanthocyanins     color        hue
## x -0.89849357    0.51522117     -0.53203867 0.3441133 -0.6840759
## y -0.02635971   -0.02507846     -0.05042644 0.7665231 -0.3780354
##     dilution    proline
## x -0.8503779 -0.6148947
## y -0.2031988  0.6717132
```

The correlation between u_k and the predictors shows roughly how much the predictor contributes to distinguishing the class.

```
maha.dist <- numeric()
for (i in 1:length(levels(y))) {
  maha.dist <- cbind(maha.dist, mahalanobis(x, as.numeric(aggregate(x~y, FUN=mean)[,-c(1)][i,]), cov(x)
}
colnames(maha.dist) <- levels(y)

head(maha.dist)
```

```
##             1         2        3
## [1,] 9.917064 15.787352 17.94001
## [2,] 7.812132 12.086900 15.17974
```

```
## [3,] 7.828124 11.936749 13.83748
## [4,] 8.575833 17.438575 15.92077
## [5,] 7.082722  8.303567 10.07814
## [6,] 3.934027 11.709468 11.73003
```

```
maha.predict <- factor(apply(maha.dist, 1, function(x) levels(y)[which.min(x)]))

table(maha.predict, y)
```

```
##             y
## maha.predict  1  2  3
##            1 59  0  0
##            2  0 71  0
##            3  0  0 48
```