Lecture 3

Statistical Learning, 2011

Niels Richard Hansen
October 12, 2011

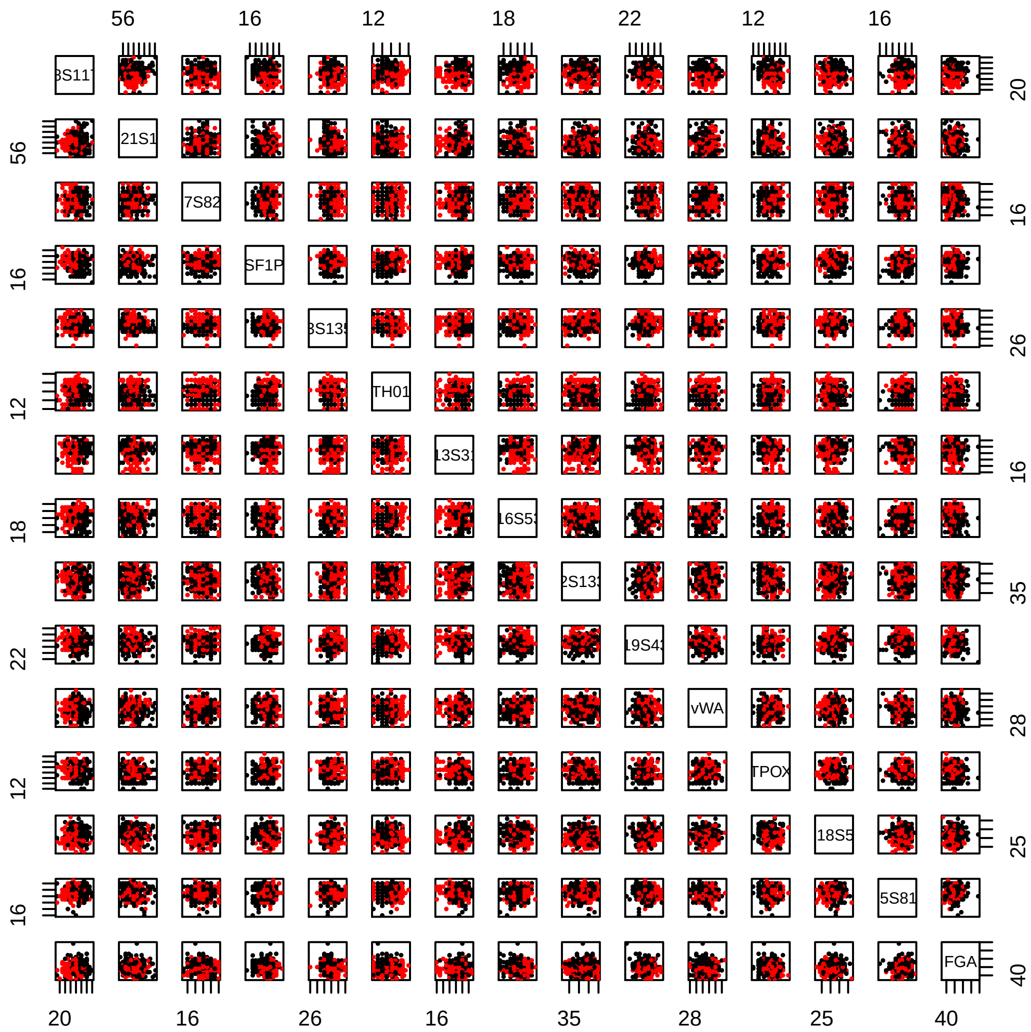
1 Solution – Prac 7

This document is produced with Sweave. For that reason all uses of ggplot2 functions for plotting need to be done inside a `print` – otherwise there will be no figures in the output.

```
> require(MASS) ## for lda
> require(ggplot2)
> require(Matrix) ## for an image of a matrix
> load("prac7.RData")
```

Question 1

```
> X <- as.matrix(prac7Train[, -16])
> y <- prac7Train[, 16]
> N <- length(y)
> p <- dim(X)[2]
> Nk <- as.numeric(table(y))
> pairs(X, col = y, pch = 20, cex = 0.4)
```



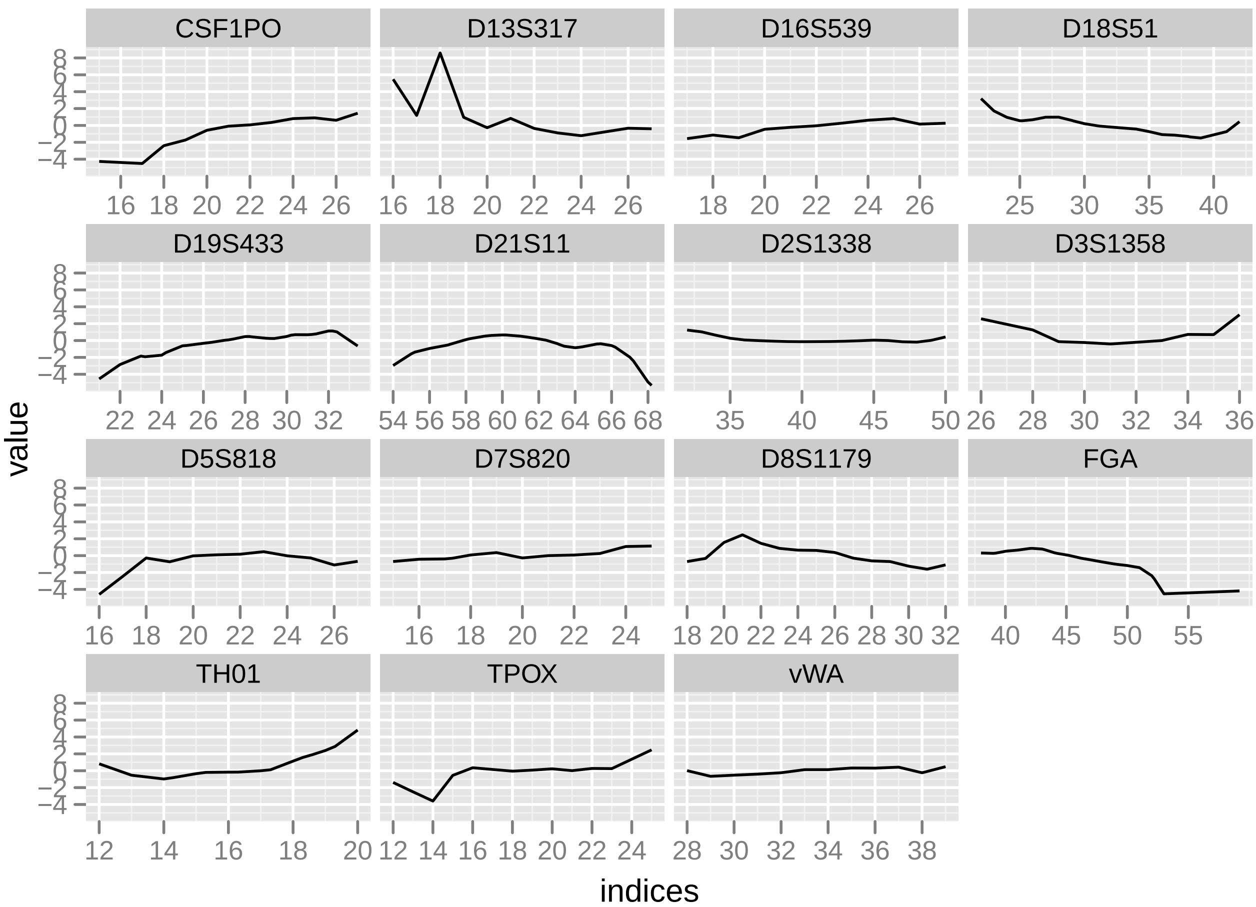
Question 2

First, we pretend that the X-variables are continuous and use `density` to automatically fit a smooth density to the marginal distributions. It is a little tricky because we need to get evaluations of the kernels at the points we are going to use later, and we need to fit the density for each column in the data frame

```
> values <- lapply(rbind(prac7Train[, -16],
+                        prac7Test[, -16]),
+                  function(x) sort(unique(x))
+                  )
> h <- list()
> for(i in seq_along(values)) {
+   dens1 <- density(X[y == "Caucasian", i])
```

```
+    fitx1 <- findInterval(values[[i]], dens1$x, all.inside = TRUE)
+    dens2 <- density(X[y == "African American", i])
+    fitx2 <- findInterval(values[[i]], dens2$x, all.inside = TRUE)
+    y1 <- as.table(dens1$y[fitx1])
+    names(y1) <- values[[i]]
+    y2 <- as.table(dens2$y[fitx2])
+    names(y2) <- values[[i]]
+    h[[i]] <- list(AA = y2, Caucasian = y1)
+ }
> names(h) <- names(values)
```

```
> logit <- sapply(h, function(x) log((x[[2]])/(x[[1]])))
> logitMelt <- melt(logit)
> print(qplot(x = indices, y = value, data = logitMelt, geom="line") +
+       facet_wrap(~ L1, scale = "free_x"))
```



Misclassification tables.

```
> yTest <- prac7Test[, 16]
```

```
> XTest <- prac7Test[, -16]
> predNaive <- function(X) {
+   yHat <- sapply(1:15,
+                  function(i) logit[[i]][as.character(X[, i])]
+                  )
+   log(Nk[2]/Nk[1]) + rowSums(yHat)
+ }
> predictionNaive <- predNaive(X)
> predictionNaiveTest <- predNaive(XTest)
```

And then the misclassification table on the test data.

```
> pred <- table(levels(y)[(predictionNaive > 0) + 1], y)
> print(pred)
```

```
                  y
                   African American Caucasian
  African American              161        18
  Caucasian                      12       158
```

```
> print(pred/sum(pred), digits=3)
```

```
                  y
                   African American Caucasian
  African American           0.4613    0.0516
  Caucasian                  0.0344    0.4527
```

```
> pred <- table(levels(yTest)[(predictionNaiveTest > 0) + 1], yTest)
> print(pred)
```

```
                  yTest
                   African American Caucasian
  African American               71        19
  Caucasian                      11        68
```

```
> print(pred/sum(pred), digits=3)
```

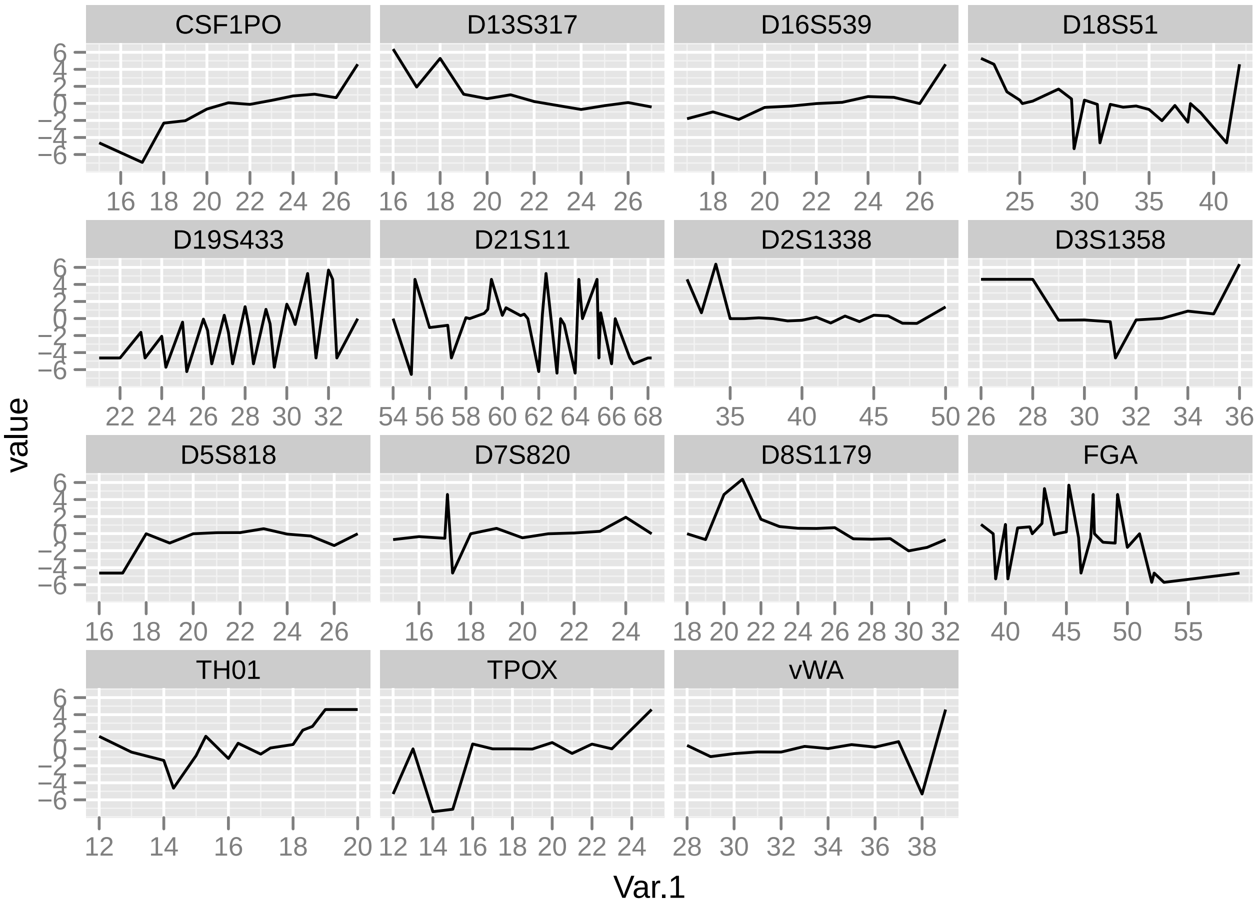
```
                  yTest
                   African American Caucasian
  African American           0.4201    0.1124
  Caucasian                  0.0651    0.4024
```

Next, we have to convert the numeric data into a form suitable for the tabulation of the discrete distributions. This is done by converting all the columns in the `X` matrix as well as the in the test data set into factors.

```
> allX <- as.data.frame(lapply(rbind(prac7Train[, -16],
+                                    prac7Test[, -16]),
+                              factor)
+                       )
> Xtrain <- allX[1:N, ]
```

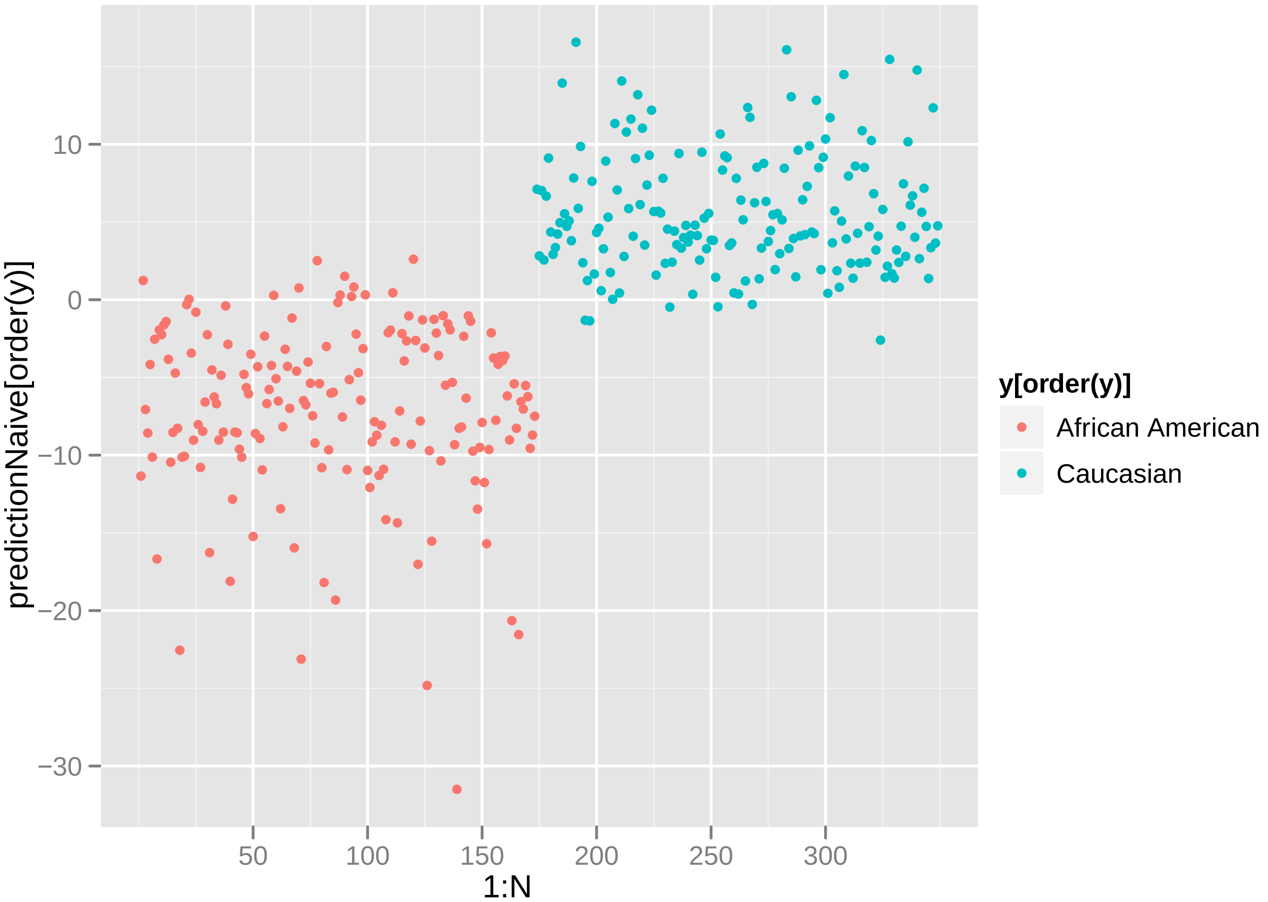
We compute the estimates of the marginal distributions and construct a plot. Along the way we modify the counts a little by adding an ϵ (pseudo-counts) and then normalize the tables to probability vectors. This is to prevent 0/0 and $\log(0)$ in subsequent computations.

```
> counts <- lapply(Xtrain, function(x) tapply(x, y, table))
> epsilon <- 0.01
> h <- lapply(counts,
+             function(x) {
+               lapply(x, function(x) (x+epsilon)/sum(x+epsilon))
+               }
+             )
> logit <- sapply(h, function(x) log((x[[2]])/(x[[1]])))
> logitMelt <- melt(logit)
> print(qplot(x = Var.1, y = value, data = logitMelt, geom="line") +
+       facet_wrap(~ L1, scale = "free_x"))
```



Computing the training and test error.

```
> predictionNaive <- predNaive(X)
> predictionNaiveTest <- predNaive(XTest)
> print(qplot(1:N, predictionNaive[order(y)],
+             shape=I(20), colour = y[order(y)]))
```



Computing the misclassification table on the training data.

```
> pred <- table(levels(y)[(predictionNaive > 0) + 1], y)
> print(pred)


                  y
                   African American Caucasian
  African American              161         6
  Caucasian                      12       170

> print(pred/sum(pred), digits=3)


                  y
                   African American Caucasian
  African American           0.4613    0.0172
  Caucasian                  0.0344    0.4871

> pred <- table(levels(yTest)[(predictionNaiveTest > 0) + 1], yTest)
```

```
> print(pred)
```

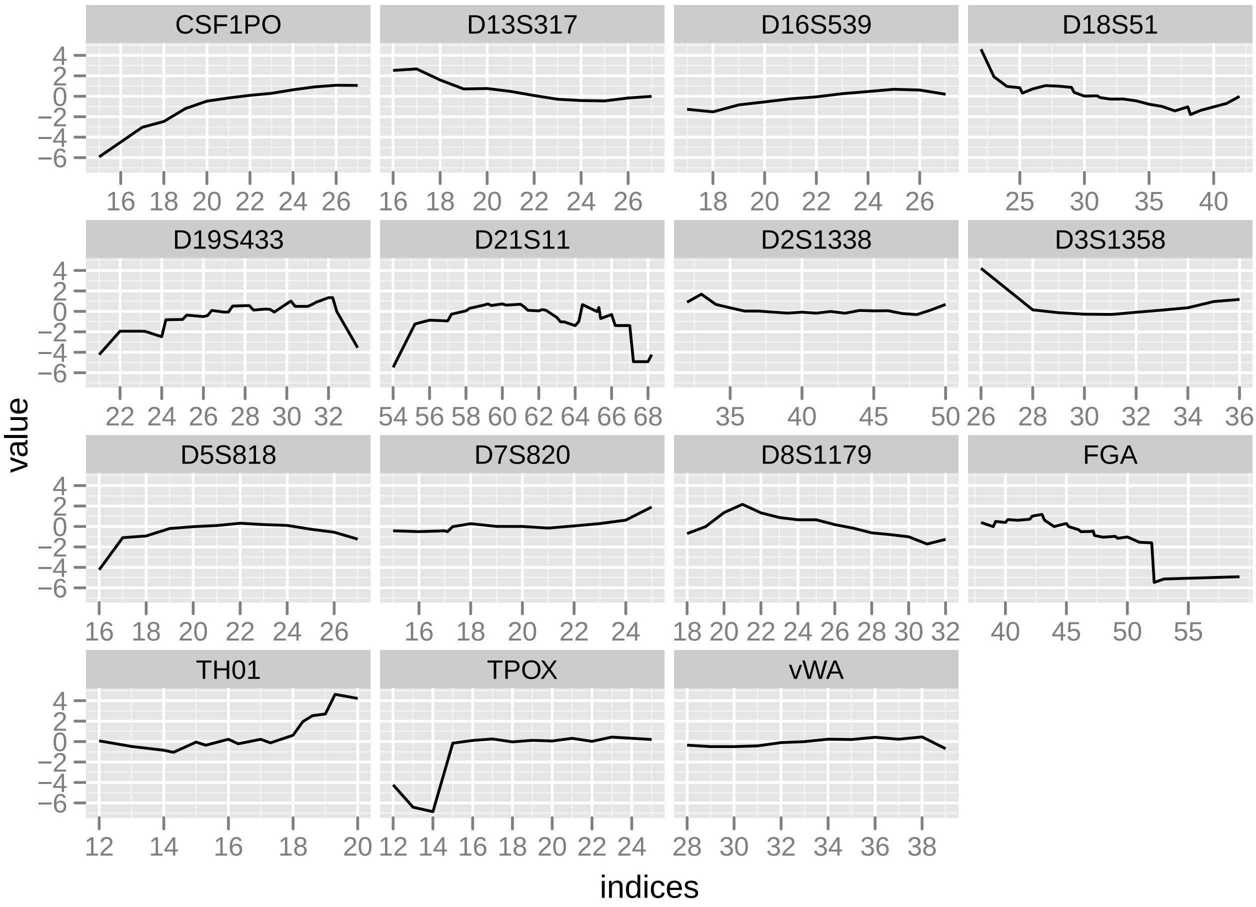
```
                  yTest
                   African American Caucasian
  African American               65        12
  Caucasian                      17        75
```

```
> print(pred/sum(pred), digits=3)
```

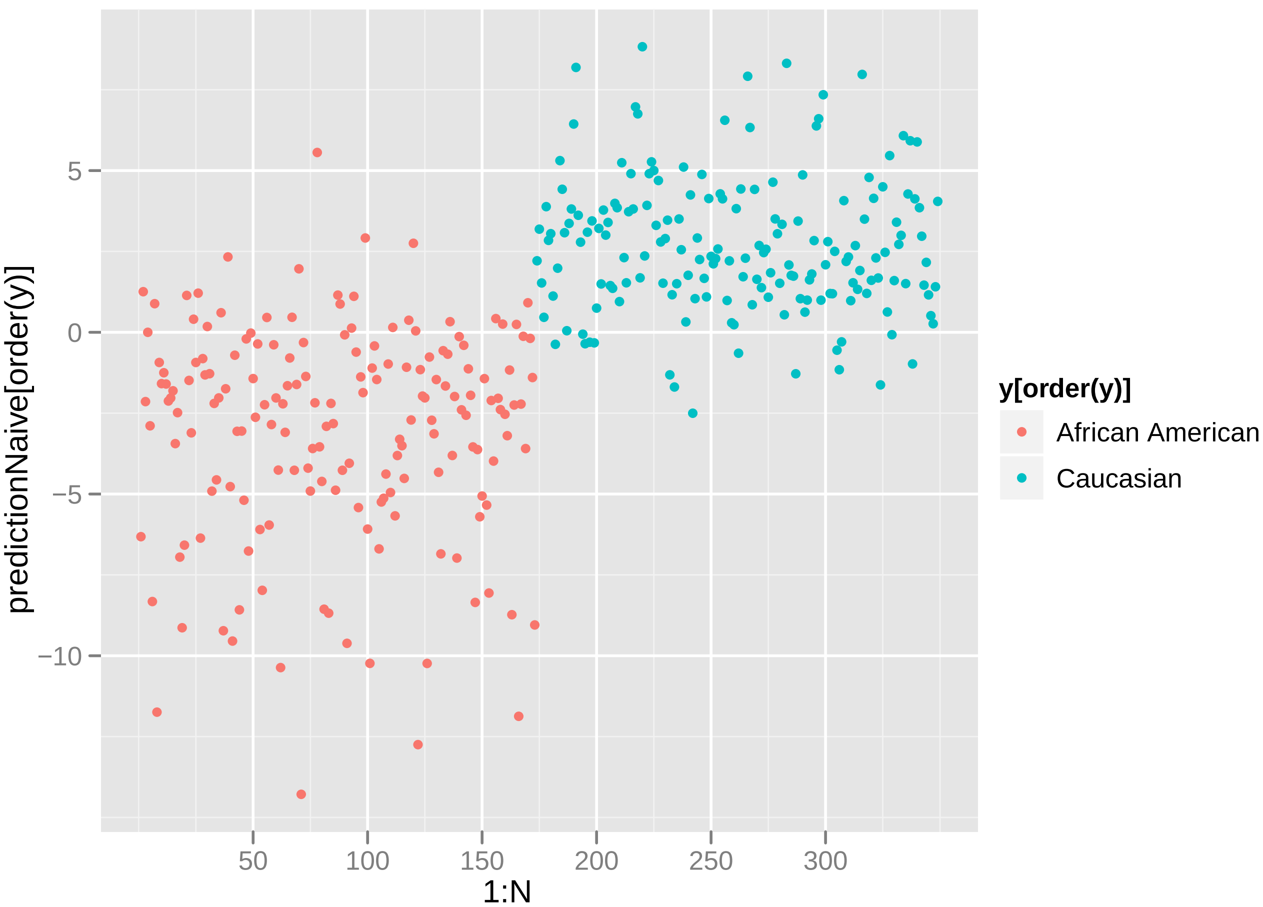
```
                  yTest
                   African American Caucasian
  African American            0.385     0.071
  Caucasian                   0.101     0.444
```

We can try to smooth the estimated probabilities for instance by convolving with a rectangular kernel.

```
> bw <- 1
> epsilon <- 0.01
> h <- lapply(counts,
+             function(x) {
+               lapply(x,
+                      function(x) {
+                        lab <- names(x)
+                        x <- convolve(c(rep(0, bw), x, rep(0, bw)),
+                                      rep(1/(2*bw+1), 2*bw+1), type =
"filter") + epsilon
+                        x <- as.table(x/sum(x))
+                        names(x) <- lab
+                        return(x)
+                        }
+                      )
+               }
+             )
> logit <- sapply(h, function(x) log((x[[2]])/(x[[1]])))
> logitMelt <- melt(logit)
> print(qplot(x = indices, y = value, data = logitMelt, geom="line") +
+       facet_wrap(~L1,scale="free_x"))
```




```
> predictionNaive <- predNaive(X)
> predictionNaiveTest <- predNaive(XTest)
> print(qplot(1:N, predictionNaive[order(y)], shape=I(20), colour =
y[order(y)]))
```



Computing the misclassification tables.

```
> pred <- table(levels(y)[(predictionNaive > 0) + 1], y)
> print(pred)


                  y
                   African American Caucasian
  African American              146        16
  Caucasian                      27       160

> print(pred/sum(pred), digits=3)


                  y
                   African American Caucasian
  African American           0.4183    0.0458
  Caucasian                  0.0774    0.4585

> pred <- table(levels(yTest)[(predictionNaiveTest > 0) + 1], yTest)
```

```
> print(pred)
```

```
                  yTest
                   African American Caucasian
  African American               60        20
  Caucasian                      22        67
```

```
> print(pred/sum(pred), digits=3)
```

```
                  yTest
                   African American Caucasian
  African American            0.355     0.118
  Caucasian                   0.130     0.396
```

Question 3

Computing group means and the estimate of the covariance matrix. The group means can be computed using "simple" R functions. For interactive use this may be useful, but it is generally slower than using linear algebra, which is to be preferred for programming.

```
> ### A simple solution
> groupMeans <-  apply(X, 2, function(x) tapply(x, y, mean))
> ### The linear algebra solution
> A <- model.matrix(~ y - 1)  ## The design matrix
> groupMeans <- 1/Nk * t(A) %*% X
> residuals <- X - groupMeans[y, ]
> ## Alternative is
> ## residuals <- X - A %*% groupMeans
> ## but the former works even if we did not compute the design matrix
> SigmaHat <- t(residuals) %*% residuals/(N - 3)
> ## Outer product (all cross-products)
> sHat <- sqrt(diag(SigmaHat)) %o% sqrt(diag(SigmaHat))
> corHat <- SigmaHat/sHat
```

	D8S1179	D21S11	D7S820	CSF1PO	D3S1358	TH01	D13S317	D16S539	D2S1338	D19S433	vWA	TPOX	D18S51	D5S818	FGA
yAfrican American	27.17	59.57	19.70	21.28	31.70	15.12	23.28	21.68	41.74	26.91	32.87	17.76	32.40	23.24	45.82
yCaucasian	25.57	59.79	20.02	22.70	32.18	15.95	22.17	22.80	41.43	27.84	33.40	18.37	30.23	23.02	43.93

Table 1: Group Means

	D8S1179	D21S11	D7S820	CSF1PO	D3S1358	TH01	D13S317	D16S539	D2S1338	D19S433	vWA	TPOX	D18S51	D5S818	FGA
D8S1179	1.00	0.00	0.05	-0.02	-0.02	0.09	0.02	0.02	-0.07	0.08	0.00	0.00	-0.05	0.03	0.00
D21S11	0.00	1.00	0.08	-0.05	-0.04	-0.02	-0.04	-0.03	0.06	-0.07	-0.00	-0.04	-0.02	-0.09	-0.06
D7S820	0.05	0.08	1.00	0.07	-0.09	0.01	0.09	-0.02	-0.07	0.03	0.00	0.02	0.08	0.04	-0.00
CSF1PO	-0.02	-0.05	0.07	1.00	-0.11	0.03	0.08	0.03	-0.05	-0.05	-0.03	0.01	0.04	0.10	0.09
D3S1358	-0.02	-0.04	-0.09	-0.11	1.00	-0.04	-0.01	-0.05	0.03	-0.03	-0.04	0.00	0.05	-0.03	-0.09
TH01	0.09	-0.02	0.01	0.03	-0.04	1.00	-0.06	0.07	-0.07	-0.07	-0.02	0.06	0.01	0.00	0.02
D13S317	0.02	-0.04	0.09	0.08	-0.01	-0.06	1.00	0.02	-0.03	-0.04	-0.03	-0.03	0.01	-0.03	0.01
D16S539	0.02	-0.03	-0.02	0.03	-0.05	0.07	0.02	1.00	-0.01	0.04	0.05	0.03	-0.01	0.01	-0.04
D2S1338	-0.07	0.06	-0.07	-0.05	0.03	-0.07	-0.03	-0.01	1.00	0.06	0.00	-0.12	-0.01	-0.05	0.06
D19S433	0.08	-0.07	0.03	-0.05	-0.03	-0.07	-0.04	0.04	0.06	1.00	-0.08	0.02	0.06	0.03	-0.03
vWA	0.00	-0.00	0.00	-0.03	-0.04	-0.02	-0.03	0.05	0.00	-0.08	1.00	0.02	-0.00	-0.04	0.05
TPOX	0.00	-0.04	0.02	0.01	0.00	0.06	-0.03	0.03	-0.12	0.02	0.02	1.00	0.01	-0.04	-0.01
D18S51	-0.05	-0.02	0.08	0.04	0.05	0.01	0.01	-0.01	-0.01	0.06	-0.00	0.01	1.00	-0.01	-0.01
D5S818	0.03	-0.09	0.04	0.10	-0.03	0.00	-0.03	0.01	-0.05	0.03	-0.04	-0.04	-0.01	1.00	0.02
FGA	0.00	-0.06	-0.00	0.09	-0.09	0.02	0.01	-0.04	0.06	-0.03	0.05	-0.01	-0.01	0.02	1.00

Table 2: Sigmahat

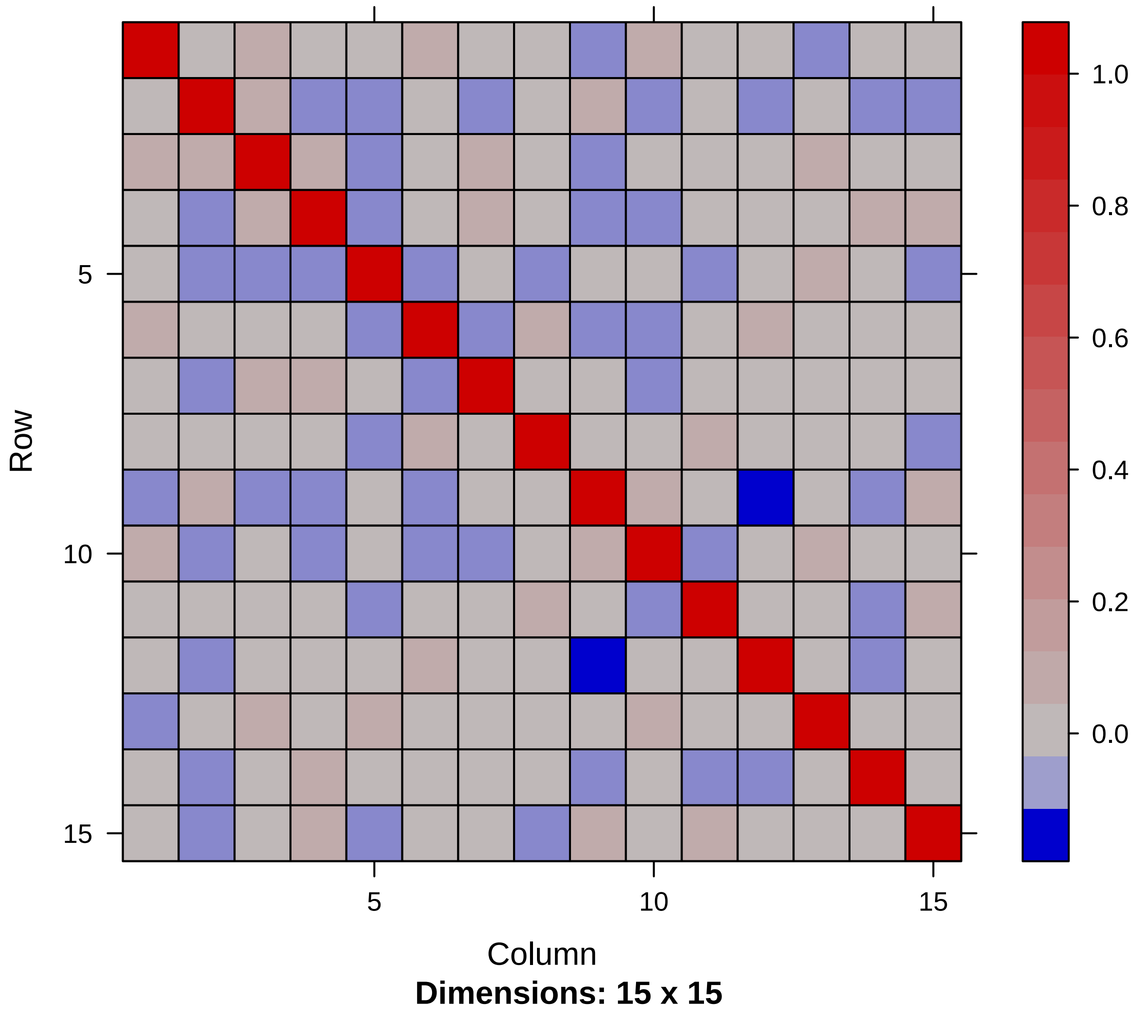


Figure 1: The correlation matrix

Question 4

Fitting lda and logistic regression models.

```
> Xlda <- lda(population ~ ., data = prac7Train)
> Xglm <- glm(population ~ ., data = prac7Train, family = binomial)
```

Computing misclassification tables for LDA.

```
> pred <- table(predict(Xlda)$class, y)
> print(pred)
```

```
                   y
                    African American Caucasian
  African American               143        22
  Caucasian                       30       154
```

```
> print(pred/sum(pred),digits=3)
```

```
                   y
                    African American Caucasian
  African American             0.410     0.063
  Caucasian                    0.086     0.441
```

```
> pred <- table(predict(Xlda, XTest)$class, yTest)
> print(pred)
```

```
                   yTest
                    African American Caucasian
  African American                61        19
  Caucasian                       21        68
```

```
> print(pred/sum(pred), digits=3)
```

```
                   yTest
                    African American Caucasian
  African American             0.361     0.112
  Caucasian                    0.124     0.402
```

Computing misclassification tables for logistic regression.

```
> predGlm <- predict(Xglm, type = "response")
> yHat <- levels(y)[as.numeric(predGlm > 0.5) + 1]
> pred <- table(yHat, y)
> print(pred)
```

```
                   y
yHat                African American Caucasian
  African American               144        24
  Caucasian                       29       152
```

```
> print(pred/sum(pred),digits=3)
```

```
                   y
yHat                African American Caucasian
  African American            0.4126    0.0688
  Caucasian                   0.0831    0.4355
```

```
> predGlm <- predict(Xglm, XTest, type = "response")
> yHat <- levels(y)[as.numeric(predGlm > 0.5) + 1]
> pred <- table(yHat, yTest)
> print(pred)
```

```
                  yTest
yHat               African American Caucasian
  African American               65        21
  Caucasian                      17        66
```

```
> print(pred/sum(pred), digits=3)
```

```
                  yTest
yHat               African American Caucasian
  African American            0.385     0.124
  Caucasian                   0.101     0.391
```