# 1 Solution – Assignment 1

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.

```
> require(MASS)
> require(ggplot2)
> setwd("~/courses/statlearn/statlearn_2009/R/")
> load("Assignment1.RData")
```

## 1.1 Question 1

```
> X <- as.matrix(Assignment1Train[, -16])
> y <- Assignment1Train[, 16]
```

Computing group means and the estimate of the covariance matrix.

```
> groupMeans <- apply(X, 2, function(x) tapply(x, y, mean))
> SigmaHat <- t(X - groupMeans[y, ]) %*% (X - groupMeans[y, ])/(dim(X)[1] -
+     3)
```

| | D8S1179 | D21S11 | D7S820 | CSF1PO | D3S1358 | TH01 | D13S317 | D16S539 | D2S1338 | D19S433 | vWA | TPOX | D18S51 | D5S818 | FGA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| African American | 27.16 | 59.57 | 19.69 | 21.16 | 31.69 | 15.11 | 23.29 | 21.70 | 41.50 | 26.92 | 32.87 | 17.74 | 32.22 | 23.23 | 45.83 |
| Caucasian | 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 |
| Hispanic | 25.99 | 60.20 | 20.59 | 22.30 | 32.01 | 15.58 | 21.68 | 22.41 | 40.16 | 27.56 | 33.04 | 18.81 | 30.61 | 22.61 | 45.30 |

Table 1: Group Means

| | D8S1179 | D21S11 | D7S820 | CSF1PO | D3S1358 | TH01 | D13S317 | D16S539 | D2S1338 | D19S433 | vWA | TPOX | D18S51 | D5S818 | FGA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D8S1179 | 5.11 | 0.01 | 0.07 | -0.05 | -0.06 | 0.29 | 0.26 | 0.02 | -0.65 | 0.17 | -0.17 | 0.31 | -0.26 | 0.13 | -0.02 |
| D21S11 | 0.01 | 6.06 | 0.30 | -0.14 | -0.21 | -0.05 | -0.29 | -0.20 | 0.13 | -0.16 | 0.07 | -0.03 | 0.10 | -0.12 | -0.15 |
| D7S820 | 0.07 | 0.30 | 3.63 | 0.38 | -0.38 | -0.14 | 0.17 | 0.00 | -0.01 | 0.14 | 0.02 | 0.14 | 0.59 | -0.03 | 0.17 |
| CSF1PO | -0.05 | -0.14 | 0.38 | 4.14 | -0.30 | 0.14 | 0.10 | -0.04 | 1.46 | -0.25 | 0.04 | 0.17 | 1.58 | 0.27 | 0.30 |
| D3S1358 | -0.06 | -0.21 | -0.38 | -0.30 | 2.55 | -0.02 | -0.12 | -0.18 | 0.44 | -0.00 | -0.19 | -0.01 | 0.46 | -0.03 | -0.37 |
| TH01 | 0.29 | -0.05 | -0.14 | 0.14 | -0.02 | 3.39 | -0.10 | 0.14 | -0.32 | -0.16 | -0.10 | 0.25 | 0.19 | 0.04 | -0.05 |
| D13S317 | 0.26 | -0.29 | 0.17 | 0.10 | -0.12 | -0.10 | 4.87 | 0.29 | -0.20 | -0.06 | -0.17 | -0.07 | -0.01 | 0.12 | -0.07 |
| D16S539 | 0.02 | -0.20 | 0.00 | -0.04 | -0.18 | 0.14 | 0.29 | 4.09 | -0.11 | 0.23 | 0.22 | -0.05 | -0.16 | 0.16 | -0.05 |
| D2S1338 | -0.65 | 0.13 | -0.01 | 1.46 | 0.44 | -0.32 | -0.20 | -0.11 | 20.00 | -0.07 | 0.13 | -0.76 | 3.03 | -0.00 | -0.04 |
| D19S433 | 0.17 | -0.16 | 0.14 | -0.25 | -0.00 | -0.16 | -0.06 | 0.23 | -0.07 | 2.86 | -0.28 | -0.10 | 0.01 | -0.10 | -0.24 |
| vWA | -0.17 | 0.07 | 0.02 | 0.04 | -0.19 | -0.10 | -0.17 | 0.22 | 0.13 | -0.28 | 3.95 | -0.08 | -0.31 | -0.17 | 0.32 |
| TPOX | 0.31 | -0.03 | 0.14 | 0.17 | -0.01 | 0.25 | -0.07 | -0.05 | -0.76 | -0.10 | -0.08 | 5.16 | 0.37 | -0.06 | -0.05 |
| D18S51 | -0.26 | 0.10 | 0.59 | 1.58 | 0.46 | 0.19 | -0.01 | -0.16 | 3.03 | 0.01 | -0.31 | 0.37 | 13.21 | 0.26 | -0.00 |
| D5S818 | 0.13 | -0.12 | -0.03 | 0.27 | -0.03 | 0.04 | 0.12 | 0.16 | -0.00 | -0.10 | -0.17 | -0.06 | 0.26 | 3.07 | 0.01 |
| FGA | -0.02 | -0.15 | 0.17 | 0.30 | -0.37 | -0.05 | -0.07 | -0.05 | -0.04 | -0.24 | 0.32 | -0.05 | -0.00 | 0.01 | 9.01 |

Table 2: Sigmahat

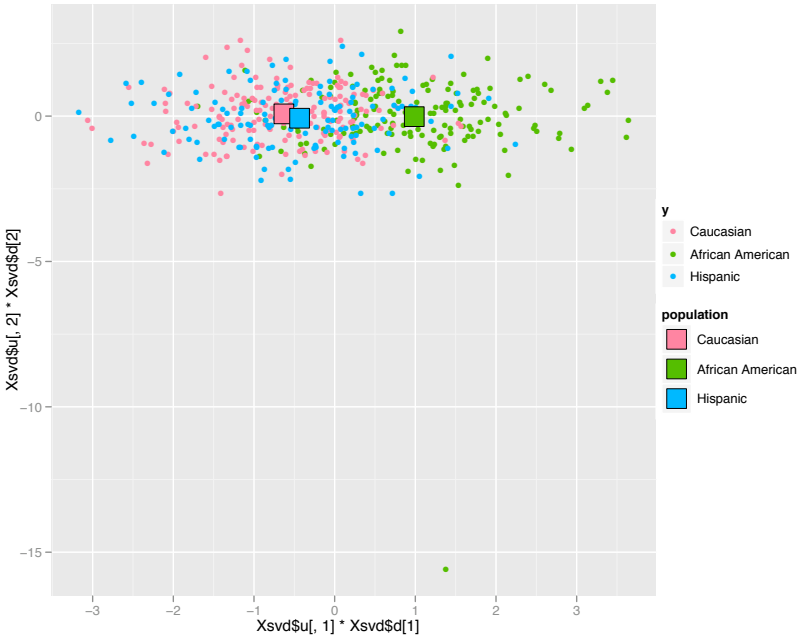
For later use in subsequent plots we compute the group means after a centering and scaling.

```
> groupMeans <- apply(scale(X), 2, function(x) tapply(x, y, mean))
```

## 1.2 Question 2

We make this plot after centering (compulsory) and scaling (optional, but generally recommended).

```
> Xsvd <- svd(scale(X))
> population <- levels(Assignment1Train[, 16])[c(2, 1, 3)]
> print(qplot(Xsvd$u[, 1] * Xsvd$d[1], Xsvd$u[, 2] * Xsvd$d[2],
+     colour = y) + geom_point(aes(groupMeans[c(2, 1, 3), ] %*%
+     Xsvd$v[, 1], groupMeans[c(2, 1, 3), ] %*% Xsvd$v[, 2], fill = population),
+     colour = I("black"), shape = 22, size = 8))
```

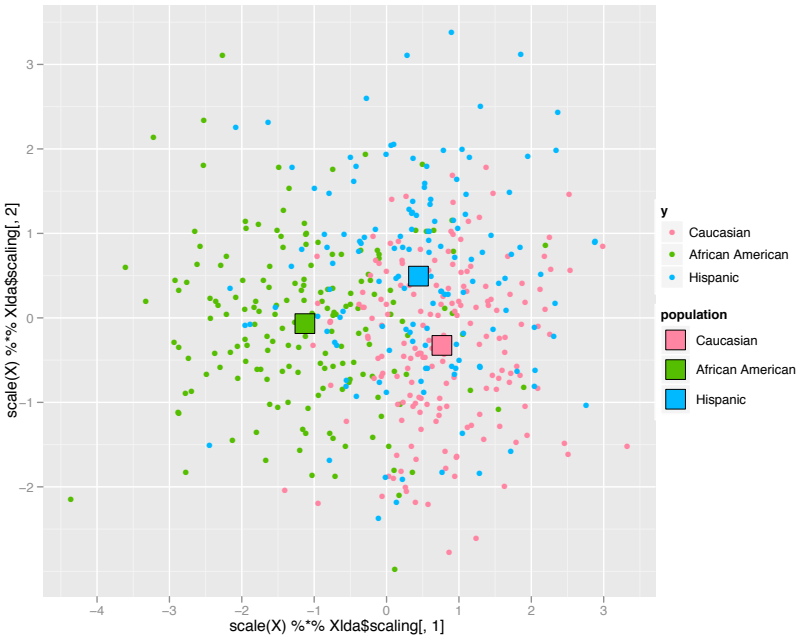



## 1.3 Question 3

We do this using centered and scaled $X$'s. The actual lda classifier is not affected by this (just remember that new observations should be centered and scaled using the same estimated means and variances) but the resulting plots are then in a standardized form, which is common.

```
> Xlda <- lda(scale(X), y)
> print(qplot(scale(X) %*% Xlda$scaling[, 1], scale(X) %*% Xlda$scaling[,
+     2], colour = y) + geom_point(aes(groupMeans[c(2, 1, 3), ] %*%
```

```
+     Xlda$scaling[, 1], groupMeans[c(2, 1, 3), ] %*% Xlda$scaling[,
+     2], fill = population), colour = I("black"), shape = 22,
+     size = 8))
```




## 1.4 Question 4

```
> X <- X[y != "Hispanic", ]
> y <- y[y != "Hispanic", drop = TRUE]
> Xlda <- lda(X, y)
```

Computing misclassification tables.

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

```
                 y
                  African American Caucasian
  African American              144        22
  Caucasian                      30       154
```

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

```
                 y
                  African American Caucasian
  African American           0.4114    0.0629
  Caucasian                  0.0857    0.4400
```

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])
```

```
[1] 52
```

```
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)
```

```
[1] 0.149
```

## 1.5 Question 5

```
> Xglm <- glm(population ~ ., data = cbind(X, data.frame(population = y)),
+     family = binomial)
```

Computing misclassification tables.

```
> pred <- table(levels(y)[round(predict(Xglm, type = "response")) +
+     1], y)
> print(pred)
```

```
                 y
                  African American Caucasian
  African American              145        24
  Caucasian                      29       152
```

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

```
                 y
                  African American Caucasian
  African American           0.4143    0.0686
  Caucasian                  0.0829    0.4343
```

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])
```

```
[1] 53
```

```
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)
```

```
[1] 0.151
```

## 1.6 Question 6

```
> mu1Hat <- Xlda$means[2, ]
> mu0Hat <- Xlda$means[1, ]
> residuals <- X - Xlda$means[y, ]
> SigmaHat <- t(residuals) %*% residuals/(dim(X)[1] - 2)
> tauHatBeta0 <- log(Xlda$prior[2]/Xlda$prior[1]) + (t(mu0Hat) %*%
+     solve(SigmaHat, mu0Hat) - t(mu1Hat) %*% solve(SigmaHat, mu1Hat))/2
> tauHatBeta <- solve(SigmaHat, mu1Hat - mu0Hat)
> hat <- data.frame(tauHat = c(tauHatBeta0, tauHatBeta), Hat = Xglm$coef)
> print(hat)
```

```
          tauHat      Hat
        -18.1177 -20.9366
D8S1179  -0.3779  -0.3948
D21S11    0.0474   0.0897
D7S820    0.1330   0.1534
CSF1PO    0.5227   0.5851
D3S1358   0.3125   0.2790
TH01      0.2934   0.2980
D13S317  -0.2497  -0.2969
D16S539   0.2369   0.2105
D2S1338  -0.0176  -0.0169
D19S433   0.4022   0.4426
vWA       0.1819   0.2094
TPOX      0.0755   0.0922
D18S51   -0.2528  -0.2391
D5S818   -0.1171  -0.1152
FGA      -0.2237  -0.2540
```

When we do two class classification the scalings from lda and the $\hat{\beta}$ coefficient from logistic regression are not directly comparable, but tauHatBeta above, which is comparable with $\hat{\beta}$, is in fact proportional to the scaling. The constant of proportionality comes from two facts. First, there is a different "centering" used for the scaling in the affine space spanned by $\hat{\mu}_1$ and $\hat{\mu}_2$, and second, the scaling vector is by definition of unit length when we use the inner product given by $\hat{\Sigma}$.

```
> tildeBeta <- solve(SigmaHat, mu1Hat - (Nk[2] * mu1Hat + Nk[1] *
+     mu0Hat)/sum(Nk))
> tildeBeta <- tildeBeta/sqrt(tildeBeta %*% SigmaHat %*% tildeBeta)
> lda(X, y)$scaling/tildeBeta
```

```
        LD1
D8S1179   1
D21S11    1
D7S820    1
CSF1PO    1
D3S1358   1
TH01      1
```

```
D13S317    1
D16S539    1
D2S1338    1
D19S433    1
vWA        1
TPOX       1
D18S51     1
D5S818     1
FGA        1
```

If you turn to classification problems with the number of groups $K \geq 3$ the $\hat{\beta}_1, \ldots, \hat{\beta}_{K-1}$ vectors from multinomial regression are in general not comparable in any direct way to the scalings computed from lda. In this situation the only way to get something out of lda that is comparable to the estimates from the multinomial regression is to compute the plug-in estimates of the parameter function $\tau$.

```
> trainPosterior <- data.frame(logis = predict(Xglm, type = "response"),
+     lda = predict(Xlda)$posterior[, 2])[order(y), ]
> print(ggplot(data = melt(cbind(data.frame(index = 1:(dim(X)[1]),
+     diff = trainPosterior[, 1] - trainPosterior[, 2] + 0.5),
+     trainPosterior), id = "index"), aes(x = index, y = value,
+     col = variable)) + geom_point())
```

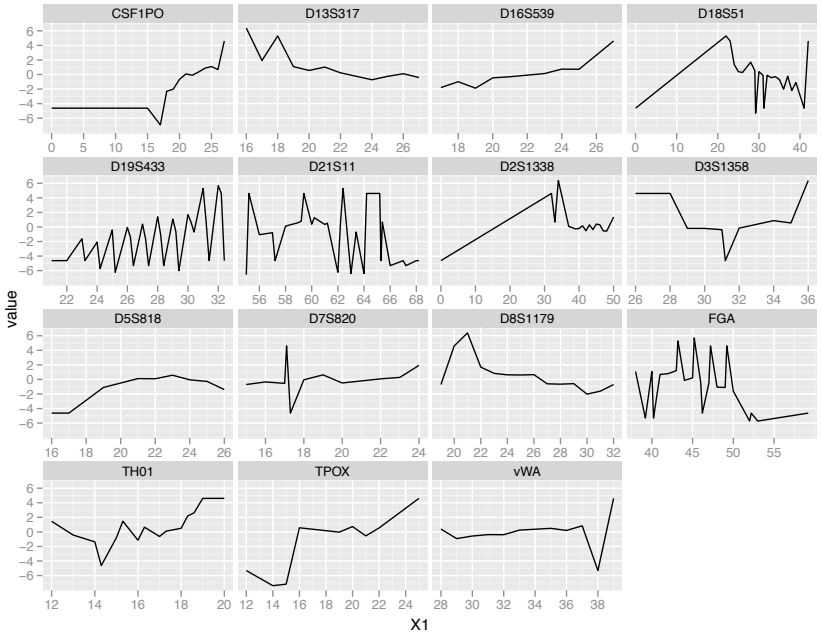

## 1.7 Question 7

```
> allLevels <- levels(factor(c(unlist(Assignment1Train[, -16]),
+     unlist(Assignment1Test[, -16]))))
```

```
> Assignment1TrainMod <- data.frame(lapply(as.data.frame(X), function(x) factor(x,
+     levels = allLevels)))
```

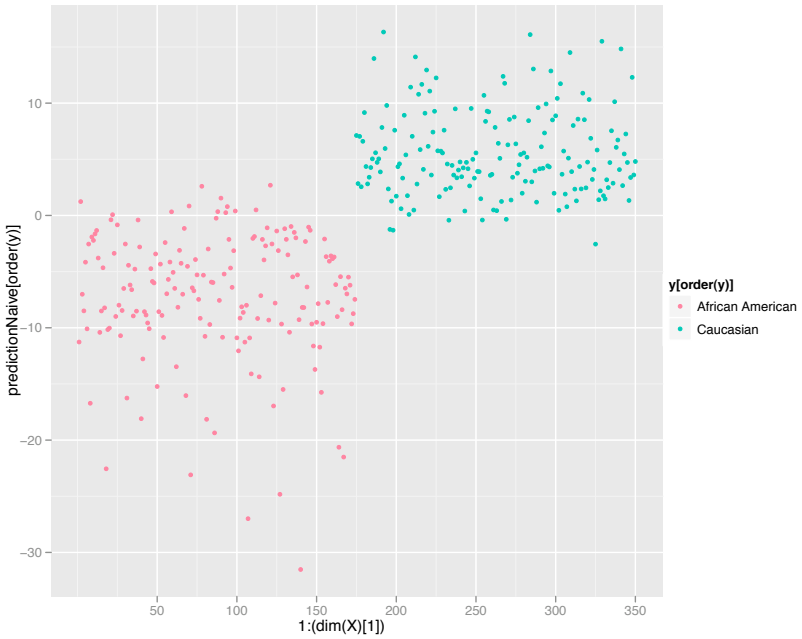
We compute the estimates of the marginal distributions.

```
> counts <- lapply(Assignment1TrainMod, 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 = X1, y = value, data = logitMelt[abs(logitMelt$value) >
+     0.02, ], geom = "line") + facet_wrap(~X2, scale = "free_x"))
```




Computing the training error.

```
> Nk <- table(y)
> intercept <- log(Nk[2]/Nk[1])
> predictionNaive <- apply(X, 1, function(x) intercept + sum(diag(logit[as.character(x),
+     ])))
> print(qplot(1:(dim(X)[1]), 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              162         6
  Caucasian                      12       170
```

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

```
                y
                 African American Caucasian
  African American           0.4629    0.0171
  Caucasian                  0.0343    0.4857
```

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])
```

```
[1] 18
```

```
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)
```

```
[1] 0.0514
```

## 1.8 Question 8

```
> yTest <- as.factor(as.vector(Assignment1Test[, 16]))
> XTest <- Assignment1Test[, -16]
> predictionNaive <- apply(XTest, 1, function(x) intercept + sum(diag(logit[as.character(x),
+     ])))
```

Naive test error:

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

```
                yTest
                 African American Caucasian
  African American               65        13
  Caucasian                      17        74
```

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

```
                yTest
                 African American Caucasian
  African American           0.3846    0.0769
  Caucasian                  0.1006    0.4379
```

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])
```

```
[1] 30
```

```
> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)
```

```
[1] 0.178
```

LDA test error:

```
> 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
```

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])

[1] 40

> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)

[1] 0.237
```

Glm test error:

```
> pred <- table(levels(y)[round(predict(Xglm, XTest, type = "response")) +
+     1], yTest)
> print(pred)

                yTest
                 African American Caucasian
  African American               65        21
  Caucasian                      17        66

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

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

Then computing the number of misclassifications and the relative number of misclassifications.

```
> print(pred[1, 2] + pred[2, 1])

[1] 38

> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)

[1] 0.225
```

The Naive Bayes method seems to be able to make the best predictions - also on the test data. The method does adapt a lot to the concrete dataset and we see a considerable increase in the estimated expected prediction error on the test data compared to misclassification rate on the training data for the Naive Bayes procedure. However, it also seems that the Naive Bayes procedure captures some important, non-linear effects that improves on the prediction.

Some have noted that there is an implementation of "naive Bayes" in the e1071 library. In effect this is a diagonal QDA – a quadratic discriminant analysis but where the covariance matrices are assumed diagonal. It turns out that is method performs surprisingly well and seems to have the smallest generalization error. It does not suffer from the many free parameters that are in my hand-crafted naive Bayes procedure above, but it incorporates quadratic terms, which seem to be very important.

## 1.9 Generalizations

If we think of include quadratic terms in a more direct way, we can do so, either explicitly using a logistic regression model or though more general non-linear effects in a generalized additive model. Using logistic regression with quadratic terms included:

```
> form <- as.formula(paste("population~", paste(colnames(X), collapse = "+"),
+     "+", paste("I(", colnames(X), "^2)", sep = "", collapse = "+")))
> XQglm <- glm(form, data = cbind(X, data.frame(population = y)),
+     family = binomial)

> pred <- table(levels(y)[round(predict(XQglm, XTest, type = "response")) +
+     1], yTest)
> print(pred)

                yTest
                 African American Caucasian
  African American               66        18
  Caucasian                      16        69

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

                yTest
                 African American Caucasian
  African American           0.3905    0.1065
  Caucasian                  0.0947    0.4083

> print(pred[1, 2] + pred[2, 1])

[1] 34

> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)

[1] 0.201
```

Using a generalized additive model

```
> require(gam)
> form <- as.formula(paste("population~", paste("s(", colnames(X),
+     ",df=3)", sep = "", collapse = "+")))
> Xgam <- gam(form, data = cbind(X, data.frame(population = y)),
+     family = binomial)

> pred <- table(levels(y)[round(predict(Xgam, XTest, type = "response")) +
+     1], yTest)
> print(pred)

                 yTest
                  African American Caucasian
  African American               66        15
  Caucasian                      16        72

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

                 yTest
                  African American Caucasian
  African American           0.3905    0.0888
  Caucasian                  0.0947    0.4260

> print(pred[1, 2] + pred[2, 1])

[1] 31

> print((pred[1, 2] + pred[2, 1])/sum(pred), digits = 3)

[1] 0.183
```