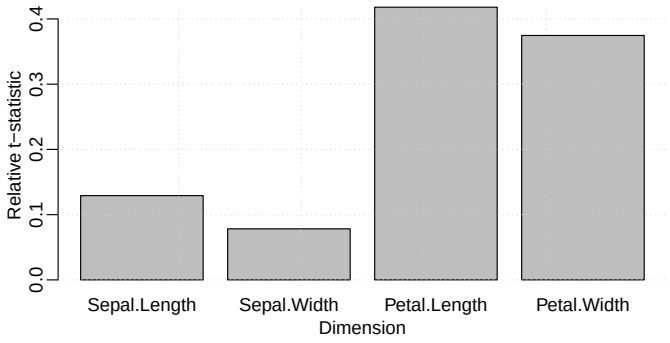
Feature selection

Evan Cummings
CSCI 548 – Douglas W. Raiford – Pattern Recognition

November 16, 2016

1 iris data



	full	***t*-test**	**seq-fwrd**
rand. acc.	97.4%	97.4%	97.4%
n-way acc.	98%	96%	96%
dims	all	3,4	1,3
J_1	6.50	15.2	7.41
J_2	41.7	22.4	24.6
J_3	31.8	19.4	22.9

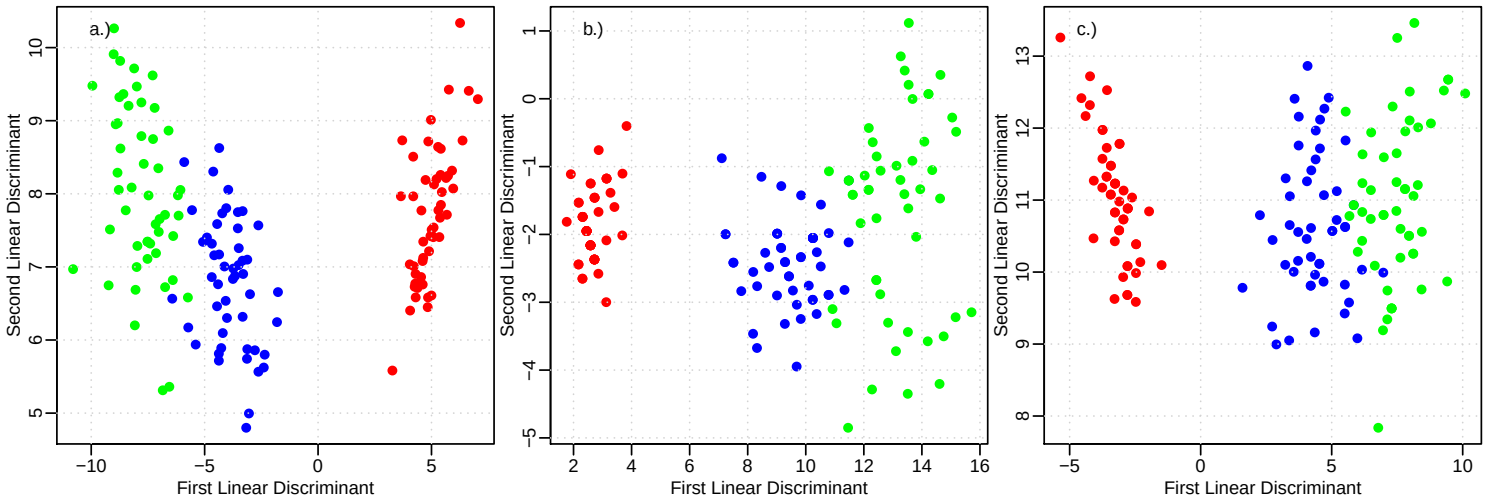
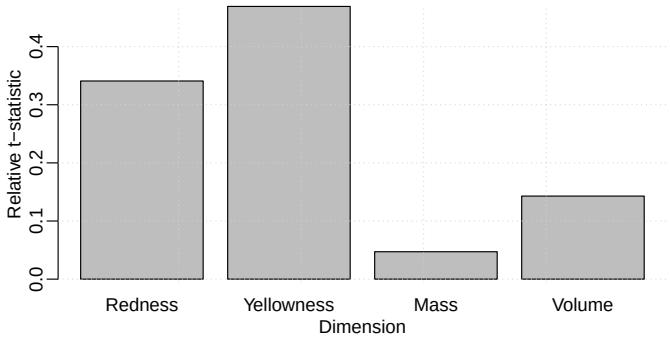


Figure 1: The `iris` data projected onto the first two linear discriminants using: a.) all the dimensions; b.) two best dimensions resulting from the relative-t-test (top left barplot); and c.), two best dimensions determined from the sequential-forward algorithm. The random 75% training data score, leave-one-out-cross-validation score, best dimension indicies, and J-scores are provided in the above right table for the full model, t-test-best-dimensions model, and sequential-forward-best-dimensions model. Note that while both the t-test and sequential-forward models resulted in identical n-way-cross-validation scores, the J_2 and J_3 values associated with the sequential-forward model are slightly improved.

2 fruit data



	full	***t*-test**	**seq-fwrd**
rand. acc.	96.4%	95.6%	95.6%
n-way acc.	95.1%	94.1%	94.1%
dims	all	1,2	1,2
J_1	0.19	9.53	9.53
J_2	33.2	21.0	21.0
J_3	21.7	20.0	20.0

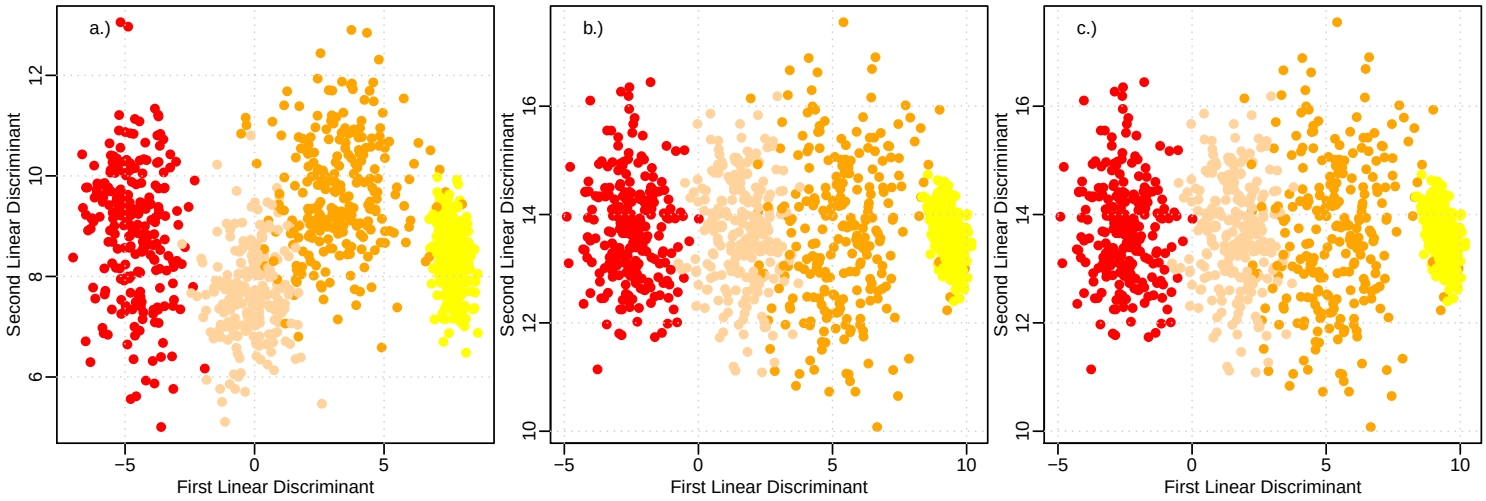
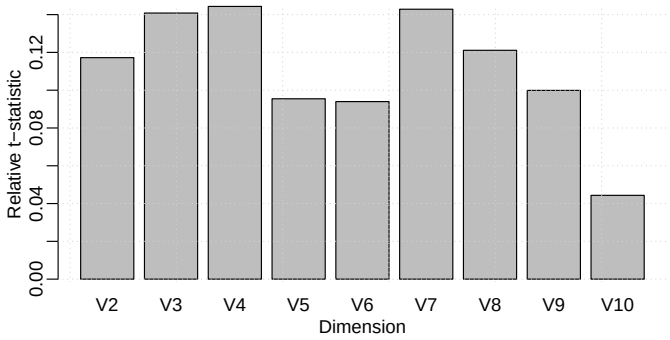


Figure 2: The `fruit` data projected onto the first two linear discriminants using: a.) all the dimensions; b.) two best dimensions resulting from the relative-t-test (top left barplot); and c.), two best dimensions determined from the sequential-forward algorithm, in this case identical to those of the t-test. The random 75% training data score, leave-one-out-cross-validation score, best dimension indicies, and J-scores are provided in the above right table for the full model, t-test-best-dimensions model, and sequential-forward-best-dimensions model. Note that the reduced-dimensional model performed slightly worse than the full-dimensional model.

3 tumor data



	full	***t*-test**	**seq-fwrd**
rand. acc.	96%	93.14%	97.1%
n-way acc.	95.7%	94.7%	95.1%
dims	all	2,3,6	1,3,6
J_1	1.30	2.01	1.67
J_2	6.11	4.96	5.23
J_3	5.11	3.96	4.23

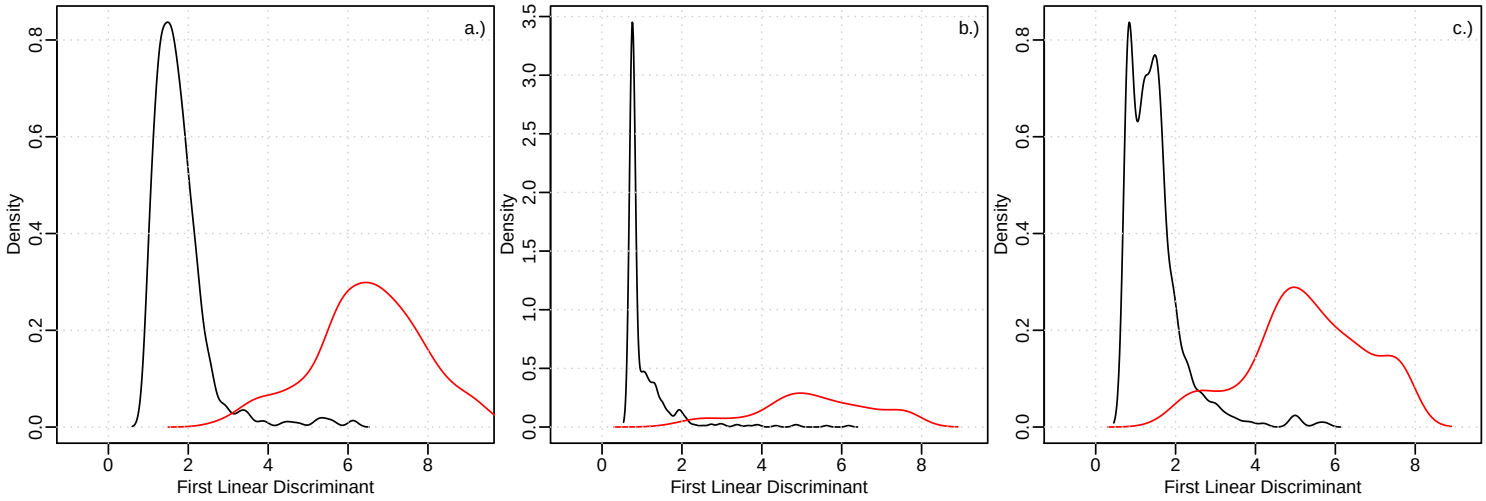
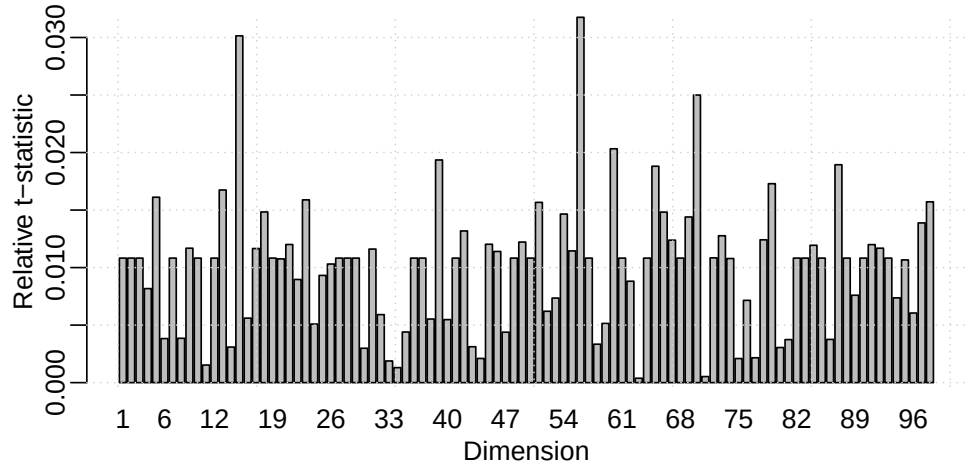


Figure 3: The density of the `tumor` data projected onto the first linear discriminant using: a.) all the dimensions; b.) three best dimensions resulting from the relative-t-test (top left barplot); and c.), three best dimensions determined from the sequential-forward algorithm. The random 75% training data score, leave-one-out-cross-validation score, best dimension indicies, and J-scores are provided in the above right table for the full model, t-test-best-dimensions model, and sequential-forward-best-dimensions model. Note that the sequential-forward-derived dimensions performed slightly better than the t-test-derived dimensions, despite having a lower J_1 score.

4 mouse data



dims	accuracy	method
5	75%	t-test
12	85%	seq-fwrd
13	80%	seq-fwrd
14	70%	seq-fwrd
15	65%	seq-fwrd

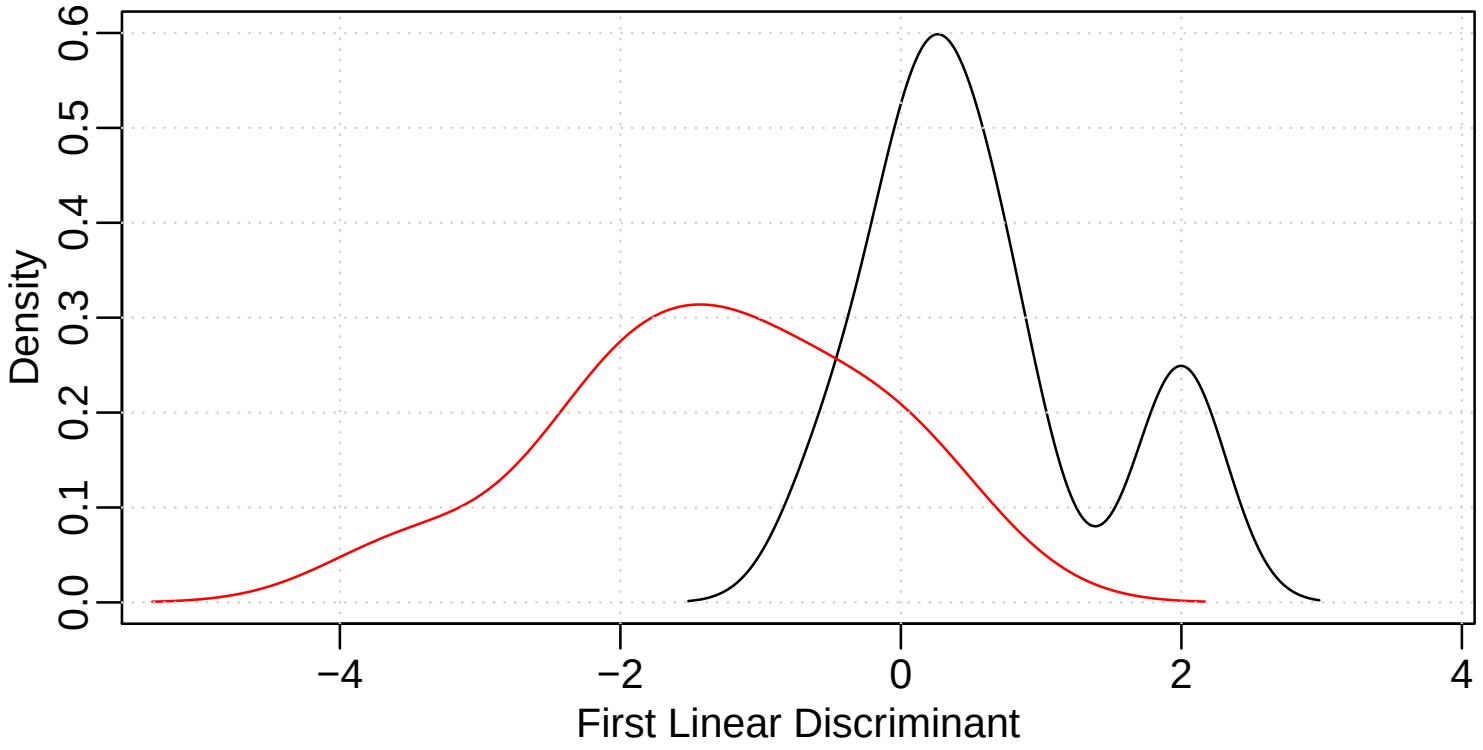
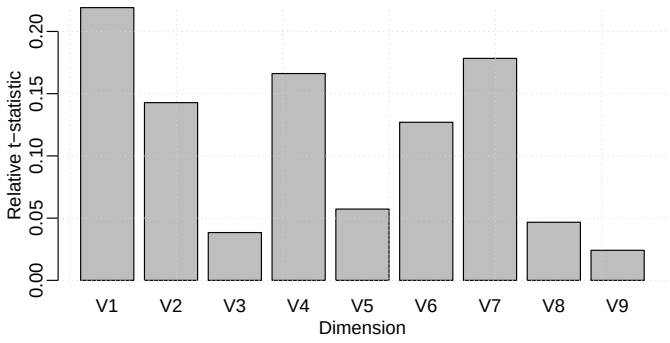


Figure 4: The density of the `mouse` data projected onto the first linear discriminant using the five-best dimensions from the relative-t-test (top left barplot). The table (upper right) gives the performance for the sequential-forward n-best dimensions. Note that as the number of dimensions increases, the n-way-cross-validation scores decrease.

5 fertility dataset



	full	***t*-test**	**seq-fwrd**
rand. acc.	92%	92%	92%
n-way acc.	85%	86%	86%
dims	all	1,2,4,6,7	1,2,4,6,7
J_1	1.57×10^{-2}	2.73×10^{-2}	2.73×10^{-2}
J_2	1.13	1.12	1.12
J_3	0.132	0.117	0.117

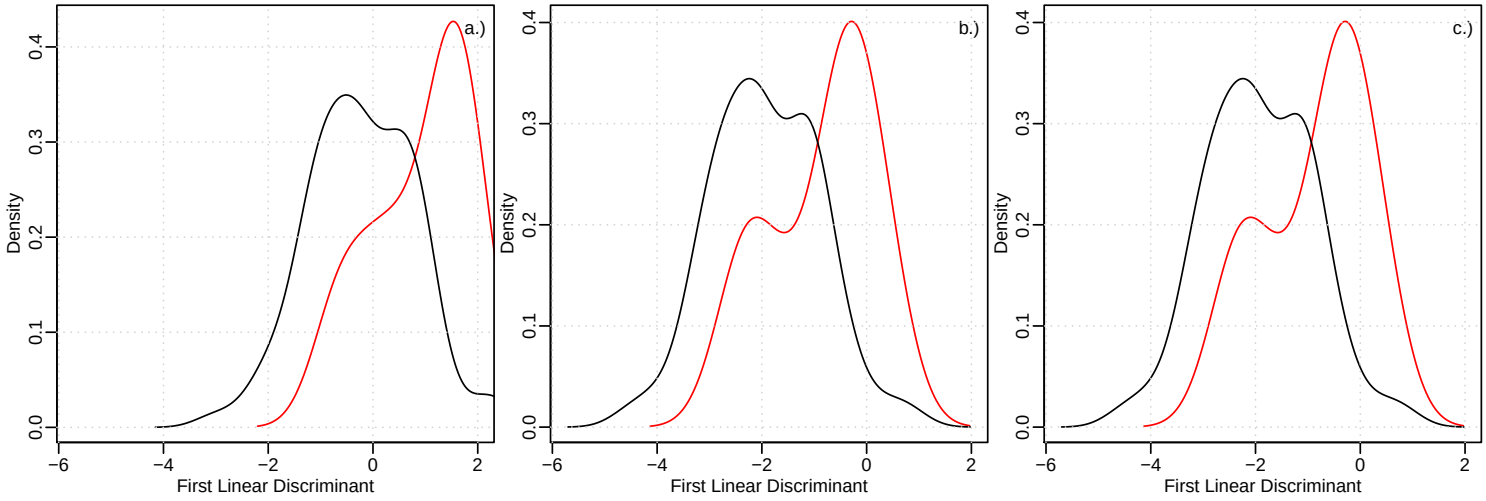


Figure 5: The density of the `fertility` data projected onto the first linear discriminant using: a.) all the dimensions; b.) five best dimensions resulting from the relative-t-test (top left barplot); and c.), five best dimensions determined from the sequential-forward algorithm – indentical to the t-test-derived dimensions. The random 75% training data score, leave-one-out-cross-validation score, best dimension indicies, and J-scores are provided in the above right table for the full model, t-test-best-dimensions model, and sequential-forward-best-dimensions model. Note that reduced-dimensional models performed slightly better than the full-dimensional model.

6 source code

6.1 functions

```
# functions used for feature-selection project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

getTscores = function(d,c)
{
  u = unique(c)
  n = length(u)
  m = dim(d)[2]
  v = rep(NA, m)
  for (k in 1:m)
  {
    s = 0
    for (i in 1:(n-1))
    {
      for (j in (i+1):n)
      {
        i_i  = which(c == u[i])
        j_i  = which(c == u[j])
        p    = t.test(d[i_i,k], d[j_i,k])
        s    = s + abs(p$statistic)
        #cat("k =", k, ": i =", i, ": j =", j, ": v[k] =", s, "\n")
      }
    }
    v[k] = s
  }
  return(v/sum(v))
}

nmax = function(x, n=2)
{
  nx = length(x)
  p  = nx - n
  xp = sort(x, partial=p)[p]
  return(which(x > xp))
}

Sb = function(x,c)
{
  u  = unique(c)
  k  = length(u)
  l  = list()

  for(i in 1:k)
  {
    l[[i]] = which(c == u[i])
  }

  m  = dim(x)[1]
  n  = dim(x)[2]
  mu = colMeans(x)
  S  = matrix(0, n, n)
  for(i in 1:k)
  {
    p    = length(l[[i]]) / m
    mu_i = colMeans(x[l[[i]],])
    S    = S + p * (mu_i - mu) %*% t(mu_i - mu)
  }
  return(S)
}

Sw = function(x,c)
{
  u  = unique(c)
  k  = length(u)
  l  = list()

  for(i in 1:k)
  {
    l[[i]] = which(c == u[i])
  }

  m  = dim(x)[1]
  n  = dim(x)[2]
  mu = colMeans(x)
  S  = matrix(0, n, n)
  for(i in 1:k)
  {
    x_i  = x[l[[i]],]
    p    = length(l[[i]]) / m
    sig  = cov(x_i)
    S    = S + p*sig
  }
  return(S)
}

J1 = function(x,c)
{
  S_b = Sb(x,c)
  S_w = Sw(x,c)
  J_1 = sum(diag(S_b)) / sum(diag(S_w))
  return(J_1)
}

J2 = function(x,c)
{
  S_b = Sb(x,c)
  S_w = Sw(x,c)
  S_m = S_w + S_b
  J_2 = det(ginv(S_w) %*% S_m)
  return(J_2)
}

J3 = function(x,c)
{
  S_b = Sb(x,c)
  S_w = Sw(x,c)
  J_3 = sum( diag( ginv(S_w) %*% S_b) )
  return(J_3)
}

seq_forward = function(x,c,k)
{
  m  = dim(x)[1]
  n  = dim(x)[2]
  v  = rep(NA, k)

  # populate the list of desired dimensions :
  for(i in 1:k)
  {
    # init the score to zero :
    s = 0

    # if the list is currently empty, just use t-test :
    if(i == 1)
    {
      u  = unique(c)
      y  = length(u)
      for(j in 1:n)
      {
        sn = 0
        for (o in 1:(y-1))
        {
          for (q in (i+1):y)
          {
            o_i  = which(c == u[o])
            q_i  = which(c == u[q])
            p    = t.test(x[o_i,j], x[q_i,j])
            sn   = sn + abs(p$statistic)
          }
        }
        if(sn > s)
        {
          s    = sn
          v[i] = j
        }
      }
    }

    # otherwise, use the J3 metric :
    else
    {
      for(j in 1:n)
      {
        # exclude any dimensions we know we want already :
        if(length(intersect(v[1:i], j)) == 0)
        {
          vn    = v[1:i]
          vn[i] = j
          sn    = J3(x[,vn],c)
          if(sn > s)
          {
            s    = sn
            v[i] = j
          }
        }
      }
    }
  }
  return(v)
}

nway_cross_validate_lda = function(x,c)
{
  s = 0
  for(i in 1:nrow(x))
  {
    zlda  = lda(x[-i,], c[-i])
    p     = predict(zlda, x[i,])
    if(p$class == c[i])
      s   = s + 1
  }
  return(100 * s/nrow(x))
}

testConstantWithinGroup = function(v,c,tol=1.0e-6)
{
  u = unique(c)
  #for each class
  for(i in 1:length(u))
  {
    i_i = which(c == u[i])
    if(mean(v[i_i]) == 0)
      return(1)
    if(var(v[i_i]) < tol)
      return(1)
  }
  return(0)
}

removeConstantRows = function(x, cl, tol=1.0e-6)
{
    r = c()
  for(i in 1:nrow(x))
  {
    b = testConstantWithinGroup(x[i,], cl, tol=tol)
    if (b)
    {
      r = c(r,i)
    }
  }
  return(r)
}
```

6.2 iris data

```
# iris data feature-selection project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

source("functs.r")

library(MASS)

# store the iris 'classes' :
c = iris[,5]

# store the iris 'data' :
d1 = iris[,seq(1,4)]
m1 = as.matrix(d1)

vt = getTscores(m1,c)
vs = seq_forward(m1,c,2)

# get reduced-dimension data :
d2 = iris[,nmax(vt,2)]
d3 = iris[,vs]
m2 = as.matrix(d2)
m3 = as.matrix(d3)

# plot them colored by class :
pdf(file = '../doc/images/iris_ttest.pdf', width=6, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

barplot(vt,
        ylab='Relative t-statistic',
        xlab='Dimension',
        main='',
        names.arg=colnames(d1))
grid()
dev.off()

# get training indicies of 75% of data :
n = dim(m1)[1]
t = sample(1:n, n*0.75)

# perform LDA on d :
zlda1 = lda(d1[t,], c[t])
zlda2 = lda(d2[t,], c[t])
zlda3 = lda(d3[t,], c[t])

# predict the test set -train :
p1 = predict(zlda1, d1[-t,])
p2 = predict(zlda2, d2[-t,])
p3 = predict(zlda3, d3[-t,])

# get the number predicted correctly :
```

```
correct1 = length(which(p1$class == c[-t]))
correct2 = length(which(p2$class == c[-t]))
correct3 = length(which(p3$class == c[-t]))

# get the proportion correct :
prop1 = correct1 / length(c[-t])
prop2 = correct2 / length(c[-t])
prop3 = correct3 / length(c[-t])

# get the J-scores :
J11 = J1(m1,c)
J21 = J2(m1,c)
J31 = J3(m1,c)

J12 = J1(m2,c)
J22 = J2(m2,c)
J32 = J3(m2,c)

J13 = J1(m3,c)
J23 = J2(m3,c)
J33 = J3(m3,c)

# perform nway-cross validation :
n1 = nway_cross_validate_lda(m1,c)
n2 = nway_cross_validate_lda(m2,c)
n3 = nway_cross_validate_lda(m3,c)

cat("accuracy of full nway =",   n1, "\n")
cat("accuracy of t-test nway =", n2, "\n")
cat("accuracy of seq nway =",    n3, "\n")

# print the result to the screen :
cat("accuracy of full dataset =",
    prop1*100, "%\n")
cat("J11 =", J11, ": J21 =", J21, ": J31 =", J31, "\n")
cat("accuracy of t-test dims",
    nmax(vt,2), "=", prop2*100, "%\n")
cat("J12 =", J12, ": J22 =", J22, ": J32 =", J32, "\n")
cat("accuracy of seq-forward for dims",
    vs, "=", prop3*100, "%\n")
cat("J13 =", J13, ": J23 =", J23, ": J33 =", J33, "\n")

# set the 1st linear discriminant :
xlda1 = m1 %*% zlda1$scaling[,1]
xlda2 = m2 %*% zlda2$scaling[,1]
xlda3 = m3 %*% zlda3$scaling[,1]

# set the 2nd linear discriminant :
ylda1 = m1 %*% zlda1$scaling[,2]
ylda2 = m2 %*% zlda2$scaling[,2]
ylda3 = m3 %*% zlda3$scaling[,2]

# store indexes of classes :
s  = which(c == 'setosa')
vc = which(c == 'versicolor')
vg = which(c == 'virginica')

# color vector :
cc     = as.vector(c)
cc[s]  = 'red'
cc[vc] = 'blue'
cc[vg] = 'green'

# plot them colored by class :
pdf('../doc/images/iris.pdf', width=9, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

plot(xlda1, ylda1, col=cc, bg=cc, type='p', pch=21,
     xlab='First Linear Discriminant',
     ylab='Second Linear Discriminant', main='')
legend("topleft", "a.)", bty="n")
grid()

plot(xlda2, ylda2, col=cc, bg=cc, type='p', pch=21,
     xlab='First Linear Discriminant',
     ylab='Second Linear Discriminant', main='')
legend("topleft", "b.)", bty="n")
grid()

plot(xlda3, ylda3, col=cc, bg=cc, type='p', pch=21,
     xlab='First Linear Discriminant',
     ylab='Second Linear Discriminant', main='')
legend("topleft", "c.)", bty="n")
grid()

dev.off()
```

6.3 fruit data

```
# fruit data feature-selection project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

source("functs.r")

library(MASS)

# read the variable back in :
f = read.csv("../../data/fruit.csv")

# store the fruit 'classes' :
c = f[,5]

# store the fruit 'data' :
d1 = f[,seq(1,4)]
m1 = as.matrix(d1)

vt = getTscores(m1,c)
vs = seq_forward(m1,c,2)

# get reduced-dimension data :
d2 = f[,nmax(vt,2)]
d3 = f[,vs]
m2 = as.matrix(d2)
m3 = as.matrix(d3)

# plot them colored by class :
pdf(file = '../doc/images/fruit_ttest.pdf', width=6, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

barplot(vt,
        ylab='Relative t-statistic',
        xlab='Dimension',
        main='',
        names.arg=colnames(d1))
grid()
dev.off()

# get training indicies of 75% of data :
n = dim(m1)[1]
t = sample(1:n, n*0.75)

# perform LDA on d :
zlda1 = lda(d1[t,], c[t])
zlda2 = lda(d2[t,], c[t])
zlda3 = lda(d3[t,], c[t])

# predict the test set -train :
p1 = predict(zlda1, d1[-t,])
p2 = predict(zlda2, d2[-t,])
p3 = predict(zlda3, d3[-t,])

# get the number predicted correctly :
correct1 = length(which(p1$class == c[-t]))
correct2 = length(which(p2$class == c[-t]))
correct3 = length(which(p3$class == c[-t]))

# get the proportion correct :
prop1 = correct1 / length(c[-t])
prop2 = correct2 / length(c[-t])
prop3 = correct3 / length(c[-t])

# get the J-scores :
J11 = J1(m1,c)
J21 = J2(m1,c)
J31 = J3(m1,c)

J12 = J1(m2,c)
J22 = J2(m2,c)
J32 = J3(m2,c)

J13 = J1(m3,c)
J23 = J2(m3,c)
J33 = J3(m3,c)

# perform nway-cross validation :
n1 = nway_cross_validate_lda(m1,c)
n2 = nway_cross_validate_lda(m2,c)
n3 = nway_cross_validate_lda(m3,c)

cat("accuracy of full nway =",   n1, "\n")
cat("accuracy of t-test nway =", n2, "\n")
cat("accuracy of seq nway =",    n3, "\n")

# print the result to the screen :
cat("accuracy of full dataset =",
    prop1*100, "%\n")
cat("J11 =", J11, ": J21 =", J21, ": J31 =", J31, "\n")
cat("accuracy of t-test dims",
    nmax(vt,2), "=", prop2*100, "%\n")
cat("J12 =", J12, ": J22 =", J22, ": J32 =", J32, "\n")
cat("accuracy of seq-forward for dims",
    vs, "=", prop3*100, "%\n")
cat("J13 =", J13, ": J23 =", J23, ": J33 =", J33, "\n")

# set the 1st linear discriminant :
xlda1 = m1 %*% zlda1$scaling[,1]
xlda2 = m2 %*% zlda2$scaling[,1]
xlda3 = m3 %*% zlda3$scaling[,1]

# set the 2nd linear discriminant :
ylda1 = m1 %*% zlda1$scaling[,2]
ylda2 = m2 %*% zlda2$scaling[,2]
ylda3 = m3 %*% zlda3$scaling[,2]

# store indexes of classes :
a = which(c == 'apple')
l = which(c == 'lemon')
o = which(c == 'orange')
z = which(c == 'peach')

# color vector :
cols  = c('red', 'yellow', 'orange', 'burlywood1')
cc    = as.vector(c)
cc[a] = cols[1]
cc[l] = cols[2]
cc[o] = cols[3]
cc[z] = cols[4]

# plot them colored by class :
pdf('../doc/images/fruit.pdf', width=9, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

plot(xlda1, ylda1, col=cc, bg=cc, type='p', pch=21,
     xlab='First Linear Discriminant',
     ylab='Second Linear Discriminant', main='')
legend("topleft", "a.)", bty="n")
grid()

plot(xlda2, ylda2, col=cc, bg=cc, type='p', pch=21,
     xlab='First Linear Discriminant',
     ylab='Second Linear Discriminant', main='')
legend("topleft", "b.)", bty="n")
grid()

plot(xlda3, ylda3, col=cc, bg=cc, type='p', pch=21,
     xlab='First Linear Discriminant',
     ylab='Second Linear Discriminant', main='')
legend("topleft", "c.)", bty="n")
grid()

dev.off()
```

6.4 tumor data

```
# tumor data feature-selection project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

source("functs.r")

library(MASS)

# read in the data :
f = read.csv("../../data/breast-cancer-wisconsin.data",
             header=FALSE, na.strings='?')

# get the sample names :
e = f[,1]

# get the classes :
c = f[,11]

# get just the data :
d = f[,seq(2,10)]

# replace the NANs with the column mean :
for(i in 1:ncol(d))
{
  na      = which(is.na(d[,i]))
  d[na,i] = mean(d[-na,i])
}
d1 = d
m1 = as.matrix(d1)

vt = getTscores(m1,c)
vs = seq_forward(m1,c,3)

# get reduced-dimension data :
d2 = d1[,nmax(vt,n=3)]
d3 = d1[,vs]
m2 = as.matrix(d2)
m3 = as.matrix(d3)
```

```
# plot them colored by class :
pdf(file = '../doc/images/tumor_ttest.pdf', width=6, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

barplot(vt,
        ylab='Relative t-statistic',
        xlab='Dimension',
        main='',
        names.arg=colnames(d1))
grid()
dev.off()

# get training indicies of 75% of data :
n = dim(m1)[1]
t = sample(1:n, n*0.75)

# perform LDA on d :
zlda1 = lda(d1[t,], c[t])
zlda2 = lda(d2[t,], c[t])
zlda3 = lda(d3[t,], c[t])

# predict the test set -train :
p1 = predict(zlda1, d1[-t,])
p2 = predict(zlda2, d2[-t,])
p3 = predict(zlda3, d3[-t,])

# get the number predicted correctly :
correct1 = length(which(p1$class == c[-t]))
correct2 = length(which(p2$class == c[-t]))
correct3 = length(which(p3$class == c[-t]))

# get the proportion correct :
prop1 = correct1 / length(c[-t])
prop2 = correct2 / length(c[-t])
prop3 = correct3 / length(c[-t])

# get the J-scores :
J11 = J1(m1,c)
J21 = J2(m1,c)
J31 = J3(m1,c)

J12 = J1(m2,c)
J22 = J2(m2,c)
J32 = J3(m2,c)

J13 = J1(m3,c)
J23 = J2(m3,c)
J33 = J3(m3,c)

# perform nway-cross validation :
n1 = nway_cross_validate_lda(m1,c)
n2 = nway_cross_validate_lda(m2,c)
n3 = nway_cross_validate_lda(m3,c)

cat("accuracy of full nway =",   n1, "\n")
cat("accuracy of t-test nway =", n2, "\n")
cat("accuracy of seq nway =",    n3, "\n")

# print the result to the screen :
cat("accuracy of full dataset =",
    prop1*100, "%\n")
cat("J11 =", J11, ": J21 =", J21, ": J31 =", J31, "\n")
cat("accuracy of t-test dims",
    nmax(vt,3), "=", prop2*100, "%\n")
cat("J12 =", J12, ": J22 =", J22, ": J32 =", J32, "\n")
cat("accuracy of seq-forward for dims",
    vs, "=", prop3*100, "%\n")
cat("J13 =", J13, ": J23 =", J23, ": J33 =", J33, "\n")

# set the 1st linear discriminant :
xlda1 = m1 %*% zlda1$scaling[,1]
xlda2 = m2 %*% zlda2$scaling[,1]
xlda3 = m3 %*% zlda3$scaling[,1]

# create density to determine range of data :
dens1   = density(xlda1)
dens2   = density(xlda2)
dens3   = density(xlda3)
xrange1 = range(dens1$x)
xrange2 = range(dens2$x)
xrange3 = range(dens3$x)

# color sample black for benign (b) and red for malignant (m) :
cols  = c('black', 'red')
cc    = as.vector(e)
b     = which(c == 2)
m     = which(c == 4)
cc[b] = cols[1]
cc[m] = cols[2]

# project just the benigns :
xb1 = m1[b,] %*% zlda1$scaling[,1]
xb2 = m2[b,] %*% zlda2$scaling[,1]
xb3 = m3[b,] %*% zlda3$scaling[,1]
db1 = density(xb1)
db2 = density(xb2)
db3 = density(xb3)

# project just the malignant :
xm1 = m1[m,] %*% zlda1$scaling[,1]
xm2 = m2[m,] %*% zlda2$scaling[,1]
xm3 = m2[m,] %*% zlda3$scaling[,1]
dm1 = density(xm1)
dm2 = density(xm2)
dm3 = density(xm3)

# plot them colored by experiment :
pdf('../doc/images/tumor.pdf', width=9, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

plot(db1$x, db1$y, col=cols[1], type='l', pch=21, xlim=xrange2,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
lines(dm1$x, dm1$y, col=cols[2], pch=21,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
legend("topright", "a.)", bty="n")
grid()

plot(db2$x, db2$y, col=cols[1], type='l', pch=21, xlim=xrange2,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
lines(dm2$x, dm2$y, col=cols[2], pch=21,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
legend("topright", "b.)", bty="n")
grid()

plot(db3$x, db3$y, col=cols[1], type='l', pch=21, xlim=xrange2,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
lines(dm2$x, dm2$y, col=cols[2], pch=21,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
legend("topright", "c.)", bty="n")
grid()

dev.off()
```

6.5 mouse data

```
# mouse data feature-selection project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

source("functs.r")

library(MASS)

# read in the data :
f = read.csv("../../data/otu_table_L6.txt", sep="\t", row.names=1)

# get the experiment names :
e = colnames(f)

# get the genera names :
g = rownames(f)

# store indexes of proximal (P) and distal (D) experiments, and mouse types
# B and C :
P = grep("[A-Z]+P[0-9]", e)
D = grep("[A-Z]+D[0-9]", e)
B = grep("^B[A-Z]+[0-9]", e)
C = grep("^C[A-Z]+[0-9]", e)

# create the classes (proximal or distal) :
c    = 1:length(e)
c[P] = 'proximal'
c[D] = 'distal'

# transpose the data so each row is an experiment :
d1 = t(f)
m1 = as.matrix(d1)

v  = getTscores(m1,c)

# get reduced-dimension data :
d2 = d1[,nmax(v,n=5)]
m2 = as.matrix(d2)

# remove and "Other" generas :
fn = f[-grep("Other", rownames(f)),]
m3 = as.matrix(fn)

# get indicies of any constant rows :
r = removeConstantRows(m3, c)
m3 = t(m3[-r,])

# do this other stuff :
v12 = seq_forward(m3,c,12)
v13 = seq_forward(m3,c,13)
v14 = seq_forward(m3,c,14)
v15 = seq_forward(m3,c,15)
n12 = nway_cross_validate_lda(m3[,v12], c)
n13 = nway_cross_validate_lda(m3[,v13], c)
n14 = nway_cross_validate_lda(m3[,v14], c)
n15 = nway_cross_validate_lda(m3[,v15], c)

# print the results of that stuff :
cat("n12 =", n12, "\n")
cat("n13 =", n13, "\n")
cat("n14 =", n14, "\n")
cat("n15 =", n15, "\n")

# plot them colored by class :
pdf(file = '../doc/images/mouse_ttest.pdf', width=6, height=3)
par(mar=c(2.5,2.5,1.0,0.1), mgp=c(1.5, .5, 0), mfrow=c(1,1))

barplot(v,
        ylab='Relative t-statistic',
        xlab='Dimension',
        main='',
        names.arg=1:dim(d1)[2])
grid()
dev.off()

zlda2    = lda(d2, c)
p2       = predict(zlda2, d2)
correct2 = length(which(p2$class == c))
prop2    = correct2 / length(c)
cat("accuracy2 =", prop2*100, "%\n")

# perform nway-cross validation :
n2 = nway_cross_validate_lda(m2,c)
cat("accuracy of t-test nway =", n2, "\n")

# set the 1st linear discriminant :
#xlda1 = m1 %*% zlda1$scaling[,1]
xlda2 = m2 %*% zlda2$scaling[,1]

# set the 2nd linear discriminant :
#ylda1 = m1 %*% zlda1$scaling[,2]
#ylda2 = m2 %*% zlda2$scaling[,2]

# create density to determine range of data :
#dens1   = density(xlda1)
dens2   = density(xlda2)
#xrange1 = range(dens1$x)
xrange2 = range(dens2$x)

# project just the benigns :
#xp1 = m1[P,] %*% zlda1$scaling[,1]
xp2 = m2[P,] %*% zlda2$scaling[,1]
#dp1 = density(xp1)
dp2 = density(xp2)

# project just the malignant :
#xd1 = m1[D,] %*% zlda1$scaling[,1]
xd2 = m2[D,] %*% zlda2$scaling[,1]
#dd1 = density(xd1)
dd2 = density(xd2)

# color vector :
cols  = c('black', 'red')
cc    = as.vector(e)
cc[P] = cols[1]
cc[D] = cols[2]

# plot them colored by experiment :
pdf('../doc/images/mouse.pdf', width=6, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

#plot(xlda1, ylda1, col=cc, bg=cc, type='p', pch=21,
#     xlab='First Linear Discriminant',
#     ylab='Second Linear Discriminant', main='')
#legend("topleft", "a.)", bty="n")
#grid()

plot(dp2$x, dp2$y, col=cols[1], type='l', pch=21, xlim=xrange2,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
lines(dd2$x, dd2$y, col=cols[2], pch=21,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
#legend("topright", "a.)", bty="n")
grid()

#plot(xlda2, ylda2, col=cc, bg=cc, type='p', pch=21,
#     xlab='First Linear Discriminant',
#     ylab='Second Linear Discriminant', main='')
```

```
#legend("topleft", "b.)", bty="n")
#grid()

dev.off()
```

6.6 fertility data

```
# fertility data feature-selection project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

source("functs.r")

library(MASS)

# read in the data :
f = read.csv("../../data/fertility_Diagnosis.txt", sep=",", header=FALSE)

# create the classes (proximal or distal) :
c = f[,10]

# transpose the data so each row is an experiment :
d1 = f[,1:9]
m1 = as.matrix(d1)

vt = getTscores(m1,c)
vs = seq_forward(m1,c,5)

# get reduced-dimension data :
d2 = f[,nmax(vt,5)]
d3 = f[,vs]
m2 = as.matrix(d2)
m3 = as.matrix(d3)

# plot them colored by class :
pdf(file = '../doc/images/fertility_ttest.pdf', width=6, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

barplot(vt,
        ylab='Relative t-statistic',
        xlab='Dimension',
        main='',
        names.arg=colnames(d1))
grid()
dev.off()

# get training indicies of 75% of data :
n = dim(m1)[1]
t = sample(1:n, n*0.75)

# perform LDA on d :
zlda1 = lda(d1[t,], c[t])
zlda2 = lda(d2[t,], c[t])
zlda3 = lda(d3[t,], c[t])

# predict the test set -train :
p1 = predict(zlda1, d1[-t,])
p2 = predict(zlda2, d2[-t,])
p3 = predict(zlda3, d3[-t,])

# get the number predicted correctly :
correct1 = length(which(p1$class == c[-t]))
correct2 = length(which(p2$class == c[-t]))
correct3 = length(which(p3$class == c[-t]))

# get the proportion correct :
prop1 = correct1 / length(c[-t])
prop2 = correct2 / length(c[-t])
prop3 = correct3 / length(c[-t])

# get the J-scores :
J11 = J1(m1,c)
J21 = J2(m1,c)
J31 = J3(m1,c)

J12 = J1(m2,c)
J22 = J2(m2,c)
J32 = J3(m2,c)

J13 = J1(m3,c)
J23 = J2(m3,c)
J33 = J3(m3,c)

# perform nway-cross validation :
n1 = nway_cross_validate_lda(m1,c)
n2 = nway_cross_validate_lda(m2,c)
n3 = nway_cross_validate_lda(m3,c)

cat("accuracy of full nway =",   n1, "\n")
cat("accuracy of t-test nway =", n2, "\n")
cat("accuracy of seq nway =",    n3, "\n")

# print the result to the screen :
cat("accuracy of full dataset =",
    prop1*100, "%\n")
cat("J11 =", J11, ": J21 =", J21, ": J31 =", J31, "\n")
cat("accuracy of t-test dims",
    nmax(vt,5), "=", prop2*100, "%\n")
cat("J12 =", J12, ": J22 =", J22, ": J32 =", J32, "\n")
cat("accuracy of seq-forward for dims",
    vs, "=", prop3*100, "%\n")
cat("J13 =", J13, ": J23 =", J23, ": J33 =", J33, "\n")

# set the 1st linear discriminant :
xlda1 = m1 %*% zlda1$scaling[,1]
xlda2 = m2 %*% zlda2$scaling[,1]
xlda3 = m3 %*% zlda3$scaling[,1]

# create density to determine range of data :
dens1   = density(xlda1)
dens2   = density(xlda2)
dens3   = density(xlda3)
xrange1 = range(dens1$x)
xrange2 = range(dens2$x)
xrange3 = range(dens3$x)

# store indexes of classes :
o = which(c == 'O')
n = which(c == 'N')

# color vector :
cols  = c('red', 'black')
cc    = as.vector(c)
cc[o] = cols[1]
cc[n] = cols[2]

# project just the benigns :
xb1 = m1[o,] %*% zlda1$scaling[,1]
xb2 = m2[o,] %*% zlda2$scaling[,1]
xb3 = m3[o,] %*% zlda3$scaling[,1]
db1 = density(xb1)
db2 = density(xb2)
db3 = density(xb3)

# project just the malignant :
xm1 = m1[n,] %*% zlda1$scaling[,1]
xm2 = m2[n,] %*% zlda2$scaling[,1]
xm3 = m2[n,] %*% zlda3$scaling[,1]
dm1 = density(xm1)
dm2 = density(xm2)
dm3 = density(xm3)

# plot them colored by experiment :
pdf('../doc/images/fertility.pdf', width=9, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

plot(db1$x, db1$y, col=cols[1], type='l', pch=21, xlim=xrange2,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
lines(dm1$x, dm1$y, col=cols[2], pch=21,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
legend("topright", "a.)", bty="n")
grid()

plot(db2$x, db2$y, col=cols[1], type='l', pch=21, xlim=xrange2,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
lines(dm2$x, dm2$y, col=cols[2], pch=21,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
legend("topright", "b.)", bty="n")
grid()

plot(db3$x, db3$y, col=cols[1], type='l', pch=21, xlim=xrange2,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
lines(dm2$x, dm2$y, col=cols[2], pch=21,
     xlab='First Linear Discriminant',
     ylab='Density', main='')
legend("topright", "c.)", bty="n")
grid()

dev.off()
```