Clustering

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

November 16, 2016

1 "Smiley" data :

1.1 k-means clustering :

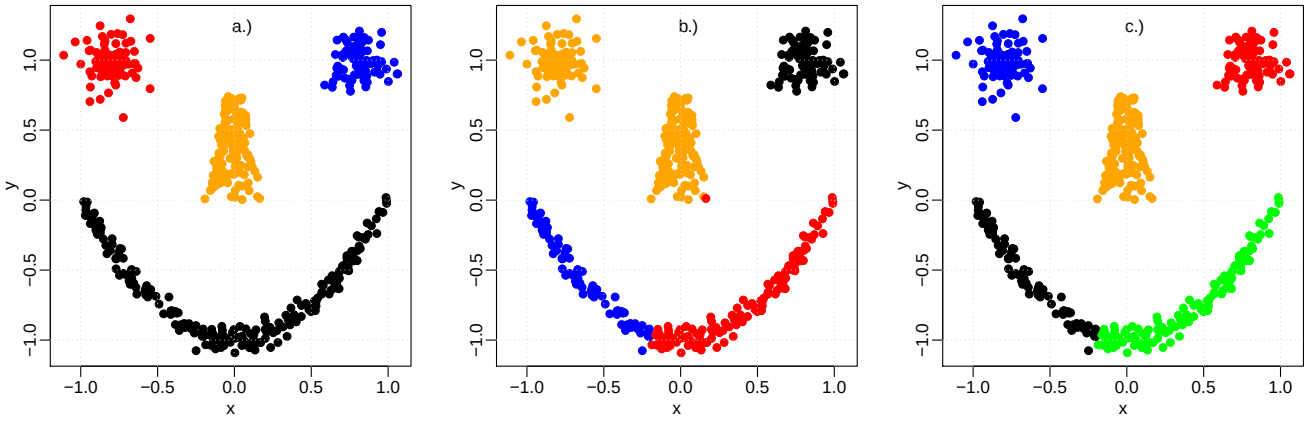

Figure 1: The original Smiley data (left) with types left eye, right eye, nose, and mouth colored arbitrarily; the k-means-derived classes with $k = 4$ (middle); and the k-means-derived classes with $k = 5$ (right). Note that k-means seeks to find k clusters with minimum within-cluster-sum-of-distances from the cluster-center squared, and hence splits the data halfway between cluster means. Because these data include a cluster, the 'mouth', with an x-mean identical to the 'nose' and directly in-between the two 'eyes', k-means performed over these data with $k = 4$ splits the 'mouth' in two and assigns the 'nose' cluster to one of the other 'eye' clusters. The phenomenon is further illustrated with the $k = 5$ data (right) which displays five somewhat-equally-sized regions around the five cluster means.

1.2 hierarchical clustering :

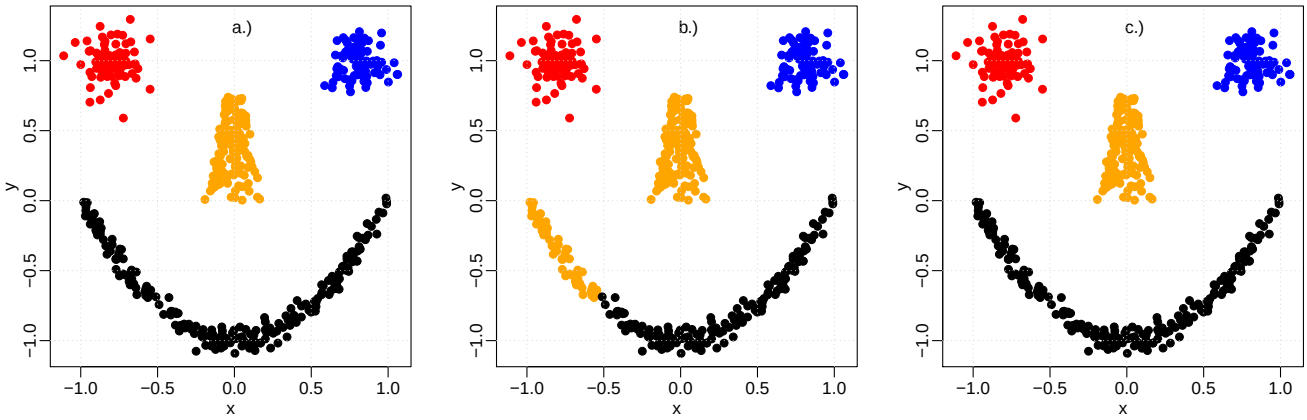

Figure 2: The original Smiley data (left) with types left eye, right eye, nose, and mouth colored arbitrarily; clusters obtained by using hierarchical clustering with the *complete linkage* method (middle); and hierarchical clustering performed with the *Ward* method (right). The complete linkage method combines clusters with minimum distance to each other, while the Ward method combines clusters which minimizes the total within-cluster variance, or *error sum of squares*. Because these data include highly-separated clusters, the Ward method is appropriate.

1.3 R source code :

```
# smiley data clustering project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

library(mlbench)

# read in the data :
f = mlbench.smiley()

# store the 'classes' :
c = f$classes

# store the 'data' :
x = f$x

# store indexes of classes :
c1 = which(c == 1)
c2 = which(c == 2)
c3 = which(c == 3)
c4 = which(c == 4)

# color vector :
cols  = c('red', 'blue', 'orange', 'black')
cc    = as.vector(c)
cc[c1] = cols[1]
cc[c2] = cols[2]
cc[c3] = cols[3]
cc[c4] = cols[4]

# plot them colored by class :
pdf(file = '../doc/images/actual_smiley.pdf', width=4, height=4)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

plot(x, col=cc, bg=cc, type='p', pch=21,
     xlab='x', ylab='y', main='')
legend("top", "a.)", bty="n")
grid()

dev.off()


#================================================================================
# k-means clustering :
km = kmeans(x, 4)

ck = km$cluster

# store indexes of classes :
c1 = which(ck == 1)
c2 = which(ck == 2)
c3 = which(ck == 3)
c4 = which(ck == 4)

# color vector :
cols  = c('red', 'blue', 'orange', 'black')
cc    = as.vector(ck)
cc[c1] = cols[1]
cc[c2] = cols[2]
cc[c3] = cols[3]
cc[c4] = cols[4]

# plot them colored by class :
pdf(file = '../doc/images/kmeans_smiley.pdf', width=4, height=4)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

plot(x, col=cc, bg=cc, type='p', pch=21,
     xlab='x', ylab='y', main='')
legend("top", "b.)", bty="n")
grid()

dev.off()

#================================================================================
# k-means clustering :
km = kmeans(x, 5)

ck = km$cluster

# store indexes of classes :
c1 = which(ck == 1)
c2 = which(ck == 2)
c3 = which(ck == 3)
c4 = which(ck == 4)
c5 = which(ck == 5)

# color vector :
cols  = c('red', 'blue', 'orange', 'black', 'green')
cc    = as.vector(ck)
cc[c1] = cols[1]
cc[c2] = cols[2]
cc[c3] = cols[3]
cc[c4] = cols[4]
cc[c5] = cols[5]

# plot them colored by class :
pdf(file = '../doc/images/kmeans_5_smiley.pdf', width=4, height=4)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

plot(x, col=cc, bg=cc, type='p', pch=21,
     xlab='x', ylab='y', main='')
legend("top", "c.)", bty="n")
grid()

dev.off()


#================================================================================
# hierarchical clustering :
hm = hclust(dist(x))
ch = cutree(hm, k=4)

# store indexes of classes :
c1 = which(ch == 1)
c2 = which(ch == 2)
c3 = which(ch == 3)
c4 = which(ch == 4)

# color vector :
cols  = c('red', 'blue', 'orange', 'black')
cc    = as.vector(ck)
cc[c1] = cols[1]
cc[c2] = cols[2]
cc[c3] = cols[3]
cc[c4] = cols[4]

# plot them colored by class :
pdf(file = '../doc/images/hierClust_smiley.pdf', width=4, height=4)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

plot(x, col=cc, bg=cc, type='p', pch=21,
     xlab='x', ylab='y', main='')
legend("top", "b.)", bty="n")
grid()

dev.off()
```

```
#================================================================================
# hierarchical clustering :
hm = hclust(dist(x), method='ward')
ch = cutree(hm, k=4)

# store indexes of classes :
c1 = which(ch == 1)
c2 = which(ch == 2)
c3 = which(ch == 3)
c4 = which(ch == 4)

# color vector :
cols  = c('red', 'blue', 'orange', 'black')
cc    = as.vector(ck)
cc[c1] = cols[1]
cc[c2] = cols[2]
cc[c3] = cols[3]
cc[c4] = cols[4]

# plot them colored by class :
pdf(file = '../doc/images/hierClust_ward_smiley.pdf', width=4, height=4)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

plot(x, col=cc, bg=cc, type='p', pch=21,
     xlab='x', ylab='y', main='')
legend("top", "c.)", bty="n")
grid()

dev.off()
```

2 Sequential-forward algorithm applied to the “Iris” data :

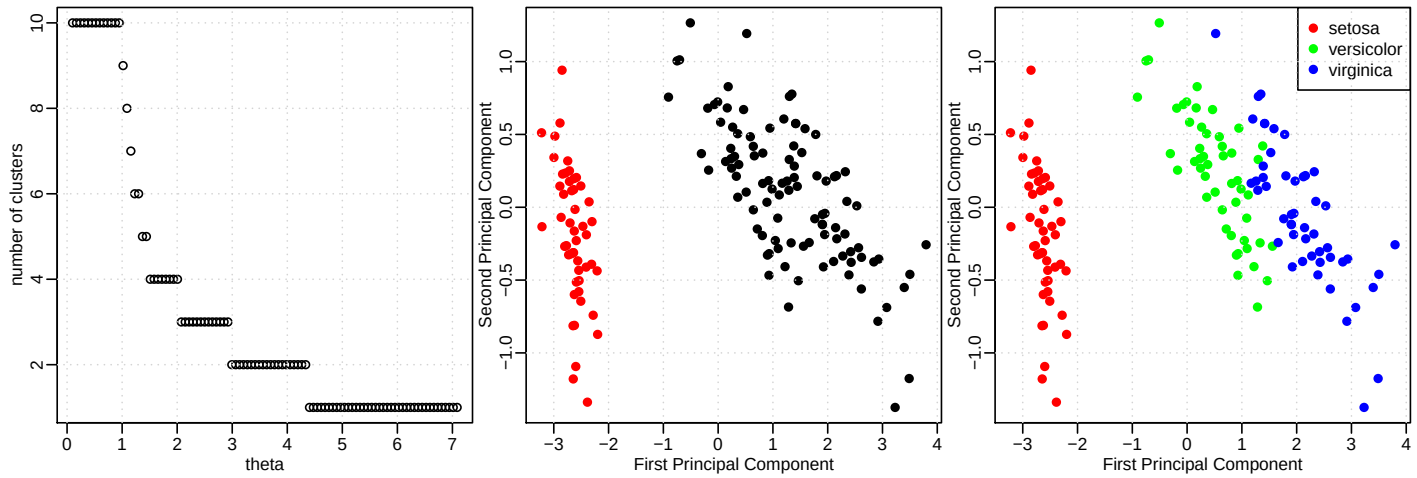

Figure 3: The number of clusters derived with a given characteristic distance for the *Iris* data using sequential clustering (left), the optimal clustering for $\hat{\theta} = 3.5$ (middle), and data plotted by class (right). Here, $\hat{\theta}$ was chosen as being representative of the most prevalent θ-value corresponding to a number of clusters greater than one. Note that while the Iris data contains three distinct classes, two of them are very close together. Due to the fact that the sequential-forward algorithm combines nearby classes, it will perform poorly when differentiating between classes separated by short distances, as is the case here.

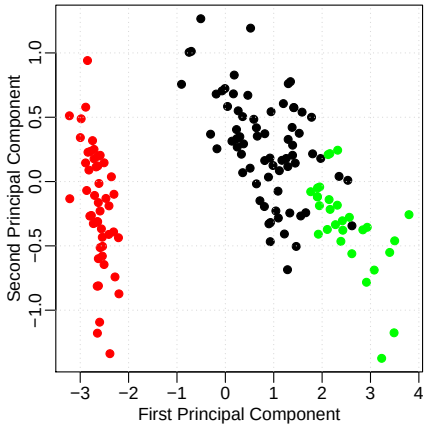

Figure 4: The resulting clusters for $\hat{\theta} = 2.5$ corresponding to three clusters. While not perfect, it appears that the sequential-forward algorithm does start to differentiate between the nearby classes *Versicolor* and *Virginica*.

2.1 R source code :

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

source("functs.r")

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

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

# get min/max of the distance matrix (complicated, right?!) :
D    = dist(m)
U    = upper.tri(D)
DU   = D[!is.na(D[U])]
tmax = max(DU)
tmin = min(DU)

# number of thetas :
k       = 100

# domain of thetas :
thetas = seq(tmin, tmax, length=k)

# range of clusters given thetas :
clusts = rep(NA, k)

# sequential clustering over range of thetas :
for (i in seq(1:k))
{
  v         = sequential_forward(m, theta=thetas[i], kmax=10)
  numClust  = length(unique(v))
  clusts[i] = numClust
}

#============================================================================
# print the data colored by most likely cluster :
v  = sequential_forward(m, theta=3.5, kmax=10)

# dimension reduction to plot in 2D :
zpca = prcomp(d)

# plot the domain vs range and optimal clusters :
pdf('../doc/images/iris_clusters.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(thetas, clusts, col='black', type='p', pch=21,
     xlab='theta', ylab='number of clusters', main='')
grid()

# store indexes of classes :
v1 = which(v == 1)
v2 = which(v == 2)

# color vector :
cc     = as.vector(v)
cc[v1] = 'red'
cc[v2] = 'black'

plot(zpca$x[,1], zpca$x[,2], col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Component',
     ylab='Second Principal Component', main='')
grid()

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

# color vector :
cc     = as.vector(c)
col    = c('red', 'green', 'blue')
cc[s]  = col[1]
cc[vc] = col[2]
cc[vg] = col[3]

plot(zpca$x[,1], zpca$x[,2], col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Component',
     ylab='Second Principal Component', main='')
legend('topright', levels(c), col=col, pt.bg=col, pch=21)
grid()

dev.off()

#============================================================================
# print the data colored with 3 clusters :
v  = sequential_forward(m, theta=2.5, kmax=10)

# dimension reduction to plot in 2D :
zpca = prcomp(d)

# plot the domain vs range and optimal clusters :
pdf('../doc/images/iris_3_clusters.pdf', width=4, height=4)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

# store indexes of classes :
v1 = which(v == 1)
v2 = which(v == 2)
v3 = which(v == 3)

# color vector :
cc     = as.vector(v)
cc[v1] = 'red'
cc[v2] = 'black'
cc[v3] = 'green'

plot(zpca$x[,1], zpca$x[,2], col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Component',
     ylab='Second Principal Component', main='')
grid()

dev.off()
```

3 E. coli data :

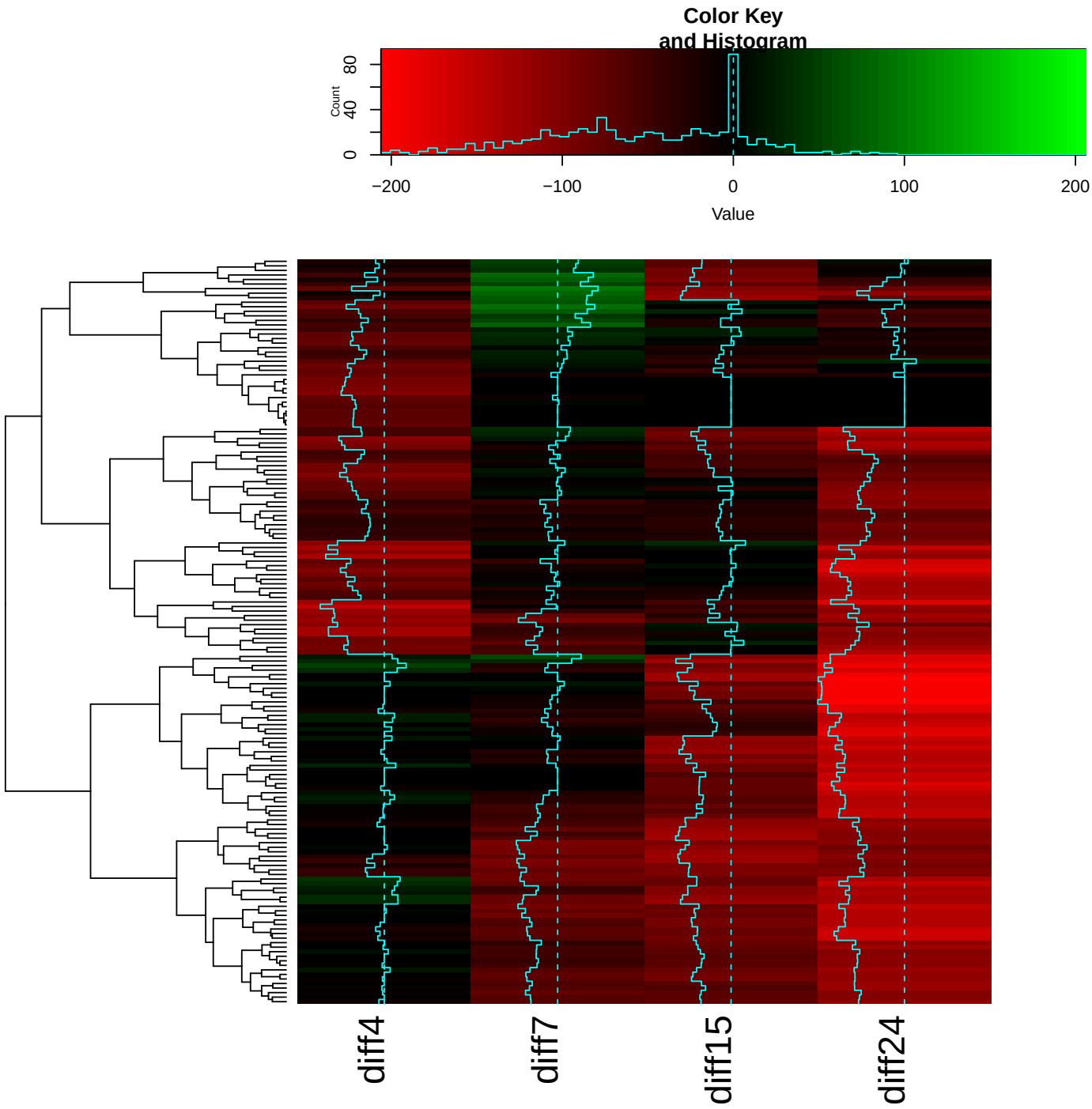

Figure 5: Gene expression samples of E. coli taken at t =4,7,15, and 24 hours, where each row represents a gene's expression profile, i.e., the extent to which the corresponding gene is expressed in biofilm vs. suspension. The rows are ordered by cluster hierarchy.

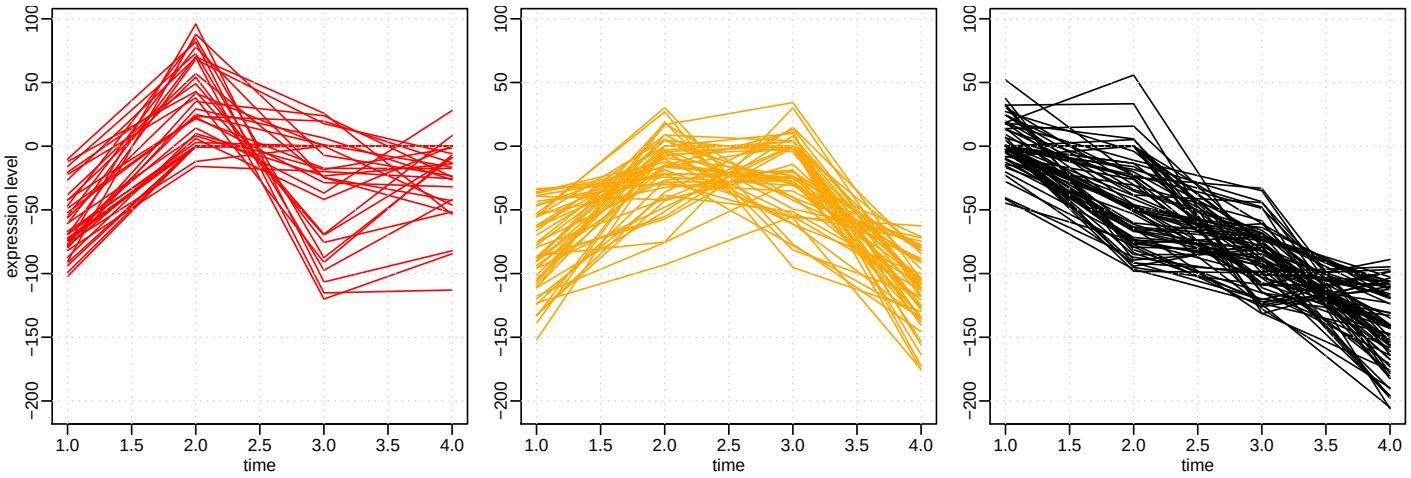

Figure 6: The individual three clusters derived as in Figure 7. It appears that one cluster exhibits oscillatory behavior (left), one goes up and then back down (middle), and one decreases (right).

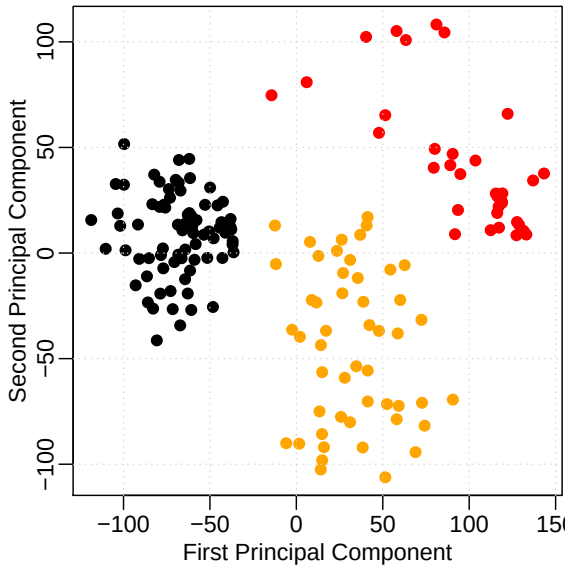

Figure 7: The E. coli data projected onto the first two principle components, colored by hierarchical clustering. It appears that there are indeed three distinct clusters here!

3.1 R source code :

```
# E. coli data sequential clustering project
# Evan Cummings
# CSCI 548 - Pattern Recognition
# Douglas Raiford, Fall 2015

library(gplots)
source("functs.r")

# read in the data :
f = read.csv("../../data/ecoliDiffExpReduced.csv", sep=",")

# store data :
d = as.matrix(f[,seq(3,6)])

# run hierarchical clustering on the data with three clusters :
hm = hclust(dist(d))
ch = cutree(hm, k=3)

# store indexes of classes :
c1 = which(ch == 1)
c2 = which(ch == 2)
c3 = which(ch == 3)

# color vector :
cols  = c('red', 'orange', 'black')
cc    = as.vector(ch)
cc[c1] = cols[1]
cc[c2] = cols[2]
cc[c3] = cols[3]

# dimension reduction to plot in 2D :
zpca = prcomp(d)

# plot the domain vs range and optimal clusters :
pdf('../doc/images/ecoli_pca.pdf', width=4, height=4)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

plot(zpca$x[,1], zpca$x[,2], col=cc, bg=cc, type='p', pch=21,
     xlab='First Principal Component',
     ylab='Second Principal Component', main='')
grid()

dev.off()

# plot the heatmap :
pdf('../doc/images/ecoli_heat.pdf', width=8, height=8)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,1))

lmat = rbind(c(3,4),c(2,1))
lwid = c(1.5,4.0)
lhei = c(1.2,4.1)

heatmap.2(d, Colv = NA, col = redgreen(75), labRow = NA,
          lmat = lmat, lwid = lwid, lhei=lhei)

dev.off()

# plot lines for each cluster :
pdf('../doc/images/ecoli_lines.pdf', width=9, height=3)
par(mar=c(2.5,2.5,.1,.1),mgp = c(1.5, .5, 0), mfrow=c(1,3))

ylim = c(min(d), max(d))

matplot(t(d[c1,]), col=cols[1], bg=cols[1], type='l', lty=1, pch=21,
        ylim=ylim, xlab='time', ylab='expression level', main='')
grid()

matplot(t(d[c2,]), col=cols[2], bg=cols[2], type='l', lty=1, pch=21,
        ylim=ylim, xlab='time', ylab='', main='')
grid()

matplot(t(d[c3,]), col=cols[3], bg=cols[3], type='l', lty=1, pch=21,
        ylim=ylim, xlab='time', ylab='', main='')
grid()

dev.off()
```