Solutions for chapter Unsupervised Machine Learning

Exercise 1

a The three-dimensional reduction is obtained by specifying the parameter `k` in the call to `sammon`.

```
> sam2 = sammon(manDist, k=3,  trace=FALSE)
```

b The R function is `cmdscale`.

```
> cmd1 = cmdscale(manDist)
```

c This is somewhat more involved and requires the use of a number of tools. Plotting the points that are computed via Sammon mapping yields a fairly good separation of the data into two groups, which we expect, because we essentially forced that to be true by our selection of the genes with large t-statistics.

```
> rtt = rowttests(ALLfilt_bcrneg, "mol.biol")
> ordtt = order(rtt$p.value)
> esTT = ALLfilt_bcrneg[ordtt[1:50],]
> dTT = dist(t(exprs(esTT)), method="manhattan")
> sTT = sammon(dTT, trace=FALSE)
```

Exercise 2

We use the `maximum` distance, but you could have chosen any other. We suggest examining some of the distance measures in the **bioDist** package.

```
> dsol = as.matrix(dist(gvals), method="maximum")
> silcheck(dsol, diss=TRUE)
[1] 2.0000 0.0997
> msscheck(as.hdist(dsol))
[1] 3.0000 0.0738
```

Exercise 3

We first use `cutree` as described above.

```
> hc13 = cutree(hc1, k=3)
> hc23 = cutree(hc2, k=3)
> hc33 = cutree(hc3, k=3)
> hc43 = cutree(hc4, k=3)
```

And now we need to compare the outputs. It is relatively easy to do that for the *hclust* objects, and unfortunately less so for comparisons with the *diana* object.

```
> table(hc13, hc33)
    hc33
hc13  1  2  3
   1 24  0 17
   2  5 13  0
   3 20  0  0
```

Exercise 4

The code to compute the cophenetic correlations, and to compute the correlation with the original distances is given below.

```
> cph2 = cophenetic(hc2)
> cor2 = cor(manDist, cph2)
> cor2
[1] 0.381
> cph3 = cophenetic(hc3)
> cor3 = cor(manDist, cph3)
> cor3
[1] 0.523
> cph4 = cophenetic(hc4)
> cor4 = cor(manDist, cph4)
> cor4
[1] 0.55
```

```
> stopifnot( cor2 == min(cor1, cor2, cor3, cor4) )
```

We see that for single-linkage clustering the cophenetic correlation is much lower than for the other three, suggesting that it is a relatively poor choice of hierarchical clustering method. The other three values are quite similar.

Exercise 5

The values returned can be examined by using `names` and by checking the manual page. One of the components returned is the cluster allocation, and we can use the `table` command to see if there are different allocations.

```
> names(km2)
[1] "cluster"  "centers"  "withinss" "size"
> table(km2$cluster, kmx$cluster)
     1  2
  1  0 18
  2 61  0
```

Exercise 6

There are 21 phenotypic variables available. Of these you can find out which are factors by simply using an apply-type function, as is shown in the code below. Notice that we are going to use either variables that are explicitly factors, or those that are implicitly factors (because a logical variable can take only two values it is really a factor). We also do a quick check to make sure that samples in the expression set are in the same order as the values in the clustering (this is harder with the `pam` output because the clustering vectors from it are not named, (at least not as of version 1.11.6). If the value returned by the second command below is not `TRUE` then all other computations are going to be incorrect and some corrective action is needed.

```
> sapply(pData(es2), function(x) is.factor(x) ||
      is.logical(x) )
          cod      diagnosis            sex
        FALSE          FALSE           TRUE
          age             BT      remission
        FALSE           TRUE           TRUE
           CR        date.cr        t(4;11)
        FALSE          FALSE           TRUE
       t(9;22)    cyto.normal          citog
         TRUE           TRUE          FALSE
     mol.biol fusion protein            mdr
         TRUE           TRUE           TRUE
        kinet            ccr        relapse
         TRUE           TRUE           TRUE
   transplant           f.u date last seen
         TRUE          FALSE          FALSE
```

Then for each of those variables, such as say, `mdr`, we can form a two-way table and use any one of your favorite tests for association. In the example below, we use the χ^2 test, but there are many others that you could use.

```
> tt1 = table(es2$mdr, km2$cluster)
> test1 = chisq.test(tt1)
> test1$p.value
[1] 0.391
```

You can then repeat this for each categorical variable and select the one with the best p-value as that variable that most closely aligns with the clustering.

Exercise 7

The answer is a little bit tricky, because the cluster labels are completely arbitrary. So we first create a two-way table, showing how the clusters align. We want to make this a bit interesting so we compute a three cluster k-means solution to compare with the three cluster PAM solution from above.

```
>  km3 = kmeans(gvals, centers=3, nstart=25)
>  table(km3$cluster, pam3$clustering)
     1  2  3
  1  0 16  0
  2 27  6  6
  3  0  5 19
```

There are 79 objects, and hence 3081 different pairs of objects. So for the comparison of interest we need to find out how many of those pairs went in the same cluster in both samples, how many pairs went in different clusters (these are both concordant values), and finally how many pairs were in one cluster, under one algorithm, but not in one cluster in the other.

```
>  outSamekm3 = outer(km3$cluster, km3$cluster,
                      FUN = function(x,y) x==y)
>  outSamepam3 = outer(pam3$clustering, pam3$clustering,
                       FUN = function(x,y) x==y)
>  inSBoth = outSamekm3 & outSamepam3
>  ##then we subtract 79, because an obs is in the same
>  ## cluster as itself this just means that the diagonal
>  ## is TRUE and divide by two
>  sameBoth = (sum(inSBoth) - 79)/2
>  ##not in the same one, in both are those FALSE entries
>  notSBoth = (!outSamekm3) & (!outSamepam3)
>  notSameBoth = sum(notSBoth)/2
>  ##those that are different, are TRUE in one and FALSE
>  ## in the other or vice versa
>
>  diffBoth = ((!outSamekm3) & outSamepam3) |
      (outSamekm3 & (!outSamepam3))
>  discordant = sum(diffBoth)/2
```

Thus we see that there are 682 pairs that are put in the same group in both clusterings. There are 1624 pairs that were not put in the same cluster, for either algorithm. And there were 775 pairs that were put in the same cluster for one of the algorithms, but not for the other.

Exercise 8

Basically repeat the steps given above, only now the table will have three groups instead of two.

Exercise 9

The first run has the samples more evenly spread among the 16 groups; there are no clusters of size 1, whereas for both other methods there are a number of clusters of size 1. In the code below we first show the distribution of samples among the 16 clusters for method 1, and then the number of clusters of different sizes for method 2. Notice that most of the clusters are of size 1.

```
> table(s1$unit.classif)
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
 7  6  4  4  6  2  5  7  2  3  5  2  6  7  4  9
> table(table(s2$unit.classif))
 1  2  3  6  9 13 17 18
 8  1  2  1  1  1  1  1
> table(table(s3$unit.classif))
 1  2  4  6  7 10 13 15
 7  2  1  1  1  1  2  1
```

Comparing clusterings is a bit hard, as there are no obvious labels to put on the clusters. We might want to devise some code that will tell us which samples were clustered together in two methods, and which number were together in one, not together in the other, and finally how many pairs were in different clusters for both outputs. But this is another topic, and one for which we do not have room.

Exercise 10

We first identify the samples, and then subset the expression values. Then we define some colors, so that we can easily tell the two groups apart.

```
> intOnes = s1$unit.classif == 13 | s1$unit.classif == 14
> gvsub = gvals[intOnes,]
> gvclasses = s1$unit.classif[intOnes]
> sideC = ifelse(gvclasses==13, "yellow", "blue")
> heatmap(t(gvsub), ColSideCol=sideC)
```

Exercise 11

So, we want to repeat our k-means analysis, as described above but with $k = 4$.

```
> km2sol = kmeans(gvals, centers=4, nstart=25)
> table(km2sol$cluster, SOMgp2)
   SOMgp2
     1  2  3  4  5  6  7
  1  0 14  0  3  1  0  4
  2  0  0  2  0  0  2 12
  3  3  5  2  0  0 12  0
  4  0  3  6  1  5  3  1
```

The results seem remarkably concordant. Next we remove those samples that correspond to the smaller groups and repeat the k-means analysis.

```
> dropInds = SOMgp2 %in% c("1", "4", "5")
> gvals2 = gvals[!dropInds,]
> km3 = kmeans(gvals2, centers=4, nstart=50)
> table(km3$cluster, SOMgp2[!dropInds])
     1  2  3  4  5  6  7
  1  0  1  0  0  0 14  0
  2  0  0  2  0  0  2 12
  3  0 14  0  0  0  0  5
  4  0  7  8  0  0  1  0
```

Exercise 12

This can be found directly from `silpam2` by mimicking the code above.

```
>   ans = silpam2[silpam2[, "sil_width"] < 0, ]
```

So there are five observations with negative silhouette widths.

Exercise 13

For this problem we make use of the output of `diana`, and hence work with `hc4`.

```
>  dcl4 = cutree(hc4, 4)
>  table(dcl4)
dcl4
 1  2  3  4
34 23  2 20
>  ## we presume the labels are in the order
>  ## given to the \indexTerm{clustering} algorithm
>
>  sild4 =  silhouette(dcl4, manDist)
```

We can compute the silhouette, and using the methods discussed in Section **??** we can plot this, or perform other operations on it.

Exercise 14

An example probe set is `33232_at`. Try

```
> t.test(exprs(esTT)["33232_at",]~esTT$mol.biol)
     Welch Two Sample t-test

data:  exprs(esTT)["33232_at", ] by esTT$mol.biol
t = 4.46, df = 76.8, p-value = 2.726e-05
alternative hypothesis: true difference in means is not
equal to 0
95 percent confidence interval:
 0.71 1.85
sample estimates:
mean in group BCR/ABL     mean in group NEG
                 8.66                  7.37
```

Exercise 15

One way to do this is to increase the value of `K` in:

```
> esTT.K = ALLfilt_bcrneg[ordtt[1:K],]
```

and then repeat the steps shown above to develop the principal components and biplot visualizations.