Practicalities of machine learning: applications to high-throughput genomics with Bioconductor

VJ Carey

August 5, 2007

Contents

1 Overview

The term *machine learning* refers to a family of computational methods for analyzing multivariate datasets. Each data point has a vector of *features* in a shared *feature space*, and may have a *class label* from some fixed finite set.

Supervised learning refers to processes that help articulate rules that map *feature vectors* to *class labels*. The class labels are known and function as supervisory information to guide rule construction. *Unsupervised learning* refers to processes that discover structure in collections of feature vectors. Typically the structure consists of a grouping of objects into clusters.

This practical introduction to machine learning will begin with a survey of a low-dimensional dataset to fix concepts, and will then address problems coming from expression arrays and array CGH data.

Some basic points to consider at the start:

- Distinguish predictive modeling from inference on model parameters. Typical work in epidemiology focuses on estimation of relative risks, and random samples are not required. Typical work with machine learning tools targets estimation (and minimization) of the misclassification rate. Representative samples are required for this task.

- "Two cultures": model fitters vs. algorithmic predictors. If statistical models are correct, parameter estimation based on the mass of data can yield optimal discriminators (e.g., LDA). Algorithmic discriminators tend to prefer to identify boundary cases and downweight the mass of data (e.g., boosting, svm).

- Different learning tools have different capabilities. There is little *a priori* guidance on matching learning algorithms to aspects of problems. While it is convenient to sift through a variety of approaches, one must pay a price for the model search.

- Data and model/learner visualization are important, but visualization in higher dimensional data structures is hard. Dynamic graphics can help; look at ggobi and Rggobi for this.

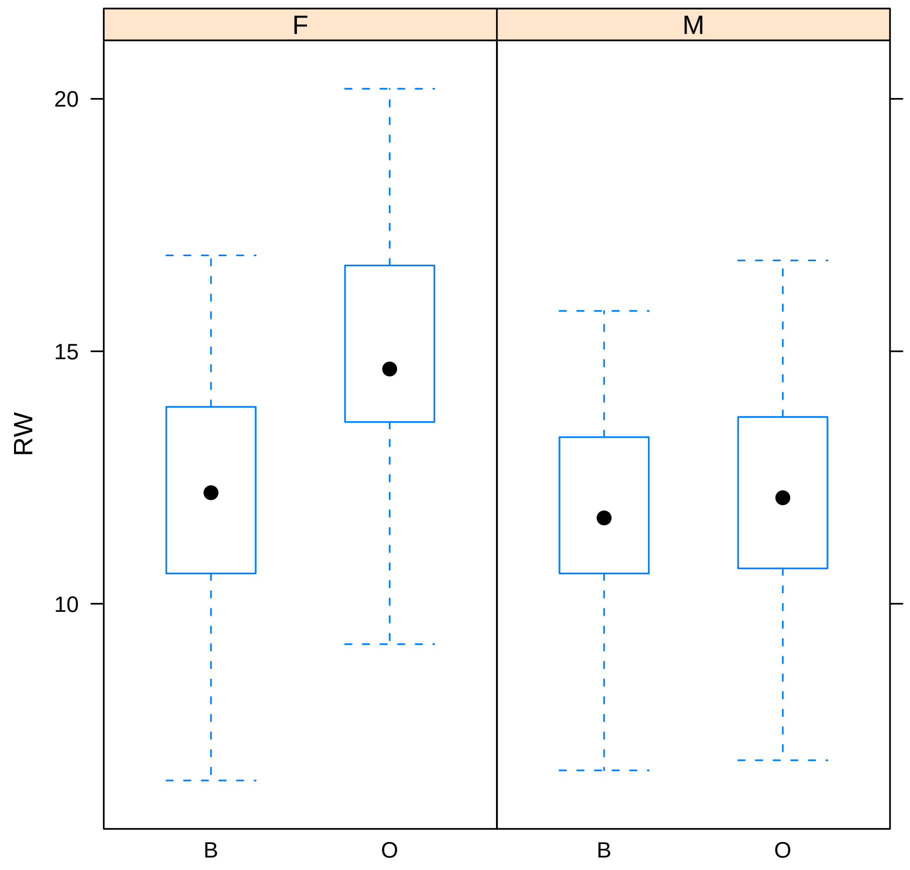
- These notes provide very little mathematical background on the methods; see for example Ripley (*Pattern recognition and neural networks*, 1995), Duda, Hart, Stork (*Pattern classification*), Hastie, Tibshirani and Friedman (2003, *Elements of statistical learning*) for copious background.

2 Getting acquainted with machine learning via the crabs data

2.1 Attaching and checking the data

The following steps bring the crabs data into scope and illustrate aspects of its structure.

```
> library(MASS)
> data(crabs)
> dim(crabs)

[1] 200   8

> crabs[1:4, ]

  sp sex index  FL  RW   CL   CW  BD
1  B   M     1 8.1 6.7 16.1 19.0 7.0
2  B   M     2 8.8 7.7 18.1 20.8 7.4
3  B   M     3 9.2 7.8 19.0 22.4 7.7
4  B   M     4 9.6 7.9 20.1 23.1 8.2

> table(crabs$sex)

  F   M
100 100

> library(lattice)
> print(bwplot(RW ~ sp | sex, data = crabs))
```



We will regard these data as providing five features (FL, RW, CL, CW, BD) and a pair of class labels (sex, sp=species). We may regard this as a four class problem, or as two two class problems.

2.2 A simple classifier derived by human reasoning

Our first problem does not involve any computations. If you want to write R code to solve the problem, do so, but use prose first.

- *Question 1.* On the basis of the lattice plot given above, comment on the prospects for predicting species on the basis of RW. State a rule for computing the predictions. Describe how to assess the performance of your rule.

2.3 Prediction via logistic regression

A simple approach to prediction involves logistic regression.

```
> m1 = glm(sp ~ RW, data = crabs, fam = binomial)
> summary(m1)
```

```
Call:
glm(formula = sp ~ RW, family = binomial, data = crabs)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.67807  -1.08840  -0.04168   1.07160   1.88030

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.44908    0.82210  -4.195 2.72e-05 ***
RW           0.27080    0.06349   4.265 2.00e-05 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 277.26  on 199  degrees of freedom
Residual deviance: 256.35  on 198  degrees of freedom
AIC: 260.35

Number of Fisher Scoring iterations: 4
```

- *Question 2.* Write down the statistical model corresponding to the R expression above. How can we derive a classifier from this model?
- *Question 3.* Perform the following computations. Discuss their interpretation. What are the estimated error rates of the two models? Is the second model better?

```
> table(predict(m1, type = "response") > 0.5, crabs$sp)
> m2 = update(m1, subset = sex == "F")
> table(predict(m2, type = "response") > 0.5, crabs$sp[crabs$sex ==
+     "F"])
```

2.4 The cross-validation concept

Cross-validation is a technique that is widely used for reducing bias in estimation of predictive accuracy. Briefly, the dataset is deterministically partitioned into a series of training and test sets. The model is built for each training set and evaluated on the test set. The accuracy measures are averaged over this series of fits. Leave-one-out cross-validation consists of N fits, with N training sets of size N-1 and N test sets of size 1.

Convenient interfaces for cross-validation with Bioconductor's MLInterfaces are only worked out for ExpressionSets. So we will convert the crabs data to such a structure in the next few commands.

```
> feat = t(data.matrix(crabs[crabs$sex == "F", -c(1:3)]))
> library(Biobase)
> pd = new("AnnotatedDataFrame", crabs[crabs$sex == "F", 1:2])
> crFES = new("ExpressionSet", exprs = feat, phenoData = pd)
> crFES

ExpressionSet (storageMode: lockedEnvironment)
assayData: 5 features, 100 samples
  element names: exprs
phenoData
  rowNames: 51, 52, ..., 200 (100 total)
  varLabels and varMetadata:
    sp: NA
    sex: NA
featureData
  featureNames: FL, RW, ..., BD (5 total)
  varLabels and varMetadata: none
experimentData: use 'experimentData(object)'
Annotation character(0)
```

First let's use `MLearn` from the *MLInterfaces* package to fit a single logistic model. `MLearn` requires you to specify a training index set. We use `c(1:30, 51:80)` to choose a training set of size 60, balanced between two species (because we know the ordering of records). This procedure also requires you to specify a probability threshold for classification. We use a typical default of 0.5. If the predicted probability of being "O" exceeds 0.5, we classify to "O", otherwise to "B".

```
> library(MLInterfaces)
> ml1 = MLearn(sp ~ RW, crFES, "logistic", c(1:30, 51:80), mlSpecials = list(thresh =
> ml1

MLOutput instance, method= logistic
Call:
 MLearn(formula = formula, data = data, method = method, trainInd = trainInd,
    mlSpecials = mlSpecials)
predicted class distribution:
TRUE
  40
summary of class membership probabilities:
          [,1]
Min.    0.7553
1st Qu. 0.8861
Median  0.9803
```

```
Mean    0.9355
3rd Qu. 0.9917
Max.    0.9997
```

```
> confuMat(ml1)

     predicted
given TRUE
    B   20
    O   20
```

- *Question 4.* What does the report on `ml1` tell you about predictions with this model? Can you reconcile this with the results in model `m2`? [Hint – non-randomness of the selection of the training set is a problem.]

- *Question 5.* Modify the MLearn call to obtain a predictor that is more successful on the test set.

Now we will illustrate cross-validation. First, we scramble the order of records in the `ExpressionSet` so that sequentially formed groups are approximately random samples.

```
> set.seed(123)
> scrFES = crFES[, sample(1:100, size = 100, replace = FALSE)]
```

We invoke the MLearn and then xvalML methods to illustrate different predictive assessment approaches.

```
> sml1 = MLearn(sp ~ RW, scrFES, "logistic", c(1:30, 51:80), mlSpecials = list(thresh
> confuMat(sml1)

     predicted
given FALSE TRUE
    B    13   10
    O     3   14

> smx1 = xvalML(sp ~ RW, scrFES, "logistic", "LOG", rep(1:5, each = 20),
+     mlSpecials = list(thresh = 0.5))
> table(scrFES$sp, smx1)

   smx1
    FALSE TRUE
  B    32   18
  O    13   37
```

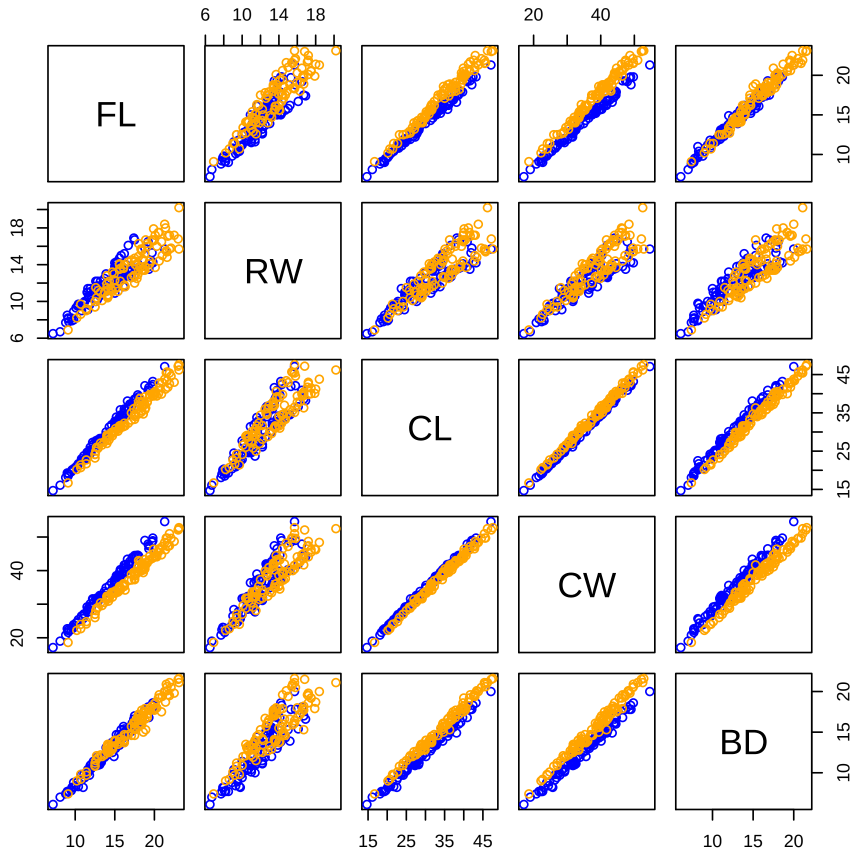
- *Question 6.* Define clearly the difference between models sml1 and smx1 and state the misclassification rate estimates associated with each model.

2.5 Exploratory multivariate analysis

2.5.1 Scatterplots

- *Question 7.* Interpret the following code. Modify it to depict the pairwise configurations with different colors for crab genders.

```
> pairs(crabs[, -c(1:3)], col = ifelse(crabs$sp == "B", "blue",
+     "orange"))
```



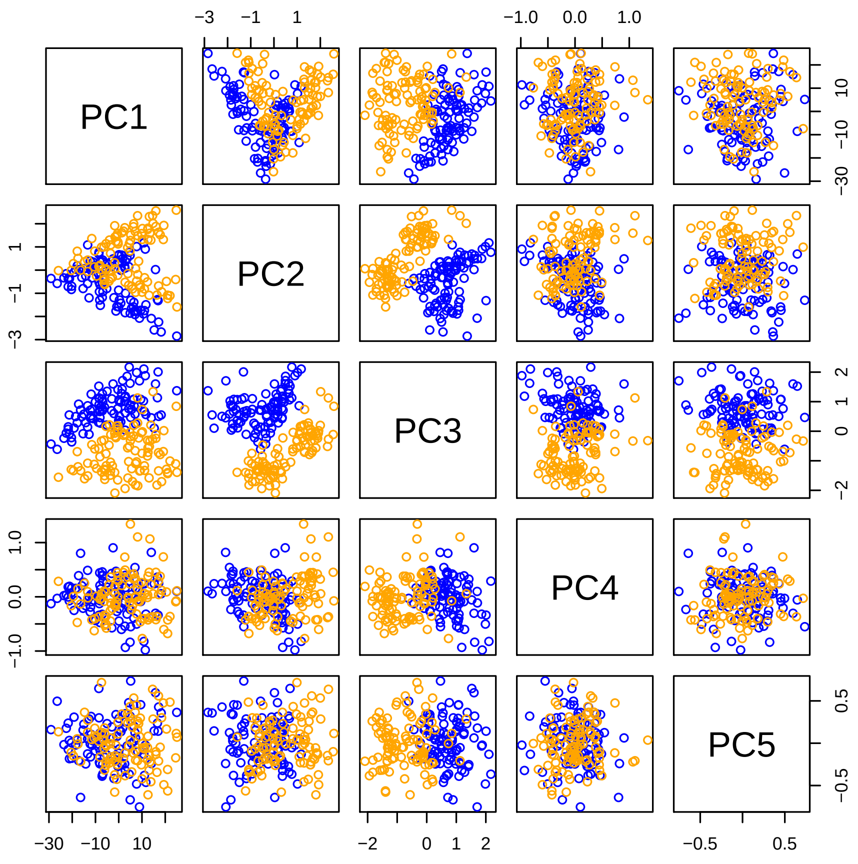
2.5.2 Principal components; biplot

Principal components analysis transforms the multivariate data X into a new coordinate system. If the original variables are X1, ..., Xp, then the variables in the new representation are denoted PC1, ..., PCp. These new variables have the properties that PC1 is the linear combination of the X having maximal variance, PC2 is the linear combination of residuals of X after projecting on PC1, and so on. If most of the variation in $X_{n\times p}$ can be captured in a low dimensional subspace of X, scatterplots of the first few principal components will depict this.

Formally, we can compute the PC using the singular value decomposition of X, in which $X = UDV^t$, where $U_{n\times p}$ and $V_{p\times p}$ are orthonormal, and D is a diagonal matrix of p nonnegative singular values. The principal components transformation is $XV = UD$, and if D is structured so that $D_{ii} \geq D_{jj}$ whenever $i > j$, then column i of XV is PCi. Note also that $D_{ii} = \sqrt{n-1}$SD PCi.

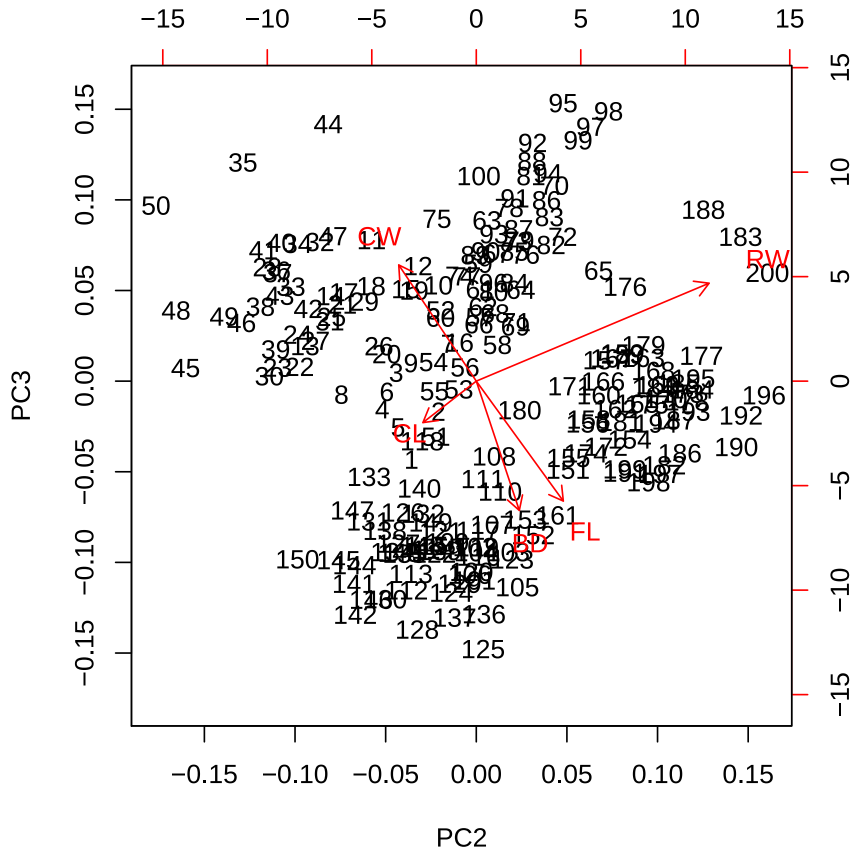
```
> pc1 = prcomp(crabs[, -c(1:3)])
```

```
> pairs(pc1$x, col = ifelse(crabs$sp == "B", "blue", "orange"))
```



The biplot shows the data in PC space and also shows the relative contributions of the original variables in composing the transformation.

```
> biplot(pc1, choices = 2:3)
```



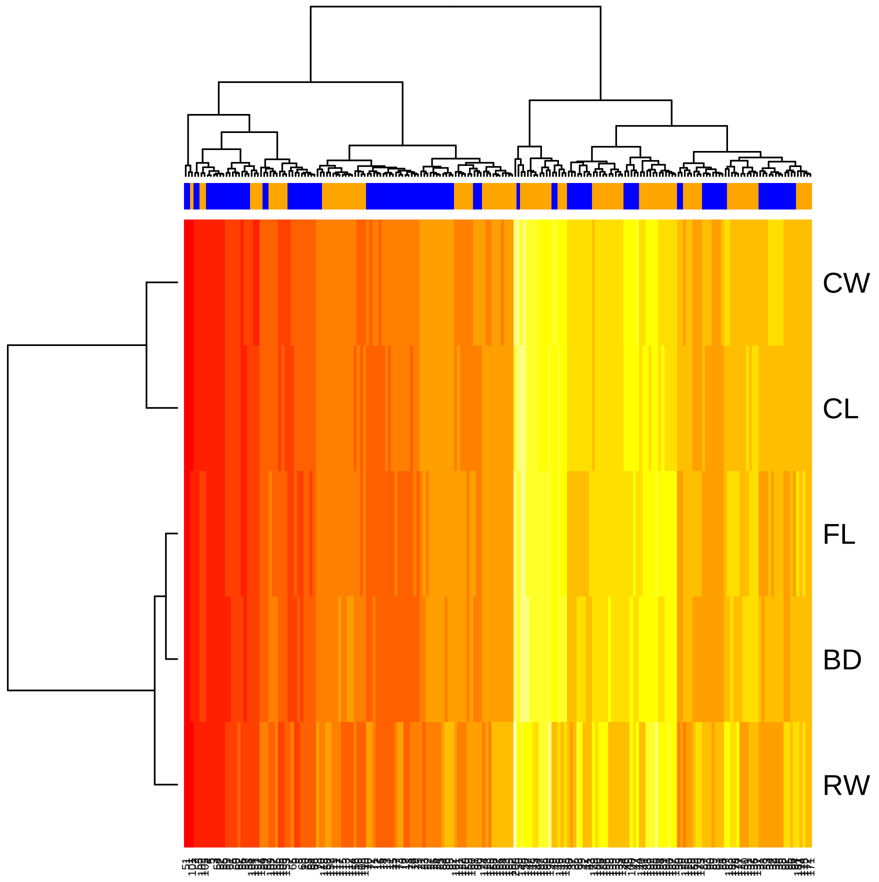
2.5.3 Clustering

A familiar technique for displaying multivariate data in high-throughput biology is called the heatmap. In this display, samples are clustered as columns, and features as rows. The clustering technique used by default is R 'hclust'. This procedure builds a clustering tree for the data as follows. Distances are computed between feature vectors for all N observations. The two closest observations are joined and are regarded as a new object, so there are $N - 1$ objects (clusters) at this point. This process is repeated until 1 cluster is formed; the clustering tree shows the process by which clusters are created via agglomeration.

There are several ways to measure distances between clusters, once these have been formed. Single-linkage clustering takes the distance between two clusters to be the shortest distance between any two members of the different clusters; average linkage averages all the distances between members; complete-linkage uses hte maximum distance between any two members of the different clusters. Other methods are available in hclust.

The following figure shows cluster trees for samples and features, using all defaults for the heatmap function. A tiled display at the top shows the species codes for the samples.

```
> X = data.matrix(crabs[, -c(1:3)])
> heatmap(t(X), ColSideColors = ifelse(crabs$sp == "O", "orange",
+     "blue"))
```



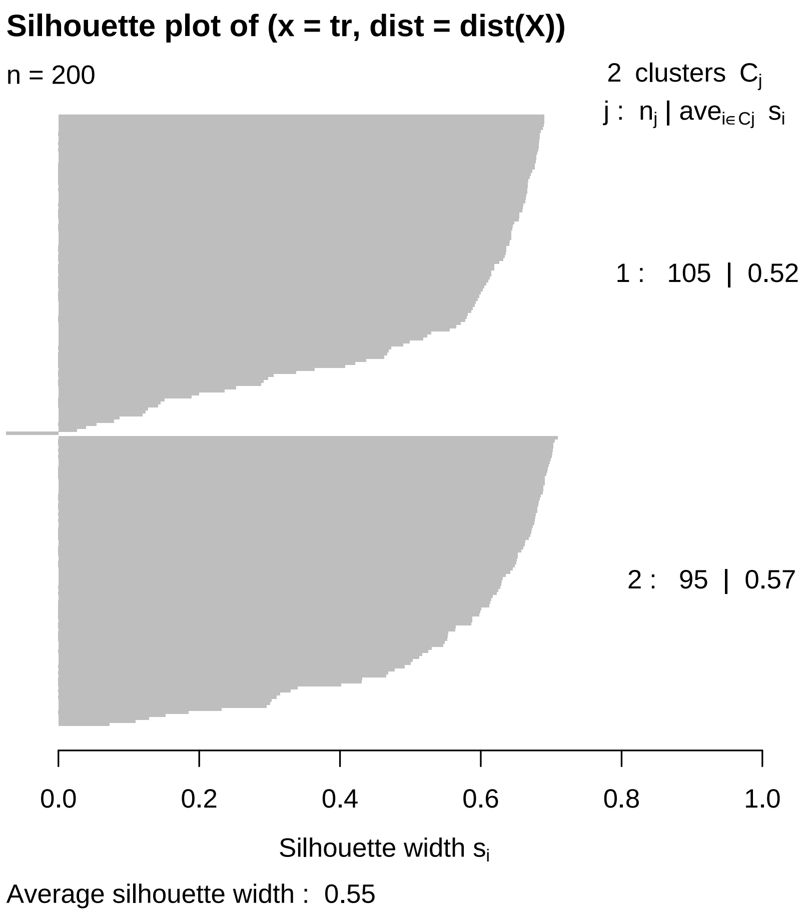
Typically clustering is done in the absence of labels – it is an example of unsupervised machine learning. We can ask whether the clustering provided is a 'good' one using the measurement of a quantity called the *silhouette*. This is defined in R documentation as follows:

```
For each observation i, the _silhouette width_ s(i) is defined as
follows:
 Put a(i) = average dissimilarity between i and all other points
of the cluster to which i belongs (if i is the _only_ observation
in its cluster, s(i) := 0 without further calculations). For all
_other_ clusters C, put d(i,C) = average dissimilarity of i to all
observations of C.  The smallest of these d(i,C) is b(i) := min_C
```

```
     d(i,C), and can be seen as the dissimilarity between i and its
     "neighbor" cluster, i.e., the nearest one to which it does _not_
     belong. Finally,

             s(i) := ( b(i) - a(i) ) / max( a(i), b(i) ).
```

We can compute the silhouette for any partition of a dataset, and can use the hierarchical clustering result to define a partition as follows:

```
> cl = hclust(dist(X))
> tr = cutree(cl, 2)
> table(tr)

tr
  1   2
105  95

> library(cluster)
> sil = silhouette(tr, dist(X))
> plot(sil)
```



- *Question 8.* In the preceding, we have used default `dist`, and default clustering algorithm for the heatmap. Investigate the impact of altering the choice of distance and clustering method on the clustering performance, both in relation to capacity to recover groups defined by species and in relation to the silhouette distribution.

- *Question 9.* The PCA shows that the data configuration in PC2 and PC3 is at least bifurcated. Apply hierarchical and K-means clustering to the two-dimensional data in this subspace, and compare results with respect to capturing the species $\times$ gender labels, and with respect to silhouette values. For example, load the exprs slot of crES [see just below for the definition of this structure] with the PCA reexpression of the features, call the result pcrES, and then:

```
> ff = kmeansB(pcrES[2:3,], k=4)
> table(ff@clustIndices, crES$spsex)
```

2.6 Supervised learning

In this section we will examine procedures for polychotomous prediction. We want to be able to use the measurements to predict both species and sex of the crab. Again we would like to use the MLInterfaces infrastructure, so an ExpressionSet container will be useful.

```
> feat2 = t(data.matrix(crabs[, -c(1:3)]))
> pd2 = new("AnnotatedDataFrame", crabs[, 1:2])
> crES = new("ExpressionSet", exprs = feat2, phenoData = pd2)
> crES$spsex = paste(crES$sp, crES$sex, sep = ":")
> table(crES$spsex)
```

```
B:F B:M O:F O:M
 50  50  50  50
```

We will permute the samples so that simple selections for training set indices are random samples.

```
> set.seed(1234)
> crES = crES[, sample(1:200, size = 200, replace = FALSE)]
```

2.6.1 RPART

A classic procedure is recursive partitioning.

```
> library(rpart)
> tr1 = MLearn(spsex ~ ., crES, "rpart", 1:140)
> tr1
```

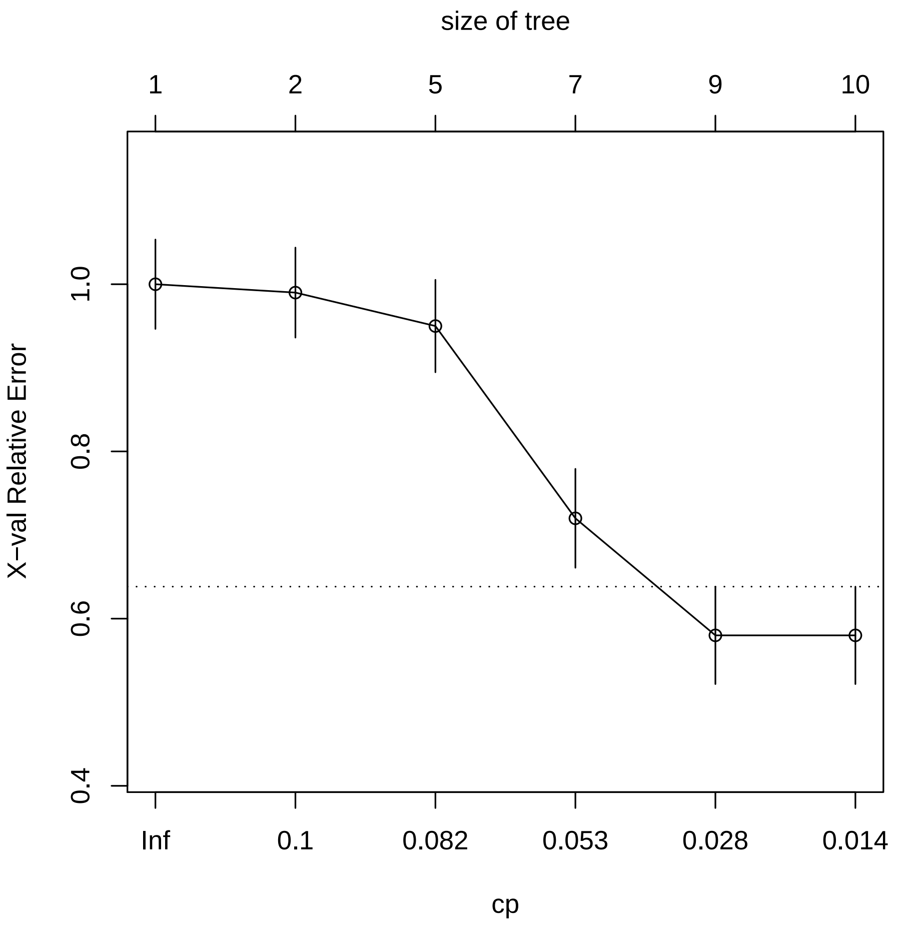
```
MLOutput instance, method= rpart
Call:
 MLearn(formula = formula, data = data, method = method, trainInd = trainInd,
    mlSpecials = mlSpecials)
predicted class distribution:
B:F B:M O:F O:M
 17  11  16  16
```

```
> confuMat(tr1)
```

```
     predicted
given B:F B:M O:F O:M
  B:F  10   4   3   0
  B:M   2   4   0   4
  O:F   3   1  11   3
  O:M   2   2   2   9
```

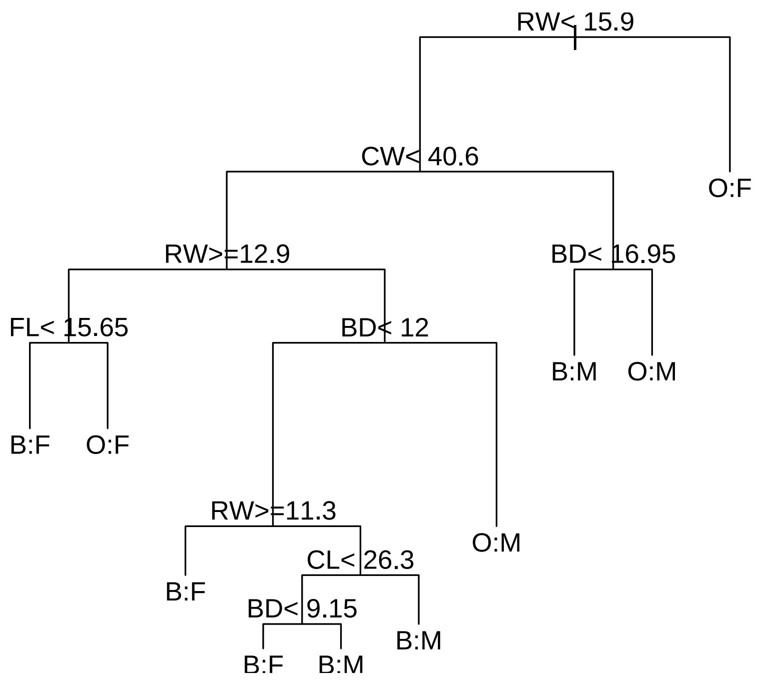
This procedure includes a diagnostic tool called the cost-complexity plot:

```
> plotcp(RObject(tr1))
```



The actual tree is

```
> plot(RObject(tr1))
> text(RObject(tr1))
```



2.6.2 Random forests

A generalization of recursive partitioning is obtained by creating a collection of trees by bootstrap-sampling cases and randomly sampling from features available for splitting at nodes.

```
> set.seed(124)
> library(randomForest)

randomForest 4.5-18
Type rfNews() to see new features/changes/bug fixes.

> rf1 = MLearn(spsex ~ ., crES, "randomForest", 1:140)
> rf1

MLOutput instance, method= randomForest
Call:
 MLearn(formula = formula, data = data, method = method, trainInd = trainInd,
    mlSpecials = mlSpecials)
predicted class distribution:
B:F B:M O:F O:M
 18  12  17  13
```

```
> cm = confuMat(rf1)
> cm
```

```
     predicted
given B:F B:M O:F O:M
  B:F  15   2   0   0
  B:M   1   9   0   0
  O:F   1   0  15   2
  O:M   1   1   2  11
```

The single split error rate is estimated at 17%.

- *Question 10.* What is the out-of-bag error rate for rf1? Obtain a cross-validated estimate of misclassification error using randomForest in xvalML.

2.6.3 Linear discriminants

```
> ld1 = MLearn(spsex ~ ., crES, "lda", 1:140)
> ld1
```

```
MLOutput instance, method= lda
Call:
 MLearn(formula = formula, data = data, method = method, trainInd = trainInd,
    mlSpecials = mlSpecials)
predicted class distribution:
B:F B:M O:F O:M
 18   9  18  15
```

```
> confuMat(ld1)
```

```
     predicted
given B:F B:M O:F O:M
  B:F  17   0   0   0
  B:M   1   9   0   0
  O:F   0   0  18   0
  O:M   0   0   0  15
```

```
> xvld = xvalML(spsex ~ ., crES, "lda", "LOG", rep(1:5, each = 40))
> table(crES$spsex, xvld)
```

```
     xvld
      B:F B:M O:F O:M
  B:F  48   2   0   0
  B:M   5  45   0   0
  O:F   0   0  47   3
  O:M   0   0   0  50
```

- *Question 11.* Use the balKfold function to generate an index set for partitions that is balanced with respect to class distribution. Check the balance and repeat the cross validation.

2.6.4 Neural net

```
> nn1 = MLearn(spsex ~ ., crES, "nnet", 1:140, size = 3, decay = 0.1)

# weights:  34
initial  value 213.055045
iter  10 value 193.520830
iter  20 value 131.748155
iter  30 value 100.092904
iter  40 value 43.061828
iter  50 value 36.883552
iter  60 value 34.146887
iter  70 value 33.420116
iter  80 value 32.781644
iter  90 value 32.506339
iter 100 value 32.448393
final  value 32.448393
stopped after 100 iterations

> nn1

MLOutput instance, method= nnet
a 5-3-4 network with 34 weights
inputs: FL RW CL CW BD
output(s): spsex
options were - softmax modelling  decay=0.1
Call:
 MLearn(formula = formula, data = data, method = method, trainInd = trainInd,
    mlSpecials = mlSpecials, size = 3, decay = 0.1)
predicted class distribution:
B:F B:M O:F O:M
 18   9  18  15
```

```
summary of class membership probabilities:
              B:F       B:M       O:F       O:M
Min.    0.0002427 0.0001249 0.0001363 0.0001787
1st Qu. 0.0099840 0.0008553 0.0131700 0.0036280
Median  0.0201700 0.0168000 0.0257600 0.0188400
Mean    0.2824000 0.1718000 0.3043000 0.2415000
3rd Qu. 0.8088000 0.0530200 0.9104000 0.2250000
Max.    0.9714000 0.9766000 0.9708000 0.9710000
```

```
> RObject(nn1)
```

```
a 5-3-4 network with 34 weights
inputs: FL RW CL CW BD
output(s): spsex
options were - softmax modelling  decay=0.1
```

```
> confuMat(nn1)
```

```
     predicted
given B:F B:M O:F O:M
  B:F  17   0   0   0
  B:M   1   9   0   0
  O:F   0   0  18   0
  O:M   0   0   0  15
```

```
> xvnn = xvalML(spsex ~ ., crES, "nnet", "LOG", rep(1:5, each = 40),
+     size = 3, decay = 0.1)
```

```
> table(crES$spsex, xvnn)
```

```
     xvnn
      B:F B:M O:F O:M
  B:F  48   2   0   0
  B:M   4  46   0   0
  O:F   0   0  49   1
  O:M   0   0   1  49
```

2.6.5 SVM

```
> sv1 = MLearn(spsex ~ ., crES, "svm", 1:140)
> sv1
```

```
MLOutput instance, method= svm
Call:
 MLearn(formula = formula, data = data, method = method, trainInd = trainInd,
    mlSpecials = mlSpecials)
predicted class distribution:
B:F B:M O:F O:M
 27   8  13  12
```

```
> RObject(sv1)
```

```
Call:
svm(formula = formula, data = sdata)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.2

Number of Support Vectors:  131
```

```
> confuMat(sv1)
```

```
     predicted
given B:F B:M O:F O:M
  B:F  17   0   0   0
  B:M   3   7   0   0
  O:F   5   0  13   0
  O:M   2   1   0  12
```

```
> xvsv = xvalML(spsex ~ ., crES, "svm", "LOG", rep(1:5, each = 40))
```

```
> table(crES$spsex, xvsv)
```

```
     xvsv
      B:F B:M O:F O:M
  B:F  43   3   4   0
  B:M  15  28   0   7
  O:F  12   0  37   1
  O:M   4   1   0  45
```

3 Learning with expression arrays

Here we will concentrate on ALL: acute lymphocytic leukemia, B-cell type.

3.1 Phenotype reduction

We will identify expression patterns that discriminate individuals with BCR/ABL fusion in B-cell leukemia.

```
> library(ALL)
> data(ALL)
> bALL = ALL[, substr(ALL$BT, 1, 1) == "B"]
> fus = bALL[, bALL$mol.biol %in% c("BCR/ABL", "NEG")]
> fus$mol.biol = factor(fus$mol.biol)
> fus

ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 79 samples
  element names: exprs
phenoData
  rowNames: 01005, 01010, ..., 84004 (79 total)
  varLabels and varMetadata:
    cod:  Patient ID
    diagnosis:  Date of diagnosis
    ...: ...
    date last seen:  date patient was last seen
    (21 total)
featureData
  featureNames: 1000_at, 1001_at, ..., AFFX-YEL024w/RIP1_at (12625 total)
  varLabels and varMetadata: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation [1] "hgu95av2"
```

3.2 Nonspecific filtering

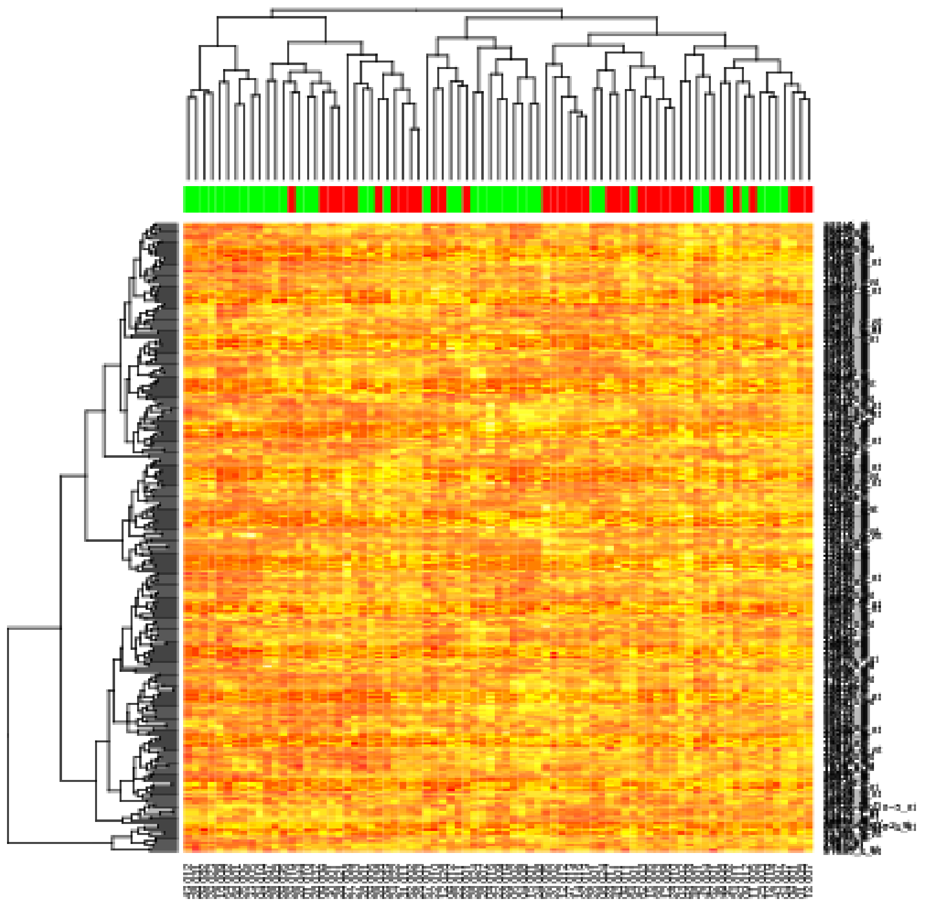
We can nonspecifically filter to 300 genes (to save computing time) with largest measures of robust variation across all samples:

```
> mads = apply(exprs(fus), 1, mad)
> fusk = fus[mads > sort(mads, decr = TRUE)[300], ]
> fcol = ifelse(fusk$mol.biol == "NEG", "green", "red")
```

3.3 Exploratory work

For exploratory data analysis, a heatmap is customary.

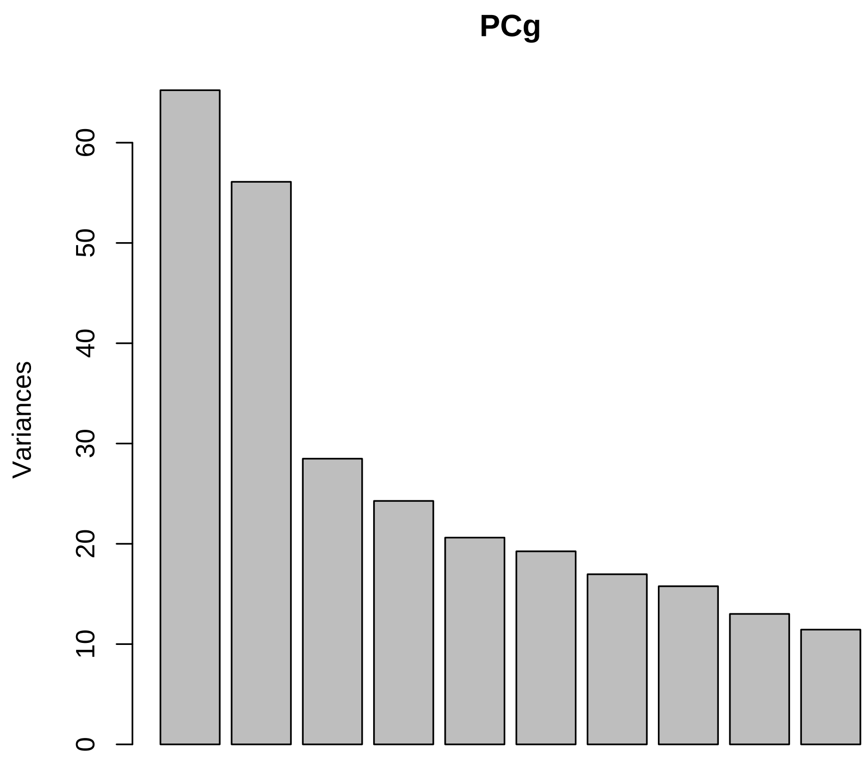
```
> heatmap(exprs(fusk), ColSideColors = fcol)
```



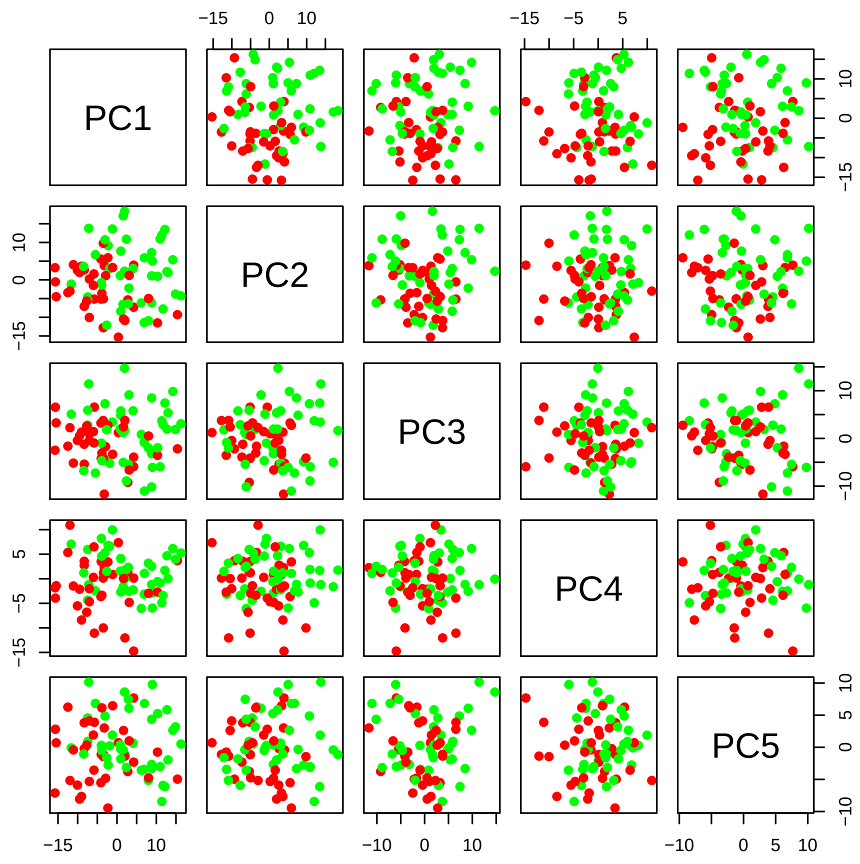
Principal components and a biplot may be more revealing. How many principal components are likely to be important?

```
> PCg = prcomp(t(exprs(fusk)))
```

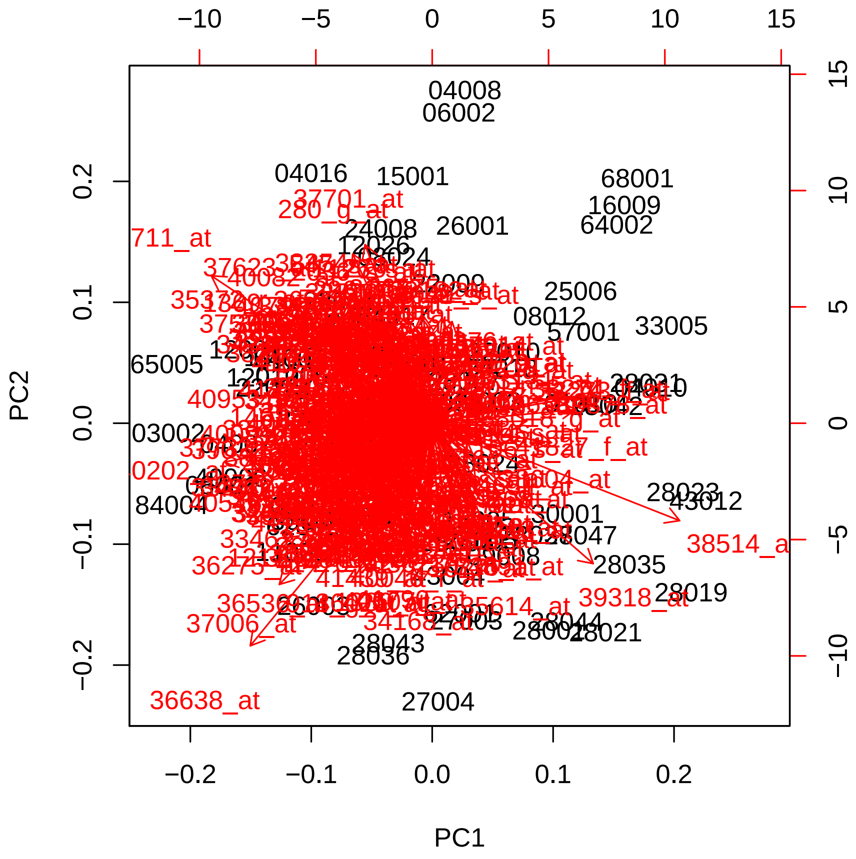
```
> plot(PCg)
```



```
> pairs(PCg$x[, 1:5], col = fcol, pch = 19)
```



```
> biplot(PCg)
```



- *Question 12.* Modify the biplot so that instead of plotting sample ID, the symbol ”O” is plotted for a NEG sample and ”+” is plotted for a BCR/ABL sample.
- *Question 13.* Consider the following code

  ```
  chkT = function (x, eset=fusk) {
  t.test(exprs(eset)[x, eset$mol.b == "NEG"], exprs(eset)[x, eset$mol.b ==
      "BCR/ABL"]) }
  ```

  Use it in conjunction with the biplot to interpret expression patterns of genes that appear to be important in defining the PCs.

3.4 Classifier construction

3.4.1 Demonstrations

Diagonal LDA has a good reputation. Let’s try it first, followed by neural net and random forests. We will not attend to tuning the latter two, defaults or guesses for key parameters are used.

```
> dld1 = MLearn(mol.biol ~ ., fusk, "dlda", 1:40)
> dld1

MLOutput instance, method= dlda
Call:
 MLearn(formula = formula, data = data, method = method, trainInd = trainInd,
    mlSpecials = mlSpecials)
predicted class distribution:
BCR/ABL     NEG
     27      12

> confuMatTrain(dld1)

         predicted
given     BCR/ABL NEG
  BCR/ABL      20   1
  NEG           3  16

> confuMat(dld1)

         predicted
given     BCR/ABL NEG
  BCR/ABL      15   1
  NEG          12  11

> nnALL = MLearn(mol.biol ~ ., fusk, "nnet", 1:40, size = 5, decay = 0.01,
+     MaxNWts = 2000)

# weights:  1506
initial  value 30.563126
iter  10 value 21.358174
iter  20 value 14.831225
iter  30 value 12.972838
iter  40 value 12.573819
iter  50 value 6.736599
iter  60 value 4.861514
iter  70 value 1.647463
iter  80 value 1.109953
iter  90 value 1.023392
iter 100 value 0.867387
final  value 0.867387
stopped after 100 iterations

> confuMatTrain(nnALL)
```

```
         predicted
given     BCR/ABL NEG
  BCR/ABL      21   0
  NEG           0  19

> confuMat(nnALL)

         predicted
given     BCR/ABL NEG
  BCR/ABL      15   1
  NEG           7  16

> rfALL = MLearn(mol.biol ~ ., fusk, "randomForest", 1:40)
> rfALL

MLOutput instance, method= randomForest
Call:
 MLearn(formula = formula, data = data, method = method, trainInd = trainInd,
    mlSpecials = mlSpecials)
predicted class distribution:
BCR/ABL     NEG
     25      14

> confuMatTrain(rfALL)

         predicted
given     BCR/ABL NEG
  BCR/ABL      21   0
  NEG           0  19

> confuMat(rfALL)

         predicted
given     BCR/ABL NEG
  BCR/ABL      15   1
  NEG          10  13
```

None of these are extremely impressive, but the problem may just be very hard. An interesting new proposal is RDA, regularized discriminant analysis (package rda, Guo, Hastie, Tibshirani 2007 Biostatistics). This algorithm recognizes the fact that covariance matrix estimation in high dimensional data is very inaccurate, so the estimator is shrunk towards the identity. We have a rudimentary interface, in which the key parameters are chosen by native cross-validation (in rda.cv) and then applied once to get the final object.

```
> rdaALL = MLearn(mol.biol ~ ., fusk, "rdacv", 1:40)

> rdaALL

MLOutput instance, method= rdacv
Call:
 MLearn(formula = formula, data = data, method = method, trainInd = trainInd,
    mlSpecials = mlSpecials)
predicted class distribution:
BCR/ABL     NEG
     16      23

> confuMatTrain(rdaALL)

         predicted
given     BCR/ABL NEG
  BCR/ABL      21   0
  NEG           1  18

> confuMat(rdaALL)

         predicted
given     BCR/ABL NEG
  BCR/ABL      13   3
  NEG           3  20
```

A by-product of the algorithm is a set of retained genes that were found to play a role in discrimination. This can be established as follows:

```
> library(hgu95av2)
> inds = as.numeric(which(RObject(rdaALL)$gene.list[, 1, ] > 0))
> saved = featureNames(fusk)[inds]
> as.list(hgu95av2GENENAME)[saved[1:4]]

$`1635_at`
[1] "v-abl Abelson murine leukemia viral oncogene homolog 1"

$`1674_at`
[1] "v-yes-1 Yamaguchi sarcoma viral oncogene homolog 1"

$`1915_s_at`
[1] "v-fos FBJ murine osteosarcoma viral oncogene homolog"

$`1916_s_at`
[1] "v-fos FBJ murine osteosarcoma viral oncogene homolog"
```

- *Question 14.* How can we assess the relative impacts of regularization (expanding the covariance model beyond that of DLDA, which was shown above to do poorly, but without relying on the full covariance) and implicit feature selection conducted in RDA?

3.4.2 Gene set appraisal

- *Question 15.* We can assess the predictive capacity of a set of genes by restricting the ExpressionSet to that set and using the best classifier appropriate to the problem. We can also assess the incremental effect of combining gene sets, relative to using them separately.

  One collection of gene sets that is straightforward to use and interpret is provided by the keggorth package (see also GSEABase). Here's how we can define the ExpressionSets for genes annotated by KEGG to Environmental (Genetic) Information Processing:

  ```
  > library(keggorth)
  > data(KOgraph)
  > adj(KOgraph, nodes(KOgraph)[1])

  $KO.June07root
  [1] "Metabolism"
  [2] "Genetic Information Processing"
  [3] "Environmental Information Processing"
  [4] "Cellular Processes"
  [5] "Human Diseases"

  > EIP = getKOprobes("Environmental Information Processing")
  > GIP = getKOprobes("Genetic Information Processing")
  > length(intersect(EIP, GIP))

  [1] 33

  > EIPi = setdiff(EIP, GIP)
  > GIP = setdiff(GIP, EIP)
  > EIP = EIPi
  > Efusk = fusk[featureNames(fusk) %in% EIP, ]
  > Gfusk = fusk[featureNames(fusk) %in% GIP, ]
  ```

  Obtain and assess the predictive capacity of the genes annotated to ”Cell Growth and Death”.

- *Question 16.* How many of the genes identified by RDA as important for discriminating fusion are annotated to Genetic Information Processing in the KEGG orthology?

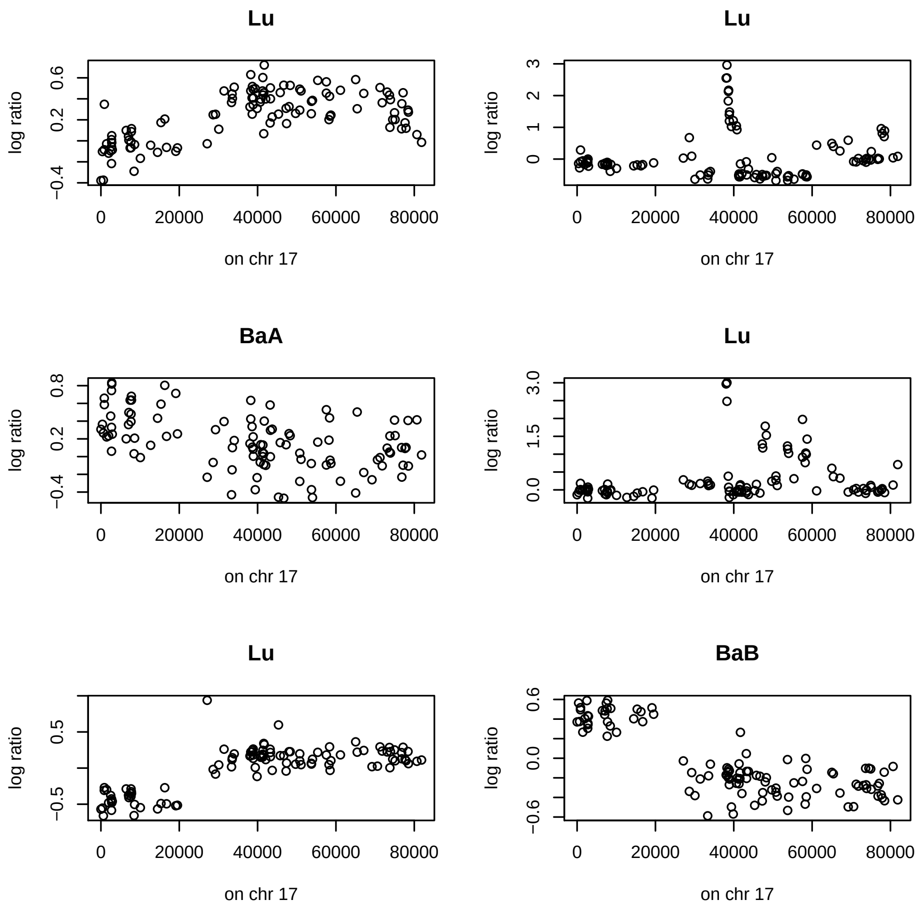
4 Learning with array CGH

This section contains two parts, one of which is left as a pair of open questions. In the first part, we use rpart as a *device* to perform segmentation of aCGH log intensity ratios. In the second part, we consider how to extract features from the CGH profile to learn about gene cluster activation and phenotype.

4.1 Working with Neve2006

We are going to examine some aCGH copy number data as published by Neve et al 2006.

```
> library(Neve2006)
> data(neveCGHmatch)
> chr = function(x) pData(featureData(x))$Chrom
> kb = function(x) pData(featureData(x))$kb
> nc17 = neveCGHmatch[chr(neveCGHmatch) == 17, ]

> par(mfrow = c(3, 2))
> for (i in 1:6) plot(logRatios(nc17)[, i] ~ kb(nc17), main = nc17$geneCluster[i],
+     ylab = "log ratio", xlab = "on chr 17")
> par(mfrow = c(1, 1))
```



4.2 Segmentation via rpart

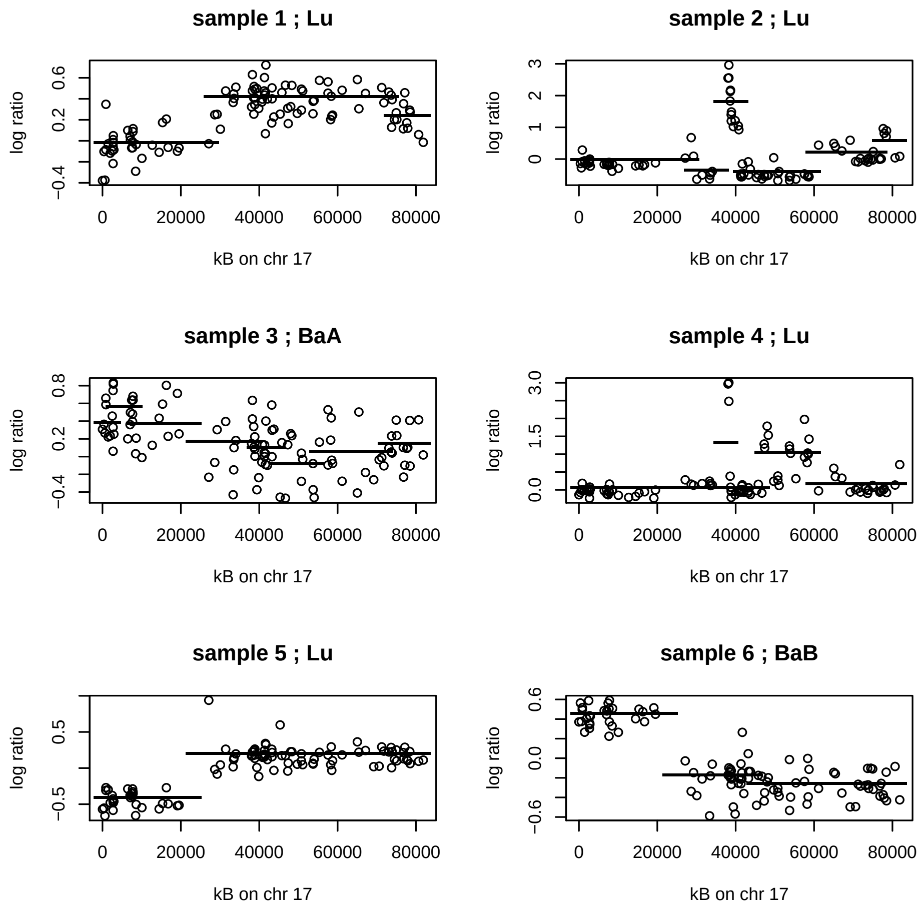
First we are going to use rpart as a device for estimating a piecewise constant model for log ratios. No optimality is claimed for the technique; it can be compared to tilingArray::segment and DNAcopy::segment. However, there are infrastructure gains obtained by using a bona fide modeling tool, which has a generic predict().

We develop additional infrastructure here:

```
> setOldClass("rpart")
> setClass("rpSeg", representation(obj = "rpart", samp = "numeric",
+     LR = "numeric", KB = "numeric", chr = "character", rpCall = "call",
+     pdLabel = "character"))
> setMethod("show", "rpSeg", function(object) {
+     cat("rpart segmentation for sample", object@samp, "phenoLabel",
+         object@pdLabel, "\n")
+     cat("range of clone locations on chr", object@chr, "\n")
+     print(range(object@KB))
+ })
> setMethod("plot", "rpSeg", function(x, y, ...) {
+     rng = range(x@KB)
+     X = seq(rng[1], rng[2], 500)
+     Y = predict(x@obj, newdata = data.frame(KB = X))
+     plot(x@KB, x@LR, xlab = paste("kB on chr", x@chr), ylab = "log ratio",
+         main = paste("sample", x@samp, ";", x@pdLabel))
+     points(X, Y, pch = "-", cex = 2)
+ })
> treeSeg = function(es, samp, chr = "17", pdv = "geneCluster",
+     ...) {
+     LR = logRatios(es)[, samp]
+     pdl = as.character(es[[pdv]][samp])
+     KB = kb(es)
+     ob = rpart(LR ~ KB, ...)
+     new("rpSeg", obj = ob, samp = samp, LR = LR, KB = KB, chr = chr,
+         rpCall = ob$call, pdLabel = pdl)
+ }
```

Now test it out:

```
> ts1 = treeSeg(nc17, 1)
> ts1

rpart segmentation for sample 1 phenoLabel Lu
range of clone locations on chr 17
[1]     0.001 81860.266

> tsL = list()
> for (i in 1:6) tsL[[i]] = treeSeg(nc17, i)
> par(mfrow = c(3, 2))
> for (i in 1:6) plot(tsL[[i]])
> par(mfrow = c(1, 1))
```



4.3 Open problems in integrative analysis

- *Question 17.* Add infrastructure to extract estimated mean log ratio at selected chromosomal offsets for all samples in a CGHset. These extracted quantities can be regarded as new predictive features, and will need names. If possible, write the infrastructure so that queries can be couched in terms of gene symbols and extents.

- *Question 18.* Is predictability of breast cancer phenotype enhanced when use is made of genomic aberration data in addition to expression data?