Lecture 12

Classification analysis – R code

MCB 416A/516A
Statistical Bioinformatics and Genomic Analysis

Prof. Lingling An
Univ of Arizona

ALL data

- preprocessed gene expression data
- Chiaretti et al., Blood (2004)
- 12625 genes (hgu95av2 Affymetrix GeneChip)
- 128 samples (arrays)
- phenotypic data on all 128 patients, including:
  - 95 B-cell cancer
  - 33 T-cell cancer

Packages needed

```
####  install the required packages for the first time ###
source("http://www.bioconductor.org/biocLite.R")
biocLite("ALL")
biocLite("genefilter")
biocLite("hgu95av2.db")
biocLite("MLInterfaces")

install.packages("gplots")
install.packages("e1071")
```

Then …

```
#### load the packages ###
library("ALL")   ## or without quotes
library("genefilter")
library("hgu95av2.db")
library("MLInterfaces")
library("gplots")
library("e1071")
```

Read in data and other practices

- `data()` ## list all data sets available in loaded packages
- `data(package="ALL")` ## list all data sets in the "ALL" package
- `data(ALL)` ## manually load the specific data into R (R doesn't automatically load everything to save memory)
- `?ALL` ## description of the ALL dataset
- `ALL` ## data set "ALL"; It is an instance of "ExpressionSet" class
- `help(ExpressionSet)`
- `exprs(ALL)[1:3,]` ## get the first three rows of the data matrix

Data manipulation

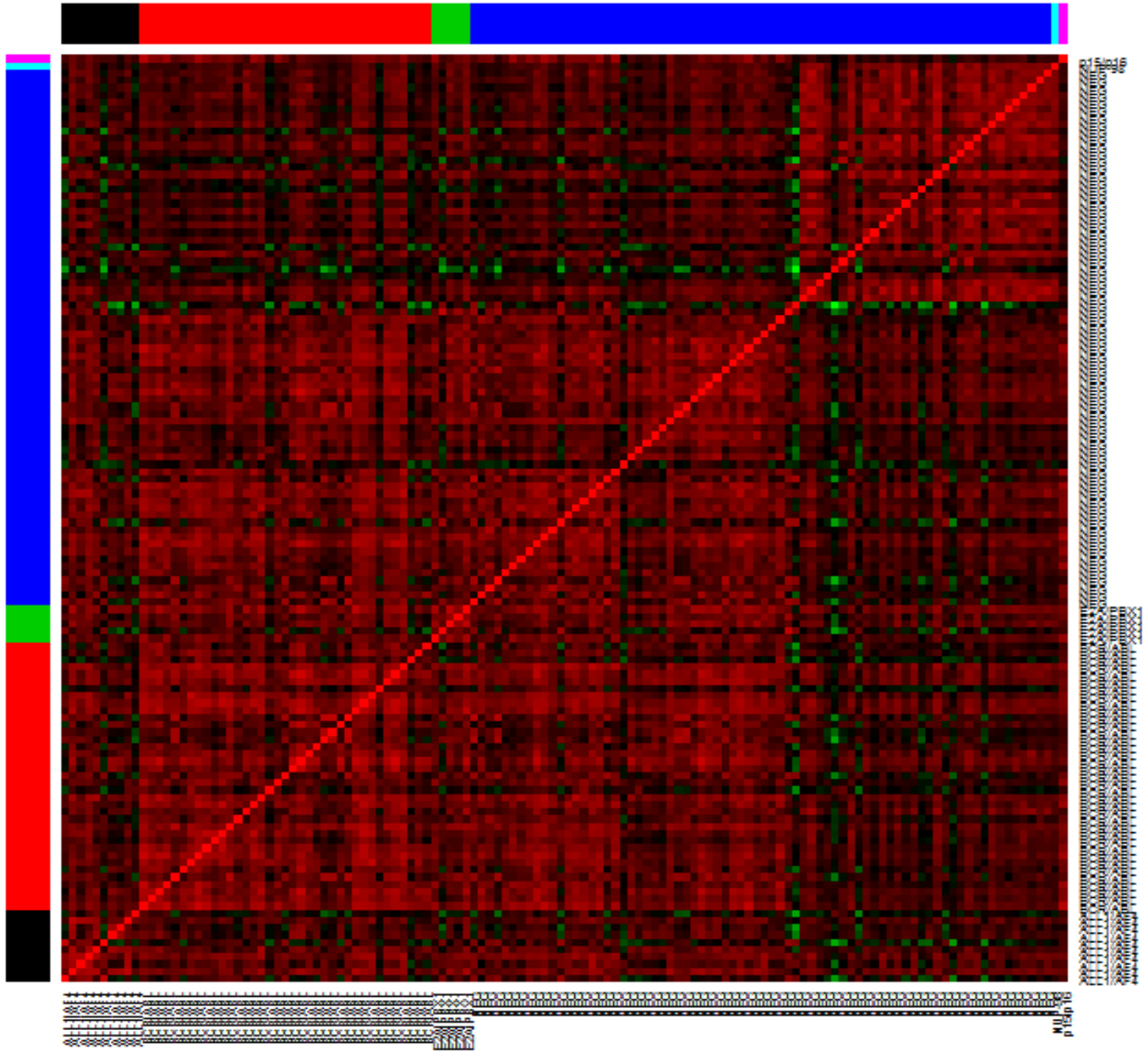
```
> pData(ALL)        ## info about the data/experiments
> names(pData(ALL)) ## column names of the info
Or     varLabels(ALL)
```

```
> ALL.1=ALL[,order(ALL$mol.bio)]   ### order samples by sample types
> ALL.1$mol.bio
```

Effect on gene selection by correlation map

plot the correlation matrix of the 128 samples using all 12625 genes

```
> library(gplots)
> heatmap( cor(exprs(ALL.1)), Rowv=NA, Colv=NA,
  scale="none", labRow=ALL.1$mol.bio, labCol= ALL.
  1$mol.bio, RowSideColors=
  as.character(as.numeric(ALL.1$mol.bio)),
  ColSideColors= as.character(as.numeric(ALL.
  1$mol.bio)), col=greenred(75))
```



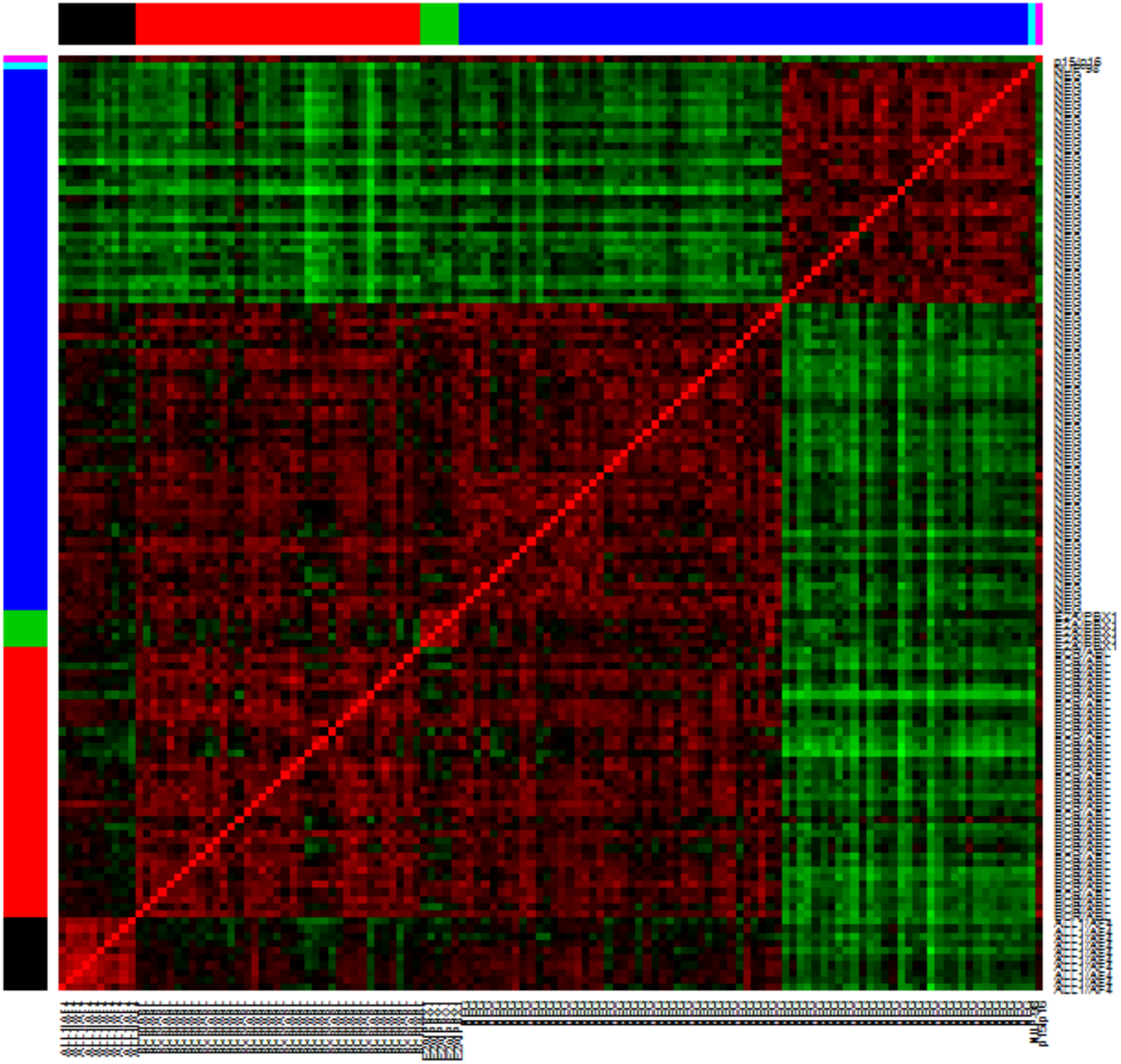
Simple gene-filtering

- #### make a simple filtering and select genes with standard deviation (i.e., sd) larger than 1 #####

```
> ALL.sd = apply(exprs(ALL.1), 1, sd)
#### 379 genes left
> ALL.new = ALL.1[ALL.sd>1, ]
> ALL.new
```

Heatmap of the selected genes

```
heatmap( cor(exprs(ALL.new)), Rowv=NA, Colv=NA,
   scale="none", labRow=ALL.new$mol.bio, labCol=
   ALL.new$mol.bio, RowSideColors=
   as.character(as.numeric(ALL.new$mol.bio)),
   ColSideColors= as.character(as.numeric(ALL.new
   $mol.bio)), col=greenred(75))
```



```
> ALL.new$BT    ## In the correlation plot, NEG has two subgroups. One is B-cell (BT=B) and the other is T-cell (BT=T).
## BT: The type and stage of the disease; B indicates B-cell ALL while a T indicates T-cell ALL
```

Exploration/illustration of why we need do gene selection is done.

Let’s go back and focus on classification analysis

Classification analysis

Preprocessing and selection of samples

1) back to the ALL data and select all B-cell lymphomas from the BT column:

```
> table(ALL$BT)
 B B1 B2 B3 B4  T T1 T2 T3 T4
 5 19 36 23 12  5  1 15 10  2
>bcell = grep("^B", as.character(ALL$BT))
> ALL_B=ALL[, bcell]   ## select 95 B-cell lymphomas samples
```

2) select BCR/ABL abnormality and negative controls from the mol.biol column in the 95 selected samples => 79 samples!

```
> table(ALL$mol.biol) ## take a look at the summary of mol.biol
ALL1/AF4  BCR/ABL E2A/PBX1      NEG   NUP-98  p15/p16
      10       37        5       74        1        1
>ALL_bcrneg = ALL_B[,ALL_B$mol.biol=="NEG"|ALL_B$mol.biol=="BCR/ABL"]
>ALL_bcrneg$mol.biol = factor(ALL_bcrneg$mol.biol) ## drop unused levels
```

Select the samples

- # structure of the new dataset

```
> head(pData(ALL_bcrneg))
> table(ALL_bcrneg$mol.biol)

BCR/ABL     NEG
     37      42
```

Gene filtering

```
>library("hgu95av2.db")
```

```
# Apply non-specific filtering
> library(genefilter)
> small = nsFilter(ALL_bcrneg, var.cutoff=0.75)$eset
> Small      ## find out how many genes are left?
> dim(small)       ## same thing
```

```
> table(small$mol.biol)

BCR/ABL      NEG
     37       42
```

#... define positive and negative cases

```
> Negs = which(small$mol.biol == "NEG")
> Bcr = which(small$mol.biol == "BCR/ABL")

> Negs
 [1]  2  4  5  6  7  8 11 12 14 19 22 24 26 28 31 35 37 38 39
[20] 43 44 45 46 49 50 51 52 54 55 56 57 58 61 62 65 66 67 68
[39] 70 74 75 77
> Bcr
 [1]  1  3  9 10 13 15 16 17 18 20 21 23 25 27 29 30 32 33 34 36 40 41
[23] 42 47 48 53 59 60 63 64 69 71 72 73 76 78 79
```

Split data into training and validation set

#... randomly sample 20 individuals from each group for training and put them together as training data

```
> set.seed(7)
> S1 = sample(Negs, 20, replace=FALSE)
> S2 = sample(Bcr, 20, replace=FALSE)
> TrainInd = c(S1, S2)
```

#... the remainder is for validation

```
> TestInd = setdiff(1:79, TrainInd)
> TestInd
 [1]  1  3  4  6 10 11 14 15 17 20 22 23 24 25 27 28 32 33 38 39 40 41 43 44
    45 46 50 51 52
[30] 53 54 58 61 64 68 71 74 75 79
```

Gene selection further

#... perform gene/feature selection on the TRAINING SET

```
> Train=small[,TrainInd]
> Traintt = rowttests(Train, "mol.biol")    # run
  two-sample t-test
> head(Traintt)    ## take a look at the top part of
  the resulting data
           statistic          dm   p.value
41654_at    0.9975424  0.22064713 0.3248114
35430_at    0.4318284  0.10770465 0.6683066
38924_s_at -0.3245558 -0.06028665 0.7472972
36023_at    1.1890007  0.19535279 0.2418165
266_s_at    0.3070361  0.12969821 0.7604922
37569_at    1.1079437  0.26620492 0.2748499
```

#... order the genes by abs. value of test statistic

```
> ordTT = order(abs(Traintt$statistic),
  decreasing=TRUE)
>
```

#... select top 50 significant genes/features for simplicity

```
> fname50 = featureNames(small[ordTT[1:50],])
```

#... create a reduced expressionSet with top 50 significant features selected on the TRAINING SET

```
> esetShort <- small[fname50,]
> dim(esetShort)
Features  Samples
      50       79
```

- So far we have selected the samples (79) with B-cell type cancer and mol.bio = BCR/AML or NEG
- And selected/filtered genes twice:

  — first ,filtering out the genes with small variation across 79 samples

  — Second, selecting the top 50 genes (via two sample t-test) from a randomly selected training data set (20 NEGs + 20 BCR/AML s)

Data set: esetShort

Now we'll use these 50 genes as marker genes and build classifiers on training data set, and calculate the classification error rate (i.e., misclassification rate) on test data set.

Classification methods

```
library(MLInterfaces)
library(help=MLInterfaces)
vignette("MLInterfaces")
?MLearn
```

Use validation set to estimate predictive accuracy

```
# ----------K nearest neighbors---------
> Knn.out = MLearn(mol.biol ~ ., data=esetShort,
  knnI(k=1, l=0), TrainInd)
> show(Knn.out)
> confuMat(Knn.out)


> bb=confuMat(Knn.out)
> Err=(bb[1,2]+bb[2,1])/sum(bb)   ## calculate
  misclassification rate
```

----Linear discriminant analysis----------

```
> Lda.out = MLearn(mol.biol ~ ., data=esetShort, ldaI, TrainInd)
> show(Lda.out)
> confuMat(Lda.out)

> bb=confuMat(Lda.out)
> Err=(bb[1,2]+bb[2,1])/sum(bb)  ## calculate misclassification rate
```

```
# ----------Support vector machine ------------
> library(e1071) ## if you haven't done this step.
> Svm.out = MLearn(mol.biol ~ ., data=esetShort, svmI, TrainInd)
> show(Svm.out)


> bb=confuMat(Svm.out)
> Err=(bb[1,2]+bb[2,1])/sum(bb)  ## calculate misclassification rate
```

```
# ----------Classification tree-----------------
> Ct.out = MLearn(mol.biol ~ ., data=esetShort,
  rpartI, TrainInd)
> show(Ct.out)
> confuMat(Ct.out)


> bb=confuMat(Ct.out)
> Err=(bb[1,2]+bb[2,1])/sum(bb)  ## calculate
misclassification rate
```

----------Logistic regression----------

it may be worthwhile using a more specialized implementation of logistic regression

```
> Lr.out = MLearn(mol.biol ~ ., data=esetShort,
  glmI.logistic(threshold=.5), TrainInd,
  family=binomial)
> show(Lr.out)
> confuMat(Lr.out)

> bb=confuMat(Lr.out)
> Err=(bb[1,2]+bb[2,1])/sum(bb)
```

Cross validation

Define function for random partition:

```
ranpart = function(K, data) {
 N = nrow(data)
 cu = as.numeric(cut(1:N, K))
 sample(cu, size = N, replace = FALSE)
 }

 ranPartition = function(K) function(data, clab,
  iternum) {
  p = ranpart(K, data)
  which(p != iternum)
 }
```

random 5-fold cross-validation

```
set.seed(1)
kk=100
error=rep(0, kk)

for (i in 1:kk){
   r1 = MLearn(mol.biol ~ ., esetShort, knnI(k = 1, l =
   0), xvalSpec("LOG",5, partitionFunc = ranPartition(5)))
   bb= confuMat(r1)
   error[i]=(bb[1,2]+bb[2,1])/sum(bb)
}
mean(error) ## get the average of the error/
   misspecification from cross-validation
boxplot(error)
```

Classification methods available in Bioconductor

"MLInterfaces" package

This package is meant to be a unifying platform for all machine learning procedures (including classification and clustering methods). Useful but use of the package easily becomes a black box!!

Separated functions:

- Linear and quadratic discriminant analysis: "lda" and "qda"
- KNN classification: "knn"
- CART: "rpart"
- Bagging and AdaBoosting: "bagging" and "logitboost"
- Random forest: "randomForest"
- Support Vector machines: "svm"
- Artifical neural network: "nnet"
- Nearest shrunken centroids: "pamr"