Statistics for bioinformatics - Tutorials
Microarray analysis - Supervised Classification

Jacques van Helden

January 12, 2012

NA http://jacques.van-helden.perso.luminy.univmed.fr/

Contents

1 Abbreviations

ALL	acute lymphoblastic leukemia
AML	acute myeloblastic leukemia
DEG	differentiall expressed genes
GEO	Gene Expression Omnibus database
KNN	K-nearest neighbours
LDA	linear discriminant analysis
QDA	quadratic discriminant analysis
PCA	principal component anlaysis

2 Prerequisites

2.1 R configuration

- Open a terminal
- Start R

We first need to define the URL of the course (`dir.base`), from which we will download some pieces of **R** code and the data sets to analyze.

```
> dir.base <- 'http://www.bigre.ulb.ac.be/Users/jvanheld/statistics_bioinformatics'
```

The following command loads a general configuration file, specifying the input directories (data, **R** utilities) and creating output directories on your computer (`dir.results`, `dir.figures`)..

```
> source(file.path(dir.base, 'R-files', 'config.R'))

[1] "Data repository http://www.bigre.ulb.ac.be/Users/jvanheld/statistics_bioinformatics/da
[1] "R scripts source http://www.bigre.ulb.ac.be/Users/jvanheld/statistics_bioinformatics/R
[1] "Results will be saved to /Users/jvanheld/course_stats_bioinfo/results"
[1] "Figures will be saved to /Users/jvanheld/course_stats_bioinfo/figures"
```

3 Goal of this tutorial

In this tutorial, we will get familiar with the basic concepts of supervised classification.

1. Distinction between unsupervised (clustering) and supervised classification
2. Training, testing and prediction
3. Cross-validation, leave-one-out
4. Over-dimensionality
5. Overfitting

4 Study case

For the practical, we will use a cohort comprized of 190 samples from patients suffering from Acute Lymphoblastic Leukemia (***ALL***) from Den Boer et al. (2009). The raw data has previously been retrieved from the Gene Expression Omnibus (***GEO***) database [1].

4.1 Data loading

We can now load the profile table and check its dimensions.

```
> ## Define the location of data directory and file containing expression profiles
> dir.denboer <- file.path(dir.base, 'data', 'gene_expression','denboer_2009')
> file.expr.table <- file.path(dir.denboer, 'GSE13425_Norm_Whole.tab')
> ## Load the expression profiles (one row per gene, one column per sample).
> ##
> ## BEWARE, the whole file makes 77Mb.
> ## It can take a few minutes to be downloaded from the Web site and loaded in R.
> expr <- read.table(file.expr.table, sep = "\t", head = T, row = 1)
> print(dim(expr))
```

[1] http://www.ncbi.nlm.nih.gov/geo/

```
[1] 22283   190
```

Once the whole data set has been loaded, the data frame "expr" should contain 22,283 rows (genes) and 190 columns (samples).

We can now load the sample descriptions ("phenoData").

```
> ## Load the table describing each sample
> ## (one row per sample, one column per description field).
> pheno <- read.table(file.path(dir.denboer,
+    'phenoData_GSE13425.tab'),
+    sep='\t', head=TRUE, row=1)
> dim(pheno)
```

```
[1] 190   4
```

We can check the content of the phenoData table by consulting its column names.

```
> names(pheno)
```

```
[1] "Sample.title"               "Sample.source.name.ch1"
[3] "Sample.characteristics.ch1" "Sample.description"
```

The column `Sample_title` indicates the cancer subtype corresponding to each sample. We can count the number of samples per subtype, and display them by decreasing group size.

```
> print(data.frame("n"=sort(table(pheno$Sample.title),decreasing=T)))
```

```
                          n
hyperdiploid             44
pre-B ALL                44
TEL-AML1                 43
T-ALL                    36
E2A-rearranged (EP)       8
BCR-ABL                   4
E2A-rearranged (E-sub)    4
MLL                       4
BCR-ABL + hyperdiploidy   1
E2A-rearranged (E)        1
TEL-AML1 + hyperdiploidy  1
```

For the sake of visualization, we will define short labels corresponding to each ALL subtype, and assign these short labels to the samples.

```
> ## Define an abbreviated name for each canceer subtype
> ## (will be used later visualization on the plots)
> group.abrev <- c(
+     'BCR-ABL + hyperdiploidy'='Bch',
+     'BCR-ABL'='Bc',
+     'E2A-rearranged (E)'='BE',
+     'E2A-rearranged (E-sub)'='BEs',
+     'E2A-rearranged (EP)'='BEp',
+     'MLL'='BM',
+     'T-ALL'='T',
+     'TEL-AML1 + hyperdiploidy'='Bth',
+     'TEL-AML1'='Bt',
+     'hyperdiploid'='Bh',
+     'pre-B ALL'='Bo'
+                   )
> sample.subtypes <- as.vector(pheno$Sample.title)
> sample.labels <- group.abrev[sample.subtypes]
> names(sample.labels) <- names(expr)
> ## Check the label for a random selection of 10 samples.
> ## Each run should give a different result
> sample(sample.labels, size=10)

GSM338668 GSM338776 GSM338710 GSM338853 GSM338839 GSM338840 GSM338669 GSM338835
      "T"      "Bh"      "Bt"      "Bo"      "Bo"      "Bo"       "T"      "Bo"
GSM338813 GSM338681
     "Bo"       "T"
```

We can also define group-specific colors and assign them to samples.

```
> ## Define group-specific colors
> group.colors <- c(
+                   'BCR-ABL + hyperdiploidy'='cyan',
+                   'BCR-ABL'='black',
+                   'E2A-rearranged (E)'='darkgray',
+                   'E2A-rearranged (E-sub)'='green',
+                   'E2A-rearranged (EP)'='orange',
+                   'MLL'='#444400',
+                   'T-ALL'='violet',
+                   'TEL-AML1 + hyperdiploidy'='#000066',
+                   'TEL-AML1'='darkgreen',
+                   'hyperdiploid'='red',
+                   'pre-B ALL'='blue'
```

```
+                   )
> ## Assign group-specific colors to patients
> sample.colors <- group.colors[as.vector(pheno$Sample.title)]
> names(sample.colors) <- names(expr)
> sample(sample.colors,size=20)
```

```
  GSM338844   GSM338716   GSM338804   GSM338735   GSM338731   GSM338790
     "blue" "darkgreen"     "black" "darkgreen" "darkgreen"  "darkgray"
  GSM338839   GSM338801   GSM338699   GSM338783   GSM338753   GSM338832
     "blue"     "green"    "violet"       "red"       "red"      "blue"
  GSM338695   GSM338672   GSM338726   GSM338693   GSM338709   GSM338791
   "violet"    "violet" "darkgreen"    "violet" "darkgreen"    "orange"
  GSM338782   GSM338769
      "red"       "red"
```

5 Reducing the dimensionality of the data - feature selection

An omnipresent problem with microarray data is the large dimensionality of the data space. The dataset we are analyzing contains 190 samples (the ***subjects***), each characterized by 22,283 gene expression values (the ***variables***).

The number of variables (also called ***features***, the genes in our case) thus exceeds by far the number of objects (samples in this case). This sitation is qualified of ***over-dimensionality***, and poses serious problems for classification. In unsupervised classification (clustering), the relationships between objects will be affected, since a vast majority of the features are likely to be uninformative, but will however contribute to the computed (dis)similarity metrics (whichever metrics is used).

5.1 A pair of genes

Figure 2 compares the expression levels of two genes selected at random (the 236th and the 1213th rows of the profile table). Each dot corresponds to one sample. Since the genes were selected at random, we don't expect them to be particularly good at discriminating the different subtypes of ALL. However, we can already see some emerging patterns: the T-ALL (labelled "T") show a trends to strongly express the 236th gene, and, to a lesser extent, several hyperdiploid B-cells (label "Bh") show a high expression of the 1213th gene. Nevertheless, it would be impossible to draw a boundary that would separate two subtypes in this plot.

5.2 Variance filter

In order to reduce the dimensionality of the data set, we will sort the genes according to their variance, and retain a defined number of the top-raking genes. Note that this ranking

```
> ## Plot the expression profiles of two arbitrarily selected genes
> g1 <- 236
> g2 <- 1213
> x <- as.vector(as.matrix(expr[g1,]))
> y <- as.vector(as.matrix(expr[g2,]))
> plot(x,y,
+      col=sample.colors,
+      type='n',
+      panel.first=grid(col='black'),
+        main=paste('PCA; Den Boer (2009); ',
+          ncol(expr), 'samples *', nrow(expr), 'genes', sep=' '),
+        xlab=paste('gene', g1), ylab=paste('gene', g2))
> text(x, y,labels=sample.labels,col=sample.colors,pch=0.5)
> legend('topleft',col=group.colors,
+        legend=names(group.colors),pch=1,cex=0.7,bg='white',bty='o')
```

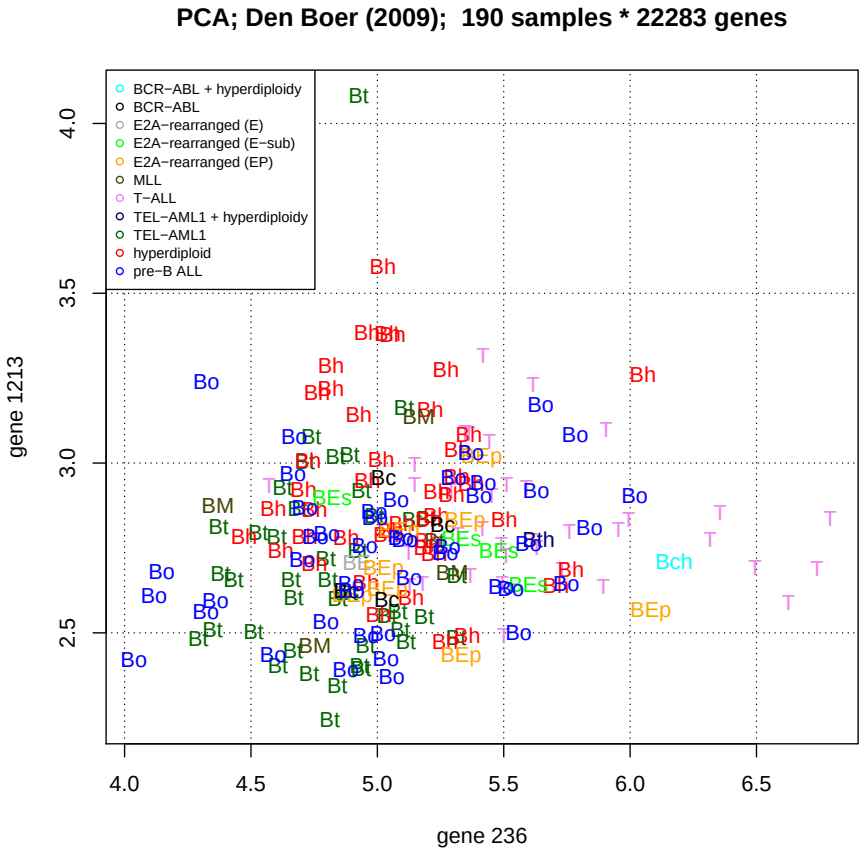

Figure 1: Comparison between expression values of two arbitrarily selected genes: 236th (abscissa) versus 1213th (ordinate) rows of the expression profiles by Den Boer (2009). Each dot represents a sample, labeled by cancer subtype.

is a very rudimentary way to select a subset of genes. Indeed, nothing guarantees us that the genes showing the largest inter-individual fluctuations of expression have anything related to cancer type. In further sections, we will present alternative methods of variable ordering, which will explicitly take into account the inter-group variance.

The **R** function `apply()` can be used to apply any function to each row (or alternatively each column) of a data table. We set the second argument (margin) to 1, thereby indicating that the function (third argument) must be applied to each row of the input table (first argument).

```
> ## Compute sample-wise variance
> var.per.sample <- apply(expr, 2, var)
> head(var.per.sample)

GSM338666 GSM338667 GSM338668 GSM338669 GSM338670 GSM338671
 2.636681  2.636490  2.749702  2.436275  2.634296  2.538188

> ## Compute gene-wise variance
> var.per.gene <- apply(expr, 1, var)
> head(var.per.gene, n=20)

  DDR1|1007_s_at     RFC2|1053_at     HSPA6|117_at      PAX8|121_at
     0.188580722      0.060546361      0.456587060      0.027817178
GUCA1A|1255_g_at     UBA7|1294_at     THRA|1316_at   PTPN21|1320_at
     0.007829279      0.190982366      0.017294454      0.010907010
  CCL5|1405_i_at   CYP2E1|1431_at    EPHB3|1438_at    ESRRA|1487_at
     0.111399737      0.014272542      0.125543355      0.022937620
CYP2A6|1494_f_at   GAS6|1598_g_at  MMP14|160020_at    TRADD|1729_at
     0.024901603      0.049478479      0.036649349      0.055674697
    FNTB|1773_at      PLD1|177_at   PMS2L11|179_at      BAD|1861_at
     0.030893133      0.024299291      0.038863238      0.142774594
```

If we sort genes by decreasing variance, we can see that there is a strong difference between the top and the bottom of the list.

```
> ## Sort genes per decreasing variance
> genes.by.decr.var <- sort(var.per.gene,decreasing=TRUE)
> ## Print the 5 genes with highest variance
> print(as.data.frame(genes.by.decr.var[1:5]))

                   genes.by.decr.var[1:5]
CD9|201005_at                    6.043017
IL23A|211796_s_at                5.928425
```

```
CD3D|213539_at                  5.696326
S100A8|202917_s_at              5.508163
KLF4|221841_s_at                5.245057
```

```
> ## Print the 5 genes with lowest variance
> n.genes <- length(genes.by.decr.var)
> print(as.data.frame(genes.by.decr.var[(n.genes-4):n.genes]))
```

```
                    genes.by.decr.var[(n.genes - 4):n.genes]
LOC730272|216686_at                                0.007022880
SCLY|59705_at                                      0.006576734
NA|AFFX-DapX-5_at                                  0.006477929
HSD17B6|37512_at                                   0.006405752
NA|AFFX-LysX-5_at                                  0.005089941
```

We can then select an arbitrary number of top-raking genes in the list.

```
> ## Select the 30 top-ranking genes in the list sorted by variance
> top.nb <- 20 ## This number can be changed for testing
> genes.selected.by.var <- names(genes.by.decr.var[1:top.nb])
> ## Check the names of the first selected genes
> head(genes.selected.by.var)
```

```
[1] "CD9|201005_at"      "IL23A|211796_s_at"  "CD3D|213539_at"
[4] "S100A8|202917_s_at" "KLF4|221841_s_at"   "HLA-DRA|208894_at"
```

```
>
```

In the next sections, we will compare methods for selecting genes on the basis of different criteria (variance, T-test, ANOVA test, step-forward procedure in Linear Discriminant Analysis). For this purpose, we will create a table indicating the values and the rank of each gene (rows of the table) according to eachs election criterion (columns).

```
> ## Create a data frame to store gene values and ranks
> ## for different selection criteria.
> gene.ranks <- data.frame(var=var.per.gene)
> head(gene.ranks)
```

```
                         var
DDR1|1007_s_at   0.188580722
RFC2|1053_at     0.060546361
HSPA6|117_at     0.456587060
PAX8|121_at      0.027817178
GUCA1A|1255_g_at 0.007829279
UBA7|1294_at     0.190982366
```

```
> ## Compute the rank of each gene in the variance-sorted list
> n.genes <- nrow(gene.ranks)
> ## Beware, we rank according to minus variance, because
> ## we want to associated the lowest ranks to the highest variances
> gene.ranks$var.rank <- rank(-gene.ranks$var, ties.method='random')
> head(gene.ranks)

                         var var.rank
DDR1|1007_s_at   0.188580722     5369
RFC2|1053_at     0.060546361    10659
HSPA6|117_at     0.456587060     1971
PAX8|121_at      0.027817178    17936
GUCA1A|1255_g_at 0.007829279    22265
UBA7|1294_at     0.190982366     5306

> ## Store the gene rank table in a text file (tab-separated columns)
> write.table(gene.ranks, file=file.path(dir.results, 'DenBoer_gene_ranks.tab'),
+             sep='\t', quote=F)
> ## Check the rank of the 5 genes with highest and lowest variance, resp.
> gene.ranks[names(genes.by.decr.var[1:5]),]

                        var var.rank
CD9|201005_at      6.043017        1
IL23A|211796_s_at  5.928425        2
CD3D|213539_at     5.696326        3
S100A8|202917_s_at 5.508163        4
KLF4|221841_s_at   5.245057        5

> gene.ranks[names(genes.by.decr.var[(n.genes-4):n.genes]),]

                            var var.rank
LOC730272|216686_at 0.007022880    22279
SCLY|59705_at       0.006576734    22280
NA|AFFX-DapX-5_at   0.006477929    22281
HSD17B6|37512_at    0.006405752    22282
NA|AFFX-LysX-5_at   0.005089941    22283
```

We have no reason to think that genes having a high variance will be specially good at discriminating ALL subtypes. Indeed, a high variance might *a priori* either reflect cell-type specificities, or differences between samples that result from any other effect (individual patient genomes, transcriptome, condition, ...). For the sake of curiosity, let us plot samples on a XY plot where the abcsissa represents the top-raking, and the ordinate the second top-ranking gene in the variance-ordered list.

```
> ## Plot the expression profiles of the two genes with highest variance
> (g1 <- names(genes.by.decr.var[1]))

[1] "CD9|201005_at"

> (g2 <- names(genes.by.decr.var[2]))

[1] "IL23A|211796_s_at"

> x <- as.vector(as.matrix(expr[g1,]))
> y <- as.vector(as.matrix(expr[g2,]))
> plot(x,y,
+      col=sample.colors,
+      type='n',
+      panel.first=grid(col='black'),
+        main=paste('PCA; Den Boer (2009); ',
+          ncol(expr), 'samples *', nrow(expr), 'genes', sep=' '),
+        xlab=paste('gene', g1), ylab=paste('gene', g2))
> text(x, y,labels=sample.labels,col=sample.colors,pch=0.5)
> legend('topright',col=group.colors,
+        legend=names(group.colors),pch=1,cex=0.7,bg='white',bty='o')
```

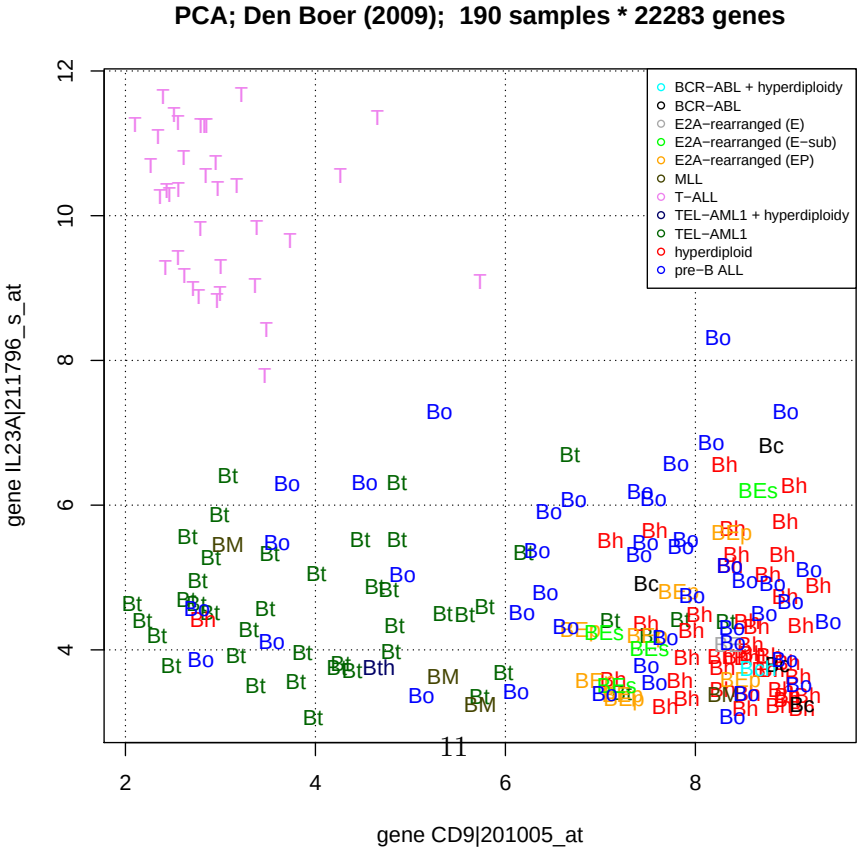

Figure 2: Comparison between expression values of the two genes with te highest variance.

5.3 Selecting differentially expressed genes (DEG) as predictor variables

5.3.1 Variable ordering by Welch t-test (2-groups test)

The T-test tests the hypothesis of equality between the means of two populations. We can use the Welch version of the t-test (which assumes that groups can have different variances) in order to select genes differentially expressed between two goups of samples.

$$H_0 : m_1 = m_2 \tag{1}$$

We thus need to define two groups of samples (multi-group comparisons will be reated by ANOVA in Section 5.3.2). For the dataset on ALL? we have several subtypes of cancer. we will perform pairwise comparisons of mean for a selection of subtypes. In each case, a given subtype (e.g. T-cells) will be compared to all other subtypes pooled together.

We will successvely run Welch's t-test for the main subtypes ("Bo", "Bh", "Bt", "T").

$$H_0^{Bo} : m_{Bo} = m_{others} \tag{2}$$
$$H_0^{Bh} : m_{Bh} = m_{others} \tag{3}$$
$$H_0^{Bt} : m_{Bt} = m_{others} \tag{4}$$
$$H_0^{T} : m_{T} = m_{others} \tag{5}$$

In each case, apply the test in to each gene, using the function t.test.mult() defined in the **R** utilities of this course.

```
> ## Load a utility to apply Welch test on each row of a table
> source(file.path(dir.util, "util_student_test_multi.R"))
> ## Define a vector indicating whether each sample
> ## belongs to the subtype of interest (e.g. "Bo") or not.
> current.group <- "Bo"
> one.group.vs.others<- sample.labels
> one.group.vs.others[sample.labels != current.group] <- "other"
> print(table(one.group.vs.others))

one.group.vs.others
   Bo other
   44   146

> ## Test the mean equality between Bo subtype and all other subtypes
> welch.one.group.vs.others <- t.test.multi(expr, one.group.vs.others)

[1] "Thu Jan 12 13:42:49 2012 - Multiple t-test started"
[1] "Thu Jan 12 13:42:52 2012 - Multiple t-test done"
```

```
> names(welch.one.group.vs.others)

 [1] "mean.other"    "mean.Bo"       "means.diff"    "var.est.other"
 [5] "var.est.Bo"    "sd.est.other"  "sd.est.Bo"     "st.err.diff"
 [9] "t.obs"         "df.welch"      "P.value"       "E.value"
[13] "sig"

> ## Update the gene rank table
> test.name <- paste(current.group, '.vs.others.sig', sep='')
> gene.ranks[,test.name] <- welch.one.group.vs.others$sig
> ## Do the same Wekch test for the 3 other majority groups
> for (current.group in c("Bh", "Bt", "T")) {
+   one.group.vs.others<- sample.labels
+   one.group.vs.others[sample.labels != current.group] <- "other"
+
+   ## Test the mean equality between Bo subtype and all other subtypes
+   welch.one.group.vs.others <- t.test.multi(expr, one.group.vs.others)
+
+   ## Update the gene rank table
+   test.name <- paste(current.group, '.vs.others.sig', sep='')
+   gene.ranks[,test.name] <- welch.one.group.vs.others$sig
+ }

[1] "Thu Jan 12 13:42:52 2012 - Multiple t-test started"
[1] "Thu Jan 12 13:42:56 2012 - Multiple t-test done"
[1] "Thu Jan 12 13:42:56 2012 - Multiple t-test started"
[1] "Thu Jan 12 13:42:59 2012 - Multiple t-test done"
[1] "Thu Jan 12 13:42:59 2012 - Multiple t-test started"
[1] "Thu Jan 12 13:43:03 2012 - Multiple t-test done"

> head(gene.ranks)

                        var var.rank Bo.vs.others.sig Bh.vs.others.sig
DDR1|1007_s_at   0.188580722     5369       -3.0644627        -2.567224
RFC2|1053_at     0.060546361    10659       -4.0038640        -3.762014
HSPA6|117_at     0.456587060     1971       -3.7553739        -1.535036
PAX8|121_at      0.027817178    17936       -3.9475569        -4.333966
GUCA1A|1255_g_at 0.007829279    22265       -3.9972656        -3.767662
UBA7|1294_at     0.190982366     5306        0.2561573         1.406475
                 Bt.vs.others.sig T.vs.others.sig
DDR1|1007_s_at         -4.1636587       6.5314387
RFC2|1053_at           -0.9093112       0.3509366
```

```
HSPA6|117_at          -3.7845427      -4.1752127
PAX8|121_at           -4.1140287      -2.9637682
GUCA1A|1255_g_at      -4.1613542      -4.2709208
UBA7|1294_at          -2.6271392      -2.9565339
```

```
> ## Plot variance against significance of the Welch test
> plot(gene.ranks[,c("var", "Bo.vs.others.sig", "Bh.vs.others.sig", "Bt.vs.others.sig", "T.
```

5.3.2 ANOVA based variable ordering (multiple subtypes)

TO BE WRITTEN

6 Principal Component Analysis

Before starting the proper process of supervised classification, we can apply a method called Principal Component Analysis (***PCA***) to evaluate the repartition of the information between the mutiple variables, and to inspect the "intrinsic" structure of the data, i.e. the structure inherent to the numbers in the data table, irrespective of the labels (cancer subtypes) attached to the various samples.

6.1 Purpose of PCA

We can do the exercise of extending this 2-dimensional plot to a 3-dimensional plot, where the third dimension represents the expression level of a third gene. With an effort of imagination, we can mentally extend this 3D plot to a 4-dimensional plot, where each dimension would represent a different gene. It is likely that the groups of genes will progressively become more separated as the number of dimension increases. However, for the sake of graphical representation, it is difficult to work with more than 2 (or at most 3) dimensions.

The purpose of ***Principal Component Analysis PCA*** is to capture the largest part of the variance of a data set in a minimal number of dimensions.

TO BE WRITTEN

6.2 Applying PCA transformation with stats::prcomp()

The **R** method `stats::prcomp()` performs a PCA transformation of an input table. We can feed it with the expression table, but we need to transpose it first (using the **R** function `t()`), in order to provide our objects of interest (the samples) as rows, and the variables (genes) as columns.

```
> ## load the stats library to use the princomp() and prcomp() function
> library(stats)
> ## Perform the PCA transformation
```

```
> expr.prcomp <- prcomp(t(expr),cor=TRUE)
> ## Analyze the content of the prcomp result:
> ## the result of the method prcomp() is an object
> ## belonging to the class ``prcomp''
> class(expr.prcomp)

[1] "prcomp"

> ## Get the field names of the promp objects
> names(expr.prcomp)

[1] "sdev"     "rotation" "center"   "scale"    "x"

> ## Get the attributes of the promp objects
> attributes(expr.prcomp)

$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

6.3 Repartition of the standard deviation along the components

A first information is the repartition of the variance (or its squared root, the standard deviation) between the components, which can be displayed on a plot.

The standard deviation barplot (Figure 3) highlights that the first component captures more or less twice as much standard deviation of the whole dataset as the second one. We can measure the relative importance of the standard deviations.

```
> ## Get the standard deviation and variance per principal component
> sd.per.pc <- expr.prcomp$sdev
> var.per.pc <- sd.per.pc^2
> ## Display the percentage of total variance explained by each
> sd.per.pc.percent <- sd.per.pc/sum(sd.per.pc)
> var.per.pc.percent <- var.per.pc/sum(var.per.pc)
```

6.4 Analysis of the first versus second component

We can generate a ***biplot*** (Figure 5), where each sample appears as a dot, and the X and Y axes respectively represent the first and second components of the PCA-transformed data. The R function stats::biplot() automatically generates the biplot, but the result is

```
> plot(expr.prcomp, main='Den Boer (2009), Variance  per component', xlab='Component')
```

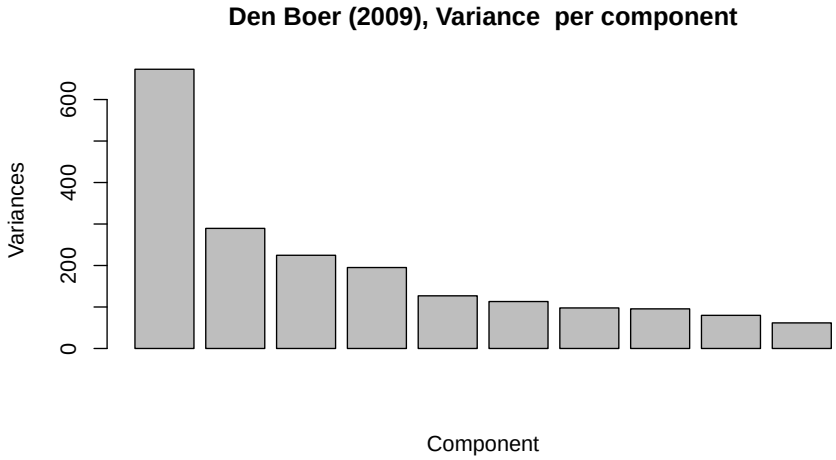

Figure 3: Parts of the variance respectively captured by the 10 first components of the PCA-transformed data from Den Boer (2009).

```
> barplot(var.per.pc.percent[1:10], main='Den Boer (2009), Percent of variance  per compone
```

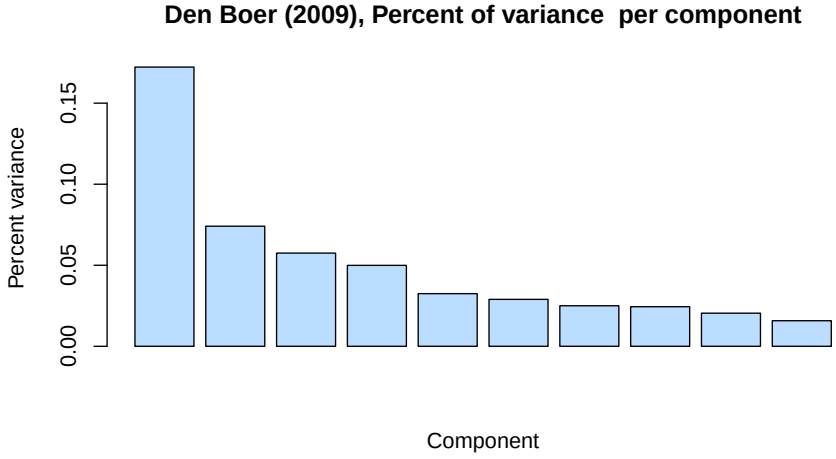

Figure 4: Percent of the variance of the whole data space captured by the 10 first components of the PCA-transformed expresion values (data from Den Boer, 2009).

```
> biplot(expr.prcomp,var.axes=FALSE,
+        panel.first=grid(col='black'),
+        main=paste('PCA; Den Boer (2009); ',
+          ncol(expr), 'samples *', nrow(expr), 'genes', sep=' '),
+        xlab='First component', ylab='Second component')
```

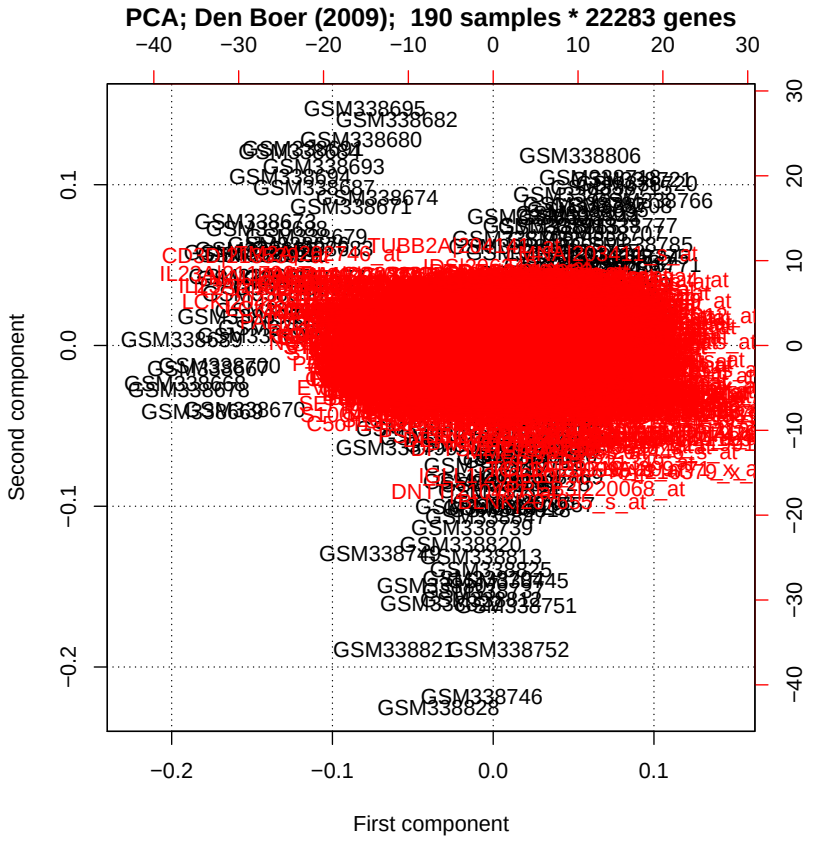

Figure 5: Biplot of the PCA-transformed data from Den Boer (2009).

somewhat confusing, because the biplot displays both the units (samples) and a projection of the variables (genes) on the two first components.

We can generate a custom plot (Figure 6), with labels and colors indicating the subtype of ALL.

The two first components of the PCA-transformed data perfectly separate the T-ALL samples from all the other subtypes. All T-ALL cells appear in one elongated cloud on the left side of the plot. The other cloud seems to contain some organization as well: subtypes are partly intermingled but there are obvious groupings.

6.5 Analysis of the second versus third component

Figure 7 shows that the second and third components capture information related to the cancer subtypes: the third component separates quite well TEL-AML1 (top) from hyperdiploid (bottom) samples, whereas the pre-B have intermediate values. The separation is however less obvious than the T-ALL versus all other subtypes that we observed in the two first components (Figure 6).

6.6 Exercise

- Generate plots of a few additional components (PC4, PC5, ...) and try to evaluate if they further separate subtypes of cancers.

6.7 Discussion of the PCA results

In the dataset from Golub et al. (1999), we saw that the three groups of samples (AML, T-ALL and B-ALL) are almost perfectly separated by a simple display of the two first components of the PCA-transformed data. The dataset from Den Boer et al. (2009) is less obvious to interpret, due to the nature of the data itself: firstly, all the samples come from the same cell type (lymphoblasts), whereas the main separation of Golub was between myeloblasts and lymphoblasts. Secondly, Den Boer and co-workers define a wider variety of subtypes. However, we see that a simple PCA transformation already reveals that the raw expression data is well structured. It is quite obvious that we will have no difficulty to find a signature discriminating T-ALL from other ALL subtypes, since T-ALL are already separated on the plane of the two first components. We would however like to further refine the diagnostics, by training a classifier to discriminate between the other subtypes as well.

Note that until now we did not apply any taining: PCA transformation is a "blind" (unsupervised) approach, performing a rotation in the variable space without using any information about pre-defined classes. Since we dispose of such information for the 190 samples of the training set, we can use a family of other approaches, generically called ***supervised classification***, that will rely on class membership of the training samples in order to train a classifier, which will further be used to assign new samples to the same classes.

```
> ## Plot components PC1 and PC2
> plot(expr.prcomp$x[,1:2],
+      col=sample.colors,
+      type='n',
+      panel.first=grid(col='black'),
+        main=paste('PCA; Den Boer (2009); ',
+          ncol(expr), 'samples *', nrow(expr), 'genes', sep=' '),
+        xlab='PC1', ylab='PC2')
> text(expr.prcomp$x[,1:2],labels=sample.labels,col=sample.colors,pch=0.5)
> legend('bottomleft',col=group.colors,
+        legend=names(group.colors),pch=1,cex=0.7,bg='white',bty='o')
```

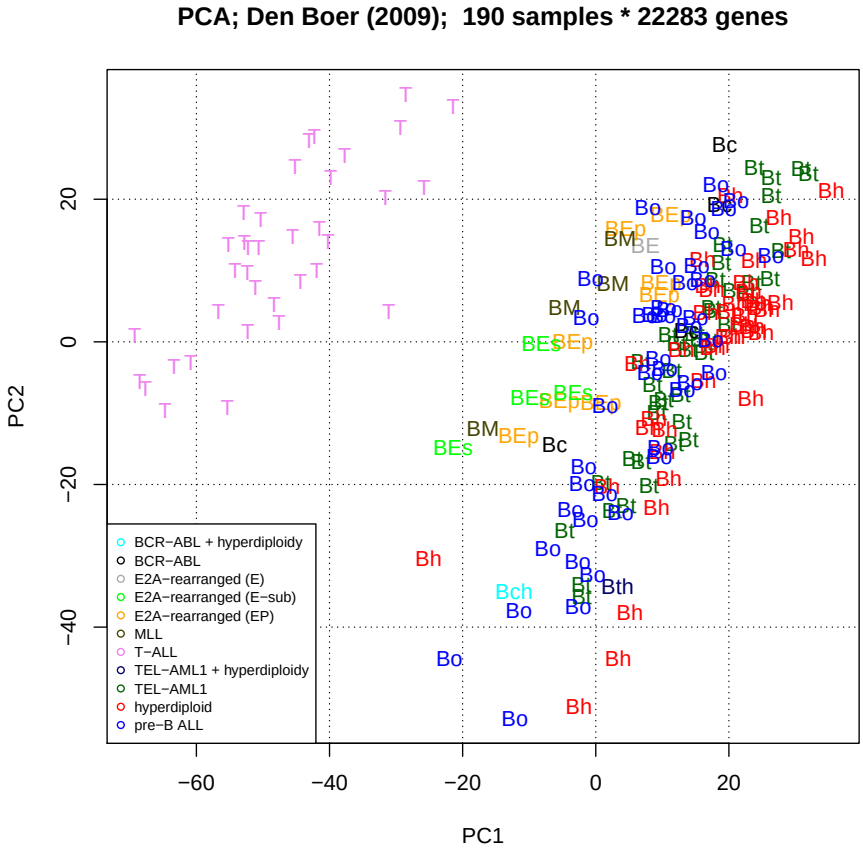

Figure 6: Two first components of the PCA-transformed data from Den Boer (2009). Colors denote ALL subtypes.

```
> ## Plot components PC2 and PC3
> plot(expr.prcomp$x[,2:3],
+      col=sample.colors,
+      type='n',
+      panel.first=grid(col='black'),
+        main=paste('PCA; Den Boer (2009); ',
+          ncol(expr), 'samples *', nrow(expr), 'genes', sep=' '),
+        xlab='PC2', ylab='PC3')
> text(expr.prcomp$x[,2:3],labels=sample.labels,col=sample.colors,pch=0.5)
> legend('bottomleft',col=group.colors,
+        legend=names(group.colors),pch=1,cex=0.7,bg='white',bty='o')
```

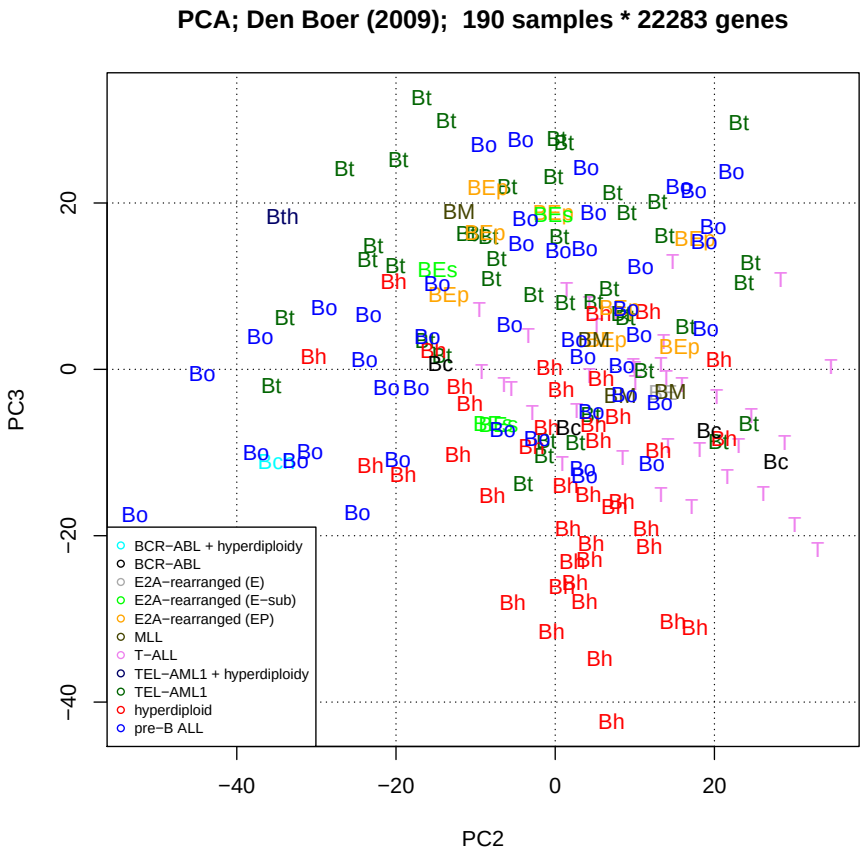

Figure 7: Second and third components of the PCA-transformed data from Den Boer (2009). Colors denote ALL subtypes.

6.8 Selecting principal components as predictor variables

TO BE WRITTEN

7 Linear Discriminant Analysis

TO BE WRITTEN

7.1 Training the classifier

As a first trial, we will train a LDA classifier using as variables the 20 top-raking genes in the list of genes sorted by variance (Section 5.2).

```
> ## Create a list of gene names sorted by decreasing variance
> sorted.names <- rownames(gene.ranks)[order(gene.ranks$var, decreasing=TRUE)]
> ## Select the 20 top-ranking genes sorted by decreasing variance
> top.variables <- 20
> selected.genes <- sorted.names[1:top.variables]
> ## Load the library containing the linear discriminant analysis function
> library(MASS)
> ## Train the classifier
> lda.classifier <- lda(t(expr[selected.genes,]),sample.labels,CV=FALSE)
>
```

NA
TO BE WRITTEN

```
>
```

7.2 Evaluating the hit rate by Leave-One-Out (LOO) cross-validation

```
> ## Use the MASS:lda() function with the cross-validation option
> lda.loo <- lda(t(expr[selected.genes,]),sample.labels,CV=TRUE)
> ## Collect the LOO prediction result in a vector
> loo.predicted.class <- as.vector(lda.loo$class)
> ## Replace <NA> valeus by a string "NA"
> ## because the table() function does not like <NA>
> ## loo.predicted.class[is.na(loo.predicted.class)] <- "NA"
> ## Well, if I do this it is not ideal either, because I have one
> ## new class (NA) in the predicted labelsand the diagonal is perturbed.
>
> ## Build a contingency table of known versus predicted class
```

```
> lda.loo.xtab <- table(sample.labels, loo.predicted.class)
> print(lda.loo.xtab)

             loo.predicted.class
sample.labels Bc BE BEp BEs Bh BM Bo Bt  T
          Bc   0  0   0   0  0  0  4  0  0
          Bch  0  0   0   0  0  0  0  0  0
          BE   0  0   0   0  0  0  0  0  0
          BEp  0  1   6   0  0  0  1  0  0
          BEs  0  0   0   3  0  0  1  0  0
          Bh   0  0   0   1 40  0  3  0  0
          BM   0  0   0   0  0  3  1  0  0
          Bo   3  0   2   2  7  1 25  4  0
          Bt   0  0   2   1  0  0  1 39  0
          Bth  0  0   0   0  0  0  0  0  0
          T    0  0   0   0  0  0  0  0 36

> ## (I should find some nice coloring scheme)
> ## Display the contingency table as a heat map
> image(lda.loo.xtab)
> library(lattice)
> levelplot(lda.loo.xtab)
> ## Compute the hit rate
> hits.per.group <- diag(lda.loo.xtab)
> total <- sum(lda.loo.xtab)
> total.hits <- sum(hits.per.group)
> print(hit.rate <- total.hits/total)

[1] 0.02673797
```

7.3 Testing the random expectation with permutation tests

7.3.1 Exercises

1. Use the **R** function `sample()` to randomly permute the training labels (store the result in a vector called "sample.labels.perm"), and test the hit rate of a classifier trained with these randomized labels.

2. Use the **R** function `sample()` to randomly permute the values of the input table (store the result in a data frame called "expr.perm"), and test the hit rate of a classifier trained with these randomized data.

7.3.2 Solutions

```
> ## Permute the training labels
> sample.labels.perm <- as.vector(sample(sample.labels))
> ## Compare original training groups and permuted labels.
> table(sample.labels, sample.labels.perm)
```

sample.labels	sample.labels.perm Bc	Bch	BE	BEp	BEs	Bh	BM	Bo	Bt	Bth	T
Bc	0	0	0	0	0	1	1	0	1	0	1
Bch	1	0	0	0	0	0	0	0	0	0	0
BE	0	0	0	0	0	0	0	0	0	0	1
BEp	0	0	0	0	0	2	0	2	3	0	1
BEs	0	0	0	0	1	2	0	1	0	0	0
Bh	1	0	0	2	0	8	0	11	12	0	10
BM	0	0	0	0	0	1	0	2	0	0	1
Bo	2	0	0	3	1	9	0	9	12	0	8
Bt	0	1	0	2	1	13	3	10	6	0	7
Bth	0	0	0	0	0	0	0	0	0	1	0
T	0	0	1	1	1	8	0	9	9	0	7

```
> ## Run LDA in cross-validation (LOO) mode with the permuted labels
> lda.loo.labels.perm <- lda(t(expr[selected.genes,]),sample.labels.perm,CV=TRUE)
> ## Build a contingency table of known versus predicted class
> loo.predicted.class.labels.perm <- as.vector(lda.loo.labels.perm$class)
> lda.loo.labels.perm.xtab <- table(sample.labels.perm, loo.predicted.class.labels.perm)
> print(lda.loo.labels.perm.xtab)
```

sample.labels.perm	loo.predicted.class.labels.perm Bc	Bch	BEp	BEs	Bh	BM	Bo	Bt	T
Bc	0	0	0	0	1	0	2	1	0
Bch	0	0	0	0	0	0	0	0	0
BE	0	0	0	0	0	0	0	0	0
BEp	0	0	1	0	1	0	4	1	1
BEs	0	0	0	0	2	0	0	1	1
Bh	0	1	0	1	19	0	6	9	8
BM	0	0	0	0	0	0	1	2	1
Bo	1	0	0	2	10	1	13	12	5
Bt	1	0	2	2	8	2	9	15	4
Bth	0	0	0	0	0	0	0	0	0
T	0	0	0	2	13	3	8	8	2

```
> ## (I should find some nice coloring scheme)
> ## Display the contingency table as a heat map
> image(lda.loo.labels.perm.xtab)
> levelplot(lda.loo.labels.perm.xtab)
> ## Compute the hit rate
> hits.per.group <- diag(lda.loo.xtab)
> total <- sum(lda.loo.xtab)
> total.hits <- sum(hits.per.group)
> print(hit.rate <- total.hits/total)

[1] 0.02673797

>
```

7.4 Impact of the number of training variables

7.4.1 Exercise: variable selection

1. Test the impact of the number of variables used for training the classifier. Draw a plot representing the hit rate (Y axis) as a function of the number of top genes used for the LDA classification, using different selection criteria:
   (a) Gene-wise variance
   (b) P-value of a Welch test (T-cells against all other types)
   (c) ANOVA

7.5 Single variable-wise ordering (by individual hit rate)

TO BE WRITTEN

7.6 Feed-forward variable selection

TO BE WRITTEN

7.7 Evaluation the classifier with an independent testing set

TO BE WRITTEN

8 Other classification approaches

8.1 K-nearest-neighbours (KNN) classifier

TO BE WRITTEN

8.2 Support vector machines

TO BE WRITTEN

References

M. L. Den Boer, M. van Slegtenhorst, R. X. De Menezes, M. H. Cheok, J. G. Buijs-Gladdines, S. T. Peters, L. J. Van Zutven, H. B. Beverloo, P. J. Van der Spek, G. Escherich, M. A. Horstmann, G. E. Janka-Schaub, W. A. Kamps, W. E. Evans, and R. Pieters. A subtype of childhood acute lymphoblastic leukaemia with poor treatment outcome: a genome-wide classification study. *Lancet Oncol*, 10(2):125–34, 2009.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–7, 1999.