Course in Practical Analysis of Microarray Data

Introduction to R

Computational Exercises

September 2002 Wolfgang Huber

1.) Installing R. Check whether R is installed on your computer. If not, download it from cran.r-project.org and install it.

2.) Reading data files. In the folder `data/alizadeh`, you find a file `lc7b048rex.DAT`.

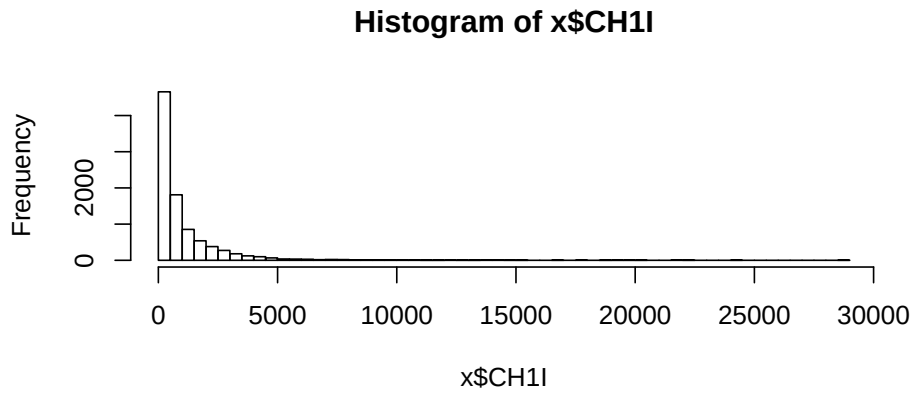
a. Open it in a text editor.
b. Read it into a data frame (use the function `read.delim`)
c. Look at the contents of the table (use the functions `dim`, `colnames`, and subsetting)

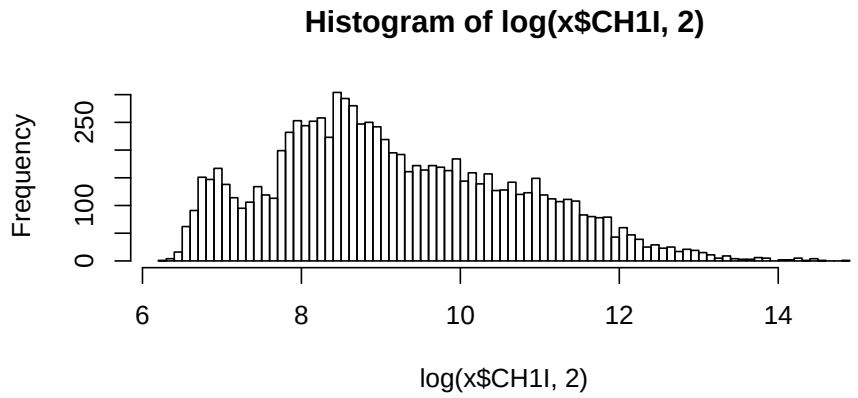
```
R> x = read.delim("../data/alizadeh/lc7b048rex.DAT")
R> dim(x)

[1] 9216   34

R> colnames(x)

 [1] "HEADER"   "SPOT"     "GRID"     "TOP"      "LEFT"     "BOT"
 [7] "RIGHT"    "ROW"      "COL"      "CH1I"     "CH1B"     "CH1AB"
[13] "CH2I"     "CH2B"     "CH2AB"    "SPIX"     "BGPIX"    "EDGE"
[19] "RAT2"     "MRAT"     "REGR"     "CORR"     "LFRAT"    "CH1GTB1"
[25] "CH2GTB1"  "CH1GTB2"  "CH2GTB2"  "CH1EDGEA" "CH2EDGEA" "FLAG"
[31] "CH1KSD"   "CH1KSP"   "CH2KSD"   "CH2KSP"

R> x[1:12, ]
```

3.) Simple plots.

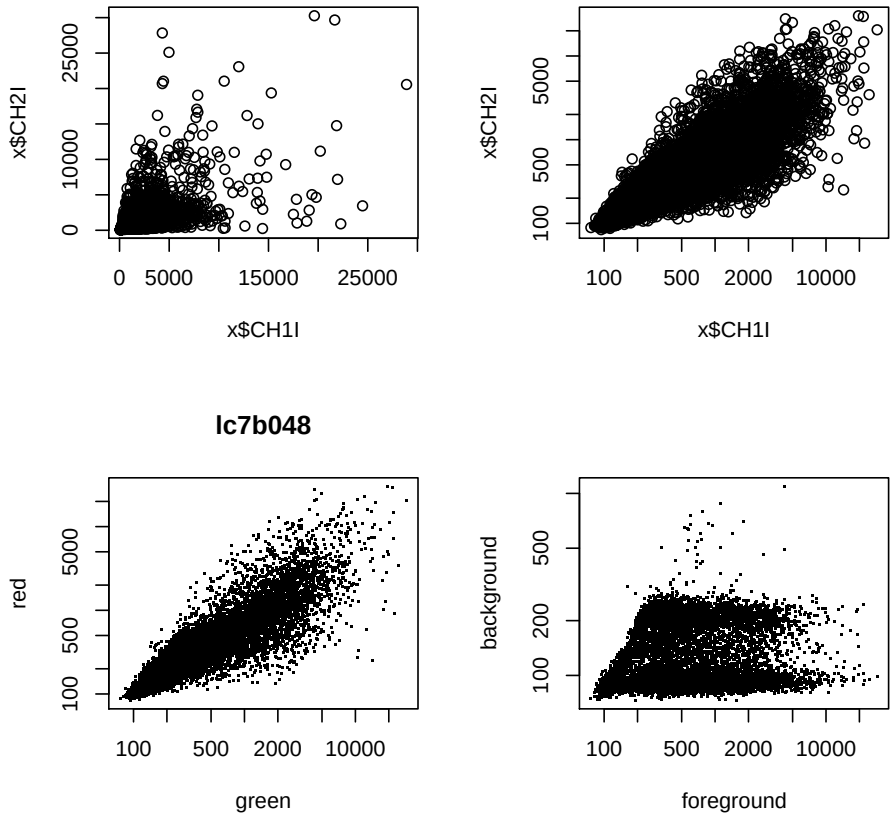
a. Make a histogram of the values in the column CH1I.
b. Produce scatterplots of CHI1 versus CH2I, once with linear axis scaling, once with double-logarithmic.
c. Find out how to decorate the plot with our own axis labels and plot title, and how to change the plot symbols.
d. Save the plots as PDF, and as Windows metafiles. Copy and paste them into MS-Office applications.

```
R> par(mfrow = c(2, 1))
R> hist(x$CH1I, breaks = 100)
R> hist(log(x$CH1I, 2), breaks = 100)
```






```
R> par(mfrow = c(2, 2))
R> plot(x$CH1I, x$CH2I)
R> plot(x$CH1I, x$CH2I, log = "xy")
R> plot(x$CH1I, x$CH2I, log = "xy", main = "lc7b048", xlab = "green",
+      ylab = "red", pch = ".")
R> plot(x$CH1I, x$CH1B, log = "xy", xlab = "foreground", ylab = "background",
+      pch = ".")
```

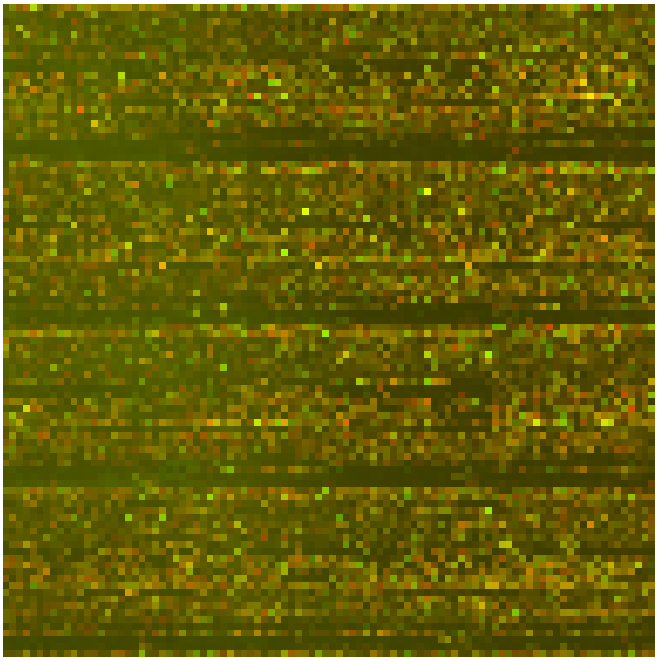


4.) Spatial distribution. The spots on these arrays are arranged in a quadratic pattern of 96 rows and 96 columns. However, the order of the 9216 rows in the file does not simply reflect the spatial arrangement row-by-row or column-by-column. In order to display the spatial distribution of measured foreground and background intensities, we first need to rearrange the data. The relationship between the x- and y-coordinates, as numbers from $1, \ldots, 96$, and the data in the files is given by:

```
R> px = x$COL + 24 * ((x$GRID - 1)%%4)
R> py = x$ROW + 24 * ((x$GRID - 1)%/%4)
```

The following piece of code displays a 2D spatial false-color representation of the CH1I and CH2I intensity data.

```
R> library(pixmap)
R> xs = x[order(px, py), ]
R> r = xs$CH1I^0.25
R> g = xs$CH2I^0.25
R> b = rep(0, nrow(xs))
R> rgb = array(c(r, g, b), dim = c(96, 96, 3))
R> plot(pixmap(rgb, type = "rgb"))
```

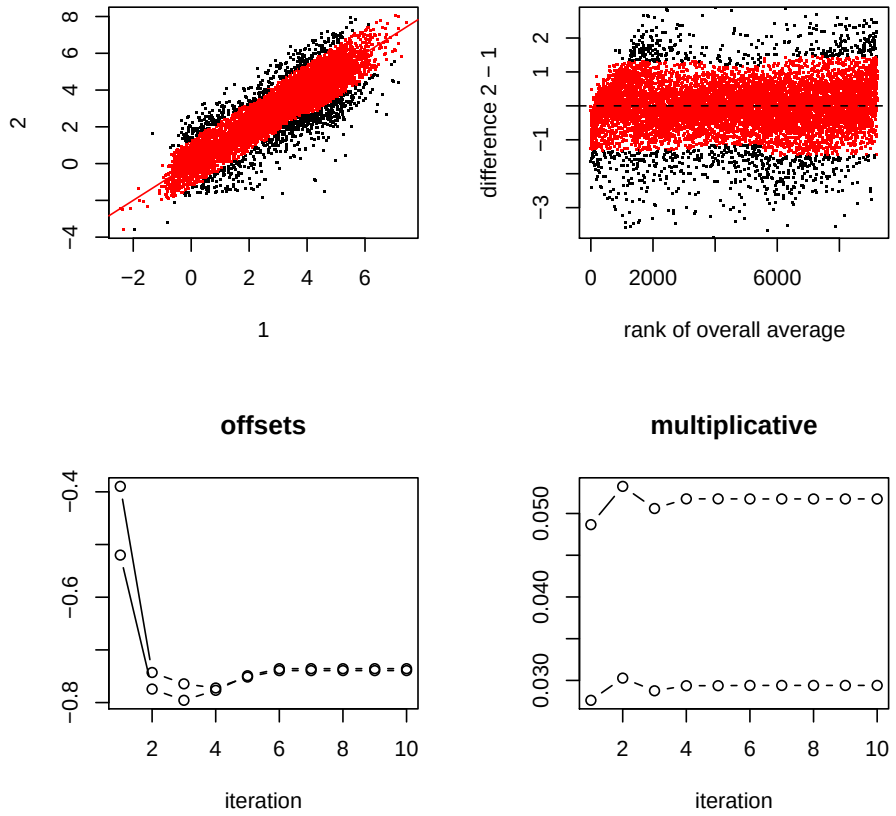


a. Read the help file for `pixmap` and understand what the above code is doing.

b. Try out other transformations than the 4-th root, like identity transformation, square root, logarithm, and observe their impact on the visual appearance of the image.

5.) Calibration and variance stabillization. Download the package VSN from http://www.dkfz.de/abt0840/whuber and install it. Subtract the background intensities CH1B, CH2B from the foreground intensities CH1I, CH2I. Use the function `vsn` to calibrate and transform, and plot the result.

```
R> library(vsn)
R> y = cbind(x$CH1I - x$CH1B, x$CH2I - x$CH2B)
R> nv = vsn(y)
R> plot(nv)
```



6.) Reading multiple data files. In the folder `data/alizadeh`, you find a file `samples.txt`.

a. Read it into a data frame (use the function `read.delim` with the `as.is=T` option)
b. Create 4 matrices of dimensions 9216×8 that contain, respectively, CH1I, CH1B, CH2I, and CH2B intensities of the 9216 spots on the 8 slides with filenames are given in `samples.txt`.
c. Save the matrices into an XDR file.
d. Note: the bioconductor packages marrayInput and affy offer more comfortable methods for reading and managing data from a series of microarrays.

```
R> datapath = "../data/alizadeh"
R> samples = read.delim(file.path(datapath, "samples.txt"), as.is = T)
R> samples
```

```
     name  sampleid
1 lc7b047    CLL-13
2 lc7b048    CLL-13
3 lc7b069    CLL-52
4 lc7b070    CLL-39
5 lc7b019 DLCL-0032
6 lc7b056 DLCL-0024
```

```
7 lc7b057 DLCL-0029
8 lc7b058 DLCL-0023

R> nrspots = 9216
R> nrsamples = nrow(samples)
R> Gf = Gb = Rf = Rb = matrix(NA, nrow = nrspots, ncol = nrsamples)
R> for (i in 1:nrsamples) {
+      filename = paste(samples$name[i], "rex.DAT", sep = "")
+      dat = read.delim(file.path(datapath, filename))
+      Gf[, i] = dat$CH1I
+      Gb[, i] = dat$CH1B
+      Rf[, i] = dat$CH2I
+      Rb[, i] = dat$CH2B
+  }
R> save(Gf, Gb, Rf, Rb, file = "intensities.RData")
```

7.) Different normalization methods. In the following, we are going to identify genes that appear to be differentially transcribed between the 4 CLL samples and the 4 DLCL samples. For this, we will apply a number of different normalization strategies to the data and compare their results.

a. Download the packages Biobase, marrayClasses, marrayNorm, and multtest from http://www.bioconductor.org and install them.

b. Create a 3D array of dimensions $9216 \times 8 \times 3$ that contains, for all spots, the value of M (that is, the log-ratio or the generalized log-ratio), for the 8 slides and the following 3 different normalization methods:

1. vsn (affine normalization and variance stabilization)
2. maNorm with global median location normalization
3. maNorm with loess for intensity- or A-dependent location normalization using the 'loess' smoother

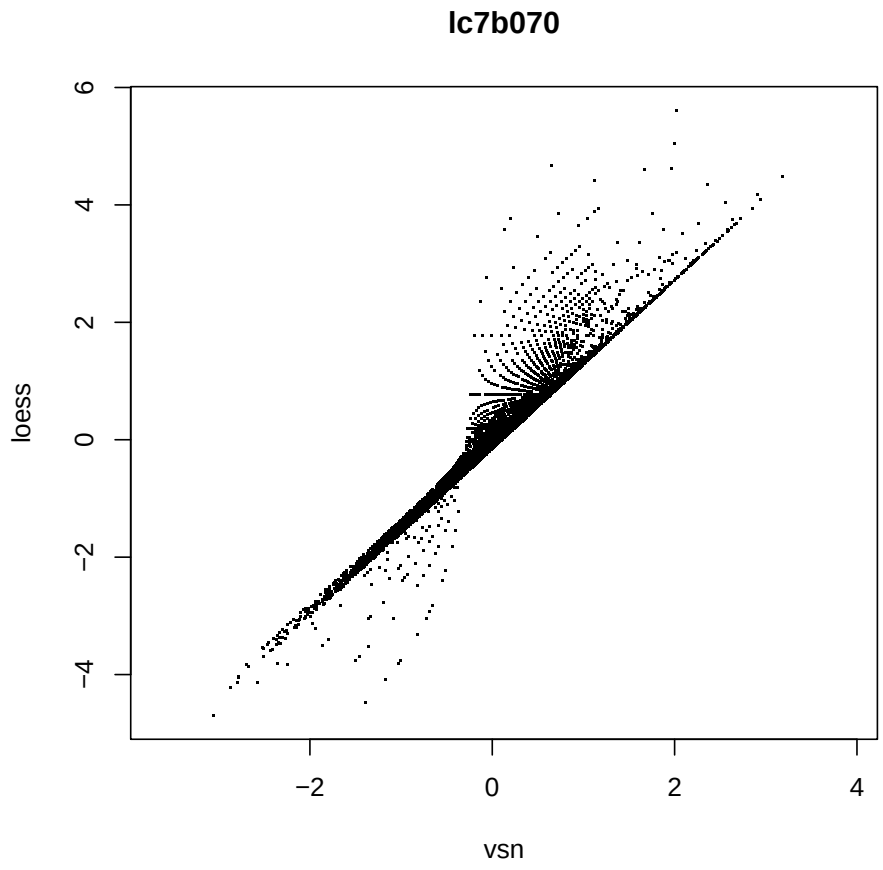
c. Save the array into an XDR file.

```
R> library(marrayNorm)

R> nrmethods = 3
R> M = array(NA, dim = c(nrspots, nrsamples, nrmethods))
R> A = array(NA, dim = c(nrspots, nrsamples, nrmethods))
R> #
R> # vsn
R> # green in columns 1:8, red in 9:16
R> nw = vsn(cbind(Gf - Gb, Rf - Rb))
R> M[, , 1] = nw$hy[, 9:16] - nw$hy[, 1:8]
R> A[, , 1] = nw$hy[, 9:16] + nw$hy[, 1:8]
R> #
R> # global median and loess
R> mar = new("marrayRaw", maGf = Gf, maGb = Gb, maRf = Rf, maRb = Rb)
R> nm = maNorm(mar, norm = "median", echo = T)
```

```
R> nl = maNorm(mar, norm = "loess", echo = T)
R> M[, , 2] = nm@maM
R> A[, , 2] = nm@maA
R> M[, , 3] = nl@maM
R> A[, , 3] = nl@maA
R> #
R> save(M, A, file = "MA.RData")
```

8.) Qualitatively compare the results. Look at scatterplots of the values of M from the same slide, calculated with different normalization methods. Do the values generally agree? How do they differ?

```
R> plot(M[, 4, 1], M[, 4, 2], pch = ".", xlab = "vsn", ylab = "loess",
+      main = samples$name[4])
```



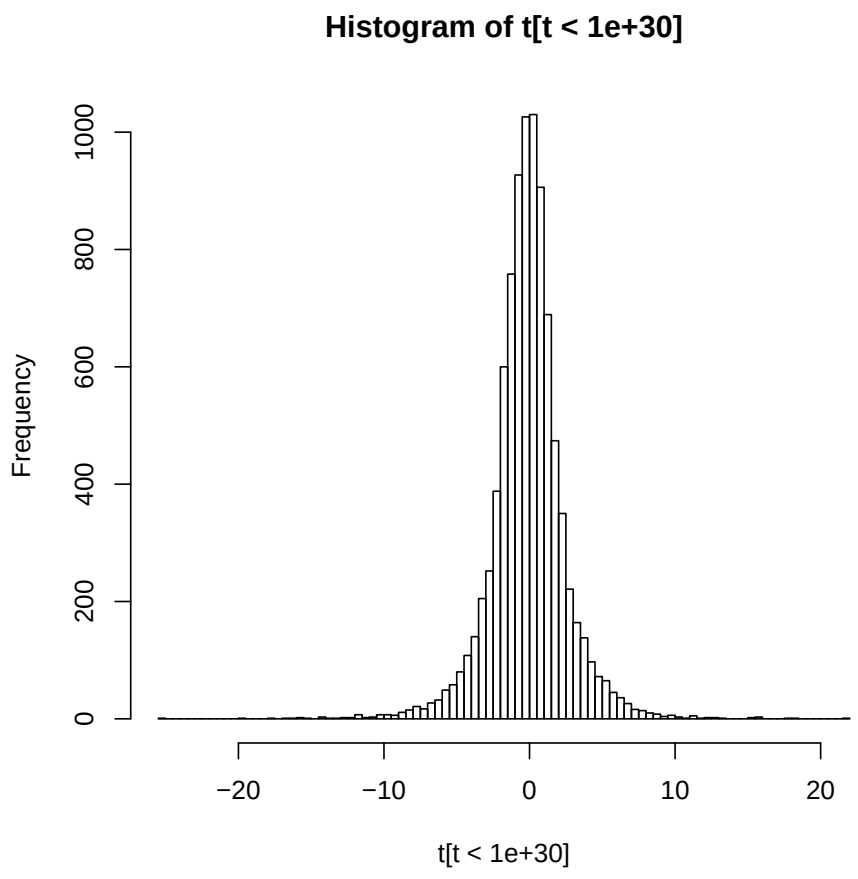
9.) Testing for differential transcription. Now we are ready to calculate test statistics and to select genes. *Note:* The number of replicates (4 versus 4) that we are considering here is very small and no solid conclusions about individual genes or individual samples will be derived from that. The full data set contains many more chips. Here we restrict ourselves to a few of them in order to keep calculations simple and not too slow for the purpose of this course.

a. Look at the function `t.test` from the package ctest (which is part of the base libraries), and at `mt.teststat` from the package multtest.

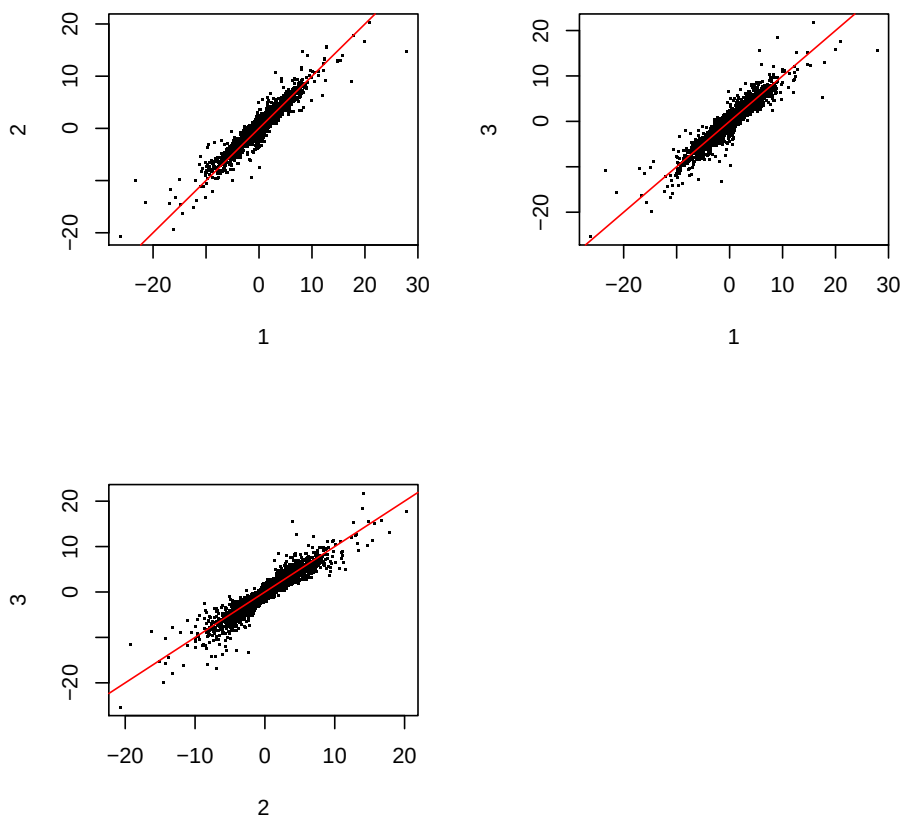
b. For each gene, and for each of the normalization methods, calculate the t-statistic for the CLL-to-DLBL class distinction. Store the result in a 9216 x 3 matrix. Which of the functions `t.test`, `mt.teststat` calculates faster? Look at the histogram of t-values that they produce; you may find extreme values like '3e38' in there. Where do they come from?

c. How do the t-statistics agree between the different normalization methods?

```
R> library(multtest)
R> classlabel = c(0, 0, 0, 0, 1, 1, 1, 1)
R> t = mt.teststat(M[, , 3], classlabel)
R> range(t, na.rm = T)

[1] -2.533819e+01  3.402823e+38

R> which(t > 1e+30)

[1] 2829 2930 2931

R> hist(t[t < 1e+30], 100)
```



```
R> calct = function(classlab, dat) {
+       t = mt.teststat(dat, classlab)
+       t[t > 1e+30] = NA
+       return(t)
+   }
R> t = matrix(NA, nrow = nrspots, ncol = nrmethods)
R> for (meth in 1:nrmethods) {
+       t[, meth] = calct(classlabel, M[, , meth])
+   }
R> #
R> par(mfrow = c(2, 2))
R> for (j in 2:nrmethods) {
+       for (i in 1:(j - 1)) {
+           plot(t[, i], t[, j], pch = ".", xlab = paste(i), ylab = paste(j))
+           lines(c(-30, 30), c(-30, 30), col = "red")
+       }
+   }
R> # alternatively: use splom from library(lattice)
```



10.) Biology. Look at the 5 top genes, and using the information in the file

.../data/alizadeh/scheme.htm, find out the curated gene names. Do they correspond to genes that are mentioned in the Alizadeh et al. paper?

```
R> csw = read.delim(file.path(datapath, "chip_spot_well.tab.txt"),
+      as.is = T, header = F)
R> colnames(csw) = c("batch", "spot", "wellid")
R> csw[1:3, ]

  batch spot wellid
1  lc3a    1  13633
2  lc3a 2026  13634
3  lc3a    2  13635

R> wcn = read.delim(file.path(datapath, "well_cloneid_name.tab.txt"),
+      as.is = T, header = F)
R> colnames(wcn) = c("wellid", "cloneid", "name")
R> wcn[c(1, 51, 158), ]

    wellid cloneid                                          name
1        1  682638 btk=Bruton agammaglobulinemia tyrosine kinase
51      51  683083                                    Cyclin T2a
158    158  683739    BRCA2=Mutated in breast and ovarian cancer

R> #
R> topspots = order(abs(t[, 1]), decreasing = TRUE)[1:5]
R> wellid = csw$wellid[csw$batch == "lc7b" & csw$spot %in% topspots]
R> wcn[wcn$wellid %in% wellid, ]

      wellid cloneid
13036  13036 1289261
13394  13394 1334260
17614  17614  289965
17670  17670  417287
19440  19440 1340277
                                                                                n
13036                                                                       Unkn
13394                                                 Unknown  UG Hs.120716  ESTs   sc_id5
17614                                       Protein tyrosine phosphatase, non-receptor type
17670 aryl hydrocarbon receptor (AhR)=AH-receptor=basic helix-loop-helix transcription fac
19440                                                                       Unkn
```

11.) ***t*-thresholds.** Designate as *differentially expressed* those clones for which the absolute value of t is larger than a certain threshold. What are the values of this threshold for our data, if we want to have a clone list length $10, 20, 50, 100, \ldots$?

```
R> # clone list lengths
R> cll = c(10, 20, 50, 100, 200, 500, 1000)
R> threshold = matrix(NA, nrow = length(cll), ncol = nrmethods)
```

```
R> rownames(threshold) = paste(cll)
R> colnames(threshold) = c("vsn", "global median", "loess")
R> #
R> for (meth in 1:nrmethods) {
+      st = sort(abs(t[, meth]), decreasing = TRUE)
+      for (j in 1:length(cll)) {
+          threshold[j, meth] = st[cll[j]]
+      }
+  }
R> threshold

           vsn global median     loess
10   16.665285     14.763376 15.796528
20   12.801105     13.274832 13.770094
50    9.857308      9.627267 10.485302
100   8.268596      8.037820  8.519700
200   6.701843      6.612137  6.921097
500   4.957255      4.793148  5.130494
1000  3.658167      3.528695  3.786914
```

12.) Permutations.

a. How many ways are there to split a set of 8 objects into two groups of 4 and 4 ? Use the function `nchoosek` from the file `nchoosek.R` to generate a numerical representation of these splits.

b. Prepare a matrix with 8 rows, corresponding to the 8 samples, and as many columns as there are splits. Set the matrix elements to 0 and 1, such that each column of the matrix represents a split.

```
R> source("nchoosek.R")
R> nck = nchoosek(7, 3)
R> nck

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
[1,]    1    1    1    1    1    1    1    1    1     1     1     1     1     1
[2,]    2    2    2    2    2    3    3    3    3     4     4     4     5     5
[3,]    3    4    5    6    7    4    5    6    7     5     6     7     6     7
     [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
[1,]     1     2     2     2     2     2     2     2     2     2     2     3
[2,]     6     3     3     3     3     4     4     4     5     5     6     4
[3,]     7     4     5     6     7     5     6     7     6     7     7     5
     [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
[1,]     3     3     3     3     3     4     4     4     5
[2,]     4     4     5     5     6     5     5     6     6
[3,]     6     7     6     7     7     6     7     7     7

R> classlabel = matrix(0, nrow = nrsamples, ncol = ncol(nck))
R> for (p in 1:ncol(nck)) {
```

```
+      classlabel[8, p] = 1
+      classlabel[nck[, p], p] = 1
+  }
R> classlabel
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
[1,]    1    1    1    1    1    1    1    1    1     1     1     1     1     1
[2,]    1    1    1    1    1    0    0    0    0     0     0     0     0     0
[3,]    1    0    0    0    0    1    1    1    1     0     0     0     0     0
[4,]    0    1    0    0    0    1    0    0    0     1     1     1     0     0
[5,]    0    0    1    0    0    0    1    0    0     1     0     0     1     1
[6,]    0    0    0    1    0    0    0    1    0     0     1     0     1     0
[7,]    0    0    0    0    1    0    0    0    1     0     0     1     0     1
[8,]    1    1    1    1    1    1    1    1    1     1     1     1     1     1
     [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
[1,]     1     0     0     0     0     0     0     0     0     0     0     0
[2,]     0     1     1     1     1     1     1     1     1     1     1     0
[3,]     0     1     1     1     1     0     0     0     0     0     0     1
[4,]     0     1     0     0     0     1     1     1     0     0     0     1
[5,]     0     0     1     0     0     1     0     0     1     1     0     1
[6,]     1     0     0     1     0     0     1     0     1     0     1     0
[7,]     1     0     0     0     1     0     0     1     0     1     1     0
[8,]     1     1     1     1     1     1     1     1     1     1     1     1
     [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35]
[1,]     0     0     0     0     0     0     0     0     0
[2,]     0     0     0     0     0     0     0     0     0
[3,]     1     1     1     1     1     0     0     0     0
[4,]     1     1     0     0     0     1     1     1     0
[5,]     0     0     1     1     0     1     1     0     1
[6,]     1     0     1     0     1     1     0     1     1
[7,]     0     1     0     1     1     0     1     1     1
[8,]     1     1     1     1     1     1     1     1     1
```

13.) False discovery rate (FDR). Now we want to apply these permutations to the data to estimate the FDR. Do the following for each normalization method:

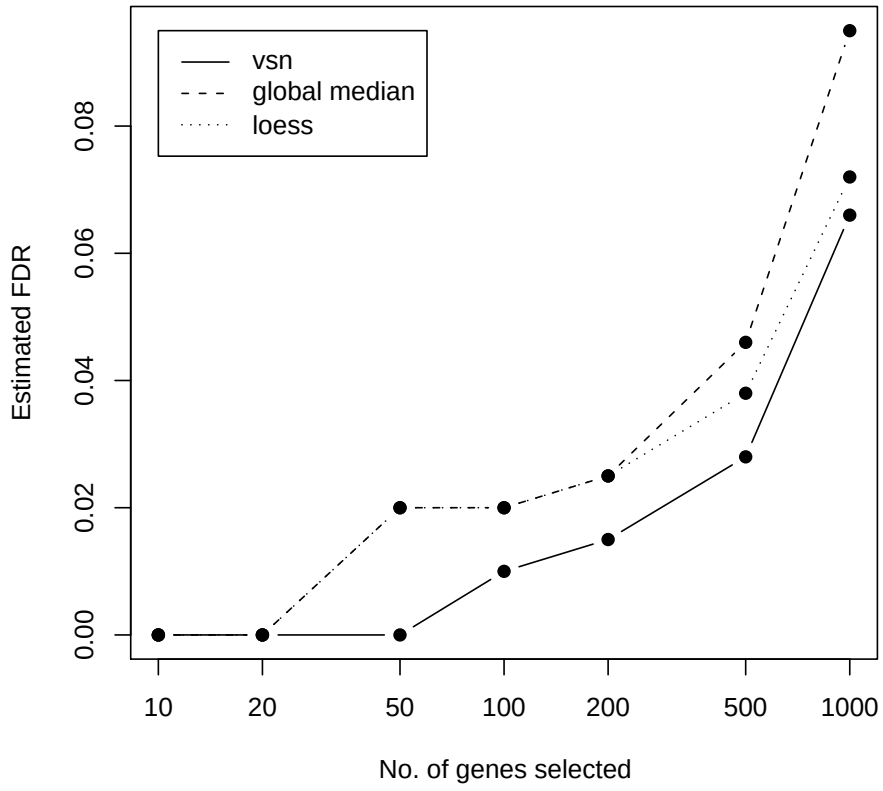
a. For each of the splits, calculate the corresponding t-statistics for all genes.

b. For each of the splits, and for each of the above choices for clone list lengths and thresholds, calculate the number of clones that have an absolute t-value greater or equal to the threshold.

c. Calculate the median of these numbers across the splits. Divide this by the clone list length to obtain an estimate of the FDR.

```
R> fdr = matrix(NA, nrow = length(cll), ncol = nrmethods, dimnames = dimnames(threshold))
R> for (meth in 1:nrmethods) {
+      # permt is a 9216 x 35 matrix of t-values, with
+      # rows corresponding to the clones
+      # and columns to the different splits
```

```
+      permt = apply(classlabel, 2, calct, M[, , meth])
+      for (j in 1:length(cll)) {
+          # pnrsel is a vector of length 35, with the
+          # number of clones that had t greater or
+          # equal to the threshold
+          pnrsel = apply(permt, 2, function(t) length(which(abs(t) >=
+              threshold[j, meth])))
+          fdr[j, meth] = median(pnrsel)/cll[j]
+      }
+  }

R> plot(range(cll), range(fdr), type = "n", log = "x", xlab = "No. of genes selected",
+      ylab = "Estimated FDR")
R> for (meth in 1:nrmethods) lines(cll, fdr[, meth], type = "b",
+      pch = 19, lty = meth)
R> legend(min(cll), max(fdr), colnames(fdr), lty = 1:nrmethods)
```



14.) In the directory ../data/Shipp, you find a number of Affymetrix CEL files and a corresponding CDF file.

a. Using the package `affy`, load them into a *probe level object* (Plob).

b. Look at the spatial distribution of intensities on the chips.
c. Normalize the data and calculate probe set summary values.

```
R> library(affy)
R> oldwd = getwd()
R> setwd("../data/Shipp")
R> dat = ReadAffy()
R> # show images of probe data
R> image(dat)
R> # calculation probe set summaries
R> e = express(dat)
R> # scatterplot first versus second sample
R> plot(exprs(e1), pch = ".")
```

15.) Prepare for the classification exercises on Thursday. Check whether the packages `e1071` and `rpam` are installed on your computer. If not, install them.