Microarray analysis exercise

Anne Kupczok

Center for Integrative Bioinformatics Vienna
Max F. Perutz Laboratories

February 6th, 2008

First Part - Linear modells

- Limma: Linear Models for Microarray Data

- Can be easily installed and loaded:
- `source(‘‘http://bioconductor.org/biocLite.R’’)`
- `biocLite(pkgs=‘‘limma’’)`
- `library (‘‘limma’’)`

Generation of target file and reading the data

- The target file contains the names of the input files and the probe-dye assignment:
- $$\begin{tikzcd} A \arrow[r, shift left, "\text{file1}"] & B \arrow[l, shift left, "\text{file2}"] \end{tikzcd}$$

```
FileNames Cy3 Cy5
file1 A B
file2 B A
```

- Limma commands:
- `targets<-readTargets(targetfilename)`
- `RG <- read.maimages(targets$FileNames)`
- Read the arrangement of blocks of spots on the arrays (print-tip groups):
- `RG$printer <-getLayout(RG$genes)`

Spot types

- There may be different types of spots on the arrays like probes and control spots:
- `spottypes <- readSpotTypes()`
- `RG$genes$Status <- controlStatus(spottypes, RG)`

- Generate the MA-plot of the unnormalized data of array `x` in `RG`:
- `plotMA(RG,array=x)`

Background normalisation

- Visualization of the green background signal over the array `x`:
- `imageplot(log2(RG$Gb[,x]),RG$printer)`

- Or save the imageplot for all arrays in png-files:
- `imageplot3by2(RG,''Gb'')`

- Background normalisation: `RG <- backgroundCorrect(RG, method=''normexp'',offset=25)`
  - `normexp` is a statistical method, see `?backgroundCorrect` for other methods available

M/A Data

- M and A values are computed from R and G together with a normalisation:
- `MA <- normalizeWithinArrays(RG)`
    - Default is print-tip loess normalisation, see `?normalizeWithinArrays` for other methods available
- Compare the distribution of M-values over the arrays:
- `boxplot(MA$M ~ col(MA$M),names=colnames(MA$M))`
- Eventually a normalisation between arrays must be applied, e.g. quantile normalisation:
- `MA<-normalizeBetweenArrays(MA,method=‘‘quantile‘‘)`

Design matrix

- The design matrix can be computed from the targets:
- `design <- modelMatrix(targets, ref=''A'')`
- Then the computed overall M values correspond to $log(B/A)$

- A dye effect can be included in the model by a linear term:
- `design2 <- cbind(Dye=1, B=design)`

Fitting a linear modell

- Estimate coefficients (*M*-values) and standard deviations for each column in the design by least squares:
- `fit <- lmFit(MA, design)`
- Compute the moderated *T*-statistic by empirical Bayes:
- `fit <- eBayes(fit)`
- Show the highest ranking genes (note: if the modell has more than one factor, the name of the factor is given by the `coef` option):
- `topTable(fit, adjust=''fdr'')`

Exercise

- Download `microarray.zip` and unzip it in a separate directory
- Start R in this directory, load the limma-library
- Read in the targets file `targets.txt` and look at the design
- Generate the RG-Data (Note: the microarray files are in the genepix-format, use option `source=''genepix''` for `read.maimages`)
- Save the print-tip groups in `RG$printer`
- Save the spottypes in `RG$genes$Status`, look at the M/A-plots
- Save the imageplots for red and green background in png-files and look at them

Exercise

- Do background correction with ``normexp'' and within-array normalization with print-tip Loess
- Generate a boxplot of the *M*-values over the arrays
- Do all further analysis only with the probes (without control): `MA <- MA[MA$genes$Status==''Probe'',]`
- Compute one design matrix with reference `minus` and one with reference `minus` and a dye effect
- Fit a linear modell with each design separately and compute the empirical Bayes statistic, compare the results

Classification

KNN with *n* neighbors:

- `knn(train,test,factors,n)`
    - `train` is the training set matrix, the row vectors are the objects
    - `test` are the objects to be classified
    - factors is a factor of the classifications of the training set
- Leave-one-out cross validation:
- `knn.cv(train,factors,n)`

Classification

KNN with *n* neighbors:

- `knn(train,test,factors,n)`
  - `train` is the training set matrix, the row vectors are the objects
  - `test` are the objects to be classified
  - factors is a factor of the classifications of the training set
- Leave-one-out cross validation:
- `knn.cv(train,factors,n)`

LDA Learn a classifier:

- `fit.lda<-lda(train,factors)`
  - `train` is a data frame where the row vectors are the objects of interest again
- Classify a new object
- `predict(fit.lda,test)`

Clustering

Heatmap of a data matrix m:

- `heatmap(m, distfun = dist, col = topo.colors(32))`
  - The distance function `distfun` is default the euclidean distance `dist`, but other distance function can be applied

Clustering

Heatmap of a data matrix m:

- `heatmap(m, distfun = dist, col = topo.colors(32))`
  - The distance function `distfun` is default the euclidean distance `dist`, but other distance function can be applied

HC First generation of a distance matrix between the row vectors of a matrix, then clustering of the distance matrix:

- `d <- dist(m); hc <- hclust(d)`
  - The dendrogram can be visualized: `plot(hc)`
  - *k* Classes can be cut out of the dendrogram: `cutree(hc,k)`

Clustering

Heatmap of a data matrix m:

- `heatmap(m, distfun = dist, col = topo.colors(32))`
  - The distance function `distfun` is default the euclidean distance `dist`, but other distance function can be applied

HC First generation of a distance matrix between the row vectors of a matrix, then clustering of the distance matrix:

- `d <- dist(m); hc <- hclust(d)`
  - The dendrogram can be visualized: `plot(hc)`
  - *k* Classes can be cut out of the dendrogram: `cutree(hc,k)`

k-means `fit.kmeans <- kmeans(m, k)`

  - `fit.kmeans$cluster` is the mapping of each vector to one of the *k* classes

Exercise - Classification

- Load the librarys MASS and class and the data set
  - 15 patients were treated with a medicament against leukemia
  - Afterwards, its measured wheter a **H**igh or **L**ow amount of leukemic cells is still present in the bones
  - Gene epxression of patients measured by Affymetrix arrays of 8793 genes
- Load the script ``scripttstat.r'' with `source` (there the feature selection and the correlation distance is implemented)
- Do KNN with $k = 3$ and cross validation and compare the inferred classes with the original ones (hint: factors can be compared with `table`)

Exercise - Feature Selection

- Compute the pvalues of a feature selection T-Test, create a new ordered matrix and truncate the matrix to the 50 most significant genes: `pval<-fs.ttest(mat,factors);` `matord<-mat[order(pval),];` `mattrunc<-matord[1:50,]`
- Repeat the KNN cross-validation with the truncated matrix and compare the results to the previous classifier
- Visualise the feature selection with a heatmap (hint: multiple graphical windows can be opened with `x11()`):
  - Compare the heatmap of the 50 most significant genes with 50 arbitrary genes, use the distance function `cordist`
  - Compare the different distance functions `cordist` and `dist` using the 50 most significant genes

Exercise - Clustering

- Do hierarchical clustering with the complete matrix and compare the two distance functions dist and cordist, therefore cut 2 classes from the dendrograms
- Repeat using only the 50 most significant genes
    - What would you decide is the number of classes each of the clustering finds
    - Look also on the structure within the classes

- Do k-means with $k = 2$ and $k = 3$ using only the 50 most significant genes and compare the results