Exercise 2015-09-21

Hubert Rehrauer

September 21, 2015

1 Introduction

A study examines a liver disease. In affected patients the liver does not work properly. Part of the liver shows acute signs other parts are affected but do not show severe symptoms. The study searches for the genes that have changed expression due to the disease. The goal is to understand at a molecular level the causes and consequences of the disease.

The study has used Affymetrix Whole Genome Microarrays of the type HG-U133-Plus2 to measure the gene expression. In total 5 sick patients were examined and from each patient a sample from acutely and moderately tissue was measured. As a reference normal tissue from 6 healthy patients were examined.

In this exercise we run some exploratory analysis of the data set. We want to check if there are outliers and/or systematic biases.

2 Loading the data

The phenotype information is in the file `SampleAnnotation.txt` that we read into a data frame,

```
>   anno = read.table("SampleAnnotation.txt", as.is=TRUE, sep="\t", quote="",
+                     row.names=1, header=TRUE)
```

that holds the sample name, tissue type, patient ID, and associated file:

```
>   anno
```

```
        TissueType PatientId        File
norm-02       norm       P02 norm-02.CEL
norm-05       norm       P05 norm-05.CEL
norm-07       norm       P07 norm-07.CEL
norm-09       norm       P09 norm-09.CEL
norm-10       norm       P10 norm-10.CEL
norm-11       norm       P11 norm-11.CEL
sick-04       sick       P04 sick-04.CEL
sick-12       sick       P12 sick-12.CEL
sick-13       sick       P13 sick-13.CEL
sick-14       sick       P14 sick-14.CEL
sick-15       sick       P15 sick-15.CEL
```

For labeling and coloring the subsequent plots we define the variables:

```
>   samples = rownames(anno)
> colors = rainbow(nrow(anno))
```

and generate boolean indices with which we can access the normal, sick and acute samples only:

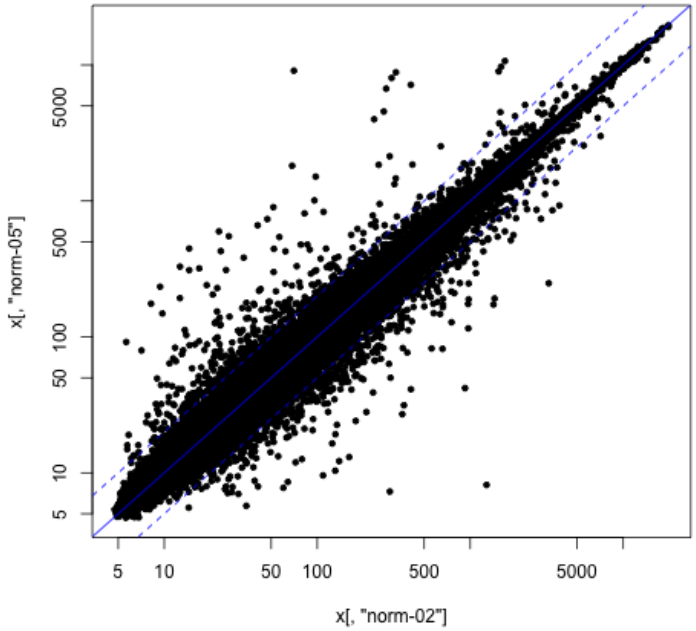
```
>   isNorm = anno$TissueType == "norm"
> isSick = anno$TissueType == "sick"
> isAcute = anno$TissueType == "acute"
```

Now we load the expression data

```
> x = read.table("expressiondata.txt", as.is=TRUE, sep="\t", quote="", row.names=1, header
> x = as.matrix(x)
```

We visually compare the expression signals from sample 1 and 2 by plotting them with the standard plot command:

```
> plot(x[ , "norm-02"], x[, "norm-05"], log="xy", pch=20)
> abline(0, 1, col="blue")
> abline(log10(2), 1, col="blue", lty=2);
> abline(-log10(2), 1, col="blue", lty=2);
```



The solid blue line gives the first diagonal and the dashed lines give the boundaries for 2-fold up- or down-regulation.

3 Checking the distribution of the intensities

A basic assumption for the subsequent analysis is, that the intensity distributions of the different arrays are similar. Do create boxplots that summarize the value distribution for each sample, and use the function plotDensities in the package limma.

4 Checking the consistency of the replicates

Do check the consistency of the replicates by computing sample correlation.

In order to check whether the replicates correlate well, we compute the correlation matrix on the logarithmic values and print it

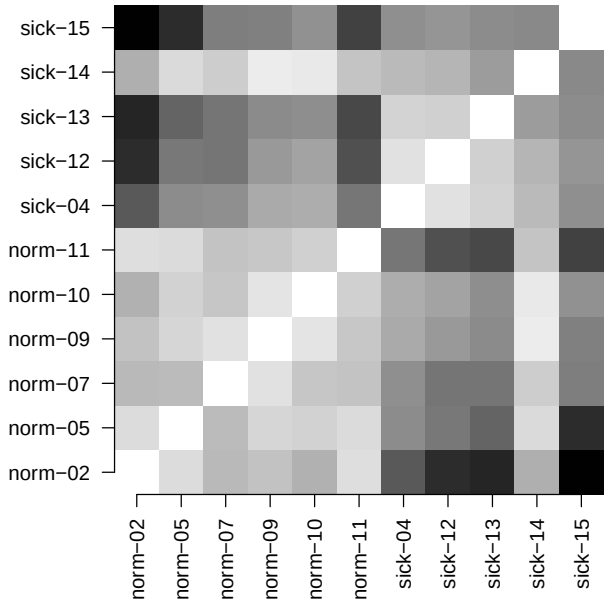
```
> corrMatrix = cor(x)
> signif(corrMatrix, digits=3)
```

	norm-02	norm-05	norm-07	norm-09	norm-10	norm-11	sick-04	sick-12	sick-13
norm-02	1.000	0.981	0.962	0.967	0.958	0.982	0.910	0.885	0.882
norm-05	0.981	1.000	0.963	0.978	0.975	0.980	0.938	0.927	0.916
norm-07	0.962	0.963	1.000	0.984	0.969	0.968	0.940	0.925	0.925
norm-09	0.967	0.978	0.984	1.000	0.985	0.970	0.954	0.945	0.938
norm-10	0.958	0.975	0.969	0.985	1.000	0.974	0.955	0.950	0.939
norm-11	0.982	0.980	0.968	0.970	0.974	1.000	0.926	0.905	0.901

```
sick-04   0.910   0.938   0.940   0.954   0.955   0.926   1.000   0.983   0.976
sick-12   0.885   0.927   0.925   0.945   0.950   0.905   0.983   1.000   0.974
sick-13   0.882   0.916   0.925   0.938   0.939   0.901   0.976   0.974   1.000
sick-14   0.957   0.980   0.973   0.990   0.988   0.968   0.962   0.960   0.946
sick-15   0.862   0.886   0.930   0.932   0.940   0.897   0.939   0.943   0.938
        sick-14 sick-15
norm-02   0.957   0.862
norm-05   0.980   0.886
norm-07   0.973   0.930
norm-09   0.990   0.932
norm-10   0.988   0.940
norm-11   0.968   0.897
sick-04   0.962   0.939
sick-12   0.960   0.943
sick-13   0.946   0.938
sick-14   1.000   0.936
sick-15   0.936   1.000
```

We visualize the matrix as an image:

```
>   par(mar=c(8,8,2,2))
> grayScale <- gray((1:256)/256)
> image(corrMatrix, col=grayScale,  axes=FALSE)
> axis(1, at=seq(from=0, to=1, length.out=length(samples)), labels=samples, las=2)
> axis(2, at=seq(from=0, to=1, length.out=length(samples)), labels=samples, las=2)
```



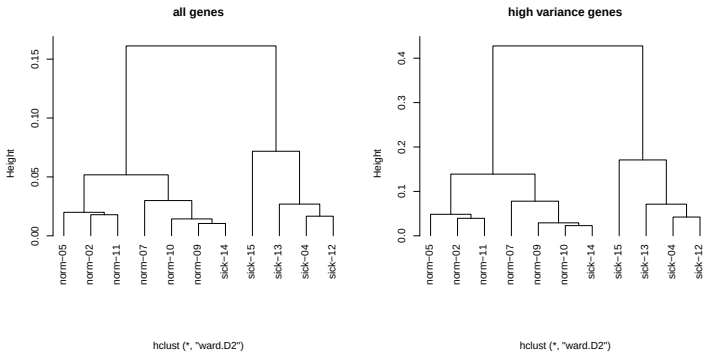
From the correlation we can see that

- normals show high correlation among each other
- the normal are very different from both sick and acute
- the sick and acute are rather similar
- `sick-14` rather looks like a normal sample
- `sick-15` has overall low correlation but rather looks like an "acute"
- `acute-04-a` which is a technical replicate of `acute-04` is more similar to `sick-04` than to `acute-04`.

5 Sample Clustering

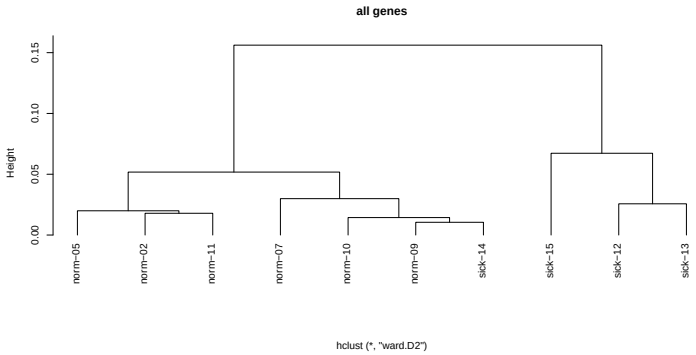
The sample clustering shows the similarities of the expression patterns of the samples in a tree. In order to compute the similarities of the samples one can use all genes or only a subset of the genes. When using all genes where is the risk that the absent genes drive the clustering of the samples. This is because in many studies the absent genes make up the majority of the measured genes. But those genes have a low intensity that is strongly influenced by the background signal measured on each chip and not by real gene expression

```
>   x.sd = apply(x, 1, sd, na.rm=TRUE)
> ord = order(x.sd, decreasing=TRUE)
> highVarGenes = ord[1:500]
> par(mfrow=c(1,2));
> d = as.dist(1-cor(x));
> c=hclust(d, method="ward.D2");
> plot(c, hang=-0.1, main="all genes", xlab="")
> d = as.dist(1-cor(x[highVarGenes, ]));
> c=hclust(d, method="ward.D2");
> plot(c, hang=-0.1, main="high variance genes", xlab="")
```



If we run the clustering without sample `sick-04`, the `acute-04` does no longer cluster in the branch with the other sick samples

```
>   sub = x[ , samples != "sick-04"]
> d = as.dist(1-cor(sub));
> c=hclust(d, method="ward.D2");
> plot(c, hang=-0.1, main="all genes", xlab="")
```



6 Apply quantile normalization

Use the function `limma::normalizeQuantiles` to normalize the data and plot again the histograms.

7 Sample Representation in Principal Component Space

Use the functions `cmdscale` and `prcomp` to create a plot that represents the sample distances in a reduced space.