Gene expression profiling of bone marrow-derived mononuclear cells (BMMC) from patients with rheumatoid arthritis (RA) vs. osteoarthritis (OA)

Nuria Renard r0368280

Nathalie Van den Berghe r0368279

Hanne Ruts r0384457

Introduction

Used article: Lee et al. Abnormal networks of immune response-related molecules in bone marow cells from patients with rheumatoid arthritis as revealed by DNA microarray analysis. Arthritis & Therapy 2011, 13:R89

https://www.ncbi.nlm.nih.gov/pubmed/21679443 (https://www.ncbi.nlm.nih.gov/pubmed/21679443)

Microarray data:

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE27390
(https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE27390)

Rheumatoid arthritis (RA) is a systemic autoimmune disease characterized by chronic synovitis that is often pathogenic and destructive to articular cartilage and bone. Accumulating evidence suggest that abnormalities in bone marrow (BM) have a significant role in RA inflammation. In this study, differentially expressed molecules in BM cells from rheumatoid arthritis (RA) and osteoarthritis (OA) patients were compared and abnormal regulatory networks were analyzed to identify the role of BM cells in RA. OA patients were chosen as controls because the OA pathology is relatively well understood and the BM cells from these patients are far more readily available than those from healthy subjects.

We start off with the analysis of a microarray dataset obtained from bone marrow samples of 9 RA and 10 OA patients. From this microarray data we generated a list of differentially expressed genes and performed a ranking-based analysis by use of GORilla. Subsequently, iRegulon analysis was performed on this list of differentially expressed genes to predict regulators and targets. From the list of predicted transcription factors and based on the literature we chose NFKB1 as a possible regulator of the upregulated genes in RA. From genemania we retrieved an association network of NFKB1, we analyzed the gene set with iRegulon and found that the motif was enriched. Furthermore a Chipseq dataset of NFKB1 was analyzed to investigate if NFKB1 directly binds the target genes (i.e. the upregulated genes in RA)

1. Microarray analysis

1.1 Libraries

First we load the following libraries: the affy package to deal with Affymetrix microarray data and the limma package for differential expression analysis.

In [1]:

```
library(limma)
library(affy)
```

```
Loading required package: BiocGenerics
Loading required package: parallel

Attaching package: ‘BiocGenerics’

The following objects are masked from ‘package:parallel’:

    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
    clusterExport, clusterMap, parApply, parCapply, parLapply,
    parLapplyLB, parRapply, parSapply, parSapplyLB

The following object is masked from ‘package:limma’:

    plotMA

The following object is masked from ‘package:stats’:

    xtabs

The following objects are masked from ‘package:base’:

    anyDuplicated, append, as.data.frame, as.vector, cbind, colnames
,
    do.call, duplicated, eval, evalq, Filter, Find, get, intersect,
    is.unsorted, lapply, Map, mapply, match, mget, order, paste, pma
x,
    pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce, rep.int
,
    rownames, sapply, setdiff, sort, table, tapply, union, unique,
    unlist, unsplit

Loading required package: Biobase
Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.
```

In [2]:

```
#we downloaded the cel files in:
celfiles = Sys.glob("/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/*.gz")
```

The original files are located in "/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/"

Under the account of r0368280

1.2 Loading the raw probe data

In [3]:

```
celfiles
```

```
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677339.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677340.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677341.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677342.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677343.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677344.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677345.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677346.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677347.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677348.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677349.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677350.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677351.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677352.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677353.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677354.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677355.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677356.CEL.gz'
'/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSM677357.CEL.gz'
```

The variable celfiles contain a list of cell file names, nothing more. The data is not loaded yet. To load the files we will use a function from the bioconductor affy package

In [4]:

```
rawcel <- ReadAffy(filenames=celfiles)
```

The variable rawcel contains an AffyBatch object.

Next step is to annotate the samples: the rawcel object only knows of the file names:

In [5]:

```
sampleNames(rawcel)
```

```
'GSM677339.CEL.gz'  'GSM677340.CEL.gz'  'GSM677341.CEL.gz'
'GSM677342.CEL.gz'  'GSM677343.CEL.gz'  'GSM677344.CEL.gz'
'GSM677345.CEL.gz'  'GSM677346.CEL.gz'  'GSM677347.CEL.gz'
'GSM677348.CEL.gz'  'GSM677349.CEL.gz'  'GSM677350.CEL.gz'
'GSM677351.CEL.gz'  'GSM677352.CEL.gz'  'GSM677353.CEL.gz'
'GSM677354.CEL.gz'  'GSM677355.CEL.gz'  'GSM677356.CEL.gz'
'GSM677357.CEL.gz'
```

We want to let R know which disease is which: rheumatoid arthritis (RA) or osteoarthritis (OA). To do this we will make a vector with the diseases.

In [2]:

```
disease = c("RA",
            "RA",
            "RA",
            "RA",
            "RA",
            "RA",
            "RA",
            "RA",
            "RA",
            "OA",
            "OA",
            "OA",
            "OA",
            "OA",
            "OA",
            "OA",
            "OA",
            "OA",
            "OA",
            "OA")
```

Now we'll merge the sample names and the disease into one data frame:

In [7]:

```
#first make a data frame
pd <- data.frame(disease = disease)
rownames(pd) <- sampleNames(rawcel)
pd
```

	disease
GSM677339.CEL.gz	RA
GSM677340.CEL.gz	RA
GSM677341.CEL.gz	RA
GSM677342.CEL.gz	RA
GSM677343.CEL.gz	RA
GSM677344.CEL.gz	RA
GSM677345.CEL.gz	RA
GSM677346.CEL.gz	RA
GSM677347.CEL.gz	RA
GSM677348.CEL.gz	OA
GSM677349.CEL.gz	OA
GSM677350.CEL.gz	OA
GSM677351.CEL.gz	OA
GSM677352.CEL.gz	OA
GSM677353.CEL.gz	OA
GSM677354.CEL.gz	OA
GSM677355.CEL.gz	OA
GSM677356.CEL.gz	OA
GSM677357.CEL.gz	OA

We will convert this dataframe to an AnnotatedDataFrame, which is a variant of a data.frame that the affy package requires for annotated data, and assign it to our rawcel object:

In [8]:

```
#this library contains the AnnotatedDataFrame:
library(Biobase)

phenoData(rawcel) <- AnnotatedDataFrame(data=pd)
```

1.3. Normalization

Next, we use frma (other option: SCAN) to normalize the data. We have 19 different samples so we have to normalize.

In [9]:

```
#load the frma library
library(frma)
```

```
Creating a generic function for ‘nchar’ from package ‘base’ in packa
ge ‘S4Vectors’
```

In [10]:

```
eset <- frma(rawcel, verbose=TRUE)
```

```
Background Correcting ...


Normalizing ...

Summarizing ...
```

In [11]:

```
save(eset, file="/mnt/storage/r0368280/jupyternotebooks/Examenopdracht/GSE27390.e
```

We now have an expression set loaded as an `ExpressionSet`. The eset objects contain the normalized data per probe and some experimental annotation.

However, the sample names are not very imformative:

In [12]:

```
sampleNames(eset)
```

```
'GSM677339.CEL.gz'  'GSM677340.CEL.gz'  'GSM677341.CEL.gz'
'GSM677342.CEL.gz'  'GSM677343.CEL.gz'  'GSM677344.CEL.gz'
'GSM677345.CEL.gz'  'GSM677346.CEL.gz'  'GSM677347.CEL.gz'
'GSM677348.CEL.gz'  'GSM677349.CEL.gz'  'GSM677350.CEL.gz'
'GSM677351.CEL.gz'  'GSM677352.CEL.gz'  'GSM677353.CEL.gz'
'GSM677354.CEL.gz'  'GSM677355.CEL.gz'  'GSM677356.CEL.gz'
'GSM677357.CEL.gz'
```

We'll fix this by assigning sample names manually:

In [13]:

```
eset.labels <- c("RA_1", "RA_2", "RA_3", "RA_4",
                 "RA_5", "RA_6", "RA_7", "RA_8",
                 "RA_9", "OA_1", "OA_2", "OA_3",
                 "OA_4", "OA_5", "OA_6", "OA_7", "OA_8", "OA_9", "OA_10")

sampleNames(eset) <- eset.labels
```

We can check the normalized expression level on a probe by probe basis:

In [14]:

```
probe.counts <- exprs(eset)
head (probe.counts)
```

	RA_1	RA_2	RA_3	RA_4	RA_5	RA_6	RA_7	RA_
1007_s_at	6.256753	6.725990	6.748881	6.627990	6.399535	6.302509	6.722324	7.06
1053_at	7.150950	6.810601	6.873038	6.775498	7.040456	6.528871	6.316756	6.75
117_at	8.380236	9.435561	9.194867	7.822858	7.818302	8.305048	8.929779	8.53
121_at	7.603966	7.499110	7.617828	7.500498	7.414912	7.698023	7.829497	7.99
1255_g_at	3.362468	3.676163	3.482711	3.383531	3.618627	3.573436	3.542474	3.62
1294_at	8.494344	8.906377	8.866384	8.540089	8.580514	8.495193	8.407988	8.54

The names in the first column (e.g. 1007_s_at) are the ID's of the probes. The names in the first row are the sample names.

1.4 Gene names

The probe names are not very informative. So, we will assign gene names to each probe. First we will load two extra libraries. The latter contains the annotation of our microarray platform (GeneChip Human Genome U133 Plus 2.0 Array (Affymetrix, Santa Clara, CA, USA))

In [15]:

```
library(annotate)
library(hgu133plus2.db)
```

```
Loading required package: AnnotationDbi
Loading required package: stats4
Loading required package: GenomeInfoDb
Loading required package: S4Vectors
Loading required package: IRanges
Loading required package: XML
Loading required package: org.Hs.eg.db
```

Now we will link the probe to its corresponding gene name using the `getSYMBOL` function

In [16]:

```
probe.names <- rownames(probe.counts)
gene.symbols <- getSYMBOL(probe.names, "hgu133plus2")
head(gene.symbols)
```

1007_s_at
'NA'
1053_at
'RFC2'
117_at
'HSPA6'
121_at
'PAX8'
1255_g_at
'GUCA1A'
1294_at
'NA'

Now we will perform a few extra steps to fix the raw expression table. The first step is to convert the raw expression to a data.frame with the gene symbols added as an extra column:

In [17]:

```
probe.counts.genesymbol <- as.data.frame(cbind(as.data.frame(probe.counts), gene.
head(probe.counts.genesymbol)
```

	RA_1	RA_2	RA_3	RA_4	RA_5	RA_6	RA_7	RA_
1007_s_at	6.256753	6.725990	6.748881	6.627990	6.399535	6.302509	6.722324	7.06
1053_at	7.150950	6.810601	6.873038	6.775498	7.040456	6.528871	6.316756	6.75
117_at	8.380236	9.435561	9.194867	7.822858	7.818302	8.305048	8.929779	8.53
121_at	7.603966	7.499110	7.617828	7.500498	7.414912	7.698023	7.829497	7.99
1255_g_at	3.362468	3.676163	3.482711	3.383531	3.618627	3.573436	3.542474	3.62
1294_at	8.494344	8.906377	8.866384	8.540089	8.580514	8.495193	8.407988	8.54

Then we will remove all probes for which we do not have a gene symbol, and ensure that the gene symbol column contains only strings:

In [18]:

```
probe.counts.genesymbol = probe.counts.genesymbol[complete.cases(probe.counts.gen
probe.counts.genesymbol$gene.symbols <- as.character(probe.counts.genesymbol$gene
```

Now, we have multiple probes per gene. There are many opinions on how to best deal with this. Differing

values may have biological meaning (splice variants) or may be noise. To keep things simple we will take the median expression here. To group the probes per gene, we will use a function from the excellent reshape2 package.

In [19]:

```
library(reshape2)
```

We will now `melt` and `dcast` the probe expression table into a gene count table.

In [20]:

```
m <- melt(probe.counts.genesymbol, id.vars = c("gene.symbols"))
```

We can check the contents of m, not how the data is now in "long" form:

In [21]:

```
head(m)
```

gene.symbols	variable	value
RFC2	RA_1	7.150950
HSPA6	RA_1	8.380236
PAX8	RA_1	7.603966
GUCA1A	RA_1	3.362468
THRA	RA_1	5.283923
PTPN21	RA_1	4.045119

And now we aggregate into a count table:

In [22]:

```
counts <- dcast(m, gene.symbols ~ variable, fun.aggregate=median)
```

If you check, there is a column called `gene.symbols`. We're turning these into the row names now:

In [23]:

```
rownames(counts) <- counts$gene.symbols
counts$gene.symbols <- NULL
```

Check the contents of the first five columns and rows:

In [24]:

```
counts[1:5,1:5]
```

	RA_1	RA_2	RA_3	RA_4	RA_5
A1BG	5.561428	5.285558	5.666805	5.854243	5.803750
A1BG-AS1	5.487833	5.243266	5.073737	5.219669	5.202499
A1CF	4.027440	3.778994	3.831084	3.691238	3.749375
A2M	5.616081	5.208048	5.849008	5.631941	5.597519
A2M-AS1	4.320441	5.195280	4.266207	5.430594	4.581357

Now we save this count table and analyse this in MeV (multiple array viewer).

In [25]:

```
write.table(counts, 'GSE27390.counts.tsv', sep="\t", quote=FALSE, col.names=NA)
```

We uploaded the 'GSE27390.counts.tsv' file into MeV and normalized the data. Subsequently we uploaded a gene cluster of genes that are known to be upregulated in rheumatoid arthritis (based on the literature). We see that the genes that are known to be upregulated are also upregulated (AREG,CXCR4,FKBP5,TPST1) in our microarray dataset, which means that our dataset is probably a good dataset.

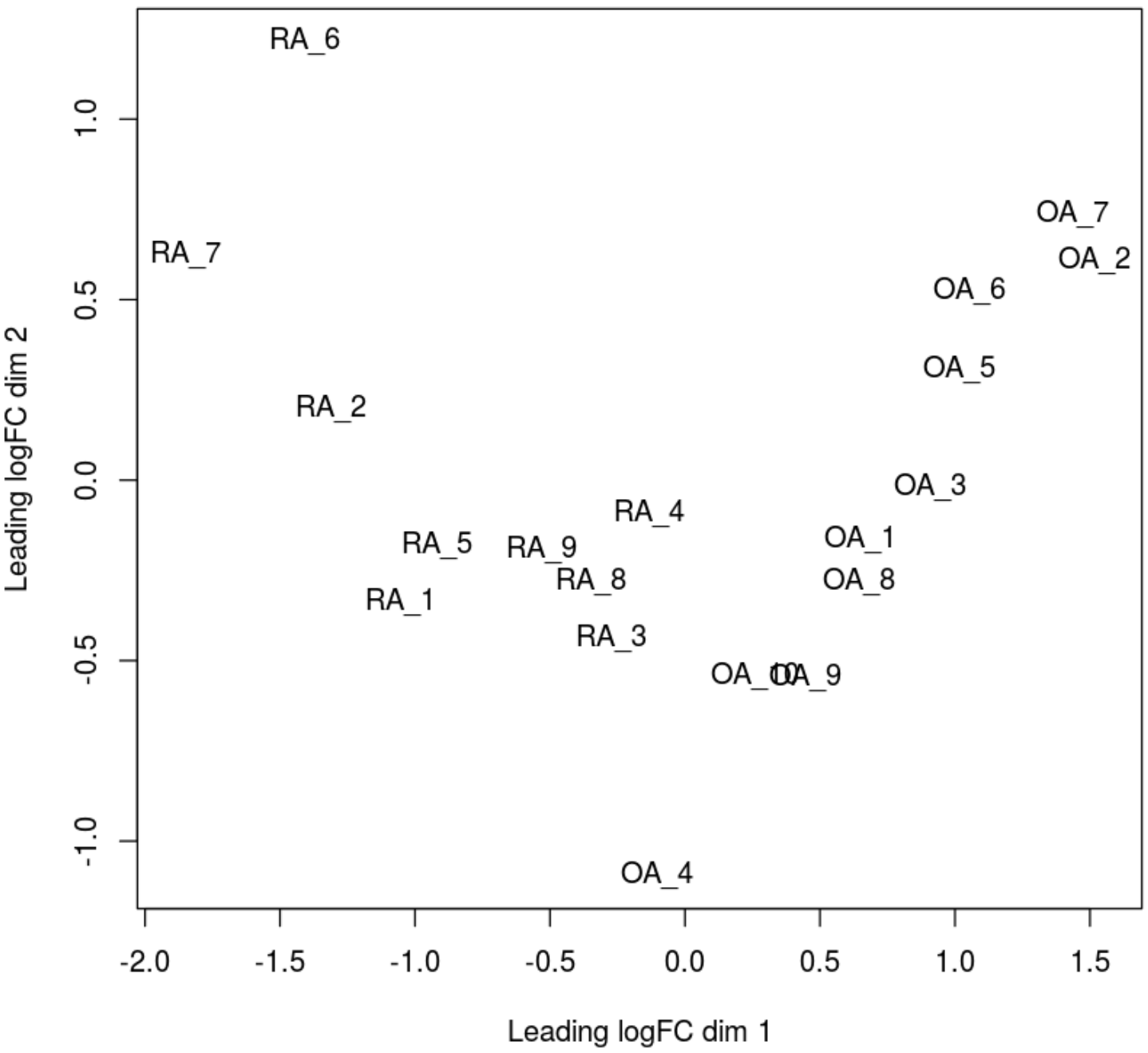


1.5 Quality control

It is very important to check the quality of your data. This can be done by the MDS tool. Basically, if two samples are similar, they are close together. In an ideal situation there is a lot of variaton between the conditions and little variation between the samples of each condition (replicates should be close together).

In [26]:

```
plotMDS(eset)
```



Samples RA_7, RA_6 and RA_2 lie further away from the other RA samples (RA_1, RA_3, RA_4, RA_5, RA_8 and RA_9, which form more or less a cluster). OA_4 lies further away from the other OA samples. Taken together there is only little variation between samples of the same condition and there is big variation between OA and RA as seen by the fact that OA and RA form two different clusters. This means that the quality of our data is good.

We now look at the genes that are already known to be upregulated in RA (same genes as checked in MeV) and genes of interest

In [27]:

```
goi <- c("AREG", "CXCR4", "FKBP5", "TPST1", "CXCL8", "IFNG", "CCL20")
counts[goi,]
```

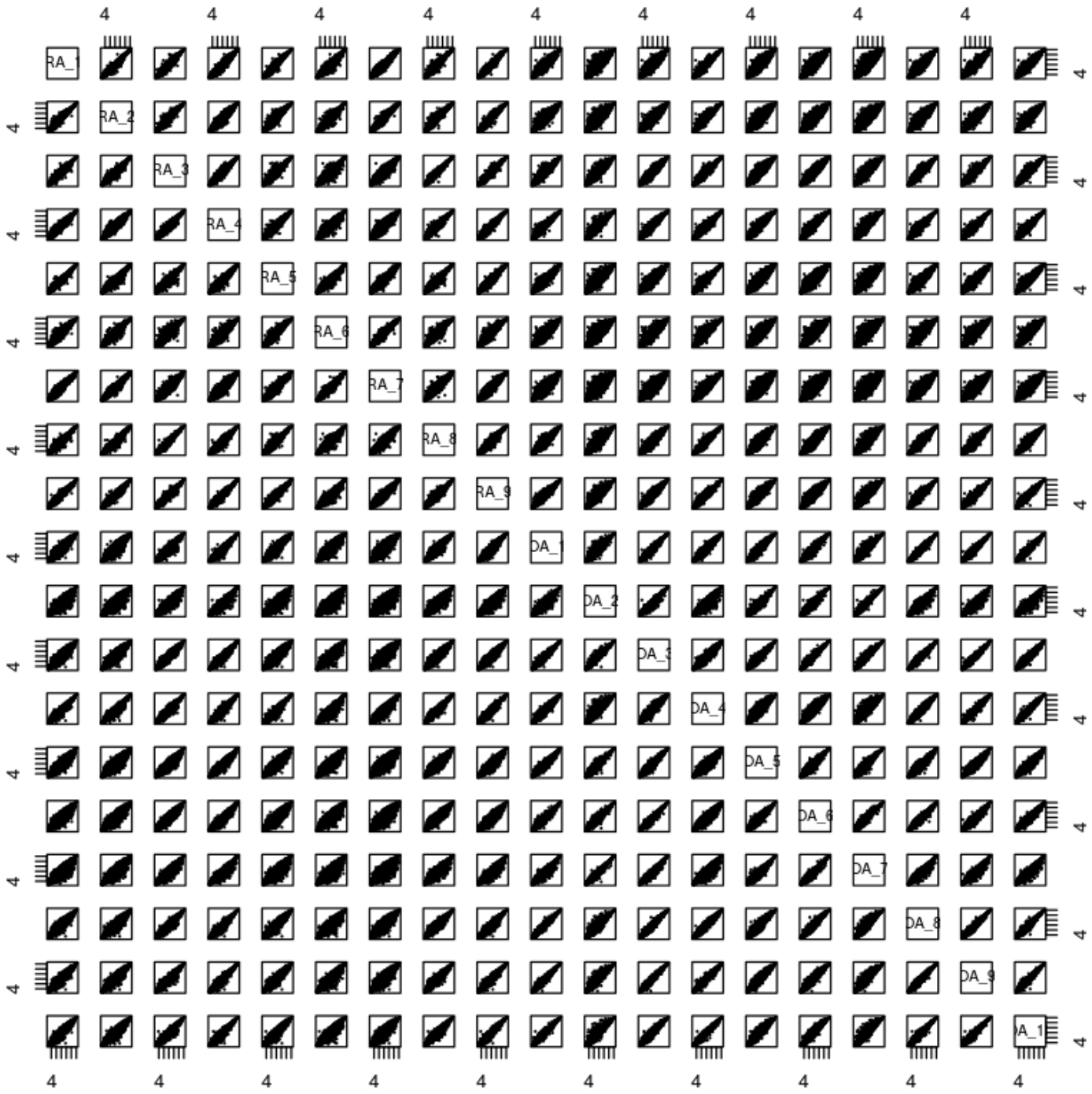
	RA_1	RA_2	RA_3	RA_4	RA_5	RA_6	RA_7
AREG	6.012298	6.014188	3.157876	4.340333	5.296133	7.273613	6.404100
CXCR4	12.390048	12.631803	11.185347	12.071187	12.182543	12.920272	12.758797
FKBP5	11.657906	11.985766	9.631348	11.408960	9.579385	10.771391	11.684850
TPST1	7.483627	8.570241	6.756177	6.503897	5.894087	7.213414	8.078765
CXCL8	12.167768	11.930142	10.464077	10.657236	11.917948	13.377361	12.386175
IFNG	7.081807	7.559429	6.375571	7.605922	8.165165	7.711508	7.929697
CCL20	6.230405	6.071133	3.988199	4.953617	6.432834	10.079633	7.528415

We chose to look at AREG, CXCR4, FKBP5 and TPST1 because these genes are already known to be upregulated in RA. CXCL8, IFNG and CCL20 are associated with inflammation and could therefore also be upregulated in RA. You see that IFNG has higher values in RA compared to OA. The same with CCL20 but there is alot of variation in the RA samples for this gene. For CXCL8 it is hard to tell.

Another nice way to look at your data is by creating a scatterplot of each gene comparing all samples against each other:

In [28]:

```
plot(data.frame(counts), cex=0.1)
```



Gene expression levels of the different samples are compared. Depending on the samples you compare with each other, the variation differs as seen by the black diagonal line. The wider the diagnonal, the more variation there is.

1.6 Differential expression

We use limma for differential expression analysis, but we already loaded the library.

The first step is to define the experimental conditions along which we want to do differential expression analaysis:

In [29]:

```
disease <- factor(c('RA', 'RA', 'RA', 'RA','RA','RA','RA','RA','RA','OA','OA','OA
                  levels=c('RA', 'OA'))
disease
```

RA RA RA RA RA RA RA RA RA OA OA OA OA OA OA
OA OA OA OA

Limma uses a linear model in the background, to this end we'll build a model matrix:

In [30]:

```
design <- model.matrix(~disease)
design
```

	(Intercept)	diseaseOA
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	1	0
8	1	0
9	1	0
10	1	1
11	1	1
12	1	1
13	1	1
14	1	1
15	1	1
16	1	1
17	1	1
18	1	1
19	1	1

In [31]:

```
fit <- eBayes(lmFit(counts, design))
```

lmFit is a `limma` function that fits a linear model and `eBayes` calculates probabilities

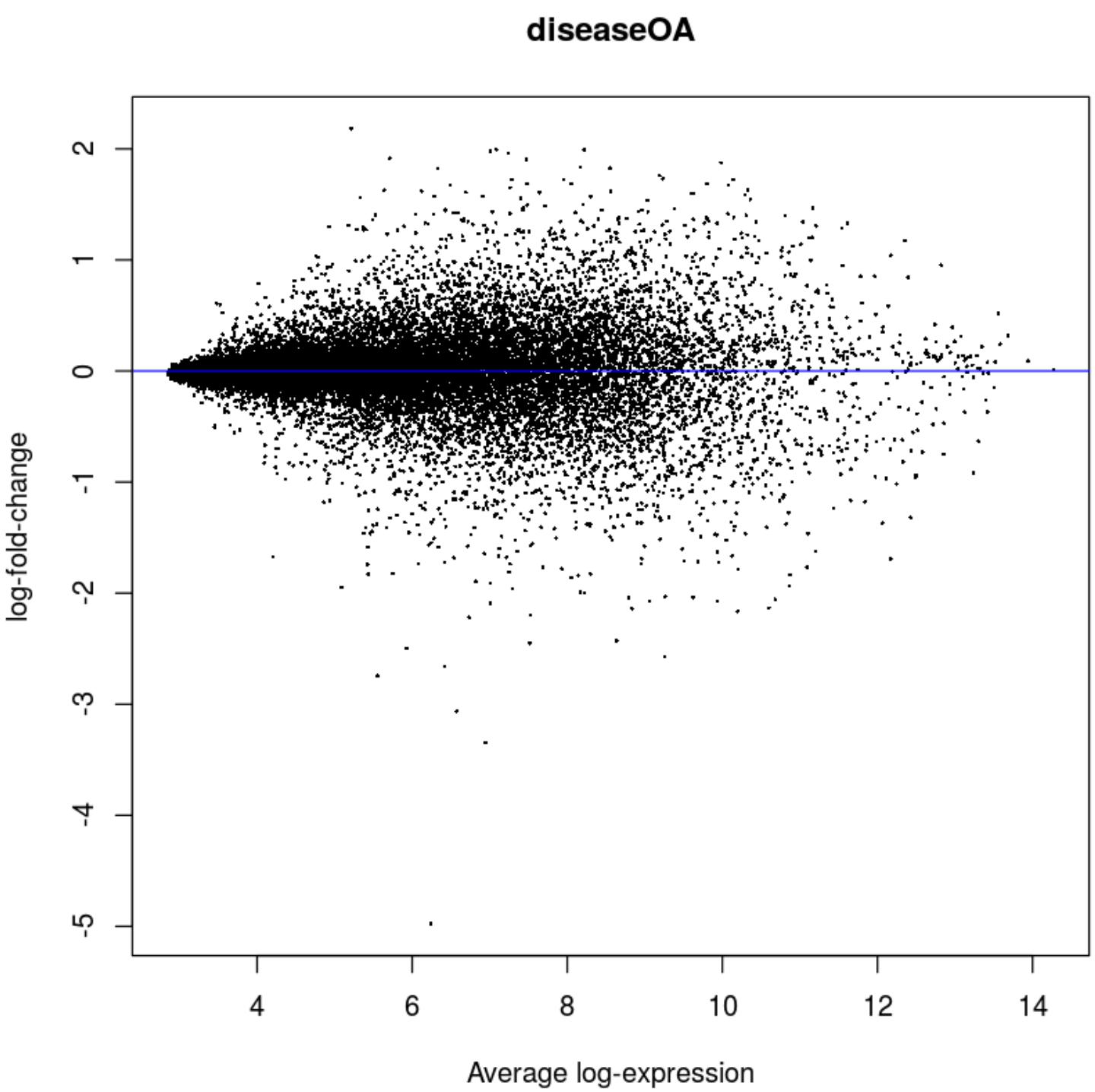
lmFit is a `limma` function that fits a linear model and `eBayes` calculates probabilities.

1.7 MA plot

An MA plot is a very common plot in differential expression analysis for the visualisation of genomic data. It shows the log ratio change of expression (M) vs. the average expression (A). It quickly gives you insight into the spread of differential expression.

In [32]:

```
limma::plotMA(fit, 2)
abline(0,0,col='blue')
```



On this MA plot you can see the differential expression of the genes in OA versus RA. The dots on the blue line represent genes that have a similar expression level in RA and in OA. Genes that are located above the blue line, are upregulated in OA compared to RA. Genes that are located under the blue line are downregulated in OA compared to RA.

The spread of differential expression is acceptable.

Now we extract the top differentially expressed genes using the `topTable` function. We choose coef=2 because we want the results from the disease (RA or OA) and not from the intercept.

In [33]:

```
topTable(fit, coef=2)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
GADD45B	-2.0389968	8.791497	-12.687179	4.488586e-11	9.151777e-07	15.057639
RABGEF1	-1.3627221	6.251218	-11.351747	3.251167e-10	3.314402e-06	13.303176
LOC728445	-1.4636083	5.456161	-10.619482	1.038041e-09	7.054876e-06	12.253016
ZFAND5	-3.0618375	6.573636	-9.658854	5.204406e-09	2.652816e-05	10.771714
SLC51A	-1.0991059	5.381464	-9.516096	6.674131e-09	2.712161e-05	10.541019
CH25H	-4.9774415	6.238996	-9.383309	8.430055e-09	2.712161e-05	10.323907
DDX50	-1.5468734	6.245126	-9.272378	1.026325e-08	2.712161e-05	10.140643
PLK2	-2.7439326	5.552549	-9.252057	1.064167e-08	2.712161e-05	10.106885
ZFAS1	-0.6359515	10.384643	-9.127480	1.330135e-08	2.876434e-05	9.898655
CCNL1	-1.5773230	8.996707	-9.038366	1.562120e-08	2.876434e-05	9.748350

NOTE: in the used commands we compared OA versus RA, so if the logFC (log fold change) is negative the gene is downregulated in OA compared to RA. This also means that the gene is upregulated in RA compared to OA.

In this table, all 10 genes are downregulated in OA compared to RA, which means that these genes are upregulated in RA. This upregulation is significant because the adjusted pvalue is less than 0.05 (which is an arbitrary chosen value).

1.8 Correct for multiple testing

Because we have so many genes we have to correct for multiple testing and this can be done through the Benjamin Hochberg (BH) method. We select again the second colum, because we are interested in the type of disease (RA or OA) and we select all the genes for which there is less than 5 % chance of false discovery.

In [34]:

```
select = p.adjust(fit[,2]$p.value, method="BH") < 0.05
```

To test if this command works we randomly select some genes:

In [35]:

```
select[20:30]
```

```
    TRUE   FALSE   TRUE   FALSE   FALSE   TRUE   FALSE   FALSE   FALSE
    FALSE   FALSE
```

In [36]:

```
summary(select)
```

```
   Mode    FALSE    TRUE    NA's
logical   15556    4833       0
```

`Summary(select)` gives us an overview of how many genes are signficant genes. In our case, 4833 are significant if you take a significance cutt-off (adjusted p-value) of 0.05 and correct for multiple testing.

In the article they used q < 0.001 as the criterion for statistical significance. So, we will also use this criterion as FDR (false discovery rate) cutoff.

In [37]:

```
select0001 = p.adjust(fit[,2]$p.value, method="BH") < 0.001
```

In [38]:

```
summary(select0001)
```

```
   Mode    FALSE    TRUE    NA's
logical   19582     807       0
```

If we take a significance cut-off of 0.001 and correct for multiple testing by the BH method we find 807 differentially expressed genes. In the article they found 2674 differentially expressed genes.

1.9 Output data

We now convert the output data generated by the topTable function in a file named "GSE27390.tophits.tsv" for further analysis. 100000 is chosen as the number of genes to ensure all genes are included. Note that the topTable function also calculates an adjusted p value and uses the BH method.

In [39]:

```
top = topTable(fit, coef=2, number=100000)
```

In [40]:

```
write.table(top, "GSE27390.tophits.tsv", sep="\t", quote=FALSE, col.names=NA)
```

We check the output data by the head function to check if the output file looks as expected (you can also upload the tsv in excell to look at the data)

In [41]:

```
head(top)
```

	logFC	AveExpr	t	P.Value	adj.P.Val	B
GADD45B	-2.038997	8.791497	-12.687179	4.488586e-11	9.151777e-07	15.05764
RABGEF1	-1.362722	6.251218	-11.351747	3.251167e-10	3.314402e-06	13.30318
LOC728445	-1.463608	5.456161	-10.619482	1.038041e-09	7.054876e-06	12.25302
ZFAND5	-3.061838	6.573636	-9.658854	5.204406e-09	2.652816e-05	10.77171
SLC51A	-1.099106	5.381464	-9.516096	6.674131e-09	2.712161e-05	10.54102
CH25H	-4.977441	6.238996	-9.383309	8.430055e-09	2.712161e-05	10.32391

2. Ranking-based analysis with GORilla

We now have a file named "GSE27390.tophits.tsv" which is a list of the most differentially expressed genes. Subsequently we are going to rank these genes in two different ways, one time on most upregulated genes in RA and the other time on most downregulated in RA. To do this, we upload the tsv file in excel and rank the genes from most upregulated genes to less upregulated by ranking genes that have a logFoldchange of less than -1 and an adjusted p-value of less than 0.05 (logFC<-1 && p.adj<0.05).

For ranking the genes from most downregulated genes to less downregulated, the genes that have a logFoldchange of more than 1 and an adjusted p-value of less than 0.05 (logFC>1 && p.adj<0.05) are ranked.

The used formulas in excel are:

Upregulated: =ALS(EN(B2<-1; F2<0.05);1;0)

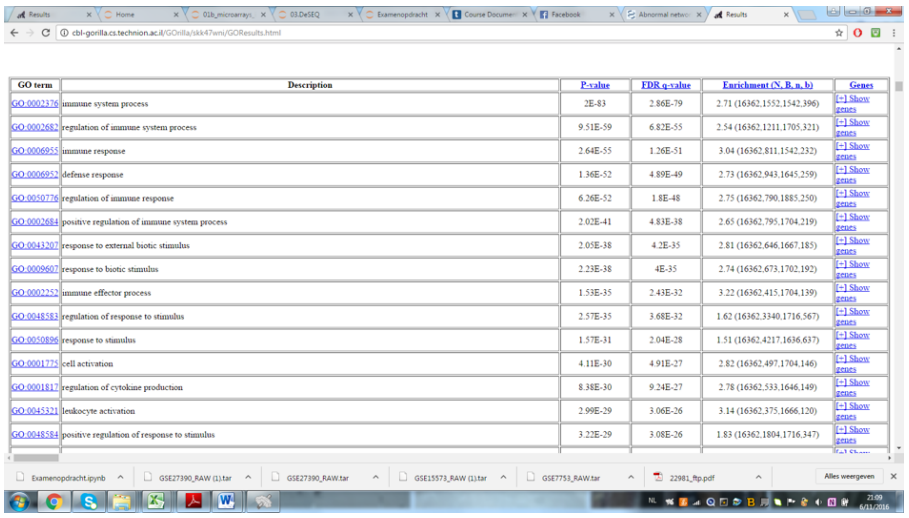
Downregulated: =ALS(EN(B2>1; F2<0.05);1;0)

In this formulas F2 stands for the adjusted p-value and B2 stands for the fold change (logFC).

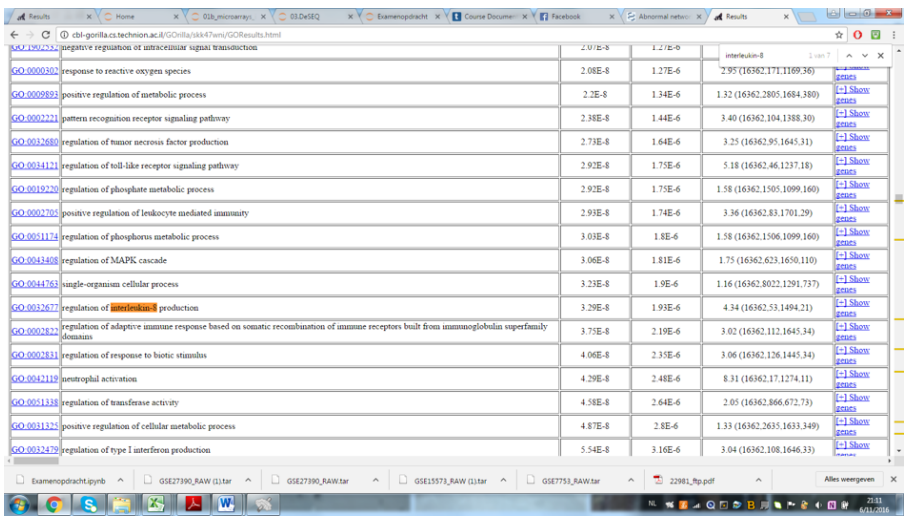
!! Notice again that the used commands of our microarray analysis are a comparison of OA versus RA so if the logFoldchange is negative the gene is downregulated in OA compared to RA which means that the gene is upregulated in RA compared to OA.

Subsequently the ranked list of most upregulated genes in RA and the ranked list of most downregulated in RA were uploaded in GORilla to obtain a gene signature.

Interpretation of the 2 pictures below: From the ranked list of most upregulated genes in RA that was uploaded in GORilla we see that the upregulated genes are involved in the regulation of immune system processes, regulation of response to stimuli, regulation of cytokine production, leukocyte activation, IL-8 production, T-cell activation etc. These are all processes involved in inflammation, which is expected in an inflammatory disease as rheumatoid arthritis.

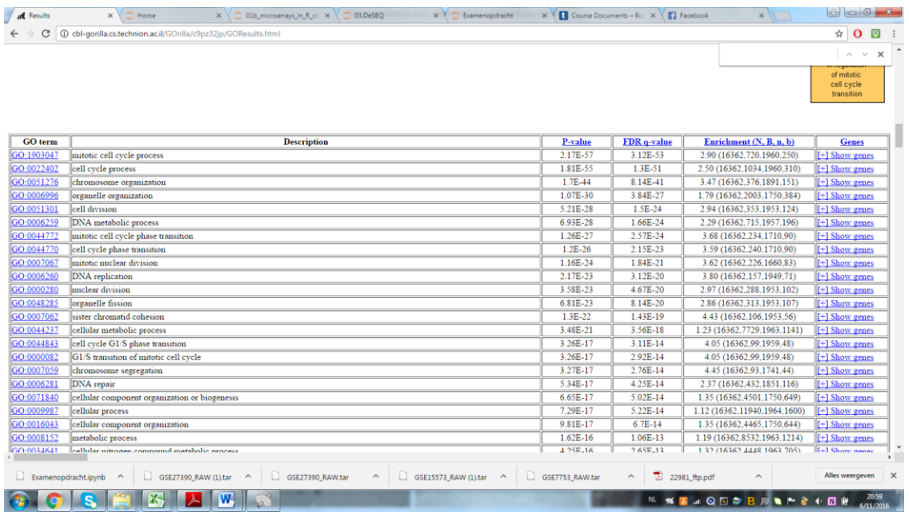


GO term	Description	P-value	FDR q-value	Enrichment (N, B, n, b)	Genes
GO:0002376	immune system process	2E-83	2.86E-79	2.71 (16362,1552,1542,396)	[+] Show genes
GO:0002682	regulation of immune system process	9.51E-59	6.82E-55	2.54 (16362,1211,1705,321)	[+] Show genes
GO:0006955	immune response	2.64E-55	1.26E-51	3.04 (16362,811,1542,232)	[+] Show genes
GO:0006952	defense response	1.36E-52	4.89E-49	2.73 (16362,943,1645,259)	[+] Show genes
GO:0050776	regulation of immune response	6.26E-52	1.8E-48	2.75 (16362,790,1885,250)	[+] Show genes
GO:0002684	positive regulation of immune system process	2.02E-41	4.83E-38	2.65 (16362,795,1704,219)	[+] Show genes
GO:0043207	response to external biotic stimulus	2.05E-38	4.2E-35	2.81 (16362,646,1667,185)	[+] Show genes
GO:0009607	response to biotic stimulus	2.23E-38	4E-35	2.74 (16362,673,1702,192)	[+] Show genes
GO:0002252	immune effector process	1.53E-35	2.43E-32	3.22 (16362,415,1704,139)	[+] Show genes
GO:0048583	regulation of response to stimulus	2.57E-35	3.68E-32	1.62 (16362,3340,1716,567)	[+] Show genes
GO:0050896	response to stimulus	1.57E-31	2.04E-28	1.51 (16362,4217,1636,637)	[+] Show genes
GO:0001775	cell activation	4.11E-30	4.91E-27	2.82 (16362,497,1704,146)	[+] Show genes
GO:0001817	regulation of cytokine production	8.38E-30	9.24E-27	2.78 (16362,533,1646,149)	[+] Show genes
GO:0045321	leukocyte activation	2.99E-29	3.06E-26	3.14 (16362,375,1666,120)	[+] Show genes
GO:0048584	positive regulation of response to stimulus	3.22E-29	3.08E-26	1.83 (16362,1804,1716,347)	[+] Show genes



GO term	Description	P-value	FDR q-value	Enrichment (N, B, n, b)	Genes
GO:0000302	response to reactive oxygen species	2.08E-8	1.27E-6	2.95 (16362,171,1169,36)	[+] Show genes
GO:0009891	positive regulation of metabolic process	2.2E-8	1.34E-6	1.32 (16362,2805,1684,380)	[+] Show genes
GO:0002221	pattern recognition receptor signaling pathway	2.38E-8	1.44E-6	3.40 (16362,104,1388,30)	[+] Show genes
GO:0032680	regulation of tumor necrosis factor production	2.73E-8	1.64E-6	3.25 (16362,95,1645,31)	[+] Show genes
GO:0034121	regulation of toll-like receptor signaling pathway	2.92E-8	1.75E-6	5.18 (16362,46,1237,18)	[+] Show genes
GO:0019220	regulation of phosphate metabolic process	2.92E-8	1.75E-6	1.58 (16362,1505,1099,160)	[+] Show genes
GO:0002705	positive regulation of leukocyte mediated immunity	2.93E-8	1.74E-6	3.36 (16362,83,1701,29)	[+] Show genes
GO:0051174	regulation of phosphorus metabolic process	3.03E-8	1.8E-6	1.58 (16362,1506,1099,160)	[+] Show genes
GO:0043408	regulation of MAPK cascade	3.06E-8	1.81E-6	1.75 (16362,623,1650,110)	[+] Show genes
GO:0044763	single-organism cellular process	3.23E-8	1.9E-6	1.16 (16362,8022,1291,737)	[+] Show genes
GO:0032677	regulation of interleukin-8 production	3.29E-8	1.93E-6	4.34 (16362,53,1494,21)	[+] Show genes
GO:0002822	regulation of adaptive immune response based on somatic recombination of immune receptors built from immunoglobulin superfamily domains	3.75E-8	2.19E-6	3.02 (16362,112,1645,34)	[+] Show genes
GO:0002831	regulation of response to biotic stimulus	4.06E-8	2.35E-6	3.06 (16362,126,1445,34)	[+] Show genes
GO:0042119	neutrophil activation	4.29E-8	2.48E-6	8.31 (16362,17,1274,11)	[+] Show genes
GO:0051338	regulation of transferase activity	4.58E-8	2.64E-6	2.05 (16362,866,672,73)	[+] Show genes
GO:0031325	positive regulation of cellular metabolic process	4.87E-8	2.8E-6	1.33 (16362,2635,1633,349)	[+] Show genes
GO:0032479	regulation of type I interferon production	5.54E-8	3.16E-6	3.04 (16362,108,1646,33)	[+] Show genes

Interpreation of the picture below: From the ranked list of most downregulated genes in RA that was uploaded in GORilla we see that the downregulated genes are involved in cell cycle processes more specifically cell division, DNA replication, chromosome organisation, G1/S transition, etc.



GO term	Description	P-value	FDR q-value	Enrichment (N, B, n, b)	Genes
GO:1903047	mitotic cell cycle process	2.17E-57	3.12E-53	2.90 (16362,720,1960,250)	[+] Show genes
GO:0022402	cell cycle process	1.81E-55	1.3E-51	2.50 (16362,1034,1960,310)	[+] Show genes
GO:0051276	chromosome organization	1.7E-44	8.14E-41	3.47 (16362,376,1891,151)	[+] Show genes
GO:0006996	organelle organization	1.07E-30	3.84E-27	1.79 (16362,2003,1750,384)	[+] Show genes
GO:0051301	cell division	5.21E-28	1.5E-24	2.94 (16362,353,1953,124)	[+] Show genes
GO:0006259	DNA metabolic process	6.93E-28	1.66E-24	2.29 (16362,715,1957,196)	[+] Show genes
GO:0044772	mitotic cell cycle phase transition	1.26E-27	2.57E-24	3.68 (16362,234,1710,90)	[+] Show genes
GO:0044770	cell cycle phase transition	1.2E-26	2.15E-23	3.59 (16362,240,1710,90)	[+] Show genes
GO:0007067	mitotic nuclear division	1.16E-24	1.84E-21	3.62 (16362,226,1660,83)	[+] Show genes
GO:0006260	DNA replication	2.17E-23	3.12E-20	3.80 (16362,157,1949,71)	[+] Show genes
GO:0000280	nuclear division	3.58E-23	4.67E-20	2.97 (16362,288,1953,102)	[+] Show genes
GO:0048285	organelle fission	6.81E-23	8.14E-20	2.86 (16362,313,1953,107)	[+] Show genes
GO:0007062	sister chromatid cohesion	1.3E-22	1.43E-19	4.43 (16362,106,1953,56)	[+] Show genes
GO:0044237	cellular metabolic process	3.48E-21	3.56E-18	1.23 (16362,7729,1963,1141)	[+] Show genes
GO:0044843	cell cycle G1/S phase transition	3.26E-17	3.11E-14	4.05 (16362,99,1959,48)	[+] Show genes
GO:0000082	G1/S transition of mitotic cell cycle	3.26E-17	2.92E-14	4.05 (16362,99,1959,48)	[+] Show genes
GO:0007059	chromosome segregation	3.27E-17	2.76E-14	4.45 (16362,93,1741,44)	[+] Show genes
GO:0006281	DNA repair	5.34E-17	4.25E-14	2.37 (16362,432,1851,116)	[+] Show genes
GO:0071840	cellular component organization or biogenesis	6.65E-17	5.02E-14	1.35 (16362,4501,1750,649)	[+] Show genes
GO:0009987	cellular process	7.29E-17	5.22E-14	1.12 (16362,11940,1964,1600)	[+] Show genes
GO:0016043	cellular component organization	9.81E-17	6.7E-14	1.35 (16362,4465,1750,644)	[+] Show genes
GO:0008152	metabolic process	1.62E-16	1.06E-13	1.19 (16362,8532,1963,1214)	[+] Show genes

3. iRegulon analysis

The next step is to upload the ranked list of downregulated genes as well as upregulated genes in iRegulon. Are these downregulated and upregulated genes co-expressed by chance or because they are regulated by the same transcription factor (TF)? iRegulon detects the TFs, the targets and the motifs/tracks from a set of

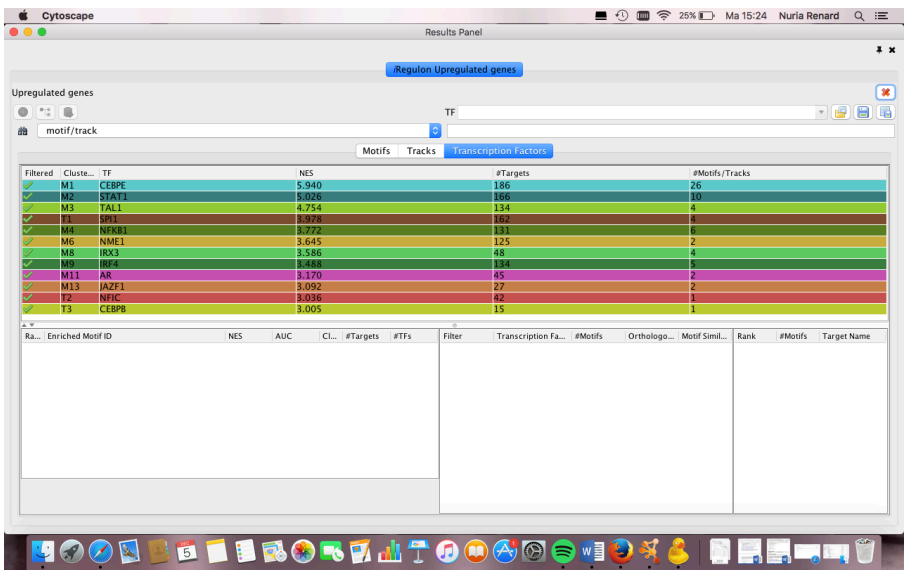
genes. The aim is to find the regulator of either the down or upgeregulated genes in RA, i.e. the transcription factor that regulates either the down or upregulated genes. Ideally, we would try to find the masterregulator of the genes and the additional regulators to establish a regulatory network but this is overreaching.

iRegulon analysis was done with the parameters left on standard settings. As a result we get a list with enriched motifs, together with a list of candidate transcription factors that can bind the motif. We obtain these results for the downregulated genes and for the upregulated genes.

3.1 Upregulated genes

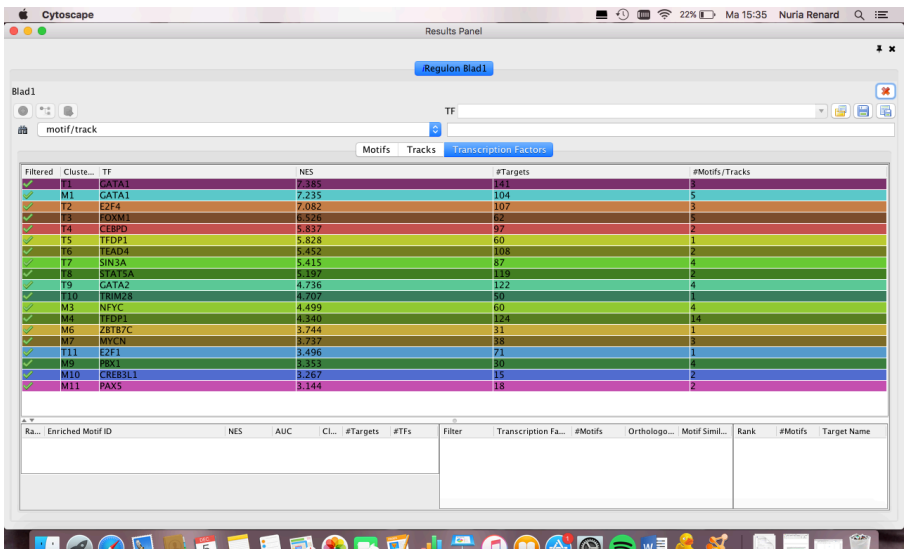
In the picture below you see the transcription factors detected by iRegulon analysis on the ranked list of upregulated genes. The top table shows the enriched TFs ranked by the maximal enrichment score of the motif or the maximal enrichment score for a given TF (=NES), presented with the number of merged targets (#Targets) found by numerous motifs (#Motifs).

CEBPE is the transcription factor with the highest NES score. It regulates 186 targets and these genes are detected by 26 motifs. The transcription factor with the fifth highest NES corresponds to NFKB1. It has 131 targets found by 6 motifs.



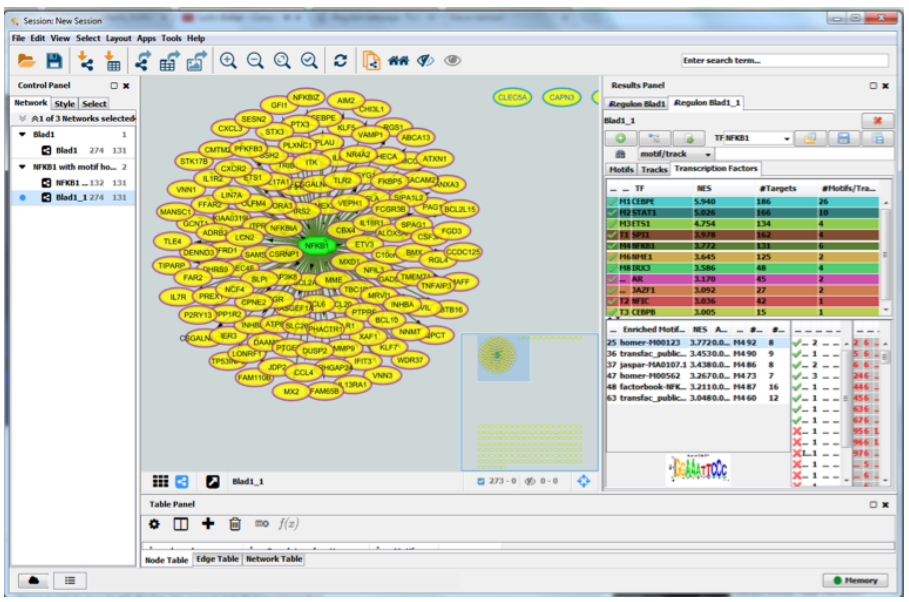
3.2 Downregulated genes

In the picture below you see the transcription factors detected by iRegulon analysis on the ranked list of downregulated genes. Several transcription factors are GATA1 (which has the highest NES), E2F4 and FOXM1.

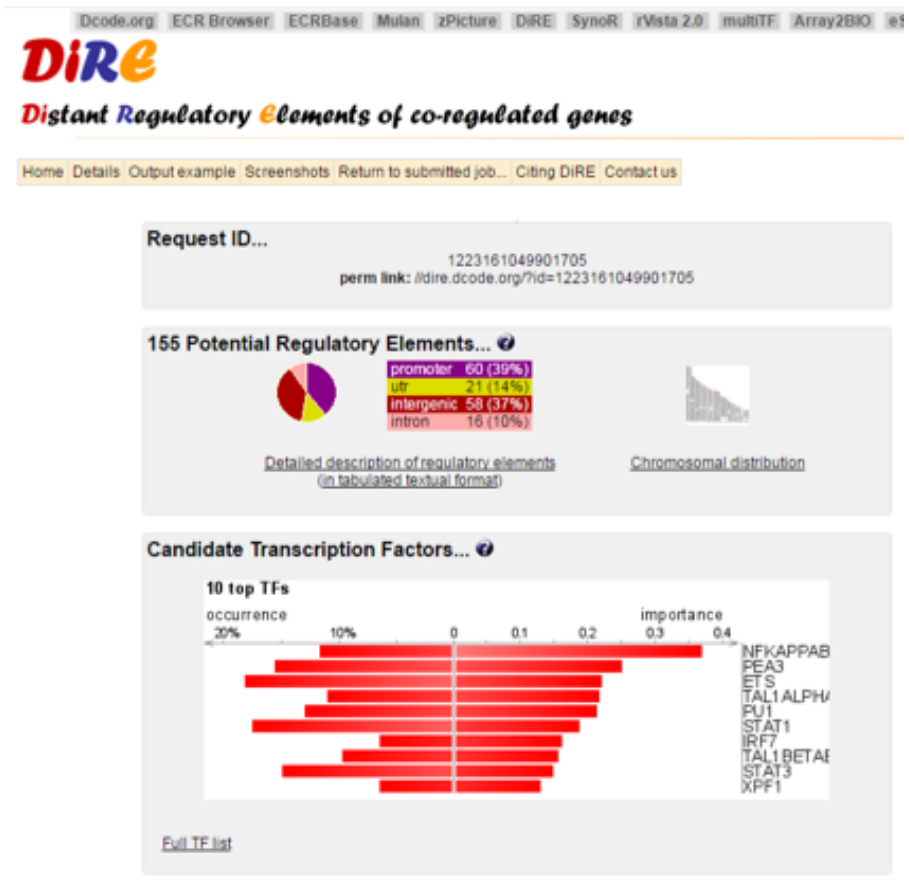


From these two lists we have to choose a transcription factor that could be the regulator of the differentially expressed genes in rheumatoid arthritis. Based on the literature and the known association of NFKB1 with inflammation we chose NFKB1 as a possible regulator of the upregulated genes in RA.

We visualised the connection of NFKB1 with its target genes in the picture below. This is a network from the co-expressed genes (i.e. upregulated genes) with the transcription factor NFKB1.

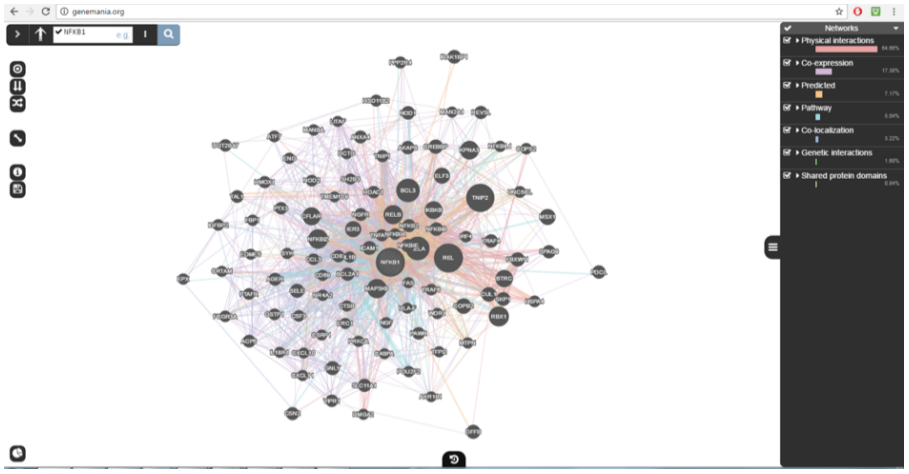


Another tool we can use for motif discovery is DIRE (identification of distant regulatory elements of co-regulated genes). We uploaded the upregulated genes in DIRE and we saw that NFKB1 is among the top TFs. This means that the NFKB1 recognition sequence is enriched in the promotors of the input genes; the input is enriched for direct targets of NFKB1.



4. GeneMANIA

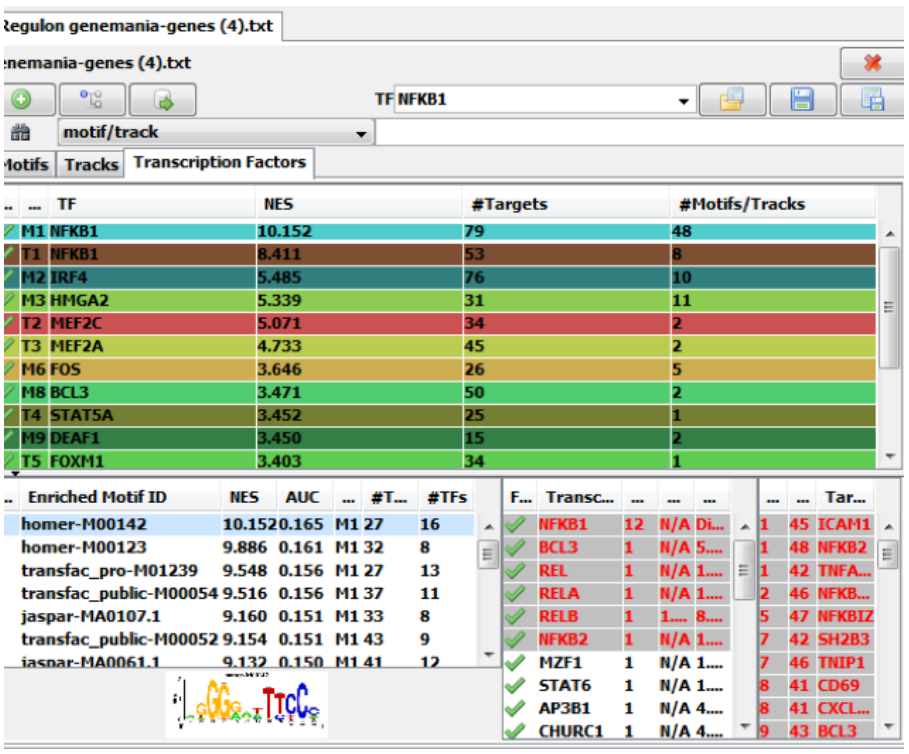
We now have the NFKB1 network that is solely based on co-expressed genes. We can also connect transcription factors with genes based on physical interactions, co-localization, genetic interactions etc. This can be done by GeneMANIA (max resultant gene was set on 100). We retrieved an association network for NFKB1 from GeneMANIA (picture below).



We now upload the list of genes generated by GeneMANIA in cytoscape and perform iRegulon analysis.

iRegulon analysis

We see that NFKB1 is still detected as a transcription factor. The direct targets are significantly enriched in the association network.



5. Chipseq analysis (other notebook: Examenopdracht_GROUP12_PART2_BASH)

The next question is if the transcription factor NFKB1 directly regulates the target genes. Does NFKB1 bind to the target genes (i.e. the upregulated genes in RA)? This can be investigated by use of Chipseq analysis. Because chipseq analysis is performed in Bash and microarray analysis in R we make a new notebook and continue working in that one.