Week_13

Hien Le

4/22/2022

Microarray data analysis

Question 1

- Microarrays relys on a pre-designed complement sequence detection probe, hence it is no able to detect all differential expresed genes. However, it is a pwerful tool of we are trying to detect novel transcripts/splice froms, go on an unbiased "fishing trip" for genes/biomarkers, detect extremelt rare transcripts, or look at changes in transcripts abundance that occur over a very wide dynamic ranges.
- RNA-seq provides several advantages over hybridization based approaches: RNA-seq has higher sensitivity for genes expressed either at low or very high and higher dynamic ranges of exression levels over which transcripts can be detected. However, upon the protocol of RNA-seq, the resutls will release more detail about bias in transcripts sequences, detected of sequence-level alterations and alternatic splicing events as well as the expression levels.

Therfore, the context of the rearch is improtant. For examples, if we want to detect only some small parts of the wider goald, then I guess microarray is not suitable. However, if we focuse on a hall picture of genes, microarray may be a good selection.

Question 2

Array manufactures originally designed single nuclotide polymorhism (SNP) arrays to genotype human DNA at thousands of SNPs across the genome simutaneously. SNP array genotyping can determine loss of heterozygosity, genomic copy number changes and DNA methylation alterations of cancer cells.

- It could be analyzed using a microarray if we design the specific probes. In thise case, specific probes must be contained all nucleotide variants.

The limited point in designing specific probes: it is to require the research, reports, or knowledge before we want to detect the single nucleotide varation.

Question 3

The differences between affymetrix genechip and cDNA microarray: probe preparation, colors, gene representation, probe length, density

- cDNA: probes are cDNA fragments, usally amplified by PCR and spotted by robots. The measure relative intensity, especially tow-color. In microarray, one probe could present one gens. The length of probe is long, hundreds to 1K bp. In one time to measure, maximum of 15000 probes.
- Affymetrix genechip: probes are short oligos synthesized using a photolithographic approach. The measure relates intensity, however, it contains only one color. In genechip. 11-16 probes pairs per genes.

Probe length is often short, 25 mers. In one time to measure, density contains a large of genes and probes: around 400000 probes.

Microarray dara proprocessing and normalization

Task 1

```
library("affy")
path = "/student_data/BBT.BI.203_2022/data/ex13/CEL_files"
file = list.files(path, full.names = TRUE)
GSM = ReadAffy(filenames = file)
```

Task 2

```
GSM_rma =rma(GSM)
name = row.names(GSM_rma)
```

Task 3

```
library("mouse4302.db")
library(help = "mouse4302.db")

x = mouse4302SYMBOL

# Convert to a list
xx = as.list(x[name])

# filter out any probes for which a matching gen symbol cannot be found
names_vector = c()
for(i in 1:length(name)){
  probe = name[i]
  gene = xx[[probe]]
  names_vector = c(names_vector,gene)
}
sum(is.na(names_vector))
filter_gene = names_vector[!is.na(names_vector)]
GSM_filter = GSM_rma[!is.na(names_vector),]
```

Task 4

```
expresso_obj = expresso(GSM, bgcorrect.method="rma", normalize.method="quantiles",pmcorrect.method="pmo
expresso_obj_filter = expresso_obj[!is.na(names_vector),]
```

Task 5

```
# Step 1: calculate IQR of each row of exprs(expresso_obj_filter)
exprs_log = exprs(expresso_obj_filter)
row.names(exprs_log) = filter_gene
row_IQR = apply(exprs_log, 1, IQR)

# Step 2: pick one row per gene, based on highest IQR

row_IQR_values <- c()
for(i in 1:length(filter_gene)){
  row_IQR_values <- c(row_IQR_values, row_IQR[[filter_gene[i]]])
}

row_IQR_values

filter_gene_unique = unique(filter_gene)
indexes <- c()
for (i in 1:length(filter_gene_unique)){
  duplicate_indexs_vector <- which(filter_gene == filter_gene_unique[i])

  values <- c()
  for( k in 1: length(duplicate_indexs_vector)){
    values = c(values,row_IQR_values[k])
  }
  indexes = c(indexes, duplicate_indexs_vector[which.max(values)])
}

# Step 3: filter expresso_obj_filter based on Step 2, should be ExpressionSet, take for further analysi
#filter_gene_unique Chua list unique gene name
filter_gene_unique
# indexes contains position(index) of row IQR with the highest values
indexes

#Filter exprs_log for taking unique name + max IQR
exprs_log_filter_normalized = exprs_log[indexes,]
expresso_obj_filter_normalized  = expresso_obj_filter[indexes,]
```

Task 6

```
expresso_obj_filter_normalized
```

Task 7

Plot two boxplots: one of the log2-transformed raw (not normalized) data matrix and one of the log2-transformed normalized data.

```
GSM_rma =rma(GSM)
name = row.names(GSM_rma)
#log2-transformed raw data
exprs_log
```

```r
boxplot(exprs_log, main = "Before normalization")
#log2-transformed normalized
exprs_log_filter_normalized
boxplot(exprs_log_filter_normalized, main = "After normalization")
```

```
## Warning: replacing previous import 'AnnotationDbi::tail' by 'utils::tail' when
## loading 'mouse4302cdf'
```

```
## Warning: replacing previous import 'AnnotationDbi::head' by 'utils::head' when
## loading 'mouse4302cdf'
```

```
## Background correcting
## Normalizing
## Calculating Expression
```

```
## background correction: rma
## normalization: quantiles
## PM/MM correction : pmonly
## expression values: medianpolish
## background correcting...done.
## normalizing...done.
## 45101 ids to be processed
## |                    |
## |####################|
```

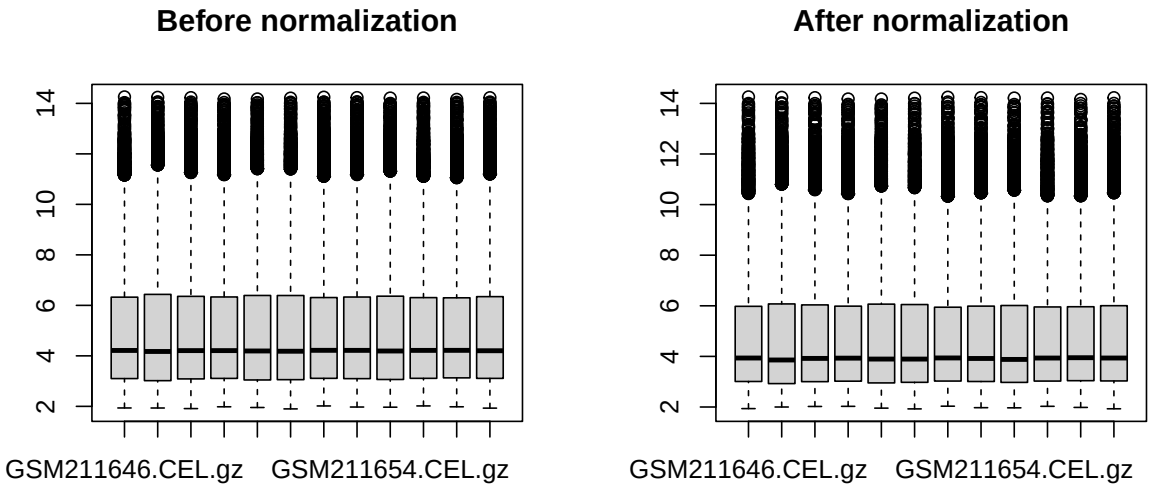

Figure 1: Boxplots of before and after normalization filter

Task 8

Make sure you plot the histogram using log2-transformed data.

```r
hist(exprs_log_filter_normalized)
```

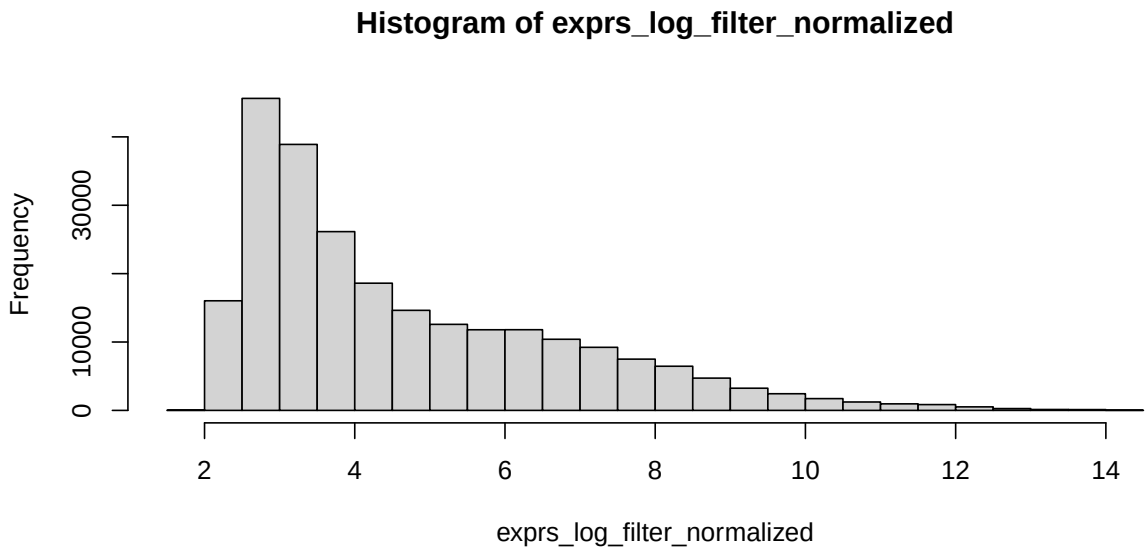

Figure 2: Histograme after normalization filter

Questions

1.

How many probes names can be matched to gene identifiers in task 3/4? Task 3: 7637 Task 4: expresso_obj_filter 37464

How many genes have more than on probe corresponding to them? 6637

2.

View and compare the normalized gene expression values of the two pre-processing and normalization methods from Tasks 3 and 4. With my opinion, the number of genes are same and i do not notice the significant difference of normalization values. I think that the reasone could be the normalization method to quantile. In addition, the reference is same in both cases.

3.

To quantile normalize two or more distributions to each other, without a reference distribution, sort as before, then set to the average (usually, arithmetical mean) of the distributions.

I think that we can not notice the difference by observing the boxes, but we can notice by dots of the plot.

4.

Too many gens have low-intensity values because the limitation of microarray experiments. In the method of microarray, the low intensity signals are retained and inaccurate signals are eliminated in the mean and median correlation in microarray analysis. Thus, the final results in here contain too much of low expression genes.

The change is not significant because

#Differential expression analysis for microarrays

##Task 1

```
library(limma)
```

#Task 2

```
#'expressoResult' is the result from Task 5, containing only probes with matching gene identifiers and
# 'sampleNames' should be a vector of the names of the mouse samples (GSM211646, GSM211647, etc.)
# 'geneNames' should be the matching gene identifiers for the probes you found in Task 5
sampleNames <- colnames(exprs_log_filter_normalized)
geneNames <- rownames(exprs_log_filter_normalized)

phenoData <- as(data.frame(sample = 1:length(sampleNames), row.names = sampleNames),"AnnotatedDataFrame
featureData <- as(data.frame(genes = 1:length(geneNames),  row.names = geneNames), "AnnotatedDataFrame"
phenoData(expresso_obj_filter_normalized) = phenoData
featureData(expresso_obj_filter_normalized) = featureData
```

#Task 3

```
design = model.matrix(~ 0+factor(c(1,1,1,1,1,1,2,2,2,2,2,2)))
eset = exprs_log_filter_normalized
fit = lmFit(eset, design)
fit = eBayes(fit)
colnames(design) = c("LF", "HF")
cont.matrix = makeContrasts(LFvsHF=LF-HF, levels=design)
fit2 = contrasts.fit(fit, cont.matrix)
fit2 = eBayes(fit2)
topTable(fit2, adjust="BH")
```

Task 4

Make a table and an expression heatmap of top 10 differntially expressed genes.

```
table = topTable(fit2, adjust="BH")
top_gene = row.names(table)
exprs_log_filter_normalized
sample_exprs_log_10 = exprs_log_filter_normalized[row.names(exprs_log_filter_normalized) %in%top_gene,
heatmap(as.matrix(sample_exprs_log_10))
```

```
## Warning in contrasts.fit(fit, cont.matrix): row names of contrasts don't match
## col names of coefficients
```

Questions

1.

The ten genes are: Thrsp, Tkt, Fasn, Scd1, Acly1, Tbx1, Hoxd9, Pcp4l1, Hoxa11, Arxes2

Among them, Fasn could be one of the potential candidates for the treatment because it encodes the protein fatty acid synthase.

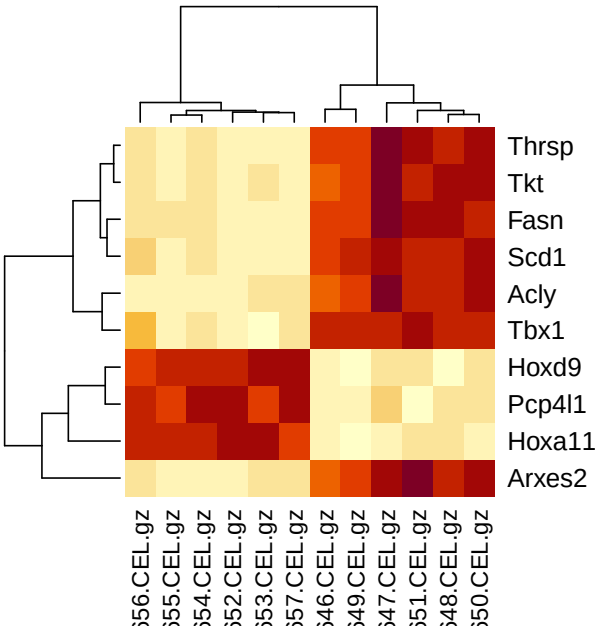

Figure 3: Heatmap of top 10 genes

Gene set enrichment analysis (GSEA)

Task 1

```
library("gage")

#Human
kegg_symbols = readList("/student_data/BBT.BI.203_2022/data/ex13/kegg_symbols.gmt")

#Check the gene ID types in the loaded list- do these match the gene identifiers in our microarray data
# -> NO. i did not notice any match in herer
```

Task 2

```
#Recall that our dataset consists of mouse samples.
mouse_samples = exprs_log_filter_normalized

#Map the mouse gene names to human gene names using the annotations from file humanMouseHomologs2.txt.
map_human_mouse = read.table("/student_data/BBT.BI.203_2022/data/ex13/humanMouseHomologs2.txt", header =

names_mouse_samples = row.names(mouse_samples) # mouse gene names
 # mapping mouse_gene_name voi map_human_mouse
mouse_samples_filter = mouse_samples[names_mouse_samples %in% map_human_mouse$Mouse_Symbol,]

#Take row names from mouse_samples_filter
names_mouse_samples_filter = row.names(mouse_samples_filter)
human_samples_names <- c()
for(i in 1: length(names_mouse_samples_filter)){
```

```
  human_name_item = map_human_mouse[map_human_mouse$Mouse_Symbol == names_mouse_samples_filter[i],]$Huma
  human_samples_names = c(human_samples_names, human_name_item)
}
human_samples_names_unique = unique(human_samples_names)
human_samples_filter = mouse_samples_filter
row.names(human_samples_filter) <- human_samples_names_unique
human_samples_filter
```

##Task 3

```
#we can perform GSEA using the gage() function.
library(gage)
#help(gage)
gage_result = gage(as.matrix(human_samples_filter), gsets=kegg_symbols, same.dir=F)
pVal = gage_result$greater[,3]
pVal_filterNA = na.omit(pVal)
```

Questions

1. What is the leading edge subset in GSEA?

The leading-edge subset can be interpreted as the core group of genes that accounts for the gene set's enrichment signal. This visualizer displays four graphs that help us visualize the overlap between the leading-edge subsets of selected gene set. This subset contains the genes that appear in the ranked gene set list between zero and the point at which the running sum statistic reaches its greatest deviation from zero.