Processing microarray data with Bioconductor

Statistical analysis of gene expression data with R and Bioconductor
University of Copenhagen

Copenhagen Biocenter

Laurent Gautier[1],[2]

August 17-21 2009

Contents

1 Affymetrix

1.1 Processing data

Processing data

1. Download and install the bioconductor packages
   - `ecolicdf`
   - `ecoliLeucine`
   - `arrayQualityMetrics`
2. Load `ecoliLeucine` into an R session
3. Control the data quality with `arrayQualityMetrics()` (package `arrayQualityMetrics`). What can you say about your arrays ? Should you perform a transformation to your data ?
4. Perform an RMA normalization of the data (using the function `rma()`).
5. Use again `arrayQualityMetrics`, this time with the object resulting from RMA normalization. Compare the outcome with the one obtained previously. What is preprocessing (here RMA) achieving ?
6. MAS5.0 is the historical way Affymetrix used to perform processing on it's arrays. Perform it using `affy::mas5()` and check the quality.
7. Using `affy::expresso`, perform a preprocessing bundle of your choosing (background correction, normalization, probe summaries)

8. *optional:* How would you save the `ExpressionSet` resulting from the RMA normalization into a file to be read by other programs ?

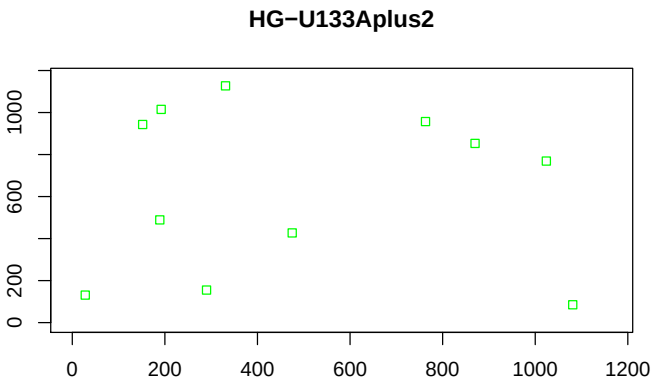
```
> library(ecoliLeucine)
> data(ecoliLeucine)

> library(arrayQualityMetrics)

> arrayQualityMetrics(ecoliLeucine,
+                     "quality/raw")
> browseURL("quality/raw/arrayQM.html")

> arrayQualityMetrics(ecoliLeucine,
+                     "quality/raw",
+                     do.logtransform = TRUE)
> browseURL("quality/raw/arrayQM.html")

> eset_rma <- rma(ecoliLeucine)
> arrayQualityMetrics(eset_rma,
+                     "quality/rma")
> browseURL("quality/rma/arrayQM.html")

> eset_mas5 <- mas5(ecoliLeucine)
> arrayQualityMetrics(eset_mas5,
+                     "quality/mas5")
> browseURL("quality/mas5/arrayQM.html")
> rm(ecoliLeucine)
```

1.2 Probeset mappings

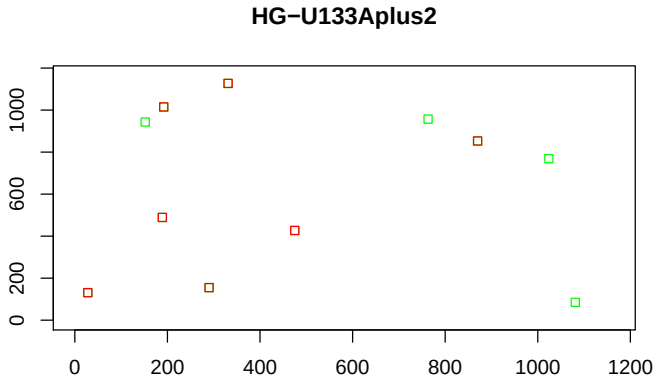
Probeset mappings

- Load the packages `altcdfenvs`, `hgu133plus2cdf`, `hgu133plus2.db`
- Using `altcdfenvs::wrapCdfEnvAffy()`, and knowing that the array size is 1164x1164, create an instance of class *CdfEnvAffy*.
- Extract indexes for the probeset *202376_at*
- Plot the position of the indexes on the array (use `plot()` and `affy::plotLocation()`)
- What is the *gene symbol* associated with the probeset *202376_at* ?
- Now load the packages `hgu133plus2hsrefseqcdf`, `hgu133plus2hsrefseq.db`
- Identify the RefSeq ID corresponding to the gene symbol *SERPINA3*, and fetch the probe indexes for that RefSeq ID in the the mapping `hgu133plus2hsrefseqcdf` (use `altcdfenvs::wrapCdfEnvAffy()` if you want to proceed like earlier). Do the original mapping and the refseq mapping differ ?

```
> library(altcdfenvs)
> library("hgu133plus2cdf")
> library("hgu133plus2.db")
> affycdf_hgu133plus2 <-
+   wrapCdfEnvAffy(hgu133plus2cdf, 1164, 1164,
+                  "HG-U133Aplus2")
> probe_i <- indexProbes(affycdf_hgu133plus2,
+                        "pm", "202376_at")[[1]]
> xy <- index2xy(affycdf_hgu133plus2, probe_i)
> plot(affycdf_hgu133plus2)
> plotLocation(xy)
> genesymbol <- hgu133plus2SYMBOL[["202376_at"]]
```




```
> library("hgu133plus2hsrefseqcdf")
> library(hgu133plus2hsrefseq.db)
> rseqcdf_hgu133plus2 <- wrapCdfEnvAffy(hgu133plus2hsrefseqcdf, 1164, 1164,
+                                       "HG-U133Aplus2")
> rseq_id <- revmap(hgu133plus2hsrefseqSYMBOL)[["SERPINA3"]]
> rseq_probe_i <- indexProbes(rseqcdf_hgu133plus2, "pm", rseq_id)[[1]]
> plot(affycdf_hgu133plus2)
> plotLocation(xy)
> plotLocation(index2xy(rseqcdf_hgu133plus2, rseq_probe_i),
+              col = "red")
```


HG−U133Aplus2
0
200
600
1000
0
200
400
600
800
1000
1200

2 Working with third-party data

Third-party data

1. Download the GEO dataset GSE12105
2. Control the data quality with `arrayQualityMetrics()` (package `arrayQualityMetrics`). What can you say about your arrays ? Should a transformation be applied to assess quality ?
3. Perform a quantile normalization of the data (using one of the functions `preprocessCore::normalize.quantiles`, `limma::normalizeQuantiles`, `affy::normalize.qspline`).
4. Plot the densities (using `affy::plotDensity` for example).
5. Build an `ExpressionSet` with the normalized intensities

```
> eset_GSE12205 <- getGEO("GSE12105")

> eset <- eset_GSE12105
> # log-transform on negative values is not possible
> exprs(eset)[exprs(eset) <= 0] <-
+   min(exprs(eset)[exprs(eset) > 0])

> arrayQualityMetrics(foo, outdir = "aqm",
+                     do.logtransform=TRUE)

> library(preprocessCore)
> exprs_n <- normalize.quantiles(exprs(eset))

> esetn <- new("ExpressionSet",
+              exprs = exprs_n,
+              phenoData = phenoData(eset),
+              featureData = featureData(eset))
```

sessionInfo()

- R version 2.9.1 Patched (2009-08-09 r49124), `i686-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8;LC_NUMERIC=C;LC_TIME=en_US.UTF-8;LC_COLLATE=en_US.UTF-8;LC_MONETARY=C;LC_MESSAGES=en_US.UTF-8;LC_PAPER=en_US.UTF-8;LC_NAME=C;LC_ADDRESS=C;LC_TEL 8;LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: affy 1.22.0, affyio 1.12.0, affyPLM 1.20.0, altcdfenvs 2.6.0, AnnotationDbi 1.6.1, arrayQualityMetrics 2.2.2, Biobase 2.4.1, Biostrings 2.12.8, DBI 0.2-4, ecolicdf 2.4.0, ecoliLeucine 1.2.3, gcrma 2.16.0, graph 1.22.2, hgu133plus2cdf 2.4.0, hgu133plus2.db 2.2.11, hgu133plus2hsrefseqcdf 12.0.0, hgu133plus2hsrefseq.db 12.0.0, hypergraph 1.16.0, IRanges 1.2.3, makecdfenv 1.22.0, matchprobes 1.16.0, preprocessCore 1.6.0, RSQLite 0.7-1, startupmsg 0.6, SweaveListingUtils 0.3.3
- Loaded via a namespace (and not attached): annotate 1.22.0, beadarray 1.12.0, genefilter 1.24.2, grid 2.9.1, hwriter 1.1, lattice 0.17-25, latticeExtra 0.6-1, limma 2.18.2, marray 1.22.0, RColorBrewer 1.0-2, simpleaffy 2.20.0, splines 2.9.1, stats4 2.9.1, survival 2.35-4, tools 2.9.1, vsn 3.12.0, xtable 1.5-5