An introduction to Microarray data analysis in using Bioconductor

Advanced Transcriptomics at SBF

Ibrahim Emam
Gabriella Rustici, PhD
Functional Genomics Team
European Bioinformatics Institute

EMBL-EBI

Lecture outline

General Introduction to R and Bioconductor
- What is R?
- What is Bioconductor?
- BioC packages

Data pre-processing using *limma*
- Reading data in
- Diagnostic plots
- Background correction
- Normalization

Data quality assessment and analysis
- Importing data from ArrayExpress with the *ArrayExpress* package
- Quality assessment with the *arrayQualityMetrics* package

Analysis of differential expression
- Linear models
- The limma approach
- Analysis results and the impact of outliers in your data

Gene set enrichment analysis
- *GSEABase* package
- Analysis results and the impact of outliers in your data

What is R?
www.r-project.org

- Free software environment for statistical computing and graphics
- Interfaces with Perl, Python, Java, C
- Active user community
- Two annual releases each year
- Download R at http://cran.r-project.org/ (precompiled binaries for Windows and Mac OS X)

What is Bioconductor?
www.bioconductor.org

- Open source and open development software project
- Initial focus on microarray data but now also including tools for proteomics, high throughput sequencing and cell-based assays
- Two annual releases each year, following the R release
- Consists of a series of R packages
- Current version: BioC 2.6, 389 packages – designed to work with R 2.11.0

- To quickly install a subset of the most common packages run:

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite()
```

Bioconductor packages

Analysis/Data type	Package name
General infrastructure	*Biobase, Biostrings, biocViews*
Pre-processing Affy/oligo data	*affy, affycomp, oligo, makecdfenv, vsn, gcrma*
Pre-processing two-color spotted DNA microarray data	*vsn, limma*
Differential gene expression	*edd, genefilter, limma, ROC, siggenes, EBArrays, factDesign*
GSEA/Hypergeometric Testing	*GSEABase, Category, GOstats, topGO*
Annotation	*annotate, annaffy, biomaRt, AnnotationDbi*
Graphs and networks	*graph, RBGL, Rgraphviz*
Graphic/GUIs	*geneplotter, hexbin, limmaGUI, exploRase*
High-throughput sequencing data	*BSgenome, Biostrings, ShortRead, IRanges, HilbertVis, HilbertVisGUI, rtracklayer*
Flow cytometry	*prada, flowCore, flowViz, flowUtils*
Protein interactions	*ppiData, ppiStats, ScISI, Rintact*
Other data	*xcms, DNAcopy, PROcess, aCGH, rsbml, SBMLR, Rdisop*

Getting help

There are a number of ways for getting help:

- Online manuals
- Commands within R
- Package vignettes

Command	Purpose
help(topic)	documentation on topic
?topic	documentation on topic
help.search('topic')	Search topic on the local help system
RSiteSearch('topic')	Search topic on http://search.r-project.org (needs an internet connection)
apropos('topic')	The names of the objects in the search list matching the regular expression topic
help.start()	Start the HTML version of help

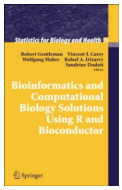
Mailing list and books

Mailing list

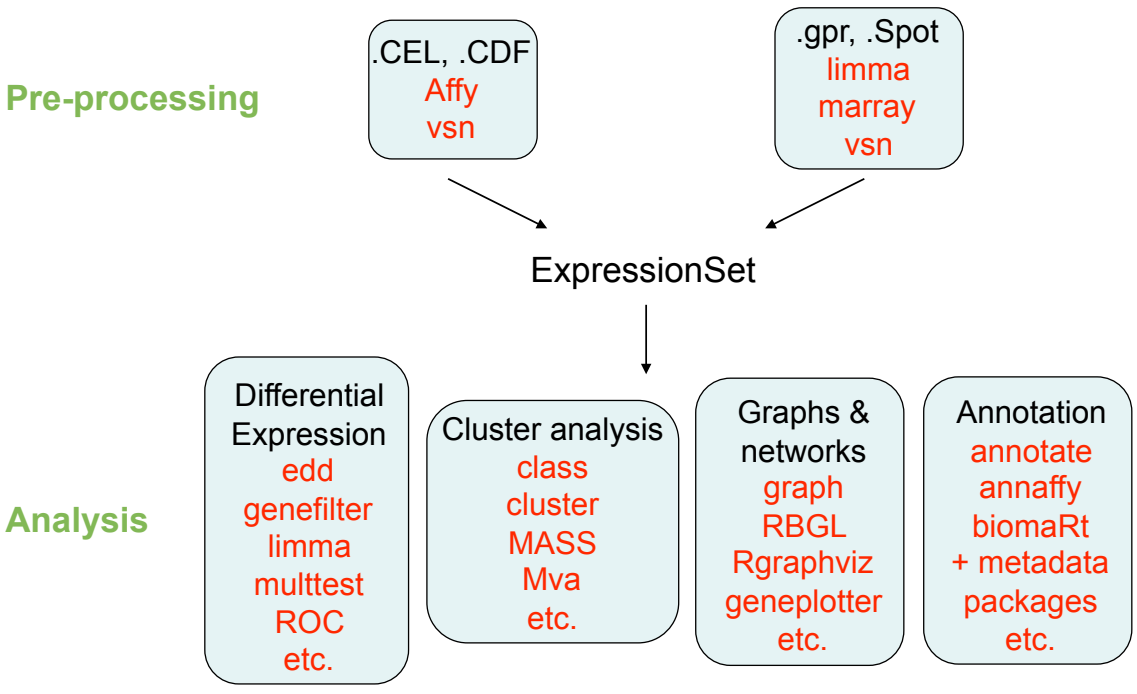
http://www.bioconductor.org/docs/mailList.html

Books

- Gentleman RC et al. (2004). Bioconductor: Open software development for computational biology and bioinformatics. Genome Biology, 5:R80
- Bioinformatics and Computational Biology Solutions Using R and Bioconductor. Series: Statistics for Biology and Health, Gentleman, R.; Carey, V.; Huber, W.; Irizarry, R.; Dudoit, S. (Eds.). Springer, 2005
- Bioconductor Case Studies. Series: Use R! Hahne, F; Huber, W.; Gentleman, R.; Falcon, S. Springer, 2008



Gene expression data analysis in Bioconductor



AffyBatch & ExpressionSet objects

Expression values (exprs)

	Sample 1	Sample 2	...	Sample i
Probe 1	$I_{1,1}$	$I_{1,2}$	...	$I_{1,i}$
Probe 2	$I_{2,1}$	$I_{2,2}$	...	$I_{2,i}$
...	...	...	...	...
Probe k	$I_{k,1}$	$I_{k,2}$	...	$I_{k,i}$

Probe annotation (featureData)

	X	Y	ID	...
Probe 1	1	1	NM_000456	...
Probe 2	2	1	NM_007294	...
...	...	...	...	...
Probe k	244	180	NM_000594	...

Sample annotation (phenoData)

	Cell type	Treatment	Replicate	...
Sample 1	WT	Yes	1	...
Sample 2	WT	Yes	2	...
...	...	...	...	...
Sample i	Mut	No	2	...

EMBL-EBI

ExpressionSet class object

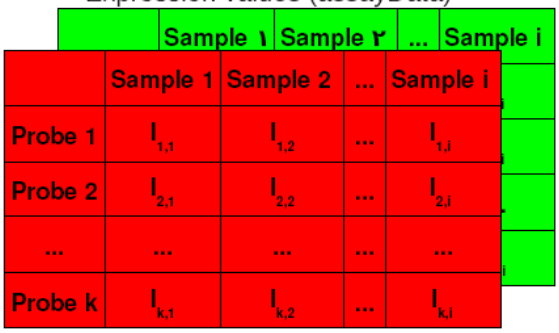
- The **ExpressionSet** class is designed to combine several different sources of information into a single convenient structure
- An **ExpressionSet** can be manipulated (e.g., subsetted, copied) conveniently, and is the input or output from many Bioconductor functions
- The data in an **ExpressionSet** consists of:
  1. expression data from microarray experiments (assayData),
  2. 'meta-data' describing samples in the experiment (phenoData),
  3. annotations and meta-data about the features on the chip or technology used for the experiment (featureData, annotation), and
  4. a flexible structure to describe the experiment (experimentData).

See **?eSet** for more details

EMBL-EBI

NChannelSet object

Expression values (assayData)



Probe annotation (featureData)

	X	Y	ID	...
Probe 1	1	1	NM_000456	...
Probe 2	2	1	NM_007294	...
...	...	...	...	...
Probe k	244	180	NM_000594	...

Sample annotation (phenoData)

	Cell type	Treatment	Replicate	...
Sample ١	WT	Yes	١	...
Sample ٢	WT	Yes	٢	...
...	...	...	...	...
Sample i	Mut	No	٢	...

The analysis pipeline

- Importing and accessing probe-level data
- Exploratory Analysis / Quality Assessment
- Preprocessing:
  - Background adjustment
  - Normalization
  - Summarization
- Quality assessment
- Differential expression
- Functional analysis of large gene lists

Importing one-channel data using *affy*

- To import CEL file data into Bioconductor

```
> library("affy")                  //loads the affy library
> myData <- ReadAffy()             //attempt to read all CEL files in the current working
```

- Probe information (CDF) are automatically loaded

- Your data is now stored in an AffyBatch Object named "myData" storing probe-level intensity matrix

A few of things you can examine at this stage

- Examine contents of the AffyBatch Object

```
> slotNames(Dilution)                    //Dilution is a an example dataset
[1] "cdfName"        "nrow"    "ncol"     "exprs"    "se.exprs"
[6] "phenoData"      "description"        "annotation"       "notes"
```

- Get information about the data object loaded

```
> Dilution
AffyBatch object
size of arrays=640x640
features (12805 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=4
number of genes=12625
annotation=hgu95av2
```

A few of things you can examine at this stage

- Description of the target samples hybridized to the chips

```
> phenoData(Dilution)
phenoData object with 3 variables and 4 cases
varLabels
        liver: amount of liver RNA hybridized to array in micrograms
        sn19: amount of central nervous system RNA hybridized to array in micrograms
        scanner: ID number of scanner used
```

```
> pData(Dilution)
      liver sn19 scanner
20A      20    0       1
20B      20    0       2
10A      10    0       1
10B      10    0       2
```


A few of things you can examine at this stage

- The *exprs* slot contains a matrix with columns corresponding to chips and rows to individual probes on the chip.

- Probe-level PM and MM intensities can be accessed using the pm and mm methods.

```
> PM <- pm(Dilution)
> PM[1:3, ]
          20A 20B 10A 10B
[1,] 468.8 282.3 433.0 198.0
[2,] 430.0 265.0 308.5 192.8
[3,] 182.3 115.0 138.0 86.3
```

- Display probset IDs and names of samples in the original CEL

```
> gnames <- geneNames(Dilution)
> gnames[1:5]
[1] "1000_at" "1001_at" "1002_f_at" "1003_s_at" "1004_at"
```

Reading data in
Two channel arrays - *limma*

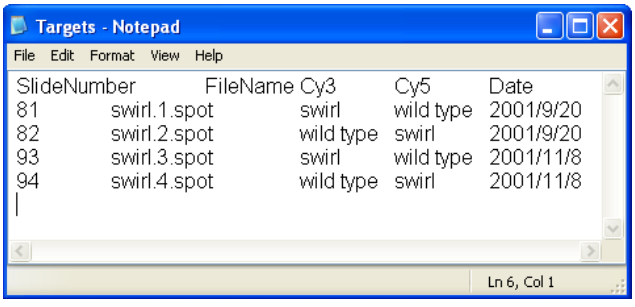
- You need image analysis output files generated by image analysis programs such as ArrayVision, ImaGene, GenePix, QuantArray or SPOT
- It also desirable to have a target file which describes which RNA sample was hybridized to each channel of each array
- An optional file is the spot type file which identifies special probes such as control spots

EMBL-EBI

Reading data in
Two channel arrays - *limma*

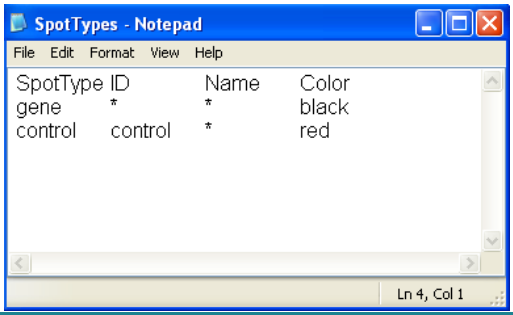
The targets file

- Tab-delimited
- Contains a row for each microarray in the experiment
- Default name 'Targets.txt'
- Read in using **readTargets()**



The spot types file

- Tab-delimited
- One row for each spot type to be distinguished
- Columns with SpotType, patterns to identify the spot type and plotting attributes
- Default name 'SportTypes.txt'
- Read in using **readSpotTypes()**



EMBL-EBI

Reading data in
Two channel arrays - *limma*

- The foreground and background intensities can be read into an RGList object using a command of the form

```
RG = read.maimages(files, source="<imageanalysisprogram>", path="<directory>")
```

- If the targets file is available, the file names are usually read from it

```
targets = readTargets()
RG = read.maimages(targets, source="spot“)
```

- An RGlist is a simple list-based class for storing red and green channel foreground and background intensities for a batch of spotted microarrays.

EMBL-EBI

Reading data in RGList

See ?read.maimages for more details on the function arguments and a description of all the RGList components

```
> RG
An object of class "RGList"
$R
       swirl.1   swirl.2   swirl.3    swirl.4
[1,] 19538.470 16138.720 2895.1600 14054.5400
[2,] 23619.820 17247.670 2976.6230 20112.2600
[3,] 21579.950 17317.150 2735.6190 12945.8500
[4,]  8905.143  6794.381  318.9524   524.0476
[5,]  8676.095  6043.542  780.6667   304.6190
8443 more rows ...

$G
       swirl.1   swirl.2   swirl.3    swirl.4
[1,] 22028.260 19278.770 2727.5600 19930.6500
[2,] 25613.200 21438.960 2787.0330 25426.5800
[3,] 22652.390 20386.470 2419.8810 16225.9500
[4,]  8929.286  6677.619  383.2381   786.9048
[5,]  8746.476  6576.292  901.0000   468.0476
8443 more rows ...

$Rb
     swirl.1 swirl.2 swirl.3 swirl.4
[1,]     174     136      82      48
[2,]     174     133      82      48
[3,]     174     133      76      48
[4,]     163     105      61      48
[5,]     140     105      61      49
8443 more rows ...

$Gb
     swirl.1 swirl.2 swirl.3 swirl.4
[1,]     182     175      86      97
[2,]     171     183      86      85
[3,]     153     183      86      85
[4,]     153     142      71      87
[5,]     153     142      71      87
8443 more rows ...

$targets
            FileName
swirl.1 swirl.1.spot
swirl.2 swirl.2.spot
swirl.3 swirl.3.spot
swirl.4 swirl.4.spot

$source
[1] "spot"
```

Data import using ArrayExpress package

The **ArrayExpress package** allows to:

- access the ArrayExpress database within Bioconductor
- query the database content
- retrieve raw and processed datasets and
- build Bioconductor data structures that can be manipulated using other Bioconductor packages

NB: use version 1.3.15 or later for full functionality

ArrayExpress package functions

The package includes several functions:

- **ArrayExpress**: produces an AffyBatch, an ExpressionSet or a NChannelSet from a raw dataset from the ArrayExpress database
- **queryAE**: queries ArrayExpress with keywords and gives a list of ArrayExpress identifiers and their availability as raw and/or processed data, as output
- **getAE**: download MAGE-TAB from ArrayExpress in a specified directory
- **magetab2bioc**: convert local MAGE-TAB files into a Bioconductor object
- **getcolproc**: return all column names from processed MAGE-TAB files
- **procset**: convert processed MAGE-TAB files into a Bioconductor object

Function's role	Raw data	Processed data
One-step function (Wrapper)	ArrayExpress	
Downloading the data	getAE	getAE
Getting the possible column names		getcolproc
Converting MAGE-TAB data into R object	magetab2bioc	procset

ArrayExpress package functions

The ArrayExpress function produces the following objects:

- AffyBatch: Affymetrix arrays
- ExpressionSet: One colour arrays
- NChannelSet: Two colours arrays

These objects will contain:

Information type	MAGE-TAB component	Bioconductor component
Expression data	Raw.zip	assayData
Sample annotation	SDRF	phenoData
Array annotation	ADF	featureData
MIAME experiment level information	IDF	experimentData

AffyBatch & ExpressionSet objects

Expression values (exprs)

	Sample 1	Sample 2	...	Sample i
Probe 1	$I_{1,1}$	$I_{1,2}$	...	$I_{1,i}$
Probe 2	$I_{2,1}$	$I_{2,2}$	...	$I_{2,i}$
...	...	...	...	...
Probe k	$I_{k,1}$	$I_{k,2}$	...	$I_{k,i}$

Probe annotation (featureData)

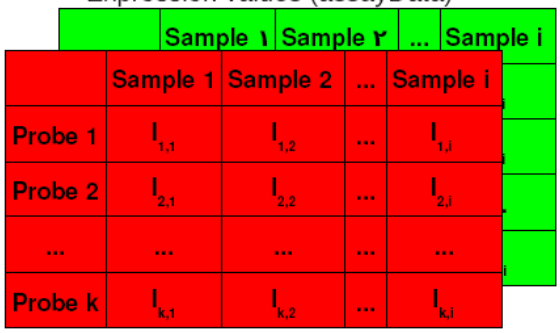
	X	Y	ID	...
Probe 1	1	1	NM_000456	...
Probe 2	2	1	NM_007294	...
...	...	...	...	...
Probe k	244	180	NM_000594	...

Sample annotation (phenoData)

	Cell type	Treatment	Replicate	...
Sample 1	WT	Yes	1	...
Sample 2	WT	Yes	2	...
...	...	...	...	...
Sample i	Mut	No	2	...

NChannelSet object

Expression values (assayData)



Probe annotation (featureData)

	X	Y	ID	...
Probe 1	1	1	NM_000456	...
Probe 2	2	1	NM_007294	...
...	...	...	...	...
Probe k	244	180	NM_000594	...

Sample annotation (phenoData)

	Cell type	Treatment	Replicate	...
Sample ١	WT	Yes	١	...
Sample ٢	WT	Yes	٢	...
...	...	...	...	...
Sample i	Mut	No	٢	...

The analysis pipeline

- Importing and accessing probe-level data
- Exploratory Analysis / Quality Assessment
- Preprocessing:
  - Background adjustment
  - Normalization
  - Summarization
- Quality assessment
- Differential expression
- Functional analysis of large gene lists

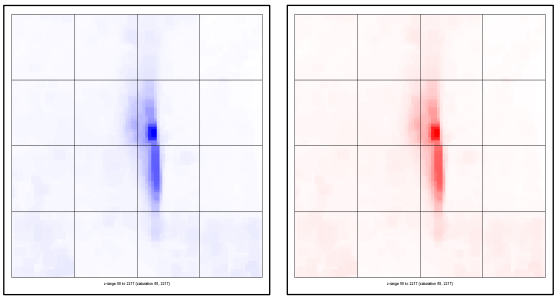
Exploratory Data Analysis

- Exploratory analysis aims to find patterns in the data that aren't predicted by the experimenter's current knowledge
- Provides a quick, simple and visual means to explore the data at hand before any further analysis is carried out.
- EDA has been the tool of choice for detection of problematic arrays

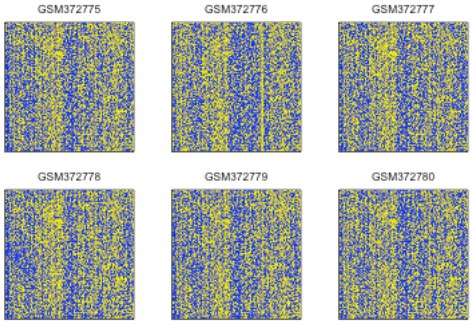
Diagnostic plots in Exploratory Analysis

- Spatial Image
- Histograms / Density plot
- Box plot
- MA plot
- Volcano Plot

Spatial image plots

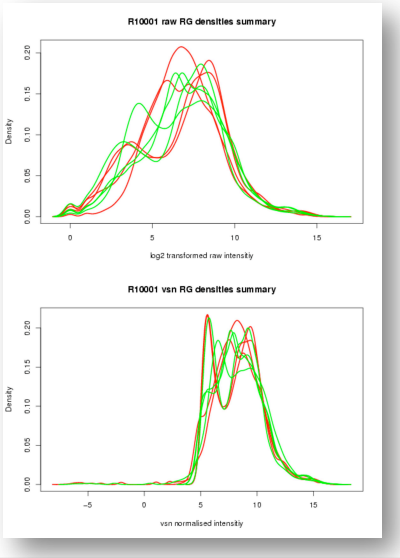


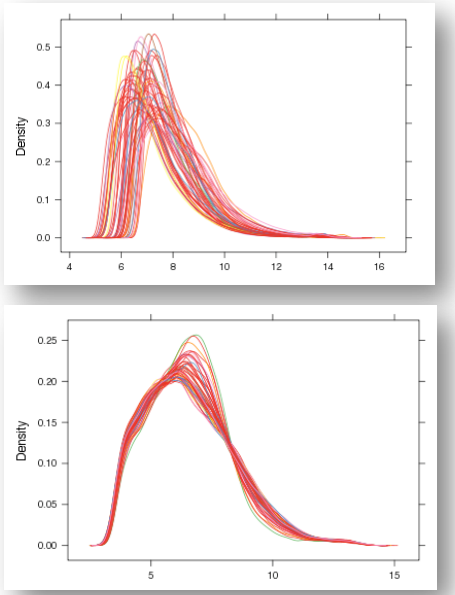
See **?imageplot** for more details



- Create chip images of the raw intensities
- Reveal hybridization artifacts such as scratches, drops or cover-slip effects
- Arrays that look poor at this stage should not be taken forward for analysis.

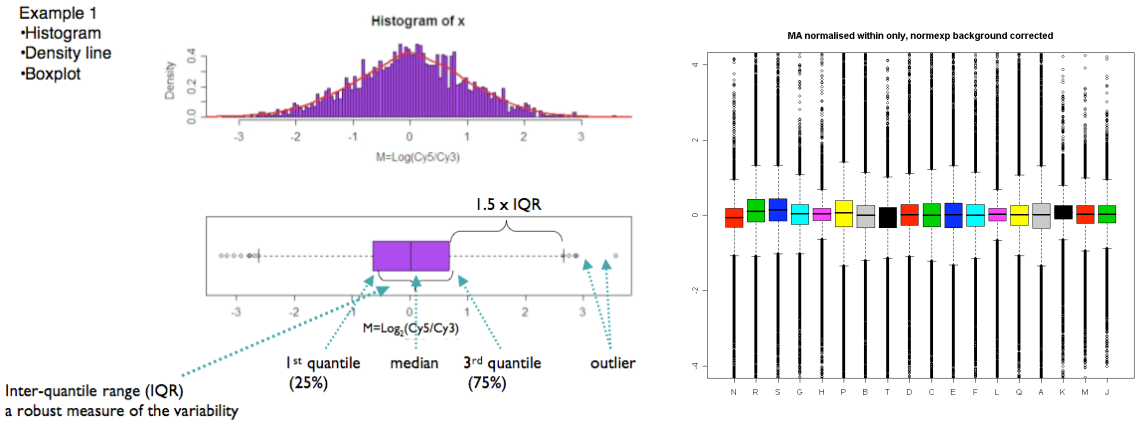
Density plot





- Identify distributions of signal intensity that show differing behavior.
- On raw data, a bimodal distribution can be indicative of an array containing a spatial artifact
- An array shifted to the right can be indicative of abnormal higher background intensities
- See **?plotDensities** for more details

Box plot



- Displays graphically the so-called 5-number summary of a dataset
- The summary consists of the median, the upper and lower quartiles, the range, and, possibly, individual extreme values

Introducing M and A

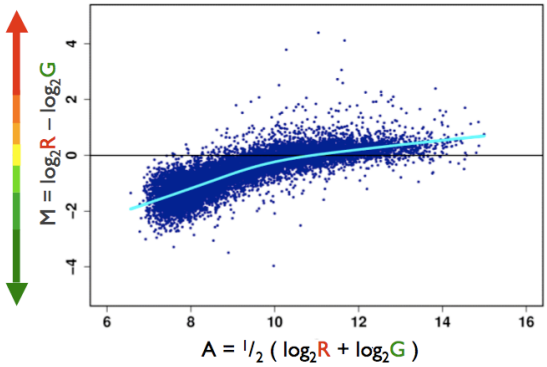
- We're ultimately interested in
  - Difference between R and G / signal and reference
  - Overall intensities of spots (the geometric mean of the Red and Green intensities)
- So we define
  - M : log differential- expression ratio (log fold change)
  - A : the average log intensity

$$M = \log_2(R/G)$$
$$= \log_2 R - \log_2 G$$

$$A = \tfrac{1}{2} \log_2 (RG)$$
$$= \tfrac{1}{2} (\log_2 R + \log_2 G)$$

MA plot

- A 45-degree rotation and axis scaling scatter plot.
- Logs stretch out region we are most interested in
- In a perfect situation, the log-ratios M in an MA-plot should be evenly distributed around zero across all intensity-values A.
- Clearly identify intensity dependent variation, and dye-bias.
- Differentially expressed genes more easily identified

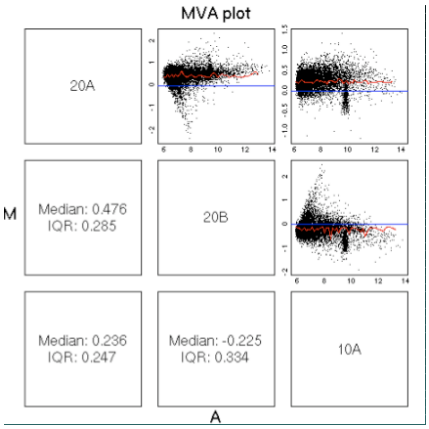


What is MA plot used for?

- For Affymetrix arrays:
  - Pairwise comparisons of all arrays in the experiment
  - Compare all arrays against a reference array
    - Reference array is created by taking probe-wise medians across all the arrays
- Compare Cy3 (G) and Cy5 (R) (dye-bias)
- Quality problems are most apparent where loess curve oscillates a great deal or if the variability in one plot differs much from other arrays

Diagnostic plots

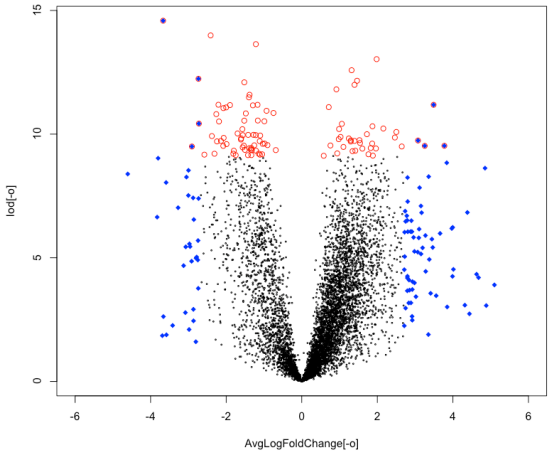
- Create MA-plots
  - mva.pairs function plots all pairwise MA-plots comparing all arrays in the AffyBatch object



  - For replicates and cases where most genes are not differentially expressed, we want the cloud of points to be around 0 and the IQR to be small.

Volcano plot

- Compares the size of the fold change to the statistical significance level
- x-axis: fold change between two groups
- y-axis: p-value for t-test of difference between samples



The analysis pipeline

- Importing and accessing probe-level data
- Exploratory Analysis / Quality Assessment
- Preprocessing:
  - Background adjustment
  - Normalization
  - Summarization
- Quality assessment
- Differential expression
- Functional analysis of large gene lists

EMBL-EBI

Data pre-processing

- Aims at adjusting for systematic effects that arise from variation in the experimental technology rather than from biological differences between RNA samples
- It is an essential requirement before carrying out any downstream analysis
- Pre-processing steps include (some are platform dependent):
  - Background correction
  - Probe summarization (Affymetrix only)
  - Treating missing values
  - Data filtering and flagging
  - Within array normalization
  - Between array normalization
  - Quality assessment

Data pre-processing
Background correction

- The default background correction action is to subtract the background intensities from the foreground intensity for each spot
- However there are many other background correction options which might be preferable in certain situations
- See **?backgroundCorrect** for a full list of methods
- Note that the background correction method can also be specified as an argument to the normalization function **normalizeWithinArrays**

Data pre-processing
Normalization

- The fundamental step in data pre-processing is **normalization**
- The goal of normalization is to remove systematic variation from the data and scale it so that comparisons can be made across studies
- The choice of normalization approach depends entirely on whether there is a problem with the data and what exactly the problem is
- Use quality assessment to determine what the problem might be
- Normalization within arrays and between arrays

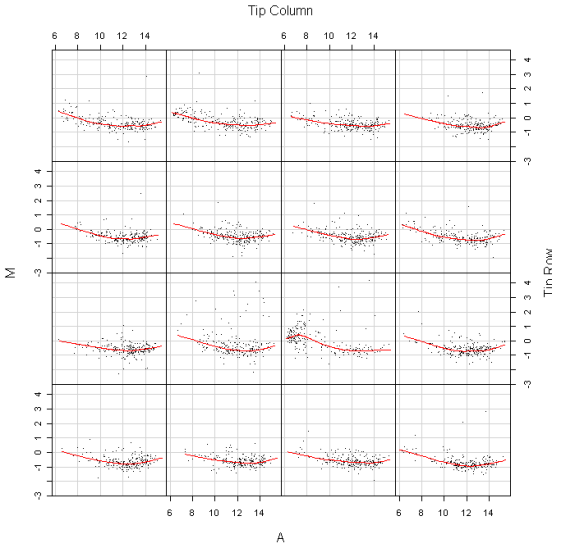
Data pre-processing
Normalization within arrays

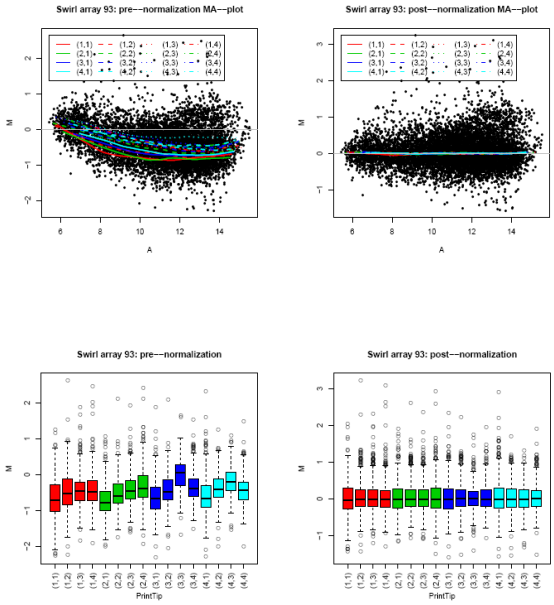
- Normalizes the M-values for each array separately
- Corrects for systematic differences between samples on the same slide which do not represent true biological variation
- How do we know it is necessary?
  By examining replicate and/or self-self hybridizations, where no true differential expression is occurring
- We find biases which vary with overall spot intensity, location on the array, dye, plate origin, pins, scanner, scanning parameters,....

Data pre-processing
Normalization within arrays

- **Print-tip loess normalization** is the default method and can be used to correct for various printing effects
- **Global loess normalization** should be used when print-tip groups information is not available
- Global loess assumes that the bulk of the probes on the array are not differentially expressed; it then detects systematic deviations in MA plots and corrects them by fitting a local weighted linear regression
- Spot quality weight information can be accounted for
- Background correction is performed automatically if the object has not already been background corrected
- See **?normalizeWithinArrays** for a full list of normalization methods

Data pre-processing
Normalization within arrays - print tip loess





From *limma* and *marray* vignettes

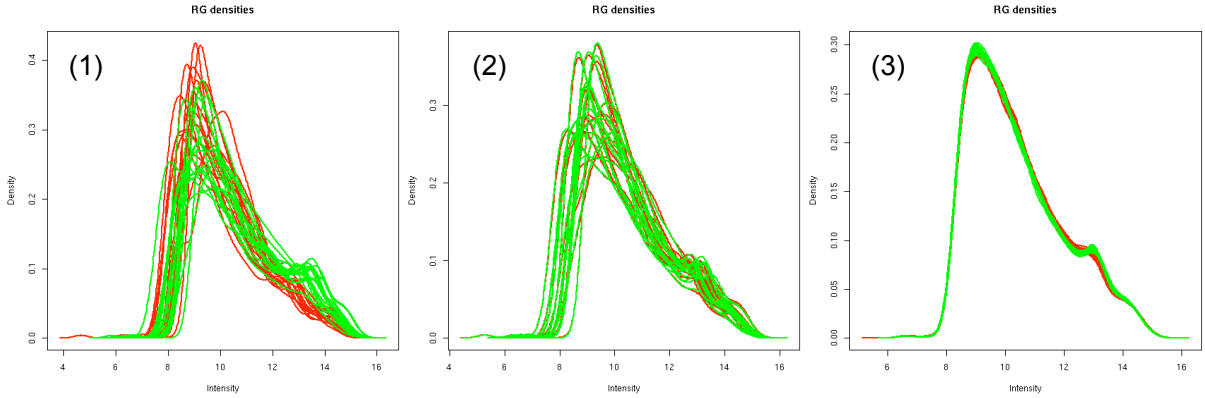
Data pre-processing
Normalization between arrays

- Normalizes expression intensities so that the intensities or log-ratios have similar distributions across a set of arrays, making them comparable

- Commonly used methods:
  - **Scale and centering normalization**: equalizes all the means or medians of all intensities of all the distributions
  - **Quantile normalization**, ensures that the intensities have the same empirical distribution across arrays and across channels
  - **Variance stabilizing normalization**, gives an interface to the variance stabilizing methods of the *vsn* package

- See **?normalizeBetweenArrays** for a full list of normalization methods

Data pre-processing
Within and between array normalization example



1) Without any normalization there is considerable variation between both channels and between arrays

2) After loess normalization of the M-values for each array the red and green distributions become essentially the same for each array, although there is still considerable variation between arrays

3) Applying quantile normalization to the A-values makes the distributions essentially the same across arrays as well as channels

Data pre-processing in practice
Affymetrix data – *affy* package

- Pre-processing for Affymetrix data involves the following 3 steps:
  - Background correction
  - Probe summarization
  - Normalization: RMA, GC-RMA and 'Li & Wong'

- The *affy* package provides implementations for a number of methods for each of these steps

```
> bgcorrect.methods
      [1] "mas" "none" "rma" "rma2"
> normalize.AffyBatch.methods
      [1] "constant" "contrasts" "invariantset" "loess" "qspline" "quantiles"
> express.summary.stat.methods
      [1] "avgdiff" "liwong" "mas" "medianpolish" "playerout"
```

See **?rma** or **?li.wong** for more details

Data pre-processing in practice

From probe-level intensities to expression measures

- **Robust Multi-Array (RMA)**
  - Convolution background correction (ignoring MM)
  - Quantile normalization e.g. the intensities are adjusted to produce identical distributions
  - Use median polish to estimate log expression robustly

- **GC-RMA**
  - GC-RMA is a modified version of RMA that takes into consideration the probes GC content

- **'Li & Wong'**
  - Uses loess correction method for non linear adjustment

EMBL-EBI

Data pre-processing in practice

From probe-level intensities to expression measures

- The main user-level pre-processing function is *expresso*:
  - It starts from raw probe- level intensities to produce gene-level expression measures

  - You plug in your desired methods for background correction, normalization and summarization to it and it produces a new object *exprSet* containing expression summary values

```
eset <- expresso(Dilution, bgcorrect.method=''rma'',
                           normalize.method="constant",
                           pmcorrect.method=''pmonly'',
                           summary.method=''medianpolish'');
```


EMBL-EBI

Data pre-processing in practice

From probe-level intensities to expression measures

- RMA is another shortcut. It is a three step procedure of predefined methods:
  - Convolution background correction
  - Quantile normalization
  - Summarization based on median polish algorithm

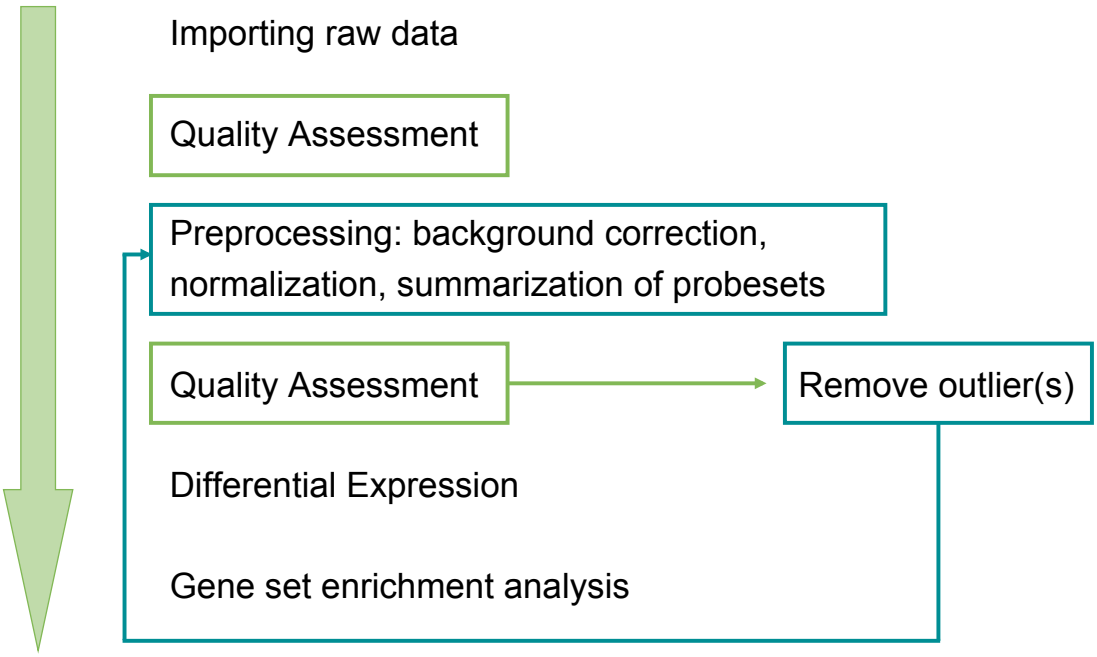
```
> rmaDil <- rma(Dilution)
   Background correcting
   Normalizing
   Calculating Expression
> head(exprs(data.rma)                                    # view expression
> write.table(exprs(data.rma), file="data.rma.txt", sep='\t')    # output to file
```

- The package "*affyPLM*" also provides "threestep" a similar summarization function that provides user with great deal of control

The analysis pipeline

- Importing and accessing probe-level data
- Exploratory Analysis / Quality Assessment
- Preprocessing:
  - Background adjustment
  - Normalization
  - Summarization
- Quality assessment
- Differential expression
- Functional analysis of large gene lists

Quality assessment



Quality assessment

What aspects do we evaluate? Which quality metrics?

Within arrays

What are we looking at?

- Intensity-dependent ratio
- Detection of spatial effects

How?

- MA plots
- Representation of the chip
- Print-tips (block), row, column

Between Arrays

What are we looking at?

- Homogeneity
- Outlier samples
- Biological meaning

How?

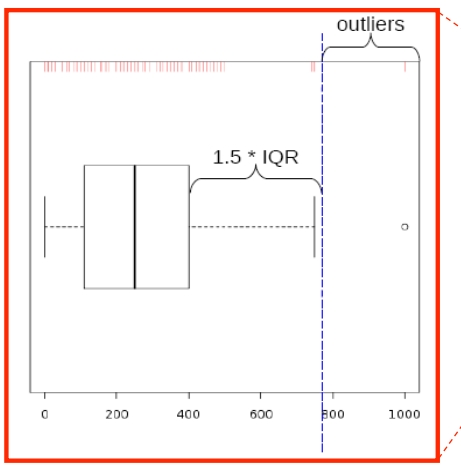
- Boxplots
- Density plots
- Empirical Cumulative Distribution Functions
- Heatmap

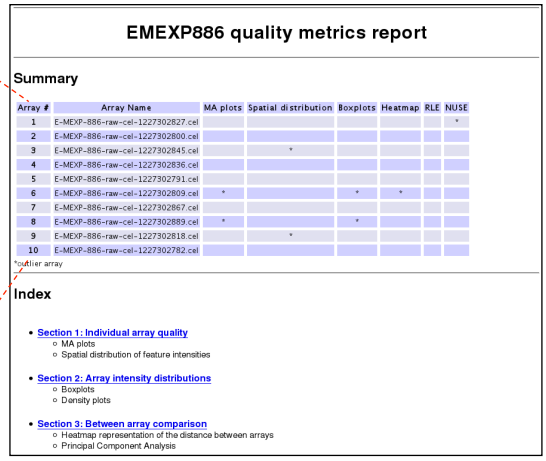
Quality assessment
arrayQualityMetrics package

- Bioconductor package to assess quality of microarrays
  - Affymetrix, Agilent, Illumina, homemade arrays etc...
  - RNA, tiling, exon, SNP arrays, etc…
- From an R object → HTML report
- Plots:
  - MA plot and spatial representations
  - Boxplots, density and ECDF
  - Heatmap
  - Variance-mean dependency
  - GC content and probe mapping studies
  - Affymetrix only: NUSE, RLE, RNA degradation, QCstats, PM/MM
- Outlier identification

Quality assessment
arrayQualityMetrics report - Outlier identification

- For each array in the experiment, we calculate S_i, for each quality metric
- If S_i is beyond the extreme whiskers of the boxplot of all the *S,* the array *i* is considered as an outlier array





EMEXP886 quality metrics report

Summary

Array #	Array Name	MA plots	Spatial distribution	Boxplots	Heatmap	RLE	NUSE
1	E-MEXP-886-raw-cel-1227302827.cel						*
2	E-MEXP-886-raw-cel-1227302800.cel						
3	E-MEXP-886-raw-cel-1227302845.cel		*				
4	E-MEXP-886-raw-cel-1227302836.cel						
5	E-MEXP-886-raw-cel-1227302791.cel						
6	E-MEXP-886-raw-cel-1227302809.cel	*		*	*		
7	E-MEXP-886-raw-cel-1227302867.cel						
8	E-MEXP-886-raw-cel-1227302889.cel	*		*			
9	E-MEXP-886-raw-cel-1227302818.cel		*				
10	E-MEXP-886-raw-cel-1227302782.cel						

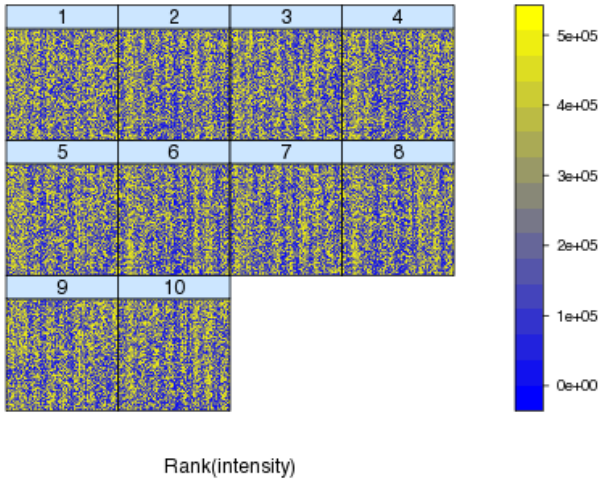
*outlier array

Index

- Section 1: Individual array quality
  - MA plots
  - Spatial distribution of feature intensities
- Section 2: Array intensity distributions
  - Boxplots
  - Density plots
- Section 3: Between array comparison
  - Heatmap representation of the distance between arrays
  - Principal Component Analysis

Quality assessment

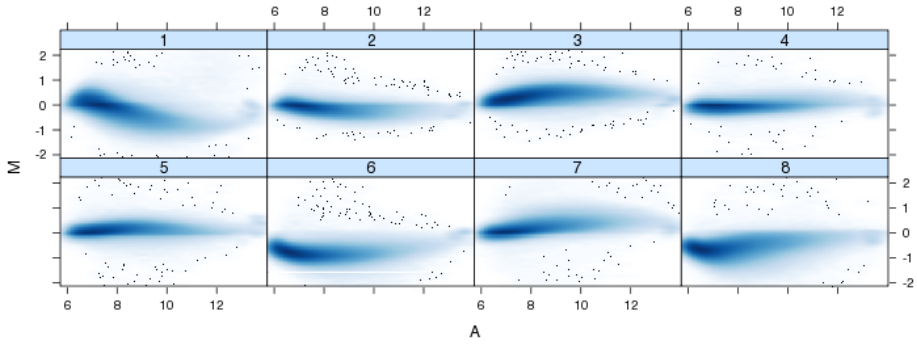
arrayQualityMetrics report – Individual array quality



Spatial plots

Quality assessment

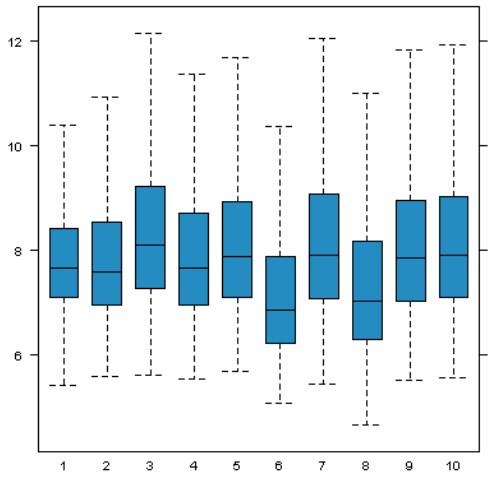
arrayQualityMetrics report – Individual array quality



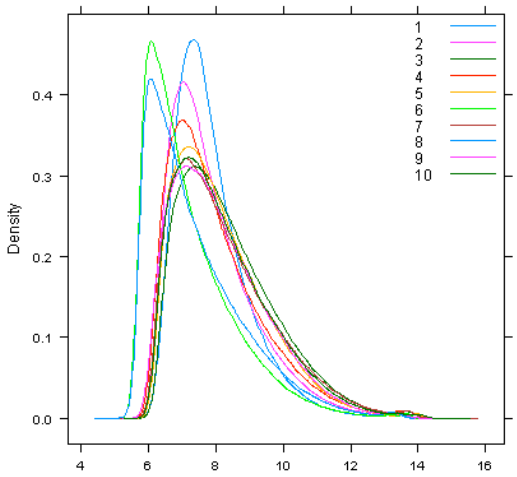
MA plots

Quality assessment

arrayQualityMetrics report – Array Intensity distribution



Boxplot

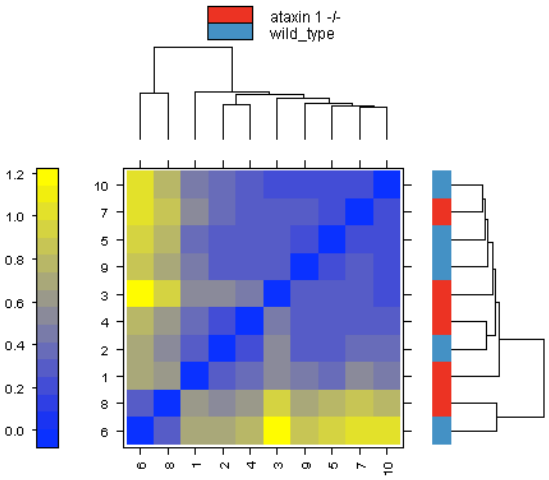


Density plot

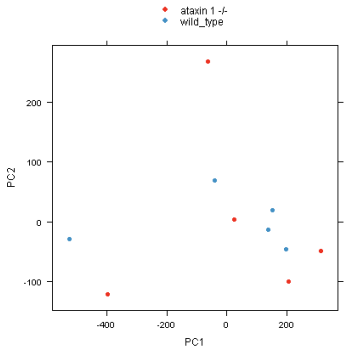
EMBL-EBI

Quality assessment

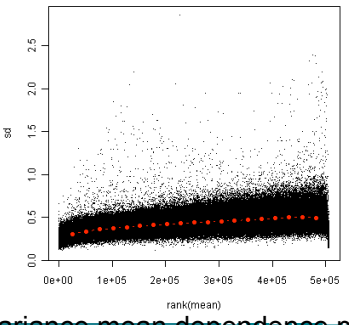
arrayQualityMetrics report – Between array comparison



Heatmap representation of the distance between arrays



Principal Component Analysis plot



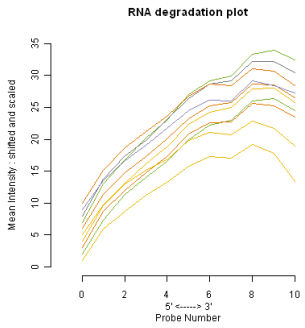
Variance mean dependence plot

EMBL-EBI

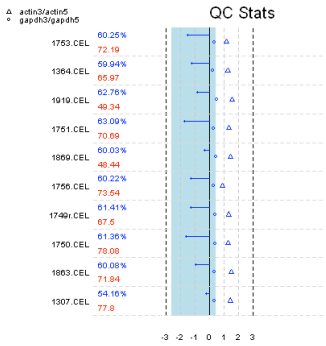
Quality assessment

arrayQualityMetrics report – Affymetrix specific plots

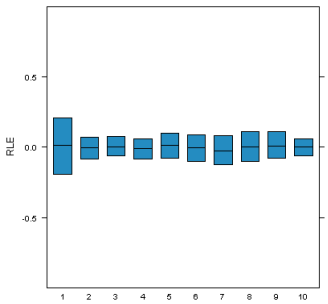
(1)



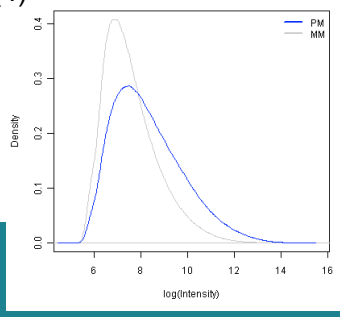
(2)



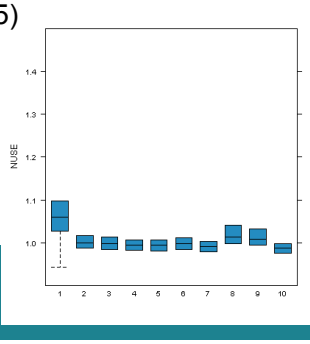
(3)



(4)



(5)



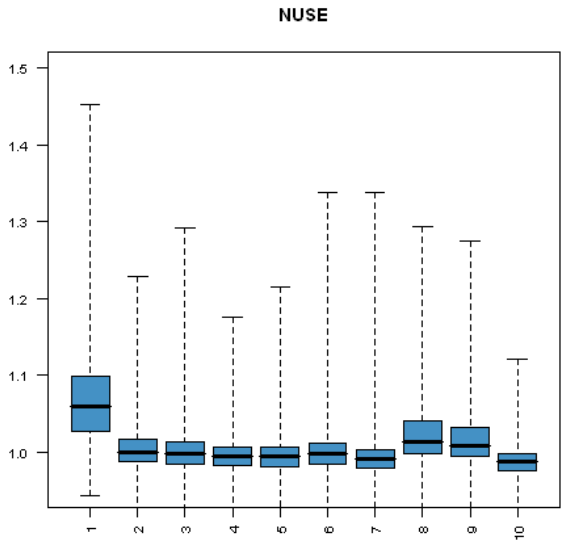
(1) RNA degradation plot
(2) QC stats
(3) Relative Log Expression (RLE) plot
(4) PM and MM plot
(5) Normalized Unscaled Standard Error (NUSE) plot

EMBL-EBI

Quality assessment

arrayQualityMetrics report – Affymetrix specific plot, NUSE

- One boxplot per array
- The boxes should have small spread and be centered at NUSE = 1
- Quality problems:
  - a box with greater spread
  - a box elevated
- These values are not comparable across data sets since NUSE is relative only within a data set
- It is the best diagnostic plot for Affymetrix arrays

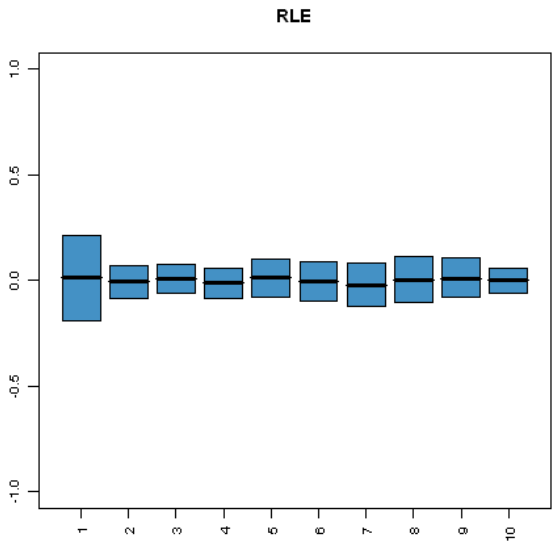


EMBL-EBI

Quality assessment

arrayQualityMetrics report – Affymetrix specific plot, RLE

- One boxplot per array
- Assumption :
  - The majority of genes are not changing in expression between tested conditions
  - The majority of these non-differential genes are displayed on the RLE plot
- How the boxes should be:
  - Small spread
  - Centered at RLE = 0
- Quality problems :
  - A box with greater spread
  - A box not centered near RLE = 0



A note to remember…

The choice of preprocessing method can have enormous influence on the quality of the ultimate results.

So…

Try to assess different methods before proceeding to more downstream analysis

Make sure you use same methods when trying to reproduce previous results

Practical 2

1- Data import:

ArrayExpress package

2- Quality assessment:

arrayQualityMetrics package

Import ArrayExpress Data into Bioconductor

- **Exercise 1**: Load the *ArrayExpress* library and import the dataset **E-MEXP-886** from the ArrayExpress database

```
> library("ArrayExpress")
> AEset = ArrayExpress("E-MEXP-886")
> AEset
```

- Dataset E-MEXP-886 consists of 10 Affymetrix arrays, comparing RNA from ataxin -/- mice (KO - 5 arrays) to wild type mice (WT - 5 arrays)

- AEset is an **AffyBatch** which is the standard Bioconductor object for Affymetrix data

Import ArrayExpress Data into Bioconductor
Sample annotation

- The sample annotation is obtained from the SDRF file from the database and is stored in the Affybatch **phenoData**. The **pData** function gives access to an object phenoData

```
> colnames(pData(AEset))
```

- We can choose a relevant column from the sample annotation to identify biological groups of interest

- In this case we want to use the genotype information available for the samples. We use the function **grep** to extract this information and store it in a separate object, **fac**

```
> fac = colnames(pData(AEset))[grep("Factor", colnames(pData(AEset)))]
> fac
```


Quality assessment
Before normalization

- **Exercise 2**: Load the *arrayQualityMetrics* library and use the **arrayQualityMetrics** function to run quality assessment before normalization

```
> library("arrayQualityMetrics")
> arrayQualityMetrics(expressionset = AEset, outdir = "QAraw",
+ force = FALSE, do.logtransform = TRUE, intgroup = fac)
```

- A report named **QMreport.html** is produced in the subdirectory QAraw.

- The report contains a series of diagnostic plots as .png images. All the images are saved in the subdirectory QAraw.

Quality assessment

After normalization

- Normalize the data using RMA (Robust Multi-Array Average expression measure) normalization.
- **Exercise 3**: Load the *affy* library and run normalization

```
> library("affy")
> rAEset = rma(AEset)
```

- As the RMA normalization summarizes the probesets, the object rAEset is now an **ExpressionSet** which is a standard object for one colour arrays.
- **Exercise 4**: Use the **arrayQualityMetrics** function to run quality assessment after normalization

```
> arrayQualityMetrics(expressionset = rAEset, outdir = "QAnorm",
+ force = FALSE, intgroup = fac)
```

- Explore the report after normalization and decide if to remove any array before proceeding with the analysis

Quality assessment

Outlier removal and normalization

- The array #1 is still identified as an outlier after normalization. This is indicated by:

1. a wider spread in the MA plot,
2. a slightly different boxplot distribution,
3. a density distribution quite dissimilar from the other arrays, and
4. very different distance matrix entries in the heatmap

- **Exercise 5**: Remove array #1 from the dataset before proceeding with the analysis

```
> cAEset = rma(AEset[, -1])
```

The analysis pipeline

Importing and accessing probe-level data

Exploratory Analysis / Quality Assessment

Preprocessing:

- Background adjustment
- Normalization
- Summarization

Quality assessment

Analysis of Differential expression

- Classical statistical tests
- Linear models

Functional analysis of large gene lists

Identifying Differential Genes

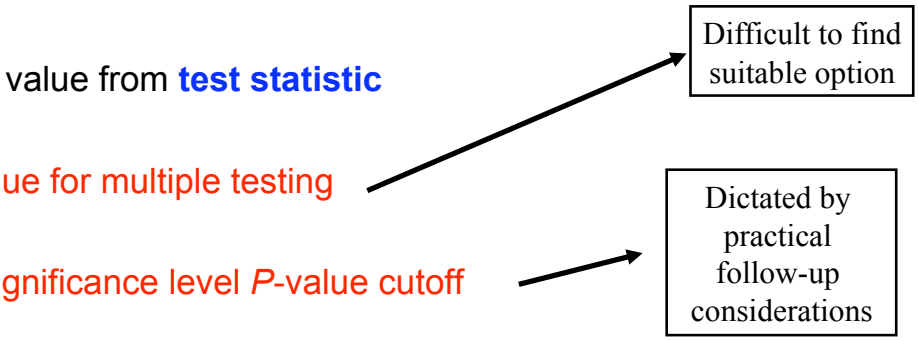
- The aim:

  To identify genes that show differential abundances.

- **Manage variation:**
  - model systematic variation,
  - quantify random noise and control error rates
  - identify relevant (non-random) *'biological'* variation

Statistical Analysis Workflow

1. Generate hypotheses
2. Perform statistical test appropriate to experimental design and hypotheses
3. Calculate *P* value from **test statistic**
4. Adjust *P* value for multiple testing
5. Choose a significance level *P*-value cutoff
6. Compare *P* value to significance level and accept or reject null hypothesis



Classical t-tests

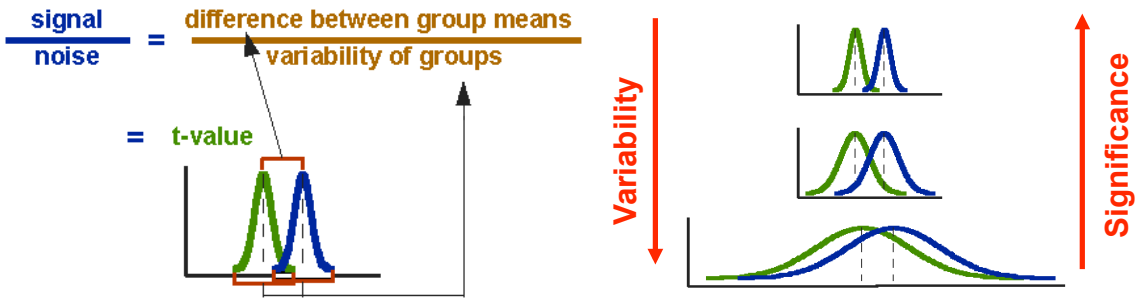
Assumptions

- Data fits a normal distribution – parametric assumption
  - Assumption is not as strong when dealing with larger n > 30.
  - Kolmogorov-Smirnov or Shapiro-Wilk tests are formal tests for normality Q-Qplot.
- Gene expression levels have equal variance s^2 across conditions
- Measures/tests are independent

- Issues for Expression Profiling
  - Small variances and small differences of mean
  - High variances and high differences of mean

t-test

- Assesses whether the means of two groups are statistically different from each other
- Calculated as difference of means relative to their variance
- Essentially measures signal-to-noise and calculates a p-value for each gene



- May suffer from intensity-dependent effects
- Apply corrections to reduce the number of false positive

Finding differentially expressed genes

- One condition with multiple replicates –which genes are significantly expressed above background/controls?
  - one-sample t-test
- Two conditions with multiple replicates -which genes are different between the two sets?
  - two-sample t-test, linear models (limma)
- Multiple conditions with multiple replicates -which genes show most variance between these sets?
  - e.g. ANOVA, linear models (limma)

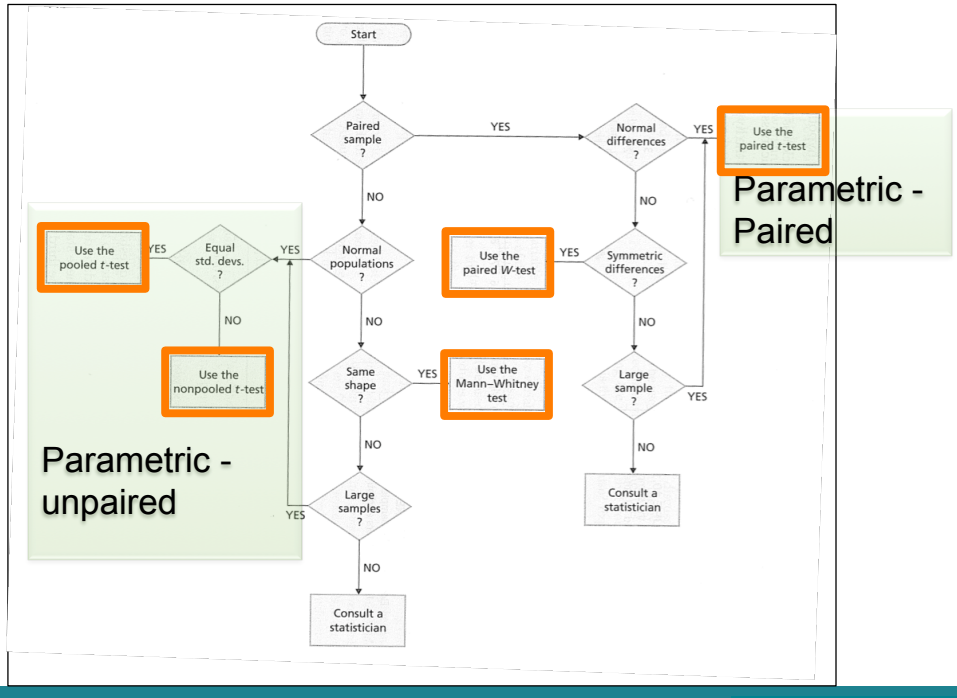
Non-parametric statistical tests

- do not assume a normal distribution
- more robust/less sensitive to outliers
- less powerful especially for small samples

- Wilcoxon signed rank test for paired data
  - ranks differences of observations between 2 samples
  - Assumptions: distribution of differences is continuous and symmetrical

- Mann-Whitney test for un-paired data
  - differences in median rank of two populations
  - Assumptions: distribution of differences is continuous

Statistical tests summary



The problem of multiple testing

- Example:
  - Genes on a chip: 10,000
  - None of them is differentially expressed
- Statistics used:
  - We consider a gene to be differentially expressed if it meets the criterion at a p-value of $p < 0.05$
- Results:
  - Even though none of the genes is differentially expressed, about 5% of the genes (i.e. 500 genes) could be erroneously concluded to be differentially expressed
  - i.e. there will be 500 false conclusions!!!!

The problem of multiple testing

- Don’t get too hung up on p-values
- Ultimately, what matters is biological relevance
- p-values should help you evaluate the strength of the evidence, rather than being used as an absolute yardstick of significance.
- Statistical significance is not necessarily the same as biological significance
- Increase in specificity is coupled with loss of sensitivity

Multiple Testing Correction methods

- Two different approaches to controlling error rates:
- Family-wise error rate (FWER):
  - Correct for the probability of at least one type I error (false positive)
- False discovery rates (FDR):
  - Correct for expected proportion of type I errors among rejected hypotheses

FPR vs FDR

	Test result		
	non-DE	DE	Total
True			
non-DE	*A* = 9025	*B* = 475	9500
DE	*C* = 100	*D* = 400	500
Total	9125	875	10 000

False Positive Rate is *B/(A + B)*
475/9500 = 5% (95% specificity)

FDR is *B/(B+D)*
475/875 = 54%,

i.e. more than half of the 'discovered' genes are bogus

FWER Multiple testing correction methods

- Reduce the probability of erroneously classifying non-significant genes as significant
- Produces an adjusted p-value

1. **Bonferroni**
   - p-value of each gene is multiplied by the number of genes in the gene list
   - adjusted p-value= p-value * n (number of genes in test) <0.05
2. **Holm**
   1. Each gene p-value is ranked from the smallest to the largest
   2. First p-value is multiplied by the number of genes present in the gene list; if the end value is less than 0.05, the gene is significant
      adjusted p-value= p-value * n < 0.05
   3. Second p-value is multiplied by the number of genes less 1
      adjusted p-value= p-value * n-1 < 0.05
   4. Third p-value is multiplied by the number of genes less 2
      adjusted p-value= p-value * n-2 < 0.05
3. **Benjamini-Hochberg**
   1. p-values of each gene are ranked from the smallest to largest
   2. Largest p-value remains as it is
   3. Second largest p-value is multiplied by the total number of genes in gene list divided by its rank. If less than 0.05, it is significant
      adjusted p-value = p-value*(n/n-1) < 0.05, if so, gene is significant
   4. Third p-value is multiplied as in step 3
      adjusted p-value = p-value*(n/n-2) < 0.05, if so, gene is significant Stringency

The analysis pipeline

Importing and accessing probe-level data

Exploratory Analysis / Quality Assessment

Preprocessing:

- Background adjustment
- Normalization
- Summarization

Quality assessment

Analysis of Differential expression

- Classical statistical tests
- Linear models

Functional analysis of large gene lists

Analysis of differential expression
Linear models

- Applying linear models to the assessment of differential expression in microarray data is a popular approach and is available through the Bioconductor package *limma*
- The moderated statistics approach in *limma* is designed to stabilize the analysis even for experiments with small number of arrays
- Linear modeling approach applies to both single channel (Affymetrix) and two-colour spotted arrays.

References:

- http://bioinf.wehi.edu.au/limma (home page)
- http://www.bioconductor.org/packages/2.3/bioc/html/limma.html (link to user guide)
- Smyth, G. K. (2005). Limma: linear models for microarray data. In: *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, W. Huber (eds.), Springer, New York, pages 397- 420

Analysis of differential expression
Linear models – the *limma* approach

1. First we need to specify a design matrix which indicates which RNA sample has been applied to each array (See ?model.matrix)
2. Based on the design matrix, a linear model is fitted for each gene on the array with the aim of modeling the systematic part of your data, so it can be distinguished from random variation (See ?lm.fit)
3. Then you need to specify a contrast matrix, which specifies which comparisons we would like to make between the RNA samples (See ? contrast.matrix)
4. Based on the contrast matrix, to obtain coefficients and standard errors for any number of contrasts of the coefficients of the original model (See ? contrasts.fit)
5. Then, for every gene, we use an empirical Bayes method for assessing differential expression and compute moderated t-statistics, moderated F-statistic, and log-odds of differential expression (See ?eBayes)

Practical

Analysis of differential expression

Linear models – the *limma* approach

Analysis of differential expression

Linear models – the *limma* approach

- **Exercise 6**: We first need to choose the biological groups of interest that we want to use to fit the model. In this case the factor we want to use is the *genotype*. Remember that we are comparing ataxin -/- mice (KO) to wild type mice (WT).

```
> library("limma")
> groups = pData(cAEset)[, fac]
```

- The names of the groups have to start with a letter, contain one word only and be without special character.

```
> groups[groups == "wild_type"] = "WT"
> groups[groups == "ataxin 1 -/-"] = "KO"
> f = factor(groups)
```

- **f** is a factor containing the groups WT and KO. We will use this factor to create the design matrix that will be used to fit a linear model.

Analysis of differential expression
Linear models – the *limma* approach

- **Exercise 7**: Create the design matrix, which indicates which RNA sample has been hybridized to each array

```
> design = model.matrix(~f)
> colnames(design) = c("WTvsRef", "KOvsWT")
> design
```

	WTvsRef	KOvsWT
1	1	1
2	1	0
3	1	0
4	1	1
5	1	1
6	1	0
7	1	0
8	1	1
9	1	1

Here the first coefficient estimates the difference between wild type and the reference for each probe while the second coefficient estimates the difference between mutant and wild type.

The matrix indicates which coefficients apply to each arrays. For the first array the fitted value will be just the WTvsRef coefficient, which is correct. For the second and third array. The fitted value will be *WTvsRef + KOvsWT,* which is equivalent to mutant vs reference.

Analysis of differential expression
Linear models – the *limma* approach

- **Exercise 8***:* Differentially expressed genes can be found by fitting a linear model (see function **lmFit**) for each gene, given the experimental design, and then computing moderated *t*-statistics of differential expression (together with moderated F-statistic and log-odds of differential expression - see ?eBayes for more information)

```
> fit = lmFit(cAEset, design)
> fit2 = eBayes(fit)
```

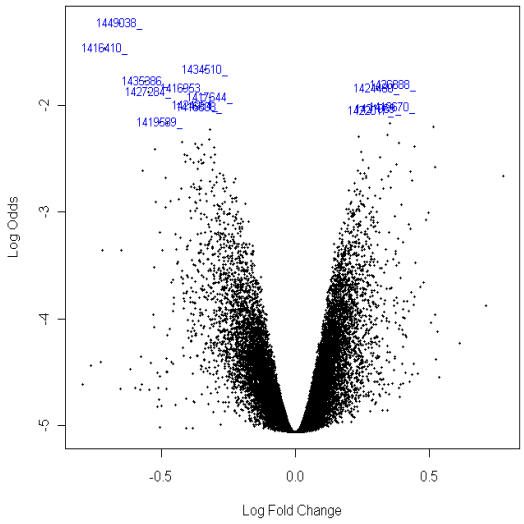
- In this example, there is no need to create a contrast matrix because the comparison of interest is already built into the fitted model

- Visualize your results using the function **volcanoplot**

```
> volcanoplot(fit2, coef="KOvsWT“, highlight=15)
```

Analysis of differential expression

Linear models – volcano plot



- A volcano plot is constructed by plotting the negative log of the *p*-value on the y-axis (usually base 10).
- This results in data points with low p-values (highly significant) appearing towards the top of the plot.
- The x-axis is the log of the fold change between the two conditions. The log of the fold-change is used so that changes in both directions (up and down) appear equidistant from the center.
- Plotting points in this way results in two regions of interest in the plot: those points that are found towards the top of the plot that are far to either the left- or the right-hand side.

Analysis of differential expression

Linear models – the *limma* approach

- **Exercise 9**: Extract a table of the top differentially expressed genes using the function **topTable** and then subset its output by selecting only the genes with a *p*-value < 0.001

- The function **topTable** is also used to perform hypothesis testing and adjust the *p*-values for multiple testing (see ?topTable arguments)

```
> results = topTable(fit2, coef = "KOvsWT", adjust = "BH", number =
  nrow(cAEset))
```

```
> topgenes = results[results[, "P.Value"] < 0.001, ]
```

Analysis of differential expression
Linear models – the *limma* approach

- **Exercise 10**: Draw a heatmap of the top differentially expressed genes that we have just extracted. First we need to get the expression values for the selected genes

```
> m = exprs(cAEset[topgenes[, "ID"], ])
> colnames(m) = 1:9
```

- **m** is a matrix containing the normalized expression values for the top differentially expressed genes

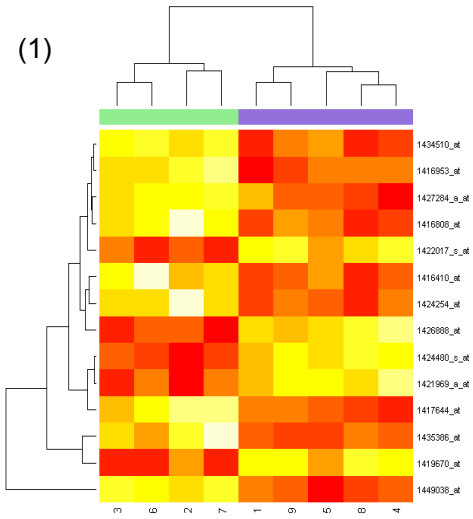
- Then we assign a different colour code (**col**) to the two biological groups of interest, WT and KO, and we draw the heatmap

```
> colours = c("lightgreen", "mediumpurple")
> col = colours[f]
> heatmap(m, ColSideColors = col, margin = c(5, 8))
```

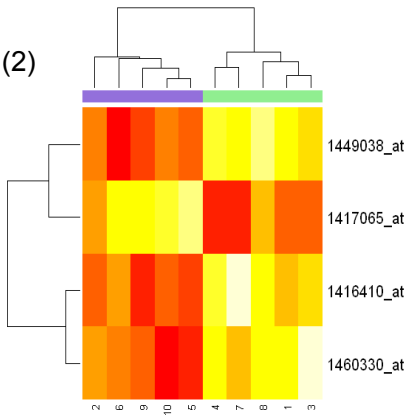

EMBL-EBI

Analysis of differential expression - results
Outlier effect on moderated *t*-statistics

(1)



(2)



(1) 9 sample (without #1)
(2) 10 samples

	# Genes	
	P < 0.01	P < 0.001
10 samples	34	4
Without array 1	190	14

EMBL-EBI

The analysis pipeline

Importing and accessing probe-level data

Exploratory Analysis / Quality Assessment

Preprocessing:

- Background adjustment
- Normalization
- Summarization

Quality assessment

Analysis of Differential expression

- Classical statistical tests
- Linear models

Functional analysis of large gene lists

EMBL-EBI

Gene lists functional analysis

- List of differentially expressed genes determined by:
  - extreme values of the univariate statistic, x
  - adjusted in some way for multiple comparison
- Many problems with this:
  - interpreting a long list of genes can be a daunting task
  - introduces an artificial distinction - ‘differentially expressed’
  - a weak effect in a group of genes may be missed when each gene is considered individually, but it may be captured when they are considered together

Huang et al (2009) *NAR* v37(1):1-13
68 different tools!

Published online 25 November 2008 *Nucleic Acids Research, 2009, Vol. 37, No. 1* **1–13**
doi:10.1093/nar/gkn923

SURVEY AND SUMMARY

Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists

Da Wei Huang, Brad T. Sherman and Richard A. Lempicki*

Laboratory of Immunopathogenesis and Bioinformatics, Clinical Services Program, SAIC-Frederick, Inc., National Cancer Institute at Frederick, Frederick, MD 21702, USA

Received September 10, 2008; Revised October 24, 2008; Accepted November 3, 2008

ABSTRACT

Functional analysis of large gene lists, derived in most cases from emerging high-throughput genomic, proteomic and bioinformatics scanning approaches, is still a challenging and daunting task. The gene-annotation enrichment analysis is a promising high-throughput strategy that increases the likelihood for investigators to identify biological processes most pertinent to their study. Approximately 68 bioinformatics enrichment tools that are currently available in the community are collected in this survey. Tools are uniquely categorized into three major classes, according to their underlying enrichment algorithms. The comprehensive collections, unique tool classifications and associated questions/issues will provide a more comprehensive and up-to-date view regarding the advantages, pitfalls and recent trends in a simpler tool-class level rather than by a tool-by-tool approach. Thus, the survey will help tool designers/developers and experienced end users understand the underlying algorithms and pertinent details of particular tool categories/tools, enabling them to make the best choices for their particular research interests.

INTRODUCTION

The traditional biological research approaches typically study one gene or a few genes at a time. In contrast, high-throughput genomic, proteomic and bioinformatics scanning approaches (such as expression microarray, promoter microarray, proteomics, ChIP-on-CHIPs, etc.) are emerging as alternative technologies that allow investigators to simultaneously measure the changes and regulation of genome-wide genes under certain biological conditions. Those high-throughput technologies usually generate large 'interesting' gene lists as their final outputs. However, the biological interpretation of large, 'interesting' gene lists (ranging in size from hundreds to thousands of genes) is still a challenging and daunting task. Over the last few decades, bioinformatics methods, using the biological knowledge accumulated in public databases [e.g. Gene Ontology (1)], make it possible to systematically dissect large gene lists in an attempt to assemble a summary of the most enriched and pertinent biology. A number of high-throughput enrichment tools, including, but not limited to Onto-Express, MAPPFinder, GoMiner, DAVID, EASE, GeneMerge and FuncAssociate, etc. (2–10), were independently developed during 2002 and 2003 as initial studies to address the challenge of functionally analyzing large gene lists. Since then, the enrichment analysis field has been very productive, resulting in more, similar tools becoming publicly available. In 2005, approximately 14 such tools were collected and reviewed by Khatri *et al.* (11) and by Curtis *et al.* (12), respectively. The activity in the field has continually grown stronger as the number of new enrichment tools (with distinct new ideas and features) has significantly increased. Approximately 68 such tools have been collected in this survey (2–10,13–73) (Table 1 and Supplementary Data 1).

During the past several years, bioinformatics enrichment tools have played a very important and successful role contributing to the gene functional analysis of large gene lists for various high-throughput biological studies, which is clearly evidenced by thousands of publications citing these tools (based on Google Scholar as of September 2008). However, these bioinformatics

Table 1. List of 68 enrichment tools

Enrichment tool name	Year of release	Key statistical method	Category
FunSpec	2002	Hypergeometric	Class I
Onto-express	2002	Fisher's exact; hypergeometric; binomial; chi-square	Class I
EASE	2003	Fisher's exact (modified as EASE score)	Class I
FatGO/FatWise/FatGO+	2003	Fisher's exact	Class I
FuncAssociate	2003	Fisher's exact	Class I
GARBAN	2003	Hypergeometric	Class I
GeneMerge	2003	Hypergeometric	Class I
GoMiner	2003	Fisher's exact	Class I
MAPPFinder	2003	Z-score; hypergeometric	Class I
CLENCH	2004	Hypergeometric; chi-square; binomial	Class I
GO::TermFinder	2004	hypergeometric	Class I
GOAL	2004	Permutation	Class I
GOArray	2004	Hypergeometric; Z-score; permutation	Class I
GOStat	2004	Fisher's exact; chi-quare	Class I
Goliath	2004	Chi-square	Class I
OntologyTraverser	2004	Hypergeometric; Fisher's exact	Class I
THEA	2004	Hypergeometric	Class I
BiNGO	2005	Hypergeometric; binomial	Class I
FACT	2005	Adopt GeneMerge and GO::TermFinder statistical modules	Class I
gfinder	2005	Fisher's exact	Class I
Golnar	2005	Hypergeometric	Class I
GOCluster	2005	Hypergeometric	Class I
GOSSIP	2005	Fisher's exact	Class I
L2L	2005	Binomial; hypergeometric	Class I
WebGestalt	2005	Hypergeometric	Class I
BayGO	2006	Bayesian; Goodman and Kruskal's gamma factor	Class I
eGOn/GeneTools	2006	Fisher's exact	Class I
Gene Class Expression	2006	Z-statistics	Class I
GOALIE	2006	Hidden Kripke model	Class I
GOFFA	2006	Fisher's inverse chi-square	Class I
GOLEM	2006	Hypergeometric	Class I
JProGO	2006	Fisher's exact; Kolmogorov-Smirnov test; student's *t*-test; Wilcoxon's test; hypergeometric	Class I
PageMan	2006	Fisher's exact; chi-square; Wilcoxon	Class I
STEM	2006	Hypergeometric	Class I
WEGO	2006	Chi-square	Class I
EasyGO	2007	Hypergeometric; chi-square; binomial	Class I
g:Profiler	2007	Hypergeometric	Class I
ProbCD	2007	Yule's Q; Goodman-Kruskal's gamma; Cramer's T	Class I
GOEAST	2008	Hypergeometric	Class I
GOHyperGAll	2008	Hypergeometric	Class I
CatMap	2004	Permutations	Class II
Godist	2004	Kolmogorov–Smirnov test	Class II
GO-Mapper	2004	Gaussian distribution; FQ-score	Class II
iGA	2004	Permutations; hypergeometric; *t*-test; Z-score	Class II
GSEA	2005	Kolmogorov–Smirnov-like statistic	Class II
MEGO	2005	Z-score	Class II
PAGE	2005	Z-score	Class II
T-profiler	2005	*t*-Test	Class II
FuncCluster	2006	Fisher's exact	Class II
FatKnn	2007	Fisher's Exact	Class II
FINA	2007	Fisher's exact	Class II
GAzer	2007	Z-statistics; permutation	Class II
GeneTrail	2007	Hypergeometric; Kolmogorov-Smirnov	Class II
MetaGP	2007	Z-score	Class II
Ontologizer	2004	Fisher's exact	Class III
POSOC	2004	PORET (a discrete math: finite partially ordered set)	Class III
topGO	2006	Fisher's exact	Class III
GO-2D	2007	Hypergeometric; binomial	Class III
GENECODIS	2007	Hypergeometric; chi-square	Class III
GOSim	2007	Resnik's similarity	Class III
PuB	2008	Percent	Class III
ProfCom	2008	Greedy heuristics	Class III
GOTM	2004	Hypergeometric	Class I,II
ermineJ	2005	Permutations; Wilcoxon rank-sum test	Class I,II
DAVID	2003	Fisher's Exact (modified as EASE score)	Class I,III
GOToolBox	2004	Hypergeometric; Fisher's exact; Binomial	Class I,III
ADGO	2006	Z-statistic	Class II,III
FunNet	2008	Unclear	Unclear

EMBL-EBI

Classification of Enrichment Tools

- *Singular* Enrichment Analysis (SEA)
- *Modular* Enrichment Analysis (MEA)
- *Gene Set* Enrichment Analysis (GSEA)

EMBL-EBI

Singular Enrichment Analysis (SEA)

- Takes user's pre-selected gene list (e.g. *t*-test and adjusted *p*-value <= 0.05 & fold change => 1.5) and then iteratively test the enrichment of each annotation term one-by-one, in a linear mode
- The individual, enriched annotation terms passing the enrichment *p*-value threshold are reported in a tabular format ordered by the enrichment probability
- May be generated from any type of high-throughput genomic studies (e.g. Microarray, ChIP-on-chip, ChIP-seq, SNP array, exon array, large scale sequence, etc)
- Output can be large and is heavily dependent on the pre-selected gene lists

Singular Enrichment Analysis (SEA)
Gene Ontology database

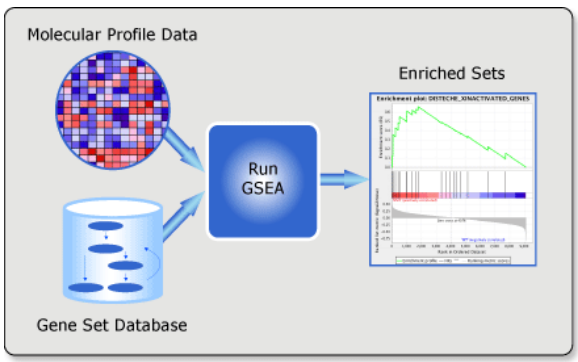
- GO is a controlled vocabulary used to describe the biology of a gene product in any organism
- There are 3 independent sets of vocabularies, or ontologies, that describe:
  - the molecular function of a gene product
  - the biological process in which the gene product participates and
  - the cellular component where the gene product can be found
- GO Consortium (2000), Gene Ontology: tool for the unification of biology, *Nat. Genetics* 25:25-29

Modular Enrichment Analysis (MEA)

- Improves discovery sensitivity and specificity by considering inter-relationships of GO terms in the enrichment calculations
- Researcher can take advantage of term–term relationships, in which joint terms may contain unique biological meaning for a given study, not held by individual terms
- Tools using MEA: GoToolBox and DAVID

Gene Set Enrichment Analysis (GSEA)

- Computational method that determines whether an *a priori* defined set of genes shows statistically significant, concordant differences between two biological states (e.g. phenotypes)



http://www.broadinstitute.org/gsea/

Gene Set Enrichment Analysis (GSEA)

- Uses a **'no cut-off' strategy** – takes all genes from a microarray experiment without selecting significant genes (i.e. expression difference, signal to noise ratio, a correlation metric, etc)
- Allows the minimally changing genes to contribute to the enrichment analysis
- Suitable for **pair-wise biological studies** (e.g. disease versus control)

Gene Set Enrichment Analysis (GSEA)
Limitations

- Requires a summarized biological value (e.g. fold change) but it can be a difficult task to summarize many biological aspects of a gene into one meaningful value when the biological study and genomic platform are complex
- Assumes that genes with large regulations (e.g. fold changes) are contributing more to the biology, but this is not always true
- Limited to comparisons of two condition groups (e.g.'disease vs. control', 'wt vs. ko', etc)

Practical

Gene Set enrichment analysis

GSEABase package

EMBL-EBI

Gene Set enrichment analysis
GSEABase package

- The GSEABase package contains the tools to create gene sets with GO terms, KEGG pathways, chromosome location and others
- **Exercise 1**: We first need to load the **GSEABase** library and the **annotation** package for cAEset

```
> library("GSEABase")
> annotation(cAEset)
> library("moe430a.db")
```

- **Exercise 2**: Organize all the gene identifiers into gene sets based on their presence into KEGG pathways, using the function **GeneSetCollection**. The incidence matrix summarizes shared membership of gene identifiers across (pairs of) gene sets

```
> gsc = GeneSetCollection(cAEset, setType = KEGGCollection())
> Am = incidence(gsc)
> dim(Am)
```


EMBL-EBI

Gene Set enrichment analysis
GSEABase package

- **Exercise 3**: We can subset our cAEset so that only the genes belonging to a KEGG pathway are kept and are stored in a smaller ExpressionSet, **nsF**

```
> nsF = cAEset[colnames(Am), ]
```

- **Exercise 4**: Compute a simple *t*-test only on each gene in **nsF.** To do this we load the *genefilter* library and use the function **rowttests**

```
> library("genefilter")
> rtt = rowttests(nsF, fac)
> rttStat = rtt$statistic
```


Gene Set enrichment analysis
GSEABase package

- **Exercise 5**: We are interested in the pathways in which at least ten genes are present; these can be extract from **Am**

```
> selectedRows = (rowSums(Am) > 10)
> Am2 = Am[selectedRows, ]
```

- We compute the *t*-test per pathway and adjust the *t*-values by the size of the pathway

```
> tA = as.vector(Am2 %*% rttStat)
> tAadj = tA/sqrt(rowSums(Am2))
> names(tA) = names(tAadj) = rownames(Am2)
```

Gene Set enrichment analysis
GSEABase package

- **Exercise 6**: We can retrieve the pathway with the highest difference between the KO for the ataxin gene and the WT

```
> library(KEGG.db)
> smPW = which(tAadj == min(tAadj))
> pwName = KEGGPATHID2NAME[[names(smPW)]]
> pwName
```

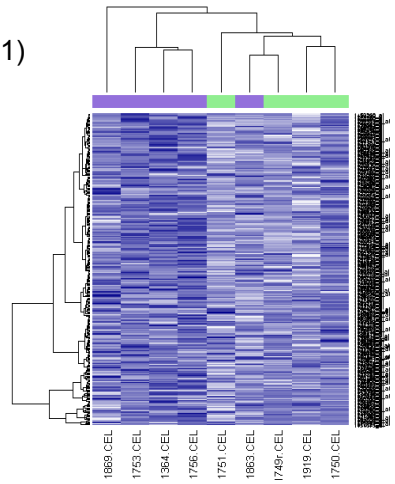
- **Exercise 7**: We can draw a heatmap of genes belonging to this pathway using the **KEGG2heatmap** function

```
> par(mfrow=c(1,1))
> KEGG2heatmap(names(smPW), nsF, "moe430a", col =
    colorRampPalette(c("white",
+ "darkblue"))(256), ColSideColors = col, margin = c(5, 8))
```

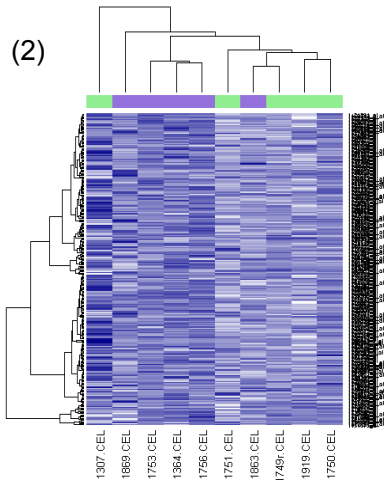

EMBL-EBI

Gene set enrichment analysis - results
Outlier effect on most enriched KEGG pathway

(1)



(2)



(1) 9 sample (without #1)
(2) 10 samples

	Neuroactive ligand-receptor interaction	
	# Significant genes	Corrected t-value
10 samples	4	-5.65
9 samples	23	-11.53

That's all folks!

Questions?

EMBL-EBI

References

Books

- Bioconductor Case Studies. Series: Use R! Hahne, F; Huber, W.; Gentleman, R.; Falcon, S. Springer, 2008, 284 p.
- Microarray Technology in Practice. Russell, S.; Meadows, L.A., Russell, R.R. Academic Press, 2008, 464 p.
- Bioinformatics and Computational Biology Solutions Using R and Bioconductor. Series: Statistics for Biology and Health, Gentleman, R.; Carey, V.; Huber, W.; Irizarry, R.; Dudoit, S. (Eds.). Springer, 2005, 460 p.

Articles

- Churchill G. (2003). Fundamentals of experimental design for cDNA microarrays. Nature Genetics review 32:490-495.
- Huang da W et al. (2009) Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. Nucleic Acids Res 37:1-13
- Kauffmann A, Huber W. (2010) Microarray data quality control improves the detection of differentially expressed genes. Genomics, in press
- Quackenbush J. (2001). Computational analysis of microarray data. Nat Rev Genet., 2(6):418-427
- Ritchie ME et al. (2007) A comparison of background correction methods for two-colour microarrays. Bioinformatics 23: 2700–2707
- Smyth GK. "Linear models and empirical bayes methods for assessing differential expression in microarray experiments". Stat Appl Genet Mol Biol, 2004;3:Article3
- Subramanian A et al. (2005) Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. PNAS 102, 15545-15550