

Applera microarrays quality control and data filtering

Francesca Cordero and Raffaele Calogero

April 25, 2006

Contents

1	Introduction	1
1.1	Applied Biosystem arrays generalities	1
2	Data structure	2
2.1	Methods of the applera-class	4
3	Subsetting applera objects	4
4	Quality Control through Data Exploration	5
4.1	Plotting probe data	5
4.2	Quality of replicates	6
5	Data normalization	8
6	Data filtering	8

1 Introduction

The `applera` package is an environment for data analysis and exploration of Applied Biosystem AB1700 oligonucleotide arrays.

1.1 Applied Biosystem arrays generalities

The Applied Biosystem arrays are single channel oligo arrays developed for human, mouse and rat genomes. Probes are 60mers oligonucleotides. Probes are located within the last 1500 bp of the transcript end. Whenever possible, a unique probe is designed to detect all different isoforms of a gene. Probe mapping to human, mouse and rat genes can be extracted from the PANTHER database: <http://panther.appliedbiosystems.com/>

In these arrays chemiluminescence (CL) is used to measure gene expression and for quality control. On the other hand fluorescence (FL) is used to auto-grid, and normalize every feature in a way independent from gene expression signal. Chemiluminescence signal is generated by cDNA/cRNA labelling with digoxigenin (DIG), by RT or by RT/IVT approach. The quantification of the amount of DIG label cDNA/cRNA hybridized on probes is based on the use of an anti-digoxigenin antibody labelled with an alkaline phosphatase. Feature (spot) characteristics (size, homogeneity, etc.) is evaluated using a unique 24mer fluorescence dye

(Liz Dye labeled oligo), co-spotted with gene specific oligo. The Liz Dye signal is used as internal control to normalize chemiluminescence.

Applera arrays are characterized by the presence of:

Labeling controls , which are used to monitor enzyme activity and DIG incorporation during the labeling protocols (RT - bacterial genes -, IVT - linearized plasmids -).

Hybridization controls , which are used to monitor mixing, stringency, and washing during the array hybridization protocol (pre labeled DIG target).

Chemiluminescent controls , which are used to demonstrate that the CL reaction chemistry is performing well during the assay (DIG labeled oligo co-spotted).

2 Data structure

The `aplera` package contains the `readAp` function that allows to read tab delimited files generated by the AB1700 instruments. Each array readout should be described by a tab delimited file containing the following columns:

ProbeID , the probe array identifier

GeneID , primary Ids associated to spotted probes

Signal , the CL signal normalized by the FL signal over the feature integration aperture. See AB1700 manual.

SDEV , which is an estimate of measurement uncertainty of Normalized Signal. See AB1700 manual.

CV , which is the fraction of uncertainty in the signal. Cv indicates the predicted spread of feature signal. See AB1700 manual.

S/N , which expresses the confidence of feature detectability. See ABI1700 manual.

Flags , which is a numeric code that identifies conditions for each feature. It allows to eliminate possibly problematic data. Features with flags greater than 100 might have quality issues. See AB1700 manual.

This array-specific tab delimited file can be easily created using the AB1700 software.

The function `readAp` needs two arguments:

- A `phenoData` file containing as rownames of the covariates the names of the files to be uploaded by the function.
- The organism identifier ("Hs.v1" or "Hs.v2" for human, "Mm" for mouse or "Rn" for rat).

The `readAp` will read all the files present in the R working directory using the information available in the `phenoData` file. The function will select the correct size of the array on the basis of the organism identifier and it will create an instance of the `aplera` class. The slots of this object are:

Organism , an `character` object

Geneid , an vector object

Signal , an `exprSet` object

Sdev , an `exprSet` object

Cv , an `exprSet` object

Sn , an `exprSet` object

Flags , an `exprSet` object

Ctrl , an list object

All slots up to **Flags** refer to gene probes as instead the **Ctrl** slot refers to all Applera controls, which can be used for quality control, although fuctions performing quality control based on **Ctrl** features are not yet implemented.

An object of the `applera` class (`test`) is available as data file:

```
> library("applera")
```

```
Loading required package: Biobase
Loading required package: affy
Loading required package: affyio
Loading required package: genefilter
Loading required package: survival
Loading required package: splines
Loading required package: limma
Loading required package: geneplotter
Loading required package: annotate
KernSmooth 2.22 installed
Copyright M. P. Wand 1997
Loading required package: RColorBrewer
Loading required package: prada
Loading required package: grid
Loading required package: combinat
```

```
> data(test)
```

```
> test
```

This is an instance of `applera-class`

Slots can be accessed using:

`signal`, `sdev`, `cv`, `sn`, `flags` methods

Each slot is an `exprSet` object made of

1000 genes

303 controls

6 samples

The `test` is a two class experiment performed on the AB1700 mouse array (condition `wk15`: 3 biological replicates; condition `wk19`: 3 biological replicates).

2.1 Methods of the `applera`-class

The `applera` class allows to access to the slot names using the following methods: *organism*, *geneid*, *signal*, *sdev*, *cvAp*, *sn*, *flags*, *ctrl*,

To access to the `exprSet` object containing the log2 CL probe intensities you can use:

```
> signal(test)
```

Expression Set (`exprSet`) with

1000 genes

6 samples

phenoData object with 3 variables and 6 cases

varLabels

num: read from file

sample: read from file

wk: read from file

As instead to access to the `phenoData` associated to the signal you can use:

```
> pData(signal(test))
```

	num	sample	wk
MA001I9	6	11	15
MA000V0	7	30	15
MA0019Y	8	31	15
MA000UR	9	13	19
MA001I7	10	14	19
MA001D0	11	15	19

3 Subsetting `applera` objects

Two function have been implemented to allow probe-specific and experiment-specific subsetting:

- `subExp`, this function subsets an `appleraSet` given a vector of `sampleNames`. It needs two parameters: the `applera` object and a character vector containing the names of the arrays to be extracted.

```
> subex <- subExp(test, rownames(pData(signal(test)))[c(1, 3, 5)])
```

- `subGenes`, this function subsets an `appleraSet` given a vector of `geneNames`. It needs two parameters: the `applera` object and a character vector containing the `geneNames` to be extracted.

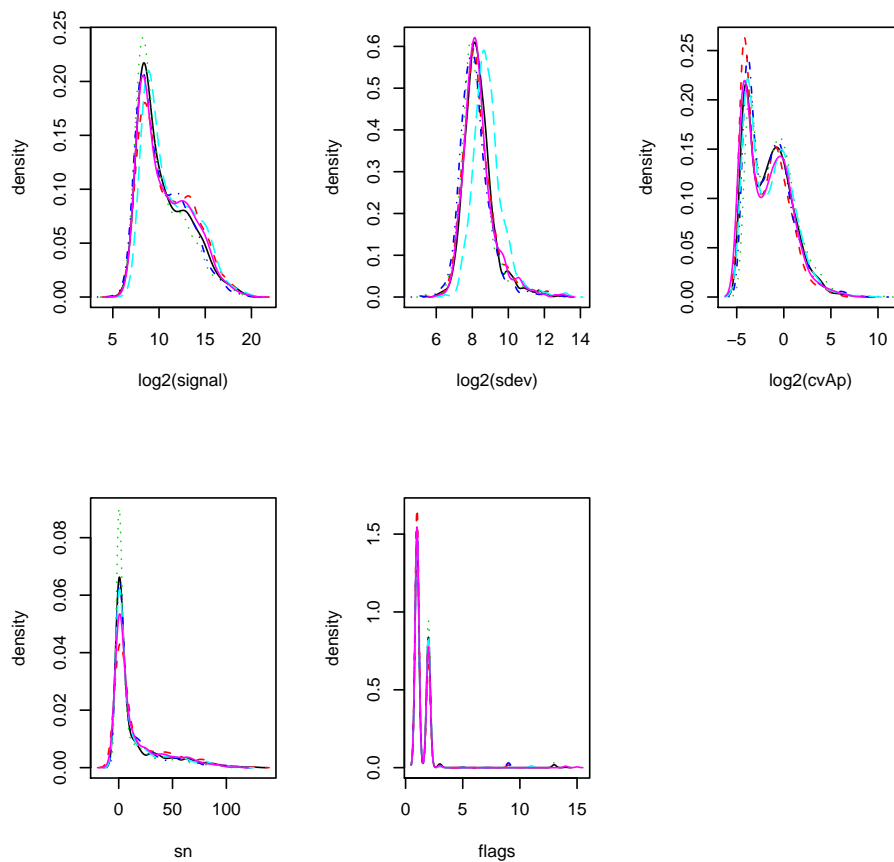
```
> subgx <- subGenes(test, geneNames(signal(test))[1:10])
```

4 Quality Control through Data Exploration

4.1 Plotting probe data

Several of the functions for plotting probe data are useful for diagnosing problems with the data. The plotting functions *boxplot* and *hist* have methods for **applera** objects. The method *hist* allows to generate density histograms for all the slots of an **applera** object.

```
> hist(test)
```



The method *boxplot* produces box-and-whisker plot(s) of all slots of an **applera** object. The method *mvaAp* allows the generation of mva.pairs plots.

```
R> samples <- sampleNames(signal(test))[which(signal(test)$wk==15)]  
R> mvaAp(subExp(test, samples))
```

These functions can be particularly useful in diagnosing problems in replicate sets of arrays.

4.2 Quality of replicates

An important issue in microarray analysis is the quality of replicates. The r-squared coefficient, which represent the fraction of variance explained by a linear model, is the parameter most frequently used to evaluate replicates homogeneity. It varies from 1 to 0, where 1 indicates that the two sets of data are identical, while 0 indicates the absence of similarity between samples. Good replicates are usually characterized by an r-squared greater than 0.8. A more efficient way to identify subtle differences within replicates has been introduced by Irizarry (http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=pubmed&dopt=Abstract&list_uids=15846361). Irizarry approach, is based on the use of a reference array against which biological replicates are compared. The homogeneity within replicates is then evaluated at the level of absolute expression fold change variation of each gene with respect to the reference. Each biological replicate is described as the sorted list (absolute fold change variation descending order) of all probe sets. Homogeneous replicates having at the top rank positions (e.g. 100-1000) a concordance in probe set composition of at least 0.4, correspond to a r-squared between replicates greater than 0.9.

This approach has been implemented in a function called *CATPlots*. The *CATPlots* accepts `applera`, `exprSet` objects and numerical matrices. It has been designed to allow the quality evaluation for two groups experiments. In the case of the `applera` object `test` the quality of wk19 condition can be compared with that of wk15 condition and viceversa.

From the *CATPlots* it is clear that the wk19, left panel, are homogeneous within each other (curves with the same colour). However, replicate quality changes depending on the wk15 reference sample used (the reference array names are located in the upper part of each panel and are associate to specific plotting colors). The option `upsideDown=T` allows to revert the comparison (right panel). From this plot it is clear that the quality of the wk15 group is much lower of wk19. It is interesting to note that the r-squared for the wk15 group is greater of 0.85, suggesting that *CATPlots* better highlight subtle differences between replicates.

```
> xx
```

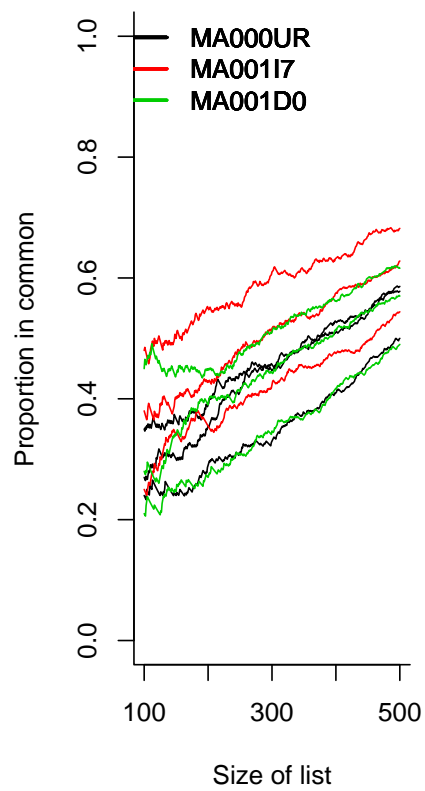
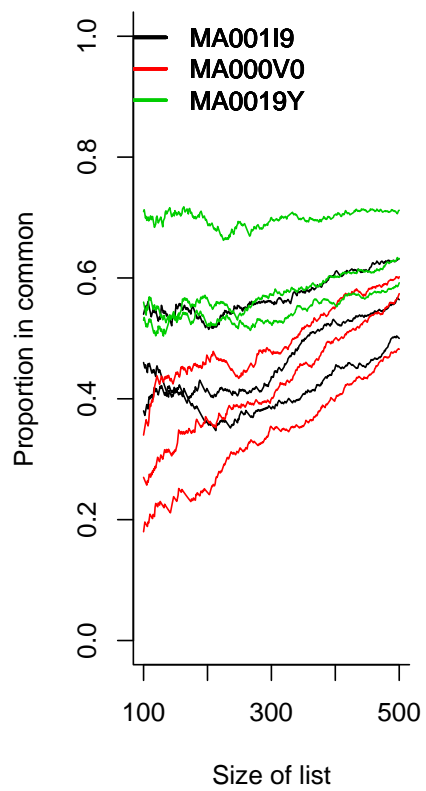
```
$resultR2Ctrl
```

	experiments	correlations
1	MA001I9 vs MA000V0	0.8673467
2	MA001I9 vs MA0019Y	0.8608140
3	MA000V0 vs MA0019Y	0.8418369

```
$resultR2Trt
```

	experiments	correlations
1	MA000UR vs MA001I7	0.8886683
2	MA000UR vs MA001D0	0.8878290
3	MA001I7 vs MA001D0	0.9210257

```
> xx <- CATPlots(test, c(0, 0, 0, 1, 1, 1), 100, 500, upsideDown = TRUE)
```



5 Data normalization

This is an important issue in microarray since it ensures that differences in intensities are indeed due to differential expression, and not to technical artifacts, and it is also required prior to any analysis which involves between-array comparisons of intensities. AB1700 instrument, after CL normalization via FL signal (see AB1700 manual), suggests as default normalization that intensities should be scaled so that each array has the same average value (constant normalization, in affy Bioconductor package). This approach, however, is less effective in the case of non-linear relationships between arrays, which are efficiently dealt with by the approaches of Li and Wong (http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=pubmed&dopt=Abstract&list_uids=11532216) and Bolstad (http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=pubmed&dopt=Abstract&list_uids=12538238).

From the normalization methods implemented in the affy Bioconductor package, two were implemented in the *aplara* package:

Cyclic loess - This approach stems from the M versus A plot, where M is the difference in log expression values and A is the average of those (http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=pubmed&dopt=Abstract&list_uids=11842121). A normalization curve is fitted to this M versus A plot by using loess, which is a method of local regression. The fits based on the normalization curve are subsequently subtracted from the M values. However, rather than being applied to two-color channels on the same array, as is done in the cDNA case, normalization is applied to probe intensities from two arrays at a time. An M versus A plot for normalized data should show a point cloud scattered about the $M = 0$ axis. To deal with more than two arrays, the method is extended to look at all distinct array pairwise combinations.

```
R> test.L<-normLoess(test) ##cyclic-loess normalization
```

Quantile normalization - The goal of the quantile method is to make the distribution of probe intensities for each of a set of arrays the same. The idea behind the method is that a quantile-quantile plot shows that the distribution of two data vectors is the same if the plot is a straight diagonal line, but not if it is other than a diagonal. This concept can be extended to n dimensions if more than two arrays are available. This suggests that an n set of data can be made to have the same distribution by projecting the points of the n dimensional quantile plot onto the diagonal. This normalization can be easily done by organizing the arrays to be normalized in a matrix, where columns represent the arrays and the rows gene-specific PMs. Each column is then sorted in a descending order. The means across rows are taken and assigned to each element of the rows. Columns are then reordered as the original matrix to generate the normalized set.

```
R> test.Q<-normQuantile(test) ##quantile normalization
```

6 Data filtering

The premise of this important step in microarray data analysis is removal of genes deemed to be not expressed according to some specific criterion under the control of the user ([http:](http://)

[//www.bepress.com/bioconductor/paper7/](http://www.bepress.com/bioconductor/paper7/)). It can also be used to eliminate genes that do not show sufficient variation in expression across all samples, since their little discriminatory power. Three filtering approaches have been implemented in **applera** package:

- The *apFilter* allows filtering on the bases of S/N or Flags. The function is an implementation of the pOverA *genefilter* function. The function needs an object of **applera** class, a p-value ranging 0 - 1. Where p-value indicates the minimum fraction of experiments satisfying a specific condition A (e.g. filtering for S/N, using p=1 and A=3; indicates that all experimental condition, for a specific gene, should be characterized by a S/N equal or greater than 3, indicating a confidence of 0.99 that the signal measurement is different from background. In the case of flags filtering A indicates the max flag value that can be accepted (High flag values indicate a low quality of the feature measurement)).

```
R>sn.subset <- apFilter(test, 0.5, 3, "sn")
R>flags.subset <- apFilter(test, 0.5, 2, "flags")
```

- The *iqrFilter* is an implementation of the IQR filtering described by von Heydebreck (<http://www.bepress.com/bioconductor/paper7/>).

```
R>iqr.sn.subset<-iqrFilter(sn.subset, 0.25)
```