Introduction to the Codelink package

Diego Diez

August 15, 2013

1 Introduction

This package implements methods to facilitate the preprocessing and analysis of Codelink microarrays. Codelink is a platform for the analysis of gene expression that uses 30 base long oligonucleotides. Codelink is currently owned by Applied Microarrays, Inc. (previously was GE Healthcare and Amersham). There is a proprietary software for reading scanned images, doing spot intensity quantification and some diagnostics. The software assigns quality flags (see Table 1) to each spot on the basis of a signal to noise ratio (SNR) computation (Eq: 1) and other morphological characteristics as irregular shape of the spots, saturation of the signal or manufacturer spots removed. By default, the software performs background correction (subtract) followed by median normalization. The results can be exported in several formats as XML, Excel, plain text, etc.

Currently two different approaches are maintained. The first uses a specially designed class `Codelink`. The second uses the new `CodelinkSet` class, which inherits from the `ExpressionSet` class in Biobase. Although the use of `CodelinkSet` is recommended, the `Codelink` infrastructure is maintained for backbard compatibility. In this document the `CodelinkSet` methods will be presented. Information about the old infrastructure can be found in the legacy vignette (Codelink_Legacy.pdf)

Flag	*Description*
G	Good signal (SNR ¿= 1)
L	Limit signal (SNR ¡ 1)
I	Irregular shape
S	Saturated signal
M	MSR spot
C	Background contaminated
X	User excluded spots

Table 1: Quality Flag description. SNR: Signal to Noise Ratio.

$$SNR = \frac{Smean}{(Bmedian + 1.5 * Bstdev)} \tag{1}$$

2 Reading data

Currently only data exported as plain text from Codelink software is supported. Unfortunately the Codelink exported text format can have arbitrary columns and header fields so depending of what has been exported reading it into a `Codelink` object may be more or less complicated. As a rule of thumb it is recommended to include in the exported files at least Spot_mean and Bkgd_median values so that background correction and normalization can be performed. In addition, Bkgd_stdev will be needed to compute the SNR. If Raw_intensity or Normalized_intensity columns are present then it is possible

Probe type	*Description*
DISCOVERY	Gene expression testing probes
POSITIVE	Positive control probes
NEGATIVE	Negative control probes
FIDUCIAL	Grid alignment probes
OTHER	Other controls and housekeeping gene probes

Table 2: Probe types for Codelink arrays.

to avoid background correction and/or normalization, and use the ones performed by the Codelink software. Feature_id column will be use to assign unique identifiers to each spot, so that CodelinkSet object can be read appropriately. To read codelink data:

```
# NOT RUN #
library(codelink)
# to read data as CodelinkSet object:
f = list.files(pattern = "TXT")
codset = readCodelinkSet(filename = f)
codset
# NOT RUN #
```

This assumes that the files have the extension "TXT" (uppercase) and are in the working directory.

```
data(codelink.exprset)
codelink.exprset

## CodelinkSet (storageMode: lockedEnvironment)
## assayData: 35129 features, 4 samples
##   element names: background, exprs, flag, snr
## protocolData: none
## phenoData
##   sampleNames: Sample-1 Sample-2 Sample-3 Sample-4
##   varLabels: sample
##   varMetadata: labelDescription
## featureData
##   featureNames: 1001 1002 ... 328112 (35129 total)
##   fvarLabels: probeName probeType ... meanSNR (5 total)
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: rwgcod
```

3 Background correction

If you have Spot_mean values Bkgd_median values the you can apply one of the several background correction methos interfaced. This is done by the function `codCorrect`. To see the different options look at ?codCorrect. For instance, if you want to apply *half* method you do:

```
codset = codCorrect(codelink.exprset, method = "half")
```

The default background correction method used is *half* and is based in the same method applied in the `limma` [1] package to two channel microarrays. In this method, the median background intensity (Bmedian) is subtracted from mean spot intensity (Smean) and any value smaller that 0.5 is shift to 0.5 to ensure no negative numbers are obtained that would prevent to transform the data into log scale. Other available methods are *none* that let the spot intensities untouched, *subtract* that is analog to the default method used in the Codelink software and *normexp* and interface to the method available in the `limma` package.

4 Normalization

Normalization of the background corrected intensities is done by the wrapper function `normalize` (or codNormalize()). The default method is *quantile* normalization that in fact call `normalizeQuantiles()` from `limma` package (allowing for NAs). There is also the possibility to use normalizeCyclicLoess() from `limma` [1] package that allow using weights. Finally, the *median* normalization allows to normalize using a method analog to the default method in the Codelink software. By default, `normalize` returns log2 intensity values. This can be controlled setting the parameter log.it to FALSE.

```
codset = codNormalize(codset, method = "quantile")
```

5 Plotting

There are some diagnostic plots available for the `Codelink` object. These are functions for producing MA plots (`plotMA`), scatterplots (`plotCorrelation`) and density plots (`plotDensities`). All functions use the available intensity value (i.e. Smean, Ri or Ni) to make the plot.

The functions `plotMA` and `plotCorrelation` can highlight points based on the Spot Type, which is the default behavior or using the SNR values. The mode is controlled with argument *label*. `plotCorrelation` requires arguments *array1* and *array2* to be set in order to select which arrays are going to be plotted. For `plotMA` if only *array1* is specified, the values are plotted againts a pseudoarray constructed with the mean of the probe intensities along all available arrays. M and A values are computed following equations 2 and 3.

$$M = Array2 - Array1 \tag{2}$$

$$A = \frac{Array2 + Array1}{2} \tag{3}$$

```
codPlot(codset)
```

The function `plotDensities` plot the density of intensity values of all arrays. It can plot only a subset of arrays if the *subset* argument is supplied.

```
codPlot(codset, what = "density")
```

When Logical_row and Logical_col columns are exported into they are stored into the *logical* slot. This information stores the physical location of each probe in the array, and can be used to plot pseudo images of the array intensities. To plot a pseudo image you should use:

```
codPlot(codset, what = "image")
```

It is possible to plot the background intensities (default), the spot mean, raw and normalized intensities and the SNR values. This images are useful to identify spatial artifact that may be affecting the analysis.

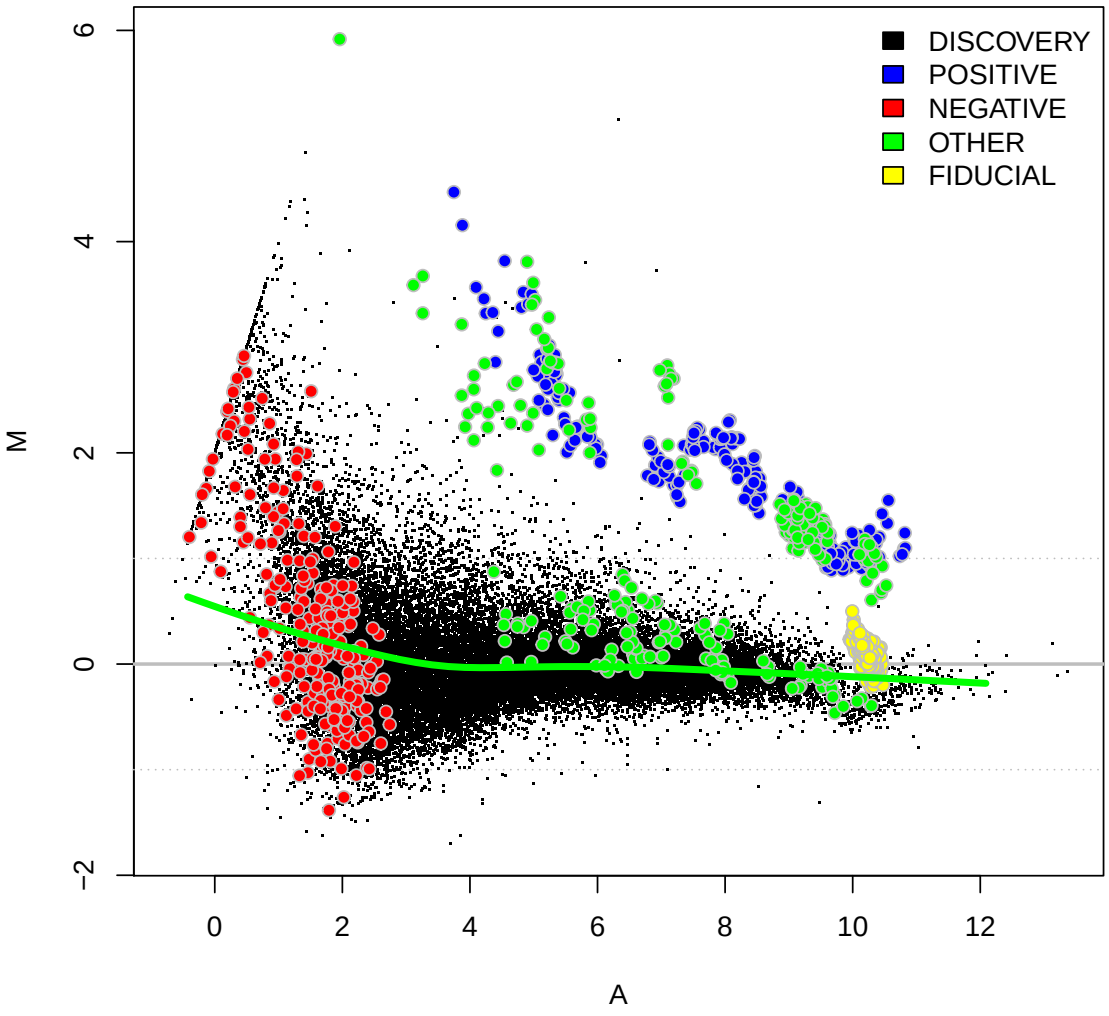

Figure 1: MA plot.

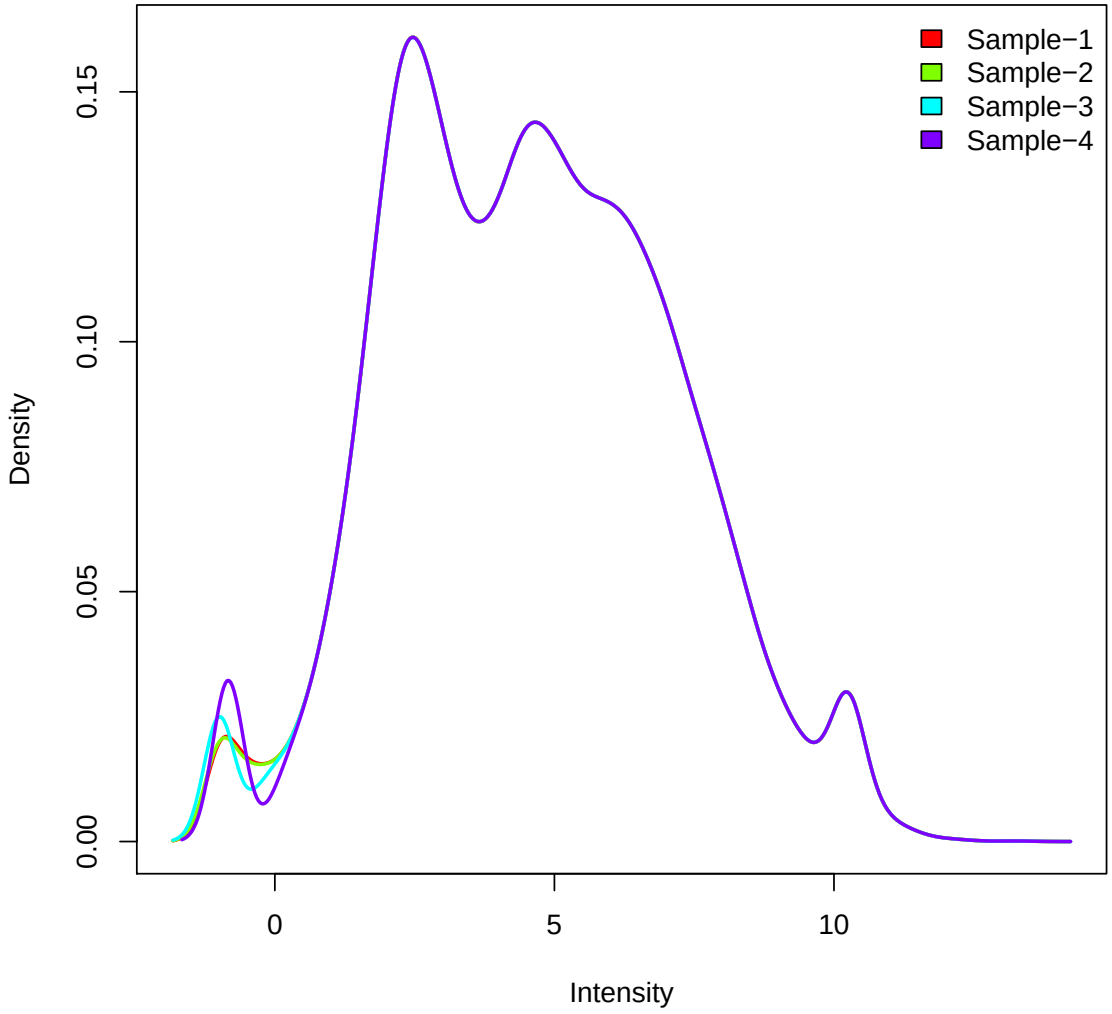

Figure 2: Density plot.

5.1 Fitting linear models

A typical analysis include the testing for differentially expressed probes between two populations. This can be performed using many different R/Bioconductor packages, but the limma package is one of the most popular options. Limma can readily use CodelinkSet objects, and can take advantage of weights generated during data reading.

```
fit = lmFit(codset, design = c(1, 1, 2, 2), weights = getWeight(codset))
fit2 = eBayes(fit)
topTable(fit2)
```

References

[1] Gordon K Smyth. *Limma: linear models for microarray data*, pages 397–420. Springer, New York, 2005.