Filtering genes using FLUSH

Stefano Calza
calza@med.unibs.it
http://www.biostatistics.it

July 21, 2009

Contents

1 Introduction

This document describes the flow-chart procedure for filtering genes in an experiment based on Affymetrix expression microarray data.

The FLUSH method for filtering genes is based on robust linear models fitted at the probe level.

After starting R, you should load the FLUSH.LVS.bundle package.

2 Fitting the robust linear models

The first thing to do is get an object of class *AffyBatch*. This can be done reading in CEL files using the function `ReadAffy` in the affy package. In this vignette we will use the data from Choe et al., available as an R data object here `http://www.biostatistics.it/library/FLUSH.RData`.

To read the data in R we can do as follows, where we assume that the "choe.RData" file is in the working directory.

```
> load("FLUSH.RData")
```

The we load the FLUSH.LVS.bundle package.

```
> library(FLUSH.LVS.bundle)
```

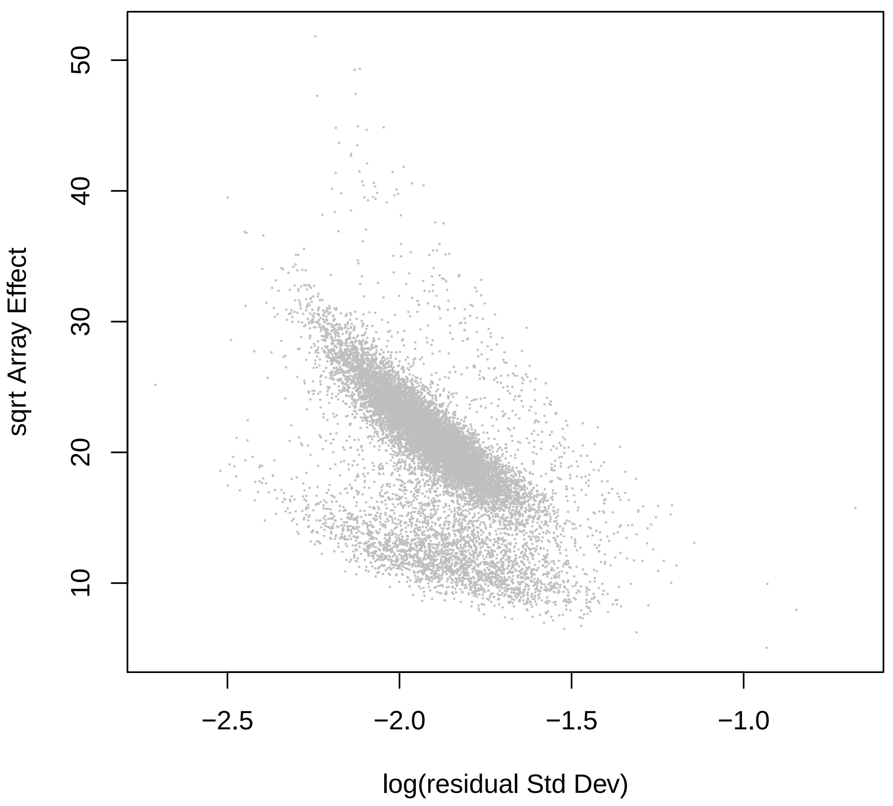
The first step is to compute the arrays effect and residual standard deviation. The following code do so.

```
> RA.fit = compute.RA(choe)
```

```
Fitting models
|                    |
|####################|
```

We can now visualize the two components.

```
> RAplot(RA.fit)
```



3 Filtering out features

The basic idea underlying the FLUSH algorithm is that features/genes with low array effect, that is with low variability among arrays, are most probably non interesting features and can be removed.

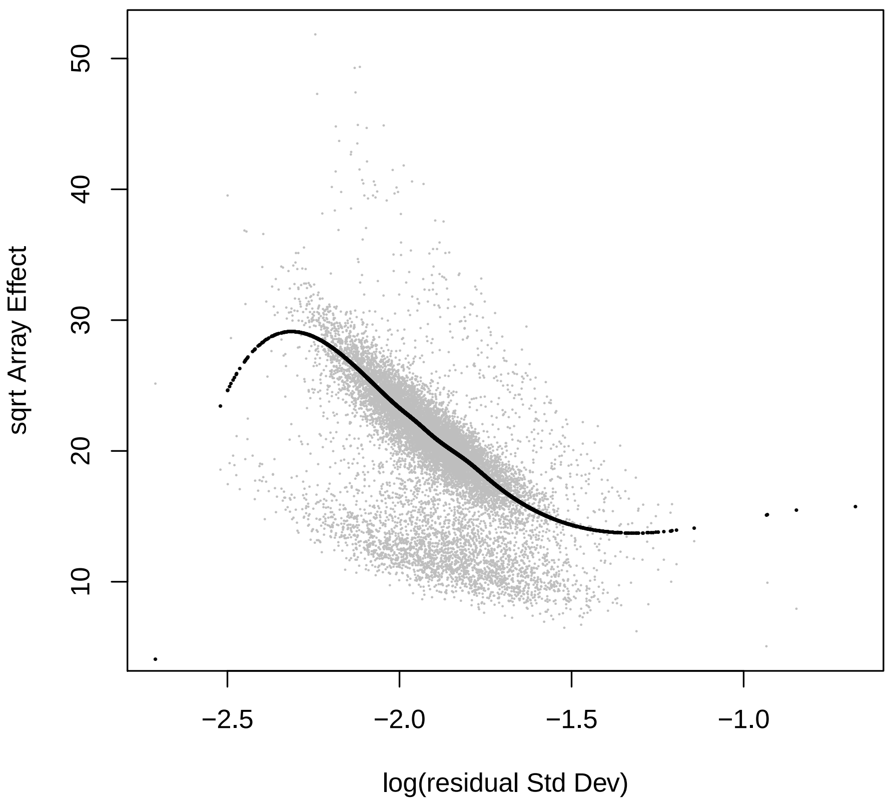
Let's assume we want to retain 40% of the features, thus we will exclude 60% of the probesets, namely those with lower inter-array variability. The following code creates an indicator object that can be used to subset the expression data.

```
> choe.fSet <- FlushSet(RA.fit, proportion = 0.6)
```

Any other choice of the proportion of genes to exclude can be specified with the `proportion` argument.

The function `FlushSet` returns an object of class *RA* that we can plot with the command `RAplot`. Setting the argument `add.rq` to TRUE, will overlay the quantile regression fit to the Residuals vs Array plot.

```
> RAplot(choe.fSet, add.rq = TRUE)
```



We are now ready to subset our expression data. The actuall expression matrix, or more generally the object of class *EpressionSet*, can be computed using any available algorithm, like RMA, MAS5, and so on.

Let's then compute expression values using the MAS5 algorithm as implemented in the function `mas5`

and subset the object to get a reduced expression matrix

```
> choe.flushed <- Flush(choe.MAS5, choe.fSet)
> dim(exprs(choe.flushed))
```

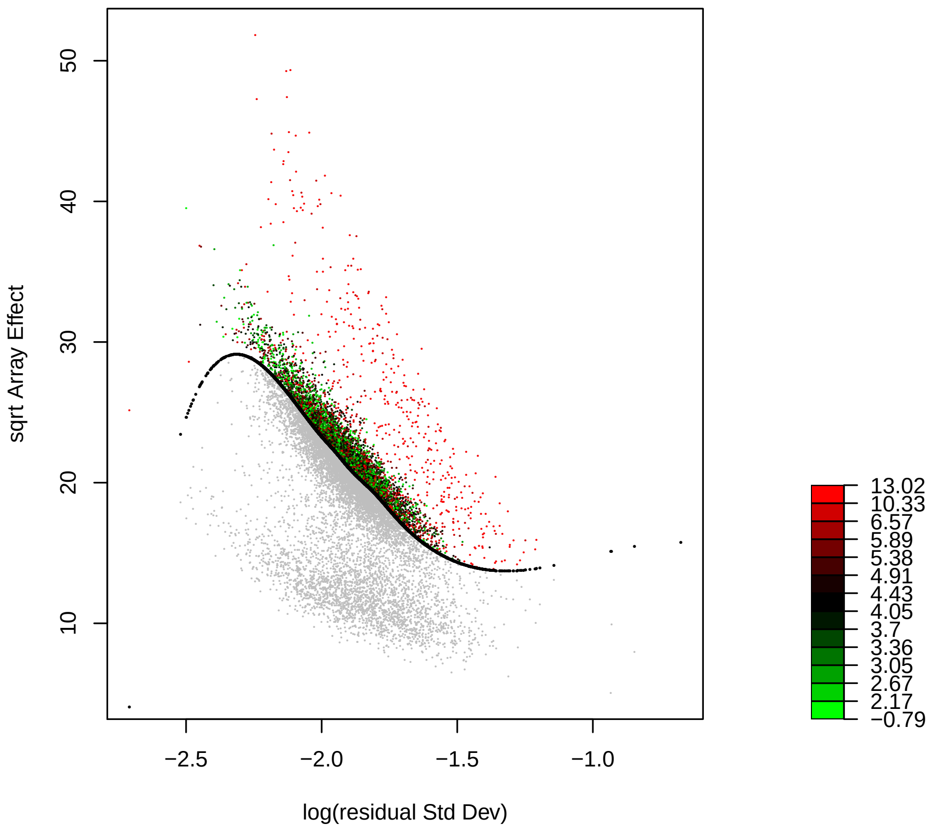
```
[1] 5607    6
```

The new *EpressionSet* object will now contain only a subset of the features.

Using the function `Flush` setting the argument `onlyExprs` to FALSE, will return an object of class *FLUSH* that can be used for plotting the selected genes in the array vs residual plot.

```
> choe.f2 <- Flush(choe.MAS5, choe.fSet, onlyExprs = FALSE)
```

```
> RAplot(choe.f2, heat = TRUE, add.rq = TRUE)
```



4 The easy way

If we are not interested in visualazing the Residual vs Array plot after choosing the proportion of genes, we can directly supply the function `Flush` an ExpressionSet object (e.g. choe.MAS5) and an object coming directly from `compute.RA` (namely `choe.RA` our example). In this case `Flush` will accept the same arguments as `FlushSet`, i.e. `proportion`, `lambda`, `delta` and `df`. The returned object will always be of class *ExpressionSet*.

```
> choe.rid <- Flush(choe.MAS5, RA.fit, proportion = 0.6)
> dim(exprs(choe.rid))

[1] 5607    6

> class(choe.rid)

[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```