Outlier detection with *nowaclean*

Einar Holsbø[*]

May 20, 2017

Abstract

This vignette shows the use of the *nowaclean R* package, which implements the standard operating procedure for detecting and removing technical outliers in the NOWAC microarray material.

1 nowaclean

1.1 Installation and loading

We'll be using the development version of *nowaclean*, which is hosted on GitHub.[1] To install from GitHub you need to install *devtools*.

```
install.packages("devtools")
```

Once you have installed *devtools*, you can use it to install *nowaclean* from its GitHub repository.

```
devtools::install_github("3inar/nowaclean", build_vignettes=T)
```

Once it is installed, you can use *nowaclean* like you would any other *R* package.

```
library(nowaclean)
```

To view *nowaclean* on github (for instance for bug reports, etc.), visit https://github.com/3inar/nowaclean.

2 Loading and Preprocessing

First to load the dataset; we have suppressed the huge text dump that happens when you load the *lumi* package:

```
library(lumi)  # Required to access LumiBatch objects
datapath <- "~/Downloads/sop_data.rda"
load(datapath)
```

This is a typical data set from the Norwegian Women and Cancer study. These are anonymized data that are freely available from the UiT Dataverse https://opendata.uit.no/. The reference is *Einar Holsbø, 2017, "Supporting data for "A Standard Operating Procedure for Outlier Removal in Large-Sample Epidemiological Transcriptomics Datasets"", doi:10.18710/FGVLKS, UiT Open Research Data Dataverse, DRAFT VERSION.*

[*] einar@cs.uit.no

[1] https://github.com

2.1 Remove blood type probes

In some situations we remove 38 probes related to genes in the human leukocyte antigen (HLA) system. These are usually expressed strongly and have high variance, which affects multivariate analyses. Specifically we have seen that they might dominate the variance-covariance pattern in the PCA transformation of the data, and as such other patterns might be obscured. The `blood_probes` function returns the nuIDs of these probes.

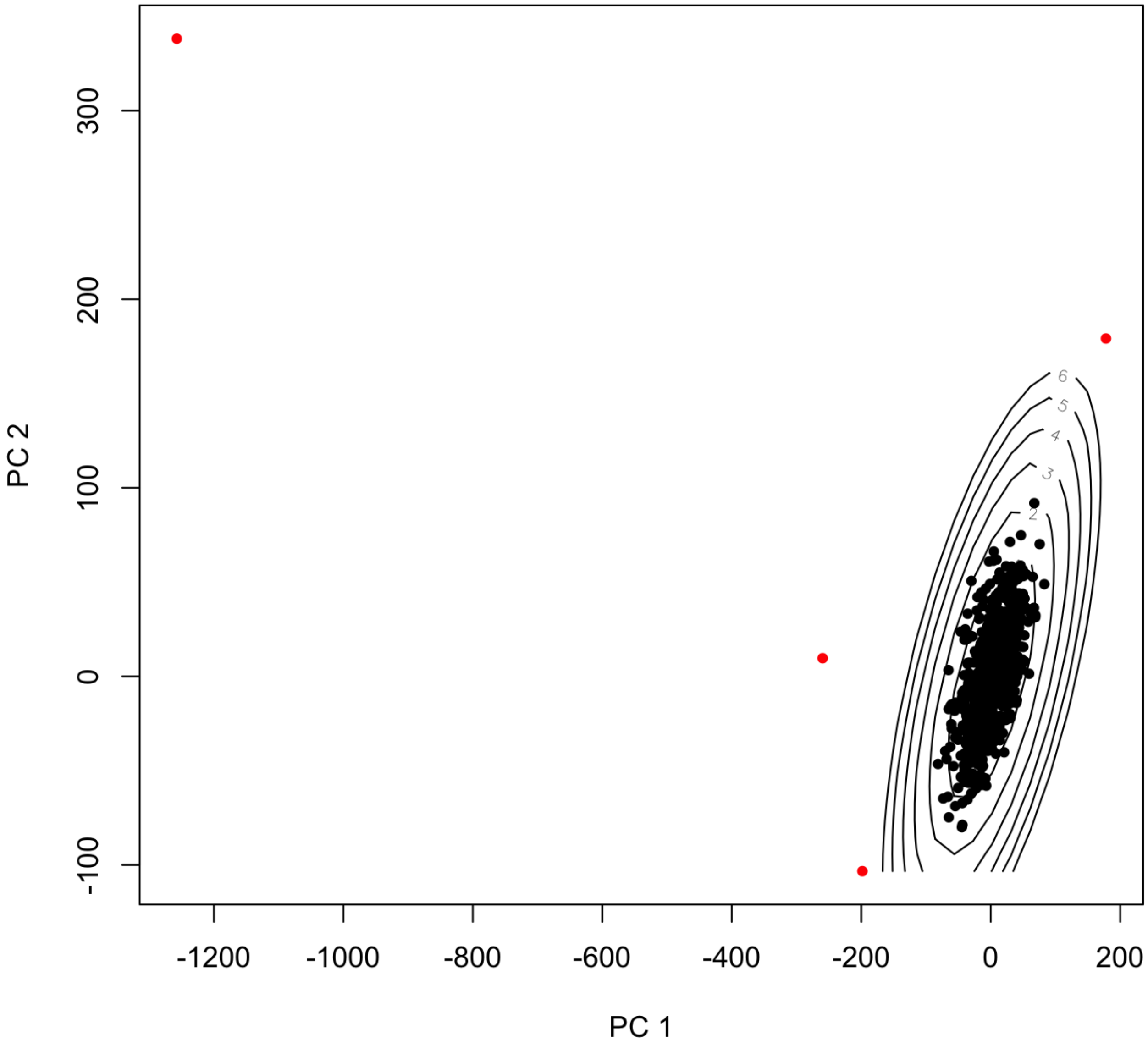
```r
gene_expression <- gene_expression[!rownames(gene_expression) %in% blood_probes(), ]
```

3 Outlier detection

We find outliers by exploratory plotting and statistical measurements described more closely in the package documentation. We will be working on log_2-transformed data to ameliorate the higher variance we usually see for higher intensities and to make the expression levels more symmetrical.

First we examine PCA-transformed data. The contour lines show distance to the center of the data in number of standard deviations.

```r
expression <- log2(t(exprs(gene_expression))) # transpose for samples by probes
prc_all <- prcout(expression)
plot(prc_all)
```

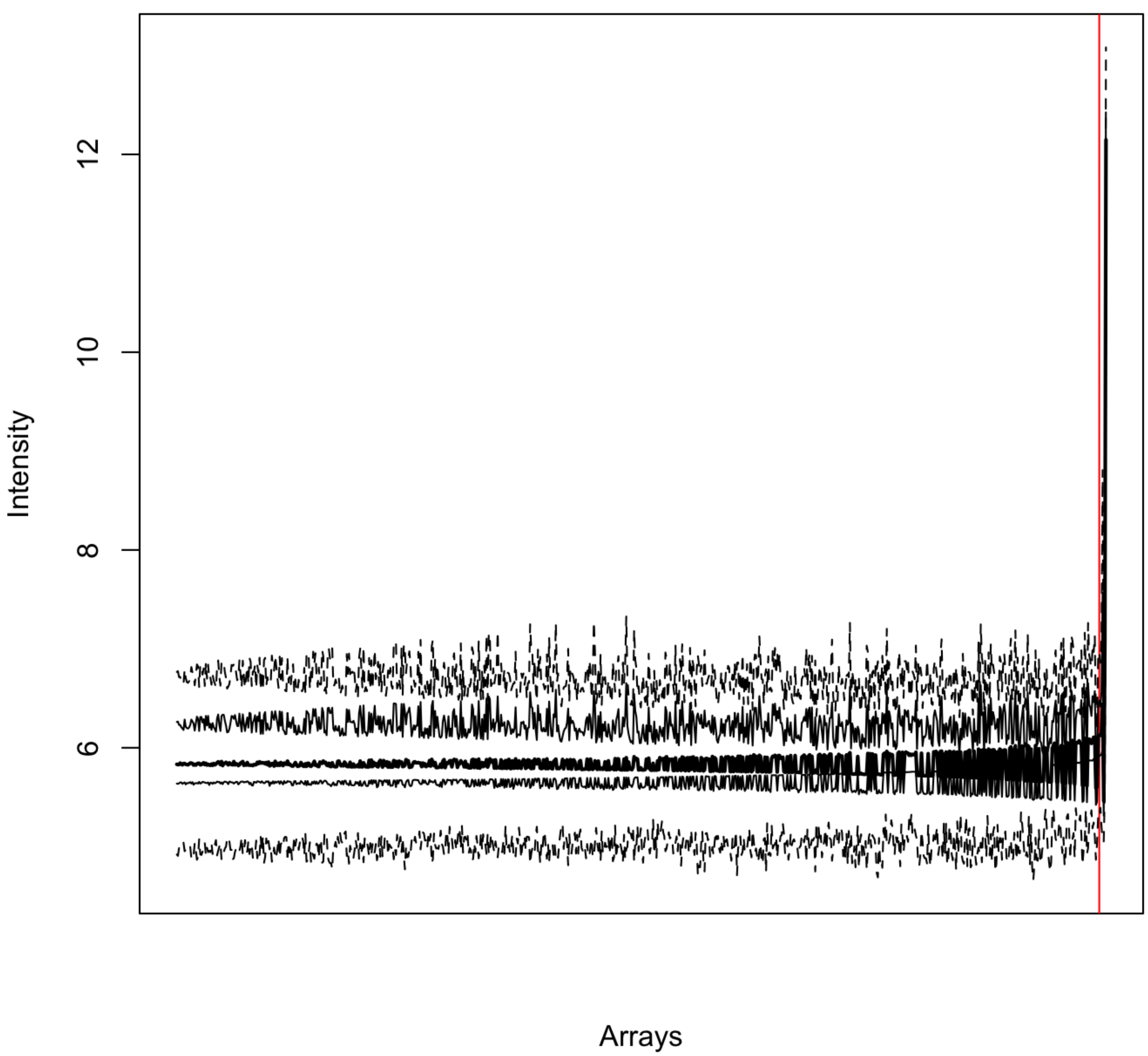


The points marked in red are three or more standard deviations away from the main bulk of the data: they look quite astonishing. Let's keep these red points as possible outliers.

```
pca_outliers <- predict(prc_all, sdev=3)
pca_outliers

## [1] "122" "511" "547" "827"
```

Next we investigate some boxplots.

```
boxo <- boxout(expression)
plot(boxo)
```



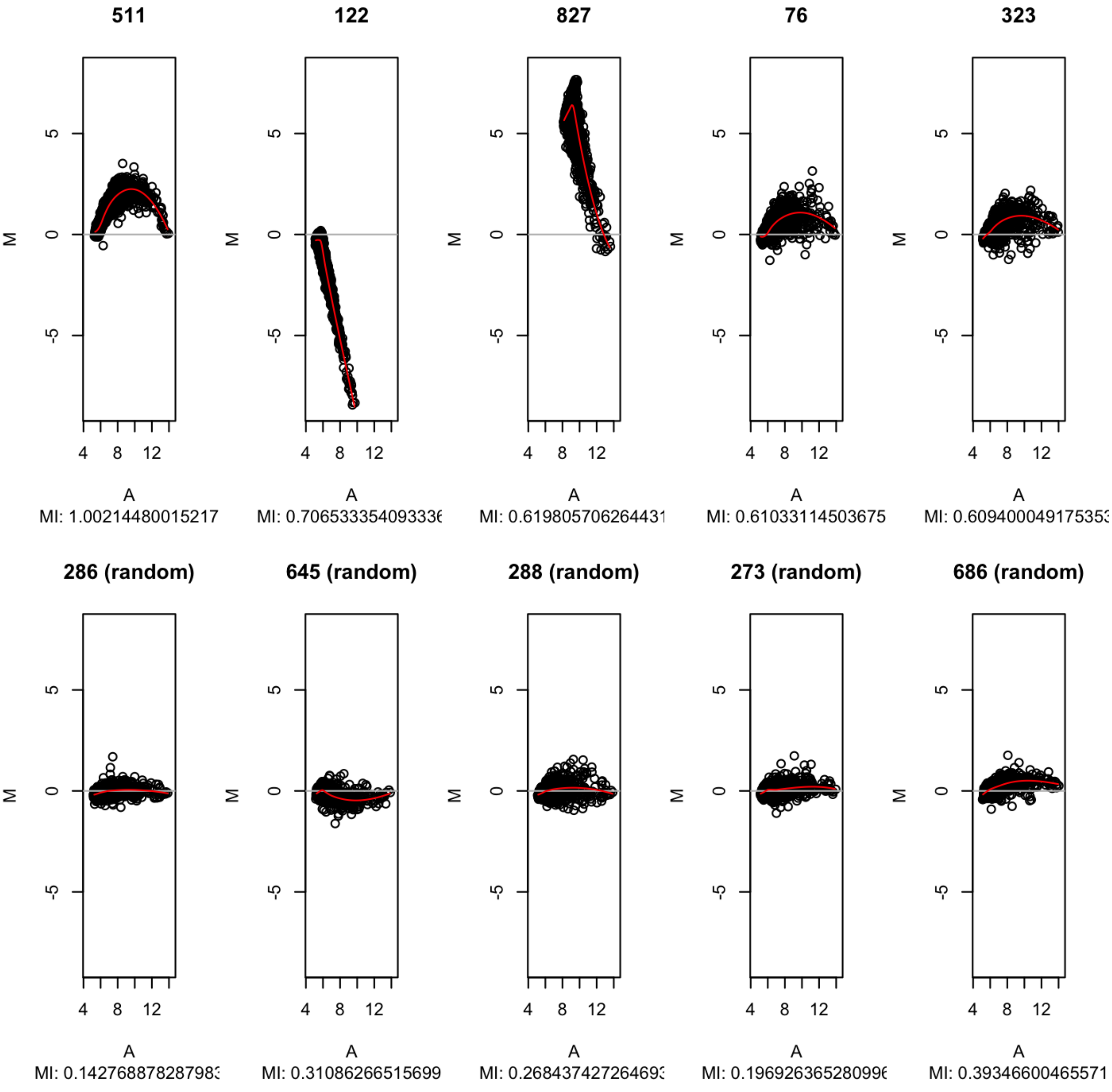
Points on the lines in this plot represent the box and whiskers of the regular `boxplot` function for your arrays. The lines represent the first and third quartiles, the median (ie the standard box), and the most extreme points that fall within 1.5 times the interquartile range (ie the whiskers/fences). As default the arrays are sorted by size of ks statistic (distance to pooled empirical distribution function).The red line demarks the cutoff for outlier or not.

```
boxplot_outliers <- predict(boxo, sdev=3)
boxplot_outliers
```

```
## [1] "122" "177" "496" "511" "547" "827"
```

The final detection method we use is the MA-plot. Let's plot the worst candidates and compare to some random samples. Badness is here defined in terms of mutual information between M and A statistics.

```
maout <- mapout(expression)
plot(maout, nout=5, lineup=T)
```



```
mapoutliers <- predict(maout, sdev=3)
mapoutliers

## [1] "76"  "122" "323" "511" "827"
```

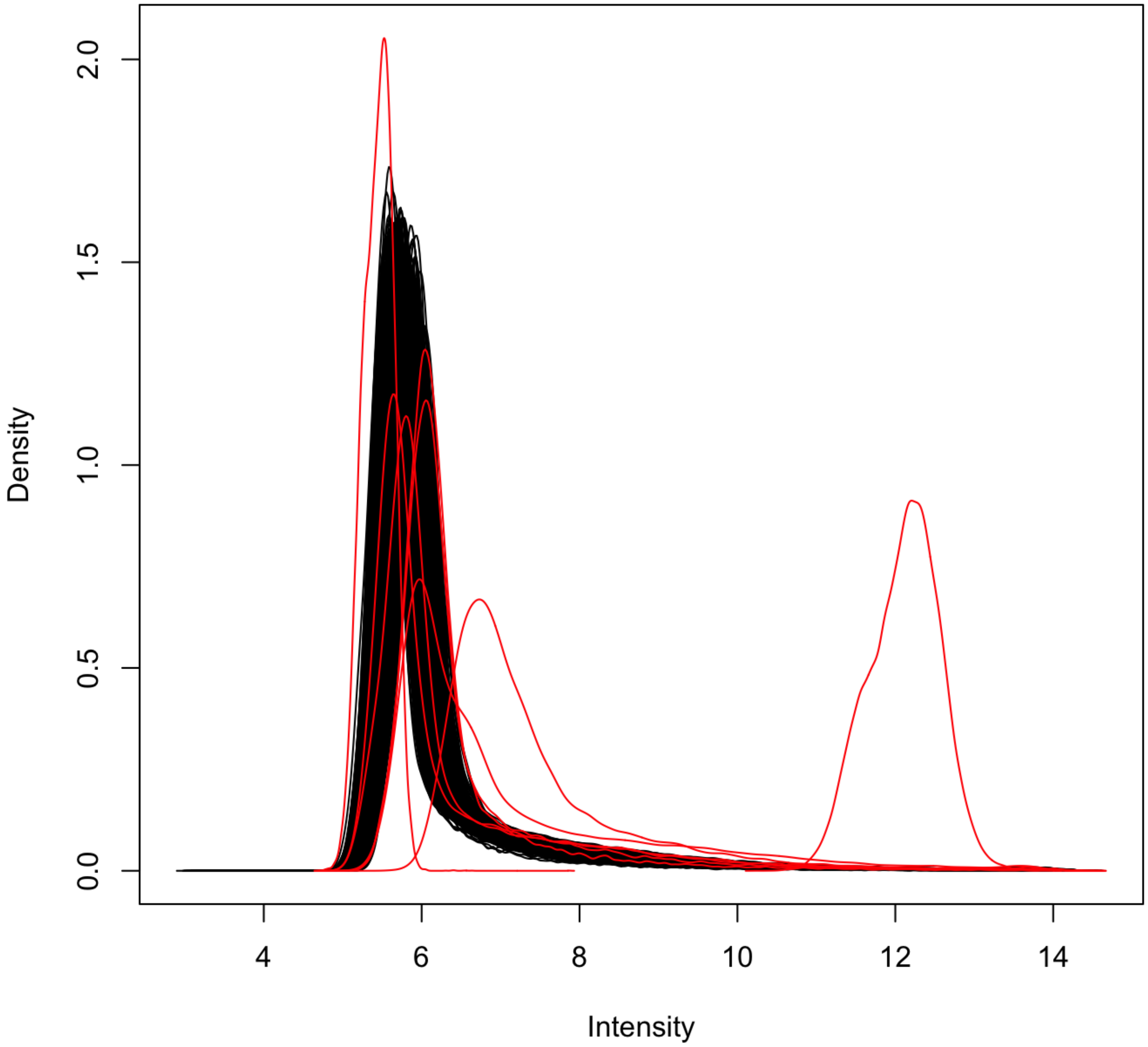
Let's now combine all outlier vectors.

```
outliers <- unique(c(mapoutliers, boxplot_outliers, pca_outliers))
outliers
```

```
## [1] "76"  "122" "323" "511" "827" "177" "496" "547"
```

These are the densities of expression values for all samples, proposed outliers in red:

```
densities <- dens(expression)
plot(densities, highlight=outliers)
```



As we can see, all of the clearly strange densities in this plot are marked as outliers; some of the candidates look fine however.

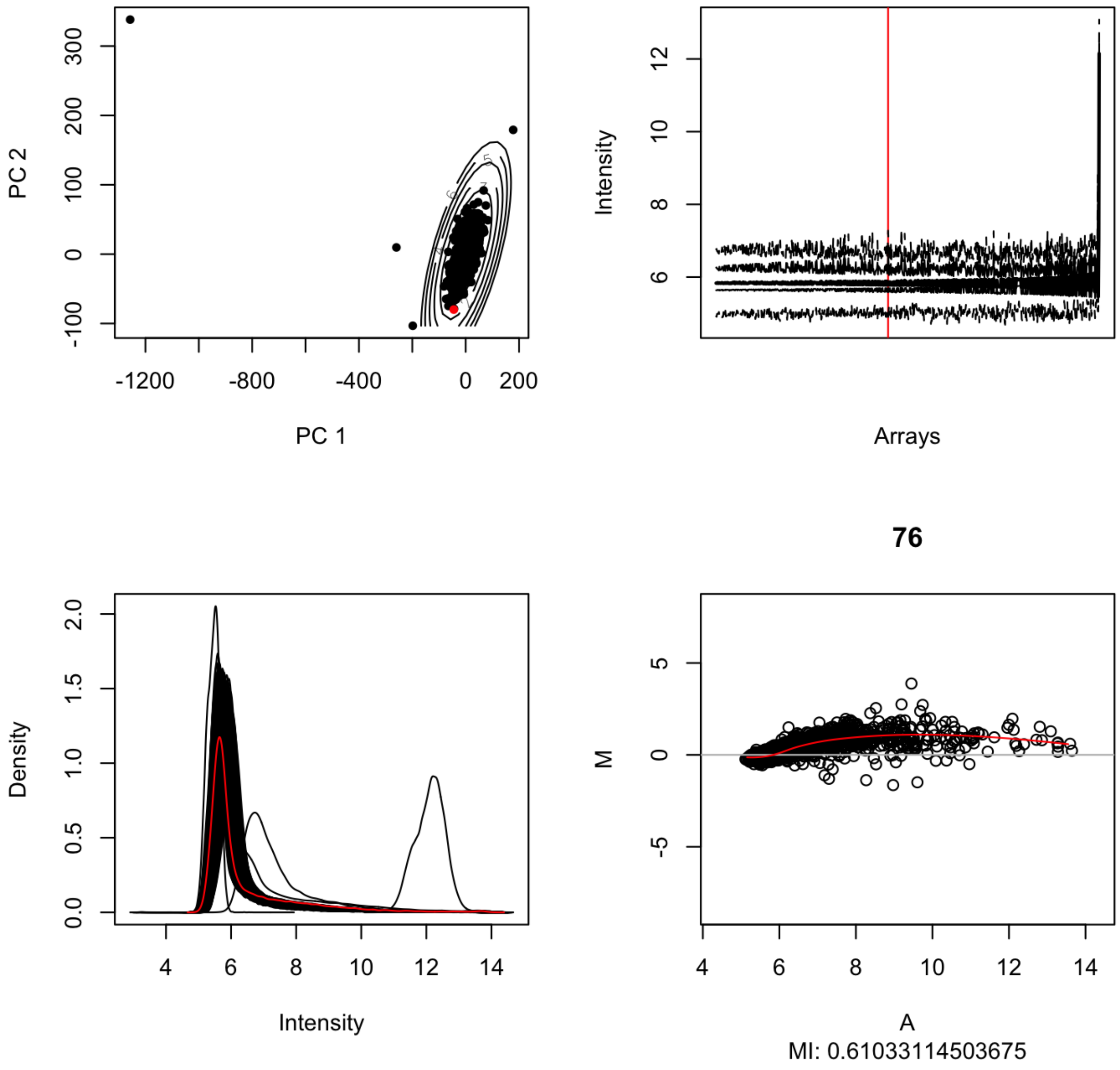
4 Outlier removal

So now we have a list of 8 candidate outliers that we suspect are technical outliers. This section will examine each of them and we'll make a decision to either keep or remove them as need be. Note that I would usually use the actual sample names instead of indexing the outlier vector with numbers. This is to be absolutely certain that I'm looking at what I think I'm looking at. I suggest others do the same. However, these data are anonyimzed, there are no sample names, and strings of row numbers will have to do.

4.1 76

This one looks fine. Maybe the MA plot is the reason it got flagged. I won't remove this.

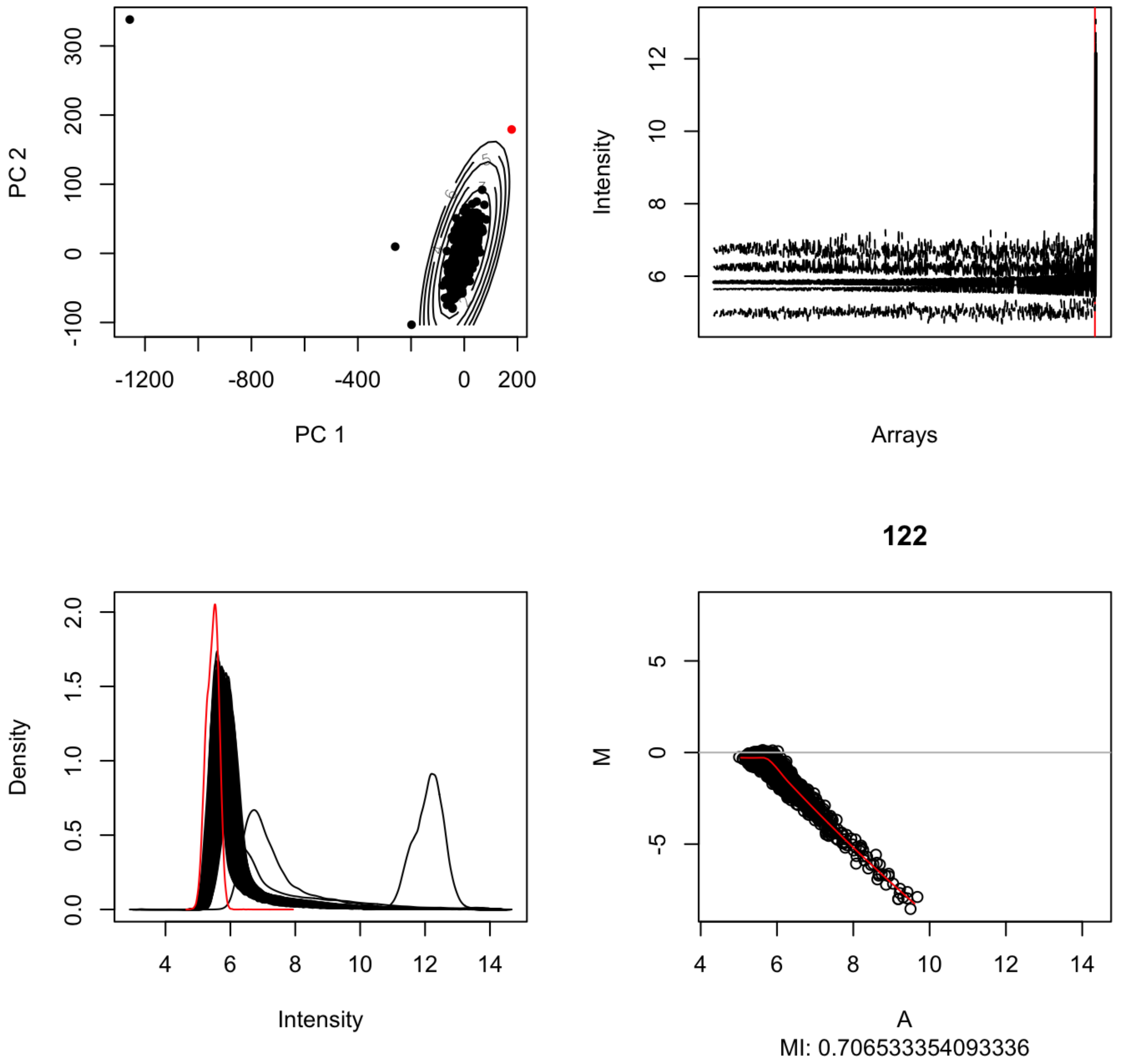
```
highlight("76", pca=prc_all, box=boxo, dens=densities, ma=maout)
```



4.2 122

This one is clearly very strange in all the plots, I will remove this.

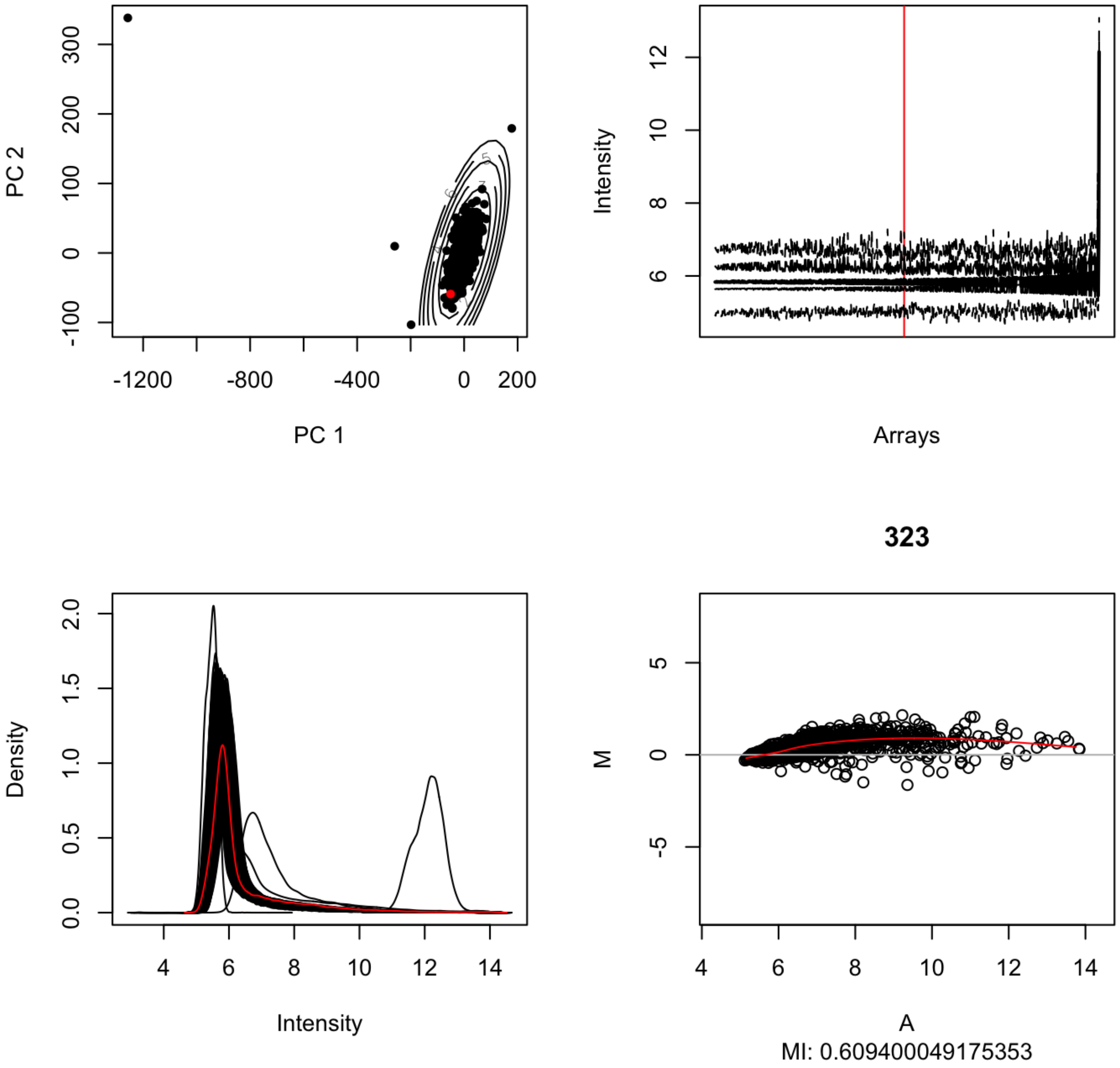
```
highlight("122", pca=prc_all, box=boxo, dens=densities, ma=maout)
```



```
for_removal <- "122"
```

4.3 323

```
highlight("323", pca=prc_all, box=boxo, dens=densities, ma=maout)
```

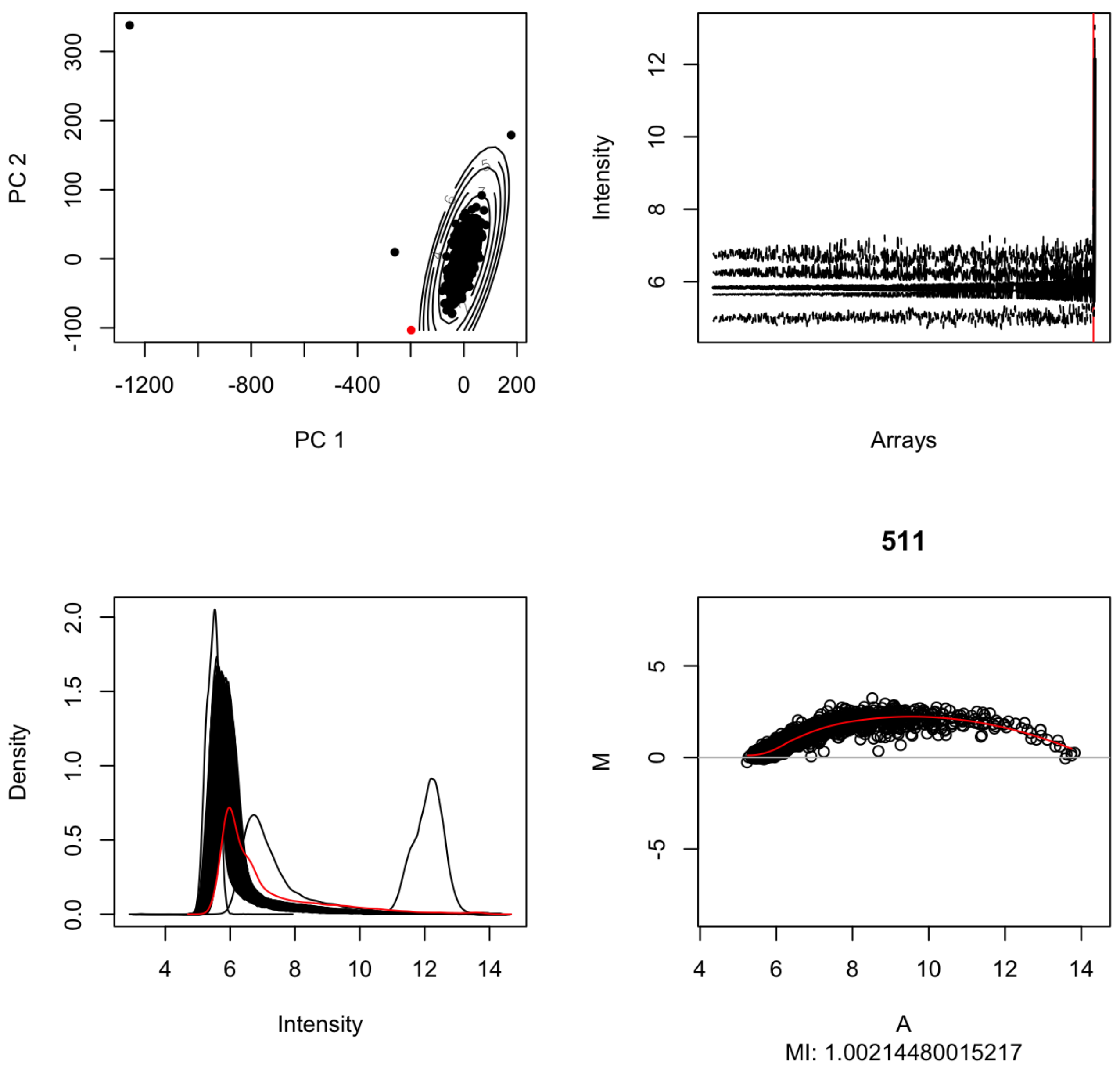


This one looks fine as well. Again it's probably the slightly high MI statistic.

4.4 511

This one once again looks strange in all the plots (maybe not all that bad in the MA plot) and I will take it out.

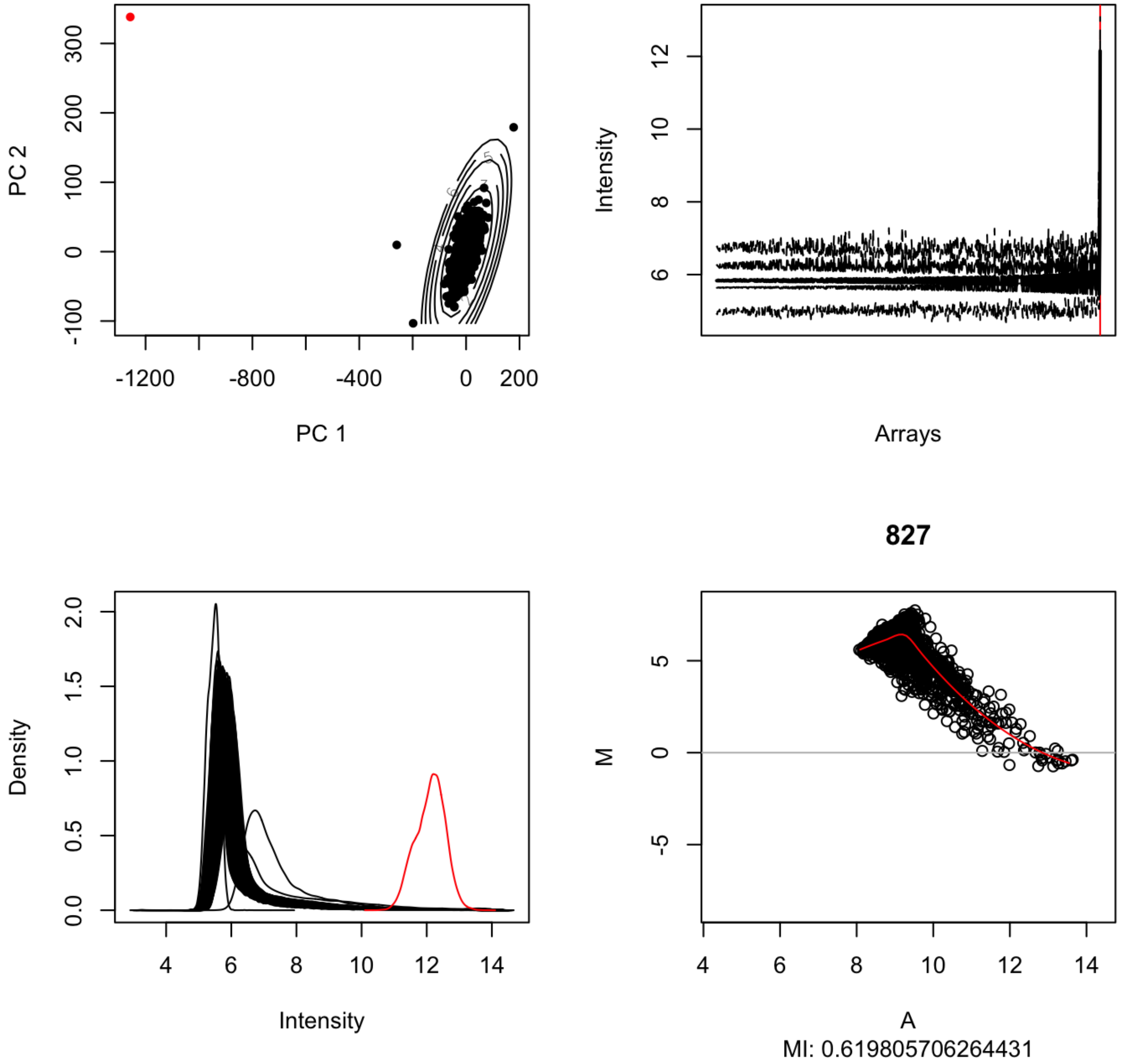
```
highlight("511", pca=prc_all, box=boxo, dens=densities, ma=maout)
```



```
for_removal <- c(for_removal, "511")
```

4.5 827

```
highlight("827", pca=prc_all, box=boxo, dens=densities, ma=maout)
```

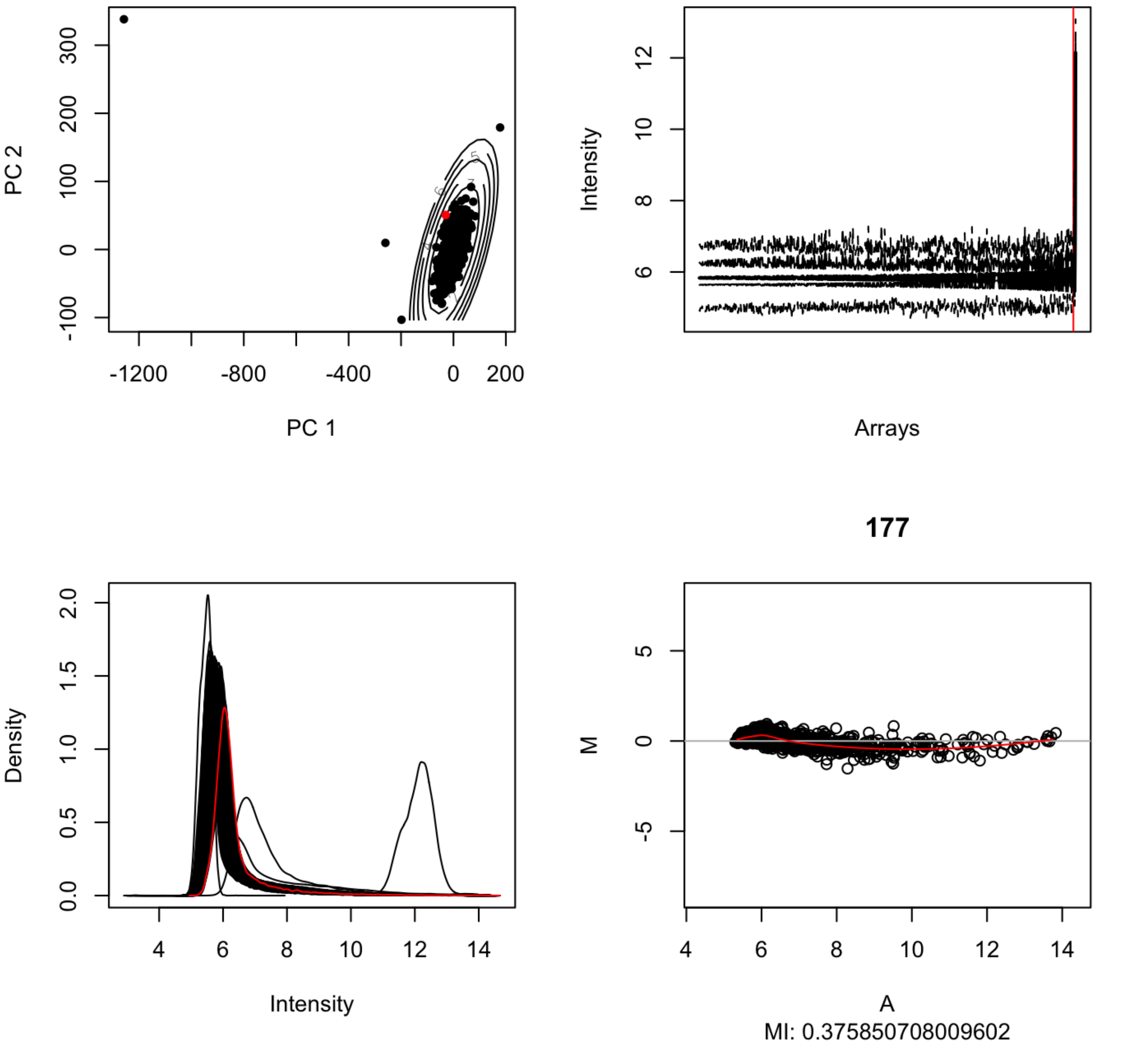


```
for_removal <- c(for_removal, "827")
```

Our most extreme point yet! Not only are the intensities pushed all the way to the right, there seems also to be some slight bimodality and other strangeness that the healthy samples don't exhibit.

4.6 177

```
highlight("177", pca=prc_all, box=boxo, dens=densities, ma=maout)
```



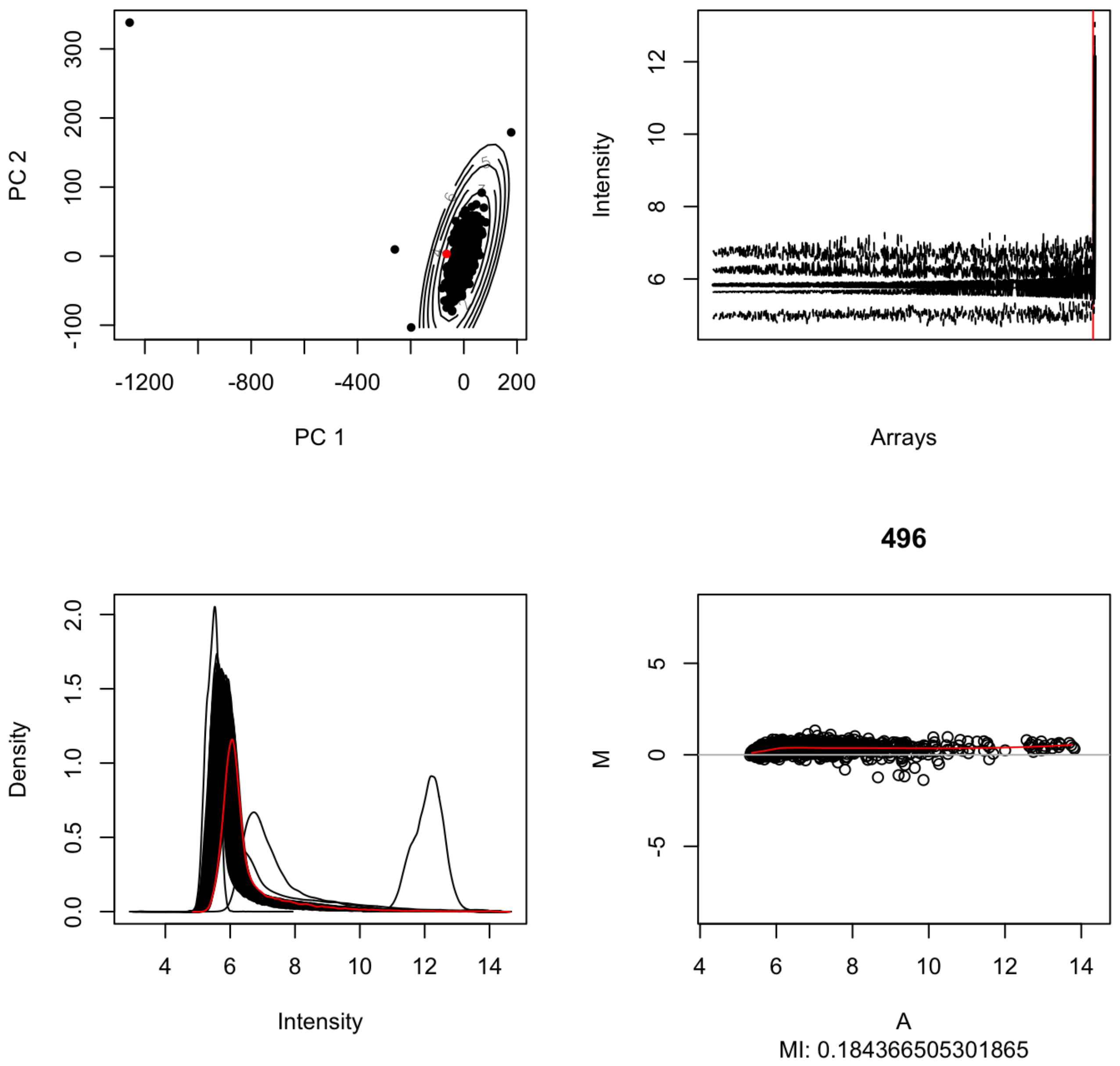
This one is more interesting, it's out there but not clearly broken. It looks as though the boxplots flagged it as outlier. Let's look at the lab info:

```
lab_info["177", ]

##     Ng/ul_RNA 260/280_RNA 260/230_RNA RIN Ng/ul_cRNA 260/280_cRNA 260/230_cRNA
## 177     55,85        2,06        1,83 8,1       1630          2,2          2,2

lab_thresholds

## [1] "Bad: RIN value < 7"           "Bad: 260/280 RNA ratio < 2"
## [3] "Bad: 260/230 RNA ratio < 1.7" "Good: 50 < Ng/ul RNA < 500"
```

It's not outside the predefined thresholds. Let's keep it.

4.7 496

```
highlight("496", pca=prc_all, box=boxo, dens=densities, ma=maout)
```

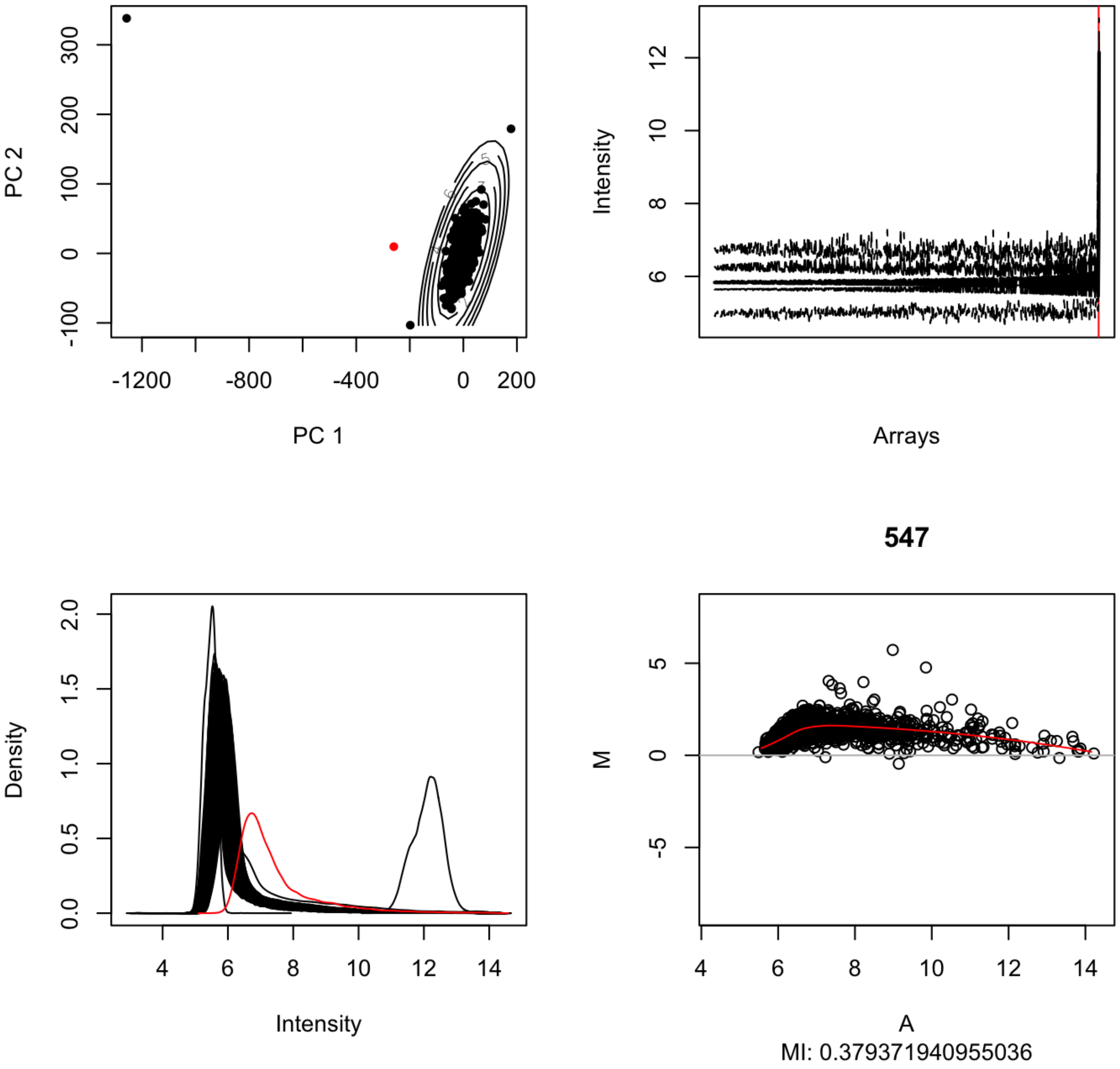


```
lab_info["496", ]

##      Ng/ul_RNA 260/280_RNA 260/230_RNA RIN Ng/ul_cRNA 260/280_cRNA 260/230_cRNA
## 496      94,44        2,08        1,72 8,2     2188,5        2,135        2,035

lab_thresholds

## [1] "Bad: RIN value < 7"             "Bad: 260/280 RNA ratio < 2"
## [3] "Bad: 260/230 RNA ratio < 1.7" "Good: 50 < Ng/ul RNA < 500"
```

This one is also slightly strange but not exactly outside the thresholds, so I'll keep it.

4.8 547

```
highlight("547", pca=prc_all, box=boxo, dens=densities, ma=maout)
```



```
lab_info["547", ]
```

```
##     Ng/ul_RNA 260/280_RNA 260/230_RNA RIN Ng/ul_cRNA 260/280_cRNA 260/230_cRNA
## 547     125,5        2,16        1,38 9,0       1625          2,2          2,2
```

```
for_removal <- c(for_removal, "547")
```

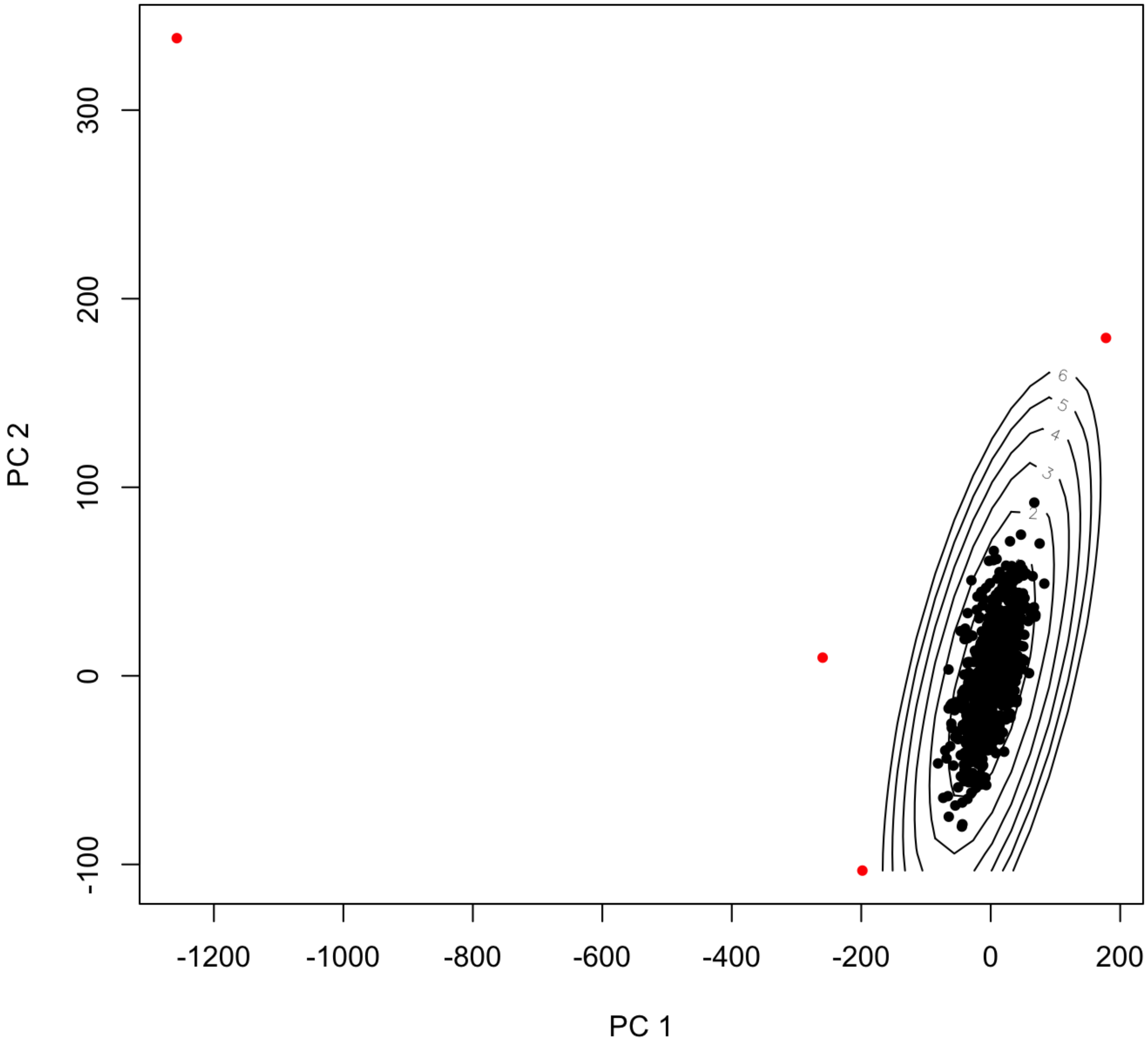
This one is strange in three out of four plots and has a too-low 260/230 ratio.

5 Summary

```
for_removal

## [1] "122" "511" "827" "547"
```

In the end we have four technical outliers. It's the ones you immediately feel strange about in the PCA plot:

```
plot(prc_all, highlight=for_removal)
```



6 Session info

- R version 3.3.1 (2016-06-21), `x86_64-apple-darwin13.4.0`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: Biobase 2.34.0, BiocGenerics 0.20.0, knitr 1.16, lumi 2.26.4, nowaclean 0.2.7
- Loaded via a namespace (and not attached): affy 1.52.0, affyio 1.44.0, annotate 1.52.1, AnnotationDbi 1.36.2, base64 2.0, beanplot 1.2, BiocInstaller 1.24.0, BiocParallel 1.8.2, BiocStyle 2.2.1, biomaRt 2.30.0, Biostrings 2.42.1, bitops 1.0-6, bumphunter 1.14.0, codetools 0.2-15, colorspace 1.3-2, data.table 1.10.4, DBI 0.6-1, digest 0.6.12, doRNG 1.6.6, entropy 1.2.1, evaluate 0.10, foreach 1.4.3, genefilter 1.56.0, GenomeInfoDb 1.10.3, GenomicAlignments 1.10.1, GenomicFeatures 1.26.4, GenomicRanges 1.26.4, GEOquery 2.40.0, grid 3.3.1, highr 0.6, httr 1.2.1, illuminaio 0.16.0, IRanges 2.8.2, iterators 1.0.8, KernSmooth 2.23-15, lattice 0.20-35, limma 3.30.13, locfit 1.5-9.1, magrittr 1.6, MASS 7.3-47, Matrix 1.2-10, matrixStats 0.52.2, mclust 5.2.3, memoise 1.1.0, methylumi 2.20.0, mgcv 1.8-17, minfi 1.20.2, multtest 2.30.0, nleqslv 3.3, nlme 3.1-131, nor1mix 1.2-2, openssl 0.9.6, pkgmaker 0.22, plyr 1.8.4, preprocessCore 1.36.0, quadprog 1.5-5, R6 2.2.1, RColorBrewer 1.1-2, Rcpp 0.12.10, RCurl 1.95-4.8, registry 0.3, reshape 0.8.6, rngtools 1.2.4, Rsamtools 1.26.2, RSQLite 1.1-2, rtracklayer 1.34.2, S4Vectors 0.12.2, siggenes 1.48.0, splines 3.3.1, stats4 3.3.1, stringi 1.1.5, stringr 1.2.0, SummarizedExperiment 1.4.0, survival 2.41-3, tools 3.3.1, XML 3.98-1.7, xtable 1.8-2, XVector 0.14.1, zlibbioc 1.20.0