Gene filtering by PCA for Affymetrix GeneChips

Jun Lu, Pierre R. Bushel

April 30, 2018

Contents

1 Introduction

Due to the nature of the array experiments which examine the expression of tens of thousands of genes (or probesets) simultaneously, the number of null hypotheses to be tested is large. Hence multiple testing correction is often necessary to control the number of false positives. However, multiple testing correction can lead to low statistical power in detecting genes that are truly differentially expressed. Filtering out non-informative genes allows for a reduction of the number of hypotheses, which potentially can reduce the impact of multiple testing corrections. While several filtering methods have been suggested [1], the best practice to filtering is still under debate. We propose a new filtering statistic for Affymetrix GeneChips, based on principal component analysis (PCA) on the probe-level gene expression data. Given that all the probes in a probeset are designed to target one or a common cluster of transcripts, the measurements of probes in a probeset

should be correlated. The degree of concordance of gene expression among probes can be approximated by the proportion of variation accounted by the first principal component (PVAC). Using a wholly defined spike-in dataset, we have shown that filtering by PVAC provides increased sensitivity in detecting truly differentially expressed genes while controlling the false discoveries. Furthermore, a data-driven approach to guide the selection of the filtering threshold value is also proposed.

The pvac package implements the method proposed in the paper: *Jun Lu, Robnet T. Kerns, Shyamal Peddada, and Pierre R. Bushel 2011 "Principal component analysis-based filtering improves detection for Affymetrix gene expression arrays" Nucleic Acids Res. 39(13):e86.*

In the sections below, we provide instructions on how to perform the PCA-based gene filtering using this package, and an example for demonstration.

2 Installation

Simply skip this section if one has been familiar with the usual Bioconductor installation process. Assume that a recent version of R has been correctly installed.

Install the packages from the Bioconductor repository, using the `biocLite` utility. Within R console, type

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

Installation using the `biocLite` utility automatically handles the package dependencies. The `pvac` package depends on the package `affy`, which can be installed in the same way as shown above. We also recommend to install the package `pbapply` (for showing a progress bar).

3 Gene filtering by PCA

PCA-based filtering requires the probe level data (*Cel* files). Also, note that filtering is performed using either all samples, or a subset of samples chosen by ignoring the sample class labels. Outlier samples should be removed before filtering. Ideally the number of samples should be at least 6 in order to make the PCA-based filtering effective.

3.1 Steps

These are a few steps for a typical analysis. Here assume all the *Cel* files are put in a directory, say /my/directory/celfiles

1. Read in the *Cel* files and store in an `AffyBatch` object

   ```
   > library(affy)
   > abatch <- ReadAffy(celfile.path="/my/directory/celfiles")
   ```

2. Summarize the probe level data into probeset level data, and store them in an `ExpressionSet` object. Here we use the method called `rma` as an example,

   ```
   > myeset <- rma(abatch)
   ```

3. Perform gene filtering and exclude probesets with low concordance in probe intensity measurements

   ```
   > library(pvac)
   > ft <- pvacFilter(abatch)
   > myeset <- eset[ft$aset,]
   ```

   Here the `myeset` is an `ExpressionSet` object containing probesets that have passed the default filtering threshold. To see additional information returned by `pvacFilter`, type `?pvacFilter`.

4. Identify the differentially expressed genes. In the case of a two-group comparison, one can use the simple t test to identify the genes of interest. Given a vector `group` which contains the group indicators for the samples (from the function call `sampleNames(myeset)`), one can do,

   ```
   > library(genefilter)
   > myres  = rowttests(exprs(myeset), as.factor(group))
   ```

3.2 Example

We use `MLL.A` dataset in the package `ALLMLL` as an example to illustrate the PCA filtering procedure. This dataset contains 20 samples from a leukemia study with RNAs hybridized on the HGU133A chip. The data have been stored in an `AffyBatch` object. First we perform the usual summarization and then filtering.

```
> library(affy)
> library(pvac)
```

```
> library(ALLMLL)
> data(MLL.A)
> myeset <- rma(MLL.A)

Background correcting
Normalizing
Calculating Expression

> ft <- pvacFilter(MLL.A)

Making absent/present calls, preprocessing data ...
Computing the PVAC scores ...
Deriving the filtering threshold ...
PVAC cutoff score (<=0.5):  0.44484

> myeset.filtered <- myeset[ft$aset,]
```

The pvac package selects the filtering threshold by first identifying a group of probesets being called "Absent" across all samples by the mas5 algorithm (in the affy package). The 99 percentile value of the PVAC scores in this (null) set of probesets is chosen as the default cutoff value. Certainly one can lower the percentile value to relax the filtering threshold if necessary. In addition, the maximum value of the threshold value is set at 0.5, which corresponds to 50% of the total variation accounted for by the first PC.

We can plot the distributions of the PVAC scores as shown in Figure 1.

```
> plot(density(ft$pvac[ft$nullset]),xlab="PVAC score",main="",
+           col="gray",cex.lab=0.5,xlim=c(0,1))
> lines(density(ft$pvac),col=1)
> abline(v=ft$cutoff,lty=2,col="gray")
```

4 Session

```
>   print(sessionInfo())

R version 3.5.0 (2018-04-23)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.4 LTS
```

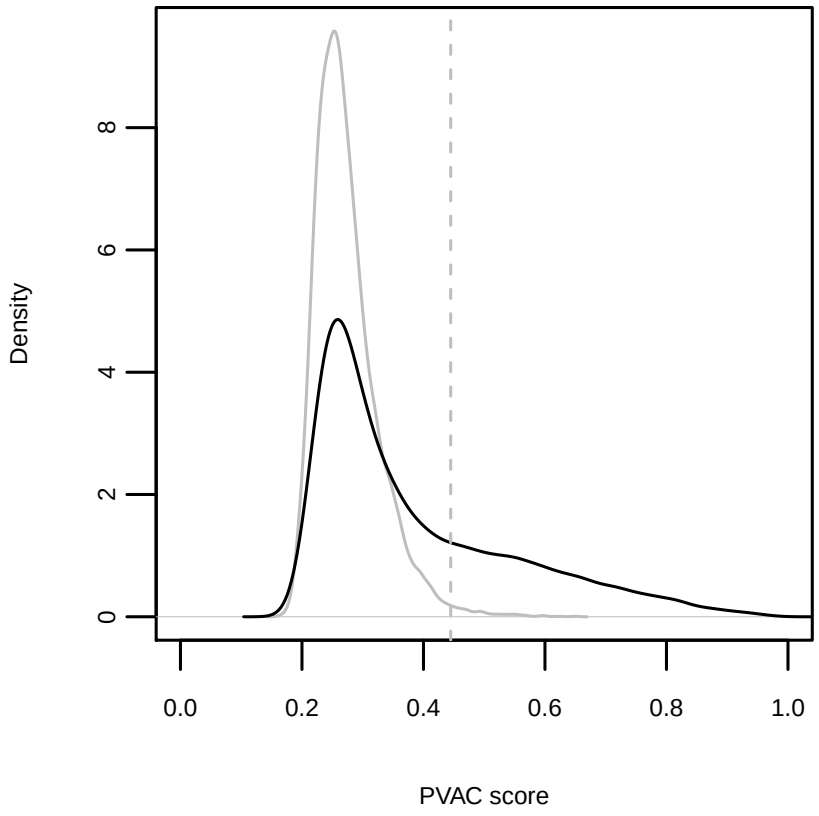

Figure 1: Density plots of PVAC scores from a group of "Absent" probesets (in gray) and the full set (in black). The vertical gray line indicates the default filtering cutoff value.

```
Matrix products: default
BLAS: /home/biocbuild/bbs-3.7-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.7-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] pbapply_1.3-4       hgu133acdf_2.18.0   ALLMLL_1.19.0
[4] pvac_1.28.0         affy_1.58.0         Biobase_2.40.0
[7] BiocGenerics_0.26.0

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.16          IRanges_2.14.0        digest_0.6.15
 [4] DBI_0.8               stats4_3.5.0          RSQLite_2.1.0
 [7] BiocInstaller_1.30.0  zlibbioc_1.26.0       affyio_1.50.0
[10] blob_1.1.1            S4Vectors_0.18.0      preprocessCore_1.42.0
[13] tools_3.5.0           bit64_0.9-7           bit_1.1-12
[16] compiler_3.5.0        AnnotationDbi_1.42.0  memoise_1.1.0
```

References

[1] Richard Bourgona, Robert Gentleman, and Wolfgang Huber. (2010) *Independent filtering increases detection power for high-throughput experiments* PNAS **107**(21), 9546-9551