Introduction to antiProfiles

Héctor Corrada Bravo hcorrada@gmail.com

Modified: March 13, 2013. Compiled: April 24, 2017

Introduction

This package implements the gene expression anti-profiles method in [1]. Anti-profiles are a new approach for developing cancer genomic signatures that specifically takes advantage of gene expression heterogeneity. They explicitly model increased gene expression variability in cancer to define robust and reproducible gene expression signatures capable of accurately distinguishing tumor samples from healthy controls.

In this vignette we will use the companion antiProfilesData package to illustrate some of the analysis in that paper.

> # these are libraries used by this vignette
> require(antiProfiles)
> require(antiProfilesData)
> require(RColorBrewer)

Colon cancer expression data

The data is stored as an `ExpressionSet`. This dataset contains colon adenomas, benign but hyperplastic growths along with the normal and tumor tissues. Let’s remove these from the remaining analysis.

```r
drop=apColonData$SubType=="adenoma"
apColonData=apColonData[,!drop]
```

Building antiprofiles

The general anti-profile idea is to find genes with hyper-variable expression in cancer with respect to normal samples and classify new samples as normal vs. cancer based on deviation from a normal expression profile built from normal training samples. Anti-profiles are built using the following general algorithm:

1. Find candidate differentially variable genes (anti-profile genes): rank by ratio of cancer to normal variance
2. Define region of normal expression for each anti-profile gene: normal median ± 5 * normal MAD
3. For each sample to classify:
   (a) count number of antiProfile genes outside normal expression region (anti-profile score)
   (b) if score is above threshold, then classify as cancer

We will use data from one of the experiments to train the anti-profile (steps 1 and 2 above) and test it on the data from the other experiment (step 3).

Computing variance ratios

The first step in building an antiprofile is to calculate the ratio of normal variance to cancer variance. This is done with the apStats function.

```r
> trainSamples=pData(apColonData)$ExperimentID=="GSE4183"
> colonStats=apStats(exprs(apColonData)[,trainSamples],
+ pData(apColonData)$Status[trainSamples],minL=5)
> head(getProbeStats(colonStats))

   affyid     SD0     SD1   stat  meds0  mads0
4895 214974_x_at 0.23369554 7.777167 5.056543 -1.5182409 0.20537749
 10 210118_s_at 0.12684950 3.991524 4.975750 -0.7449072 0.08599552
1318 205719_s_at 0.09529811 2.817550 4.885850 -0.8532822 0.08670758
2707 205863_at 0.12430282 3.431335 4.786839 -0.5508941 0.15574925
4896 215101_s_at 0.24540888 6.577807 4.744406 -0.9257671 0.15043660
2272 227140_at 0.24976921 5.718607 4.516996 -1.8026488 0.13726594
```

We can see how that ratio is distributed for these probesets:

```r
> hist(getProbeStats(colonStats)$stat, nc=100,
+     main="Histogram of log variance ratio", xlab="log2 variance ratio")
```

![Histogram of log variance ratio](image)

**Building the anti-profile**

Now we construct the anti-profile by selecting the 100 probesets most hyper-variable probesets

```r
> ap=buildAntiProfile(colonStats, tissueSpec=FALSE, sigsize=100)
> show(ap)
```
AntiProfile object with 100 probes

Normal medians

Min. 1st Qu.  Median  Mean  3rd Qu.  Max.
-1.8026  -0.8542  -0.5102  0.6835  -0.1187  23.6885

Using cutoff 5

Computing the anti-profile score

Given the estimated anti-profile, we can get anti-profile scores for a set of testing samples.

```R
> counts=apCount(ap, exprs(apColonData)[,!trainSamples])
> palette(brewer.pal(8,"Dark2"))
> # plot in score order
> o=order(counts)
> dotchart(counts[o],col=pData(apColonData)$Status![trainSamples][o]+1,
+       labels="",pch=19,xlab="anti-profile score",
+       ylab="samples",cex=1.3)
> legend("bottomright", legend=c("Cancer","Normal"),pch=19,col=2:1)
```
The anti-profile score measures deviation from the normal expression profile obtained from the training samples. We see in this case that the anti-profile score can distinguish the test samples perfectly based on deviation from the normal profile.

References


SessionInfo

- R version 3.4.0 (2017-04-21), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Running under: Ubuntu 16.04.2 LTS
- Matrix products: default
- BLAS: /home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so
- LAPACK: /home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.36.0, BiocGenerics 0.22.0, RColorBrewer 1.1-2, antiProfiles 1.16.0, antiProfilesData 1.11.0, locfit 1.5-9.1, matrixStats 0.52.2
- Loaded via a namespace (and not attached): compiler 3.4.0, grid 3.4.0, lattice 0.20-35, tools 3.4.0