

How To Filter Genes

October 3, 2009

Introduction

The *genefilter* package can be used to filter (select) genes from a microarray experiment according to a wide variety of different filtering mechanisms. To be concrete we will consider the artificial experiment contained in the `sample.ExpressionSet` example from the *Biobase* package. This experiment has 26 samples, there are 500 genes and 3 covariates. The covariates are named `sex`, `type` and `score`. The first two have two levels and the last one is continuous.

```
> library("Biobase")
> library("genefilter")
> data(sample.ExpressionSet)
> varLabels(sample.ExpressionSet)

[1] "sex"  "type" "score"

> table(sample.ExpressionSet$sex)

Female  Male
    11   15

> table(sample.ExpressionSet$type)

Case Control
    15     11
```

One dichotomy that is of interest is to split filters into the non-specific and the specific. Here, specific means that we are filtering with reference to some particular variable. For example if we want to select genes that are differentially expressed in the two groups defined by `cov1` that is a specific filter. If on the otherhand we want to select genes that are expressed in more than 5 samples then that is an example of a non-specific filter.

First let's see how to perform a non-specific filter. Suppose we want to select genes that are expressed above 200 in at least 5 samples. To do that we use the function `kOverA`.

There are three steps that must be performed.

1. Create the filters function(s).
2. Assemble them into a filtering function.
3. Apply the filtering function to the expression matrix.

```
> f1 <- kOverA(5, 200)
> ffun <- filterfun(f1)
> which <- genefilter(exprs(sample.ExpressionSet), ffun)
> sum(which)
```

[1] 159

Here `f1` is a filter, the function `ffun` is a filtering function and we finally apply it using `genefilter`. There were 154 genes that had a level of more than 200 in at least 5 samples.

To perform a specific filter lets select genes that are differentially expressed in the groups defined by `type`.

```
> tf1 <- ttest(sample.ExpressionSet$type, 0.1)
> ff2 <- filterfun(tf1)
> wh2 <- genefilter(exprs(sample.ExpressionSet), ff2)
> sum(wh2)
```

[1] 88

Now we see that there are 31 genes that pass this selection procedure. In this example we chose a very large p -value because we have a very small sample and the covariate was made up.

Suppose that we want to combine the two filters. We want those genes for which at least 5 are over 200 and which also are differentially expressed in the groups defined by `type`.

```
> ff3 <- filterfun(f1, tf1)
> wh3 <- genefilter(exprs(sample.ExpressionSet), ff3)
> sum(wh3)
```

[1] 35

Now we see that there are only 10 genes that satisfy both conditions.

1 Aggregate Vignette

The function given below performs k -nearest neighbour classification using leave-one-out cross-validation. At the same time it aggregates the genes that were selected. The function returns the predicted classifications as its returned value. However, there is an additional side effect. The number of times that each gene was used (provided it was at least one) are recorded and stored in the environment of the aggregator `Agg`. These can subsequently be retrieved and used for other purposes.

```
> knnCV <- function(EXPR, selectfun, cov, Agg, pselect = 0.01,
+   Scale = FALSE) {
+   nc <- ncol(EXPR)
+   outvals <- rep(NA, nc)
+   for (i in 1:nc) {
+     v1 <- EXPR[, i]
+     expr <- EXPR[, -i]
+     glist <- selectfun(expr, cov[-i], p = pselect)
+     expr <- expr[glist, ]
+     if (Scale) {
+       expr <- scale(expr)
+       v1 <- as.vector(scale(v1[glist]))
+     }
+     else v1 <- v1[glist]
+     out <- paste("iter ", i, " num genes= ", sum(glist),
+       sep = "")
+     print(out)
```

```

+     Aggregate(row.names(expr), Agg)
+     if (length(v1) == 1)
+       outvals[i] <- knn(expr, v1, cov[-i], k = 5)
+     else outvals[i] <- knn(t(expr), v1, cov[-i], k = 5)
+   }
+   return(outvals)
+ }

> gfun <- function(expr, cov, p = 0.05) {
+   f2 <- ttest(cov, p = p)
+   ffun <- filterfun(f2)
+   which <- genefilter(expr, ffun)
+ }

```

Next we show how to use this function on the system dataset, `geneData`.

```

> library("class")
> geneData <- genescale(exprs(sample.ExpressionSet)[1:75, ], 1)
> Agg <- new("aggregator")
> testcase <- knnCV(geneData, gfun, sample.ExpressionSet$type,
+   Agg, pselect = 0.05)

[1] "iter 1 num genes= 3"
[1] "iter 2 num genes= 5"
[1] "iter 3 num genes= 5"
[1] "iter 4 num genes= 6"
[1] "iter 5 num genes= 7"
[1] "iter 6 num genes= 5"
[1] "iter 7 num genes= 8"
[1] "iter 8 num genes= 5"
[1] "iter 9 num genes= 4"
[1] "iter 10 num genes= 5"
[1] "iter 11 num genes= 5"
[1] "iter 12 num genes= 6"
[1] "iter 13 num genes= 5"
[1] "iter 14 num genes= 5"
[1] "iter 15 num genes= 6"
[1] "iter 16 num genes= 6"
[1] "iter 17 num genes= 6"
[1] "iter 18 num genes= 4"
[1] "iter 19 num genes= 6"
[1] "iter 20 num genes= 4"
[1] "iter 21 num genes= 5"
[1] "iter 22 num genes= 5"
[1] "iter 23 num genes= 6"
[1] "iter 24 num genes= 6"
[1] "iter 25 num genes= 5"
[1] "iter 26 num genes= 6"

> genes.used <- as.list(aggenv(Agg))
> genes.counts <- as.numeric(genes.used)
> names(genes.counts) <- names(genes.used)
> sort(genes.counts)

```

AFFX-HUMRGE/M10098_M_at	AFFX-DapX-5_at	AFFX-TrpnX-5_at
1	1	1
AFFX-BioDn-5_at	AFFX-PheX-5_at	31312_at
1	1	3
AFFX-BioC-3_st	31308_at	AFFX-PheX-M_at
3	15	15
AFFX-YEL018w/_at	AFFX-MurIL4_at	AFFX-TrpnX-M_at
20	26	26
AFFX-hum_alu_at		
26		

The variable `genes.counts` now contains, for each gene, the number of times it was selected in the cross-validation.

One may also compare testcase with `geneCov` to see how well the procedure worked at predicting.

2 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 2.9.2 (2009-08-24), x86_64-unknown-linux-gnu
- Locale: LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=en_US;
- Base packages: base, datasets, graphics, grDevices, methods, stats, tools, utils
- Other packages: Biobase 2.4.1, class 7.2-49, genefilter 1.24.3
- Loaded via a namespace (and not attached): annotate 1.22.0, AnnotationDbi 1.6.1, DBI 0.2-4, RSQLite 0.7-2, splines 2.9.2, survival 2.35-7, xtable 1.5-5