Using the genefilter function to filter genes from a microarray dataset

May 9, 2014

Introduction

The *genefilter* package can be used to filter (select) genes from a microarray dataset according to a variety of different filtering mechanisms. Here, we will consider the example dataset in the `sample.ExpressionSet` example from the *Biobase* package. This experiment has 26 samples, and 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 can be of interest for subsequent analyses is whether the filter is *specific* or *non-specific*. Here, specific means that we are filtering with reference to sample metadata, for example, `type`. For example, if we want to select genes that are differentially expressed in the two groups defined by `type`, that is a specific filter. If on the other hand we want to select genes that are expressed in more than 5 samples, that is an example of a non–specific filter.

First, let us see how to perform a non–specific filter. Suppose we want to select genes that have an expression measure 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 function(s) implementing the filtering criteria.
2. Assemble it (them) into a (combined) filtering function.
3. Apply the filtering function to the expression matrix.

```
> f1 <- kOverA(5, 200)
> ffun <- filterfun(f1)
> wh1 <- genefilter(exprs(sample.ExpressionSet), ffun)
> sum(wh1)

[1] 159
```

Here `f1` is a function that implies our "expression measure above 200 in at least 5 samples" criterion, the function `ffun` is the filtering function (which in this case consists of only one criterion), and we apply it using `genefilter`. There were 159 genes that satisfied the criterion and passed the filter.

As an example for a specific filter, let us select genes that are differentially expressed in the groups defined by `type`.

```
> f2 <- ttest(sample.ExpressionSet$type, p=0.1)
> wh2 <- genefilter(exprs(sample.ExpressionSet), filterfun(f2))
> sum(wh2)

[1] 88
```

Here, `ttest` is a function from the `genefilter` package which provides a suitable wrapper around `t.test` from package *stats*. Now we see that there are 88 genes that satisfy the selection criterion.

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

```
> ffun_combined <- filterfun(f1, f2)
> wh3 <- genefilter(exprs(sample.ExpressionSet), ffun_combined)
> sum(wh3)

[1] 35
```

Now we see that there are only 35 genes that satisfy both conditions.

Selecting genes that appear useful for prediction

The function `knnCV` defined 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 dataset `geneData`.

```
>   library("class")
>   ##scale the genes
>   ##genescale is a slightly more flexible "scale"
>   ##work on a subset -- for speed only
>   geneData <- genescale(exprs(sample.ExpressionSet)[1:75,], 1)
>   Agg <- new("aggregator")
>   testcase <- knnCV(geneData, gfun, sample.ExpressionSet$type,
+          Agg, pselect=0.05)

> sort(sapply(aggenv(Agg), c), decreasing=TRUE)

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

The environment `Agg` contains, for each gene, the number of times it was selected in the cross-validation.

Session Information

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

- R version 3.1.0 (2014-04-10), `x86_64-apple-darwin13.1.0`

- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.24.0, BiocGenerics 0.10.0, class 7.3-10, genefilter 1.46.1
- Loaded via a namespace (and not attached): AnnotationDbi 1.26.0, DBI 0.2-7, GenomeInfoDb 1.0.2, IRanges 1.22.6, RSQLite 0.11.4, XML 3.98-1.1, annotate 1.42.0, splines 3.1.0, stats4 3.1.0, survival 2.37-7, tools 3.1.0, xtable 1.7-3