A note on esApply

exprSets are complex objects. We will think of them as linked arrays: the exprs element of an *exprSet* is $G \times N$, where G is the number of genes on a chip and N is the number of tissues analyzed, and the pData element of the associated phenoData element is $N \times p$, where p is the number of phenotypic or demographic, etc., variables collected.

Abstractly, we are often interested in evaluating functions $f(y; x)$ where y is an N-vector of expression results for a specific gene and x is an N-dimensional structure, coordinated with y, that distinguishes elements of y for processing in the function f. A basic problem is to guarantee that the jth element of y is correctly associated with the jth component of x.

As an example, let's consider sample.exprSet which is an *exprSet* supplied with Biobase. We will print a little report, then the first N-vector of gene expressions and some covariate data:

```
> print(sample.exprSet)

Expression Set (exprSet) with
        500 genes
        26 samples
        phenoData object with 3 variables and 26 cases
        varLabels
                sex: Female/Male
                type: Case/Control
                score: Testing Score

> print(exprs(sample.exprSet)[1, ])

       A        B        C        D        E        F        G
192.7420  85.7533 176.7570 135.5750  64.4939  76.3569 160.5050
       H        I        J        K        L        M        N
 65.9631  56.9039 135.6080  63.4432  78.2126  83.0943  89.3372
       O        P        Q        R        S        T        U
 91.0615  95.9377 179.8450 152.4670 180.8340  85.4146 157.9890
       V        W        X        Y        Z
146.8000  93.8829 103.8550  64.4340 175.6150

> print(pData(sample.exprSet)[1:2, 1:3])

     sex    type score
A Female Control  0.75
B   Male    Case  0.40
```

Now let's see how expressions and a covariate are related:

```
> print(rbind(exprs(sample.exprSet[1, ]), sex = t(pData(sample.exprSet))[1,
+     ]))
```

```
               A         B         C         D         E
AFFX-MurIL2_at "192.742" "85.7533" "176.757" "135.575" "64.4939"
sex            "Female"  "Male"    "Male"    "Male"    "Female"
               F         G         H         I         J
AFFX-MurIL2_at "76.3569" "160.505" "65.9631" "56.9039" "135.608"
sex            "Male"    "Male"    "Male"    "Female"  "Male"
               K         L         M         N         O
AFFX-MurIL2_at "63.4432" "78.2126" "83.0943" "89.3372" "91.0615"
sex            "Male"    "Female"  "Male"    "Male"    "Female"
               P         Q         R         S         T
AFFX-MurIL2_at "95.9377" "179.845" "152.467" "180.834" "85.4146"
sex            "Female"  "Female"  "Male"    "Male"    "Female"
               U         V        W         X         Y
AFFX-MurIL2_at "157.989" "146.8"  "93.8829" "103.855" "64.434"
sex            "Male"    "Female" "Male"    "Male"    "Female"
               Z
AFFX-MurIL2_at "175.615"
sex            "Female"
```

A function that evaluates the difference in median expression across strata defined using an abstract covariate `x` is

```
> medContr <- function(y, x) {
+     ys <- split(y, x)
+     median(ys[[1]]) - median(ys[[2]])
+ }
```

We can apply this to a small *exprSet* that gives back the data listed above:

```
> print(apply(exprs(sample.exprSet[1, , drop = F]), 1,
+     medContr, pData(sample.exprSet)[["sex"]]))
```

```
AFFX-MurIL2_at
     -12.7935
```

That's a bit clumsy. This is where `esApply` comes in. We pay for some simplicity by following a strict protocol for the definition of the statistical function to be applied.

```
> medContr1 <- function(y) {
+     ys <- split(y, sex)
+     median(ys[[1]]) - median(ys[[2]])
+ }
> print(esApply(sample.exprSet, 1, medContr1)[1])
```

```
AFFX-MurIL2_at
      -12.7935
```

The manual page on `esApply` has a number of additional examples that show how applicable functions can be constructed and used. The important thing to note is that the applicable functions *know* the names of the covariates in the `pData` dataframe.

This is achieved by having an environment populated with all the variables in the `phenoData` component of the *exprSet* put in as the environment of the function that will be applied. If that function already has an environment we retain that but in the second position. Thus, there is some potential for variable shadowing.

1 Session Information

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

```
R version 2.5.1 (2007-06-27)
x86_64-unknown-linux-gnu

locale:
LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=en_US;LC_MESSAGE

attached base packages:
[1] "tools"     "stats"     "graphics"  "grDevices" "utils"
[6] "datasets"  "methods"   "base"

other attached packages:
 Biobase
"1.14.1"
```