Supplementary Analysis For: The sva package for removing batch effects and other unwanted variation in high-throughput experiments

Jeffrey Leek[1]*, W. Evan Johnson[2], Andrew Jaffe[1], Hilary Parker[1], John Storey[3]
[1]Johns Hopkins Bloomberg School of Public Health
[2]Boston University
[3]Princeton University
email: jleek@jhsph.edu

Modified: December 22, 2011 Compiled: December 22, 2011

Contents

1 Setting up the data from an ExpressionSet

The analyses performed in this experiment are based on gene expression measurements from a bladder cancer study [1]. The data can be loaded from the bladderbatch data package. The relevant packages for this Supplement can be loaded with the code:

```
> options(width = 10)
> library(sva)
> library(bladderbatch)
```

```
> library(RColorBrewer)
> data(bladderdata)
> library(pamr)
> library(limma)
```

For the bladder cancer study, the variable of interest is cancer status. To begin we will assume no adjustment variables. The bladder data are stored in an expression set - a Bioconductor object used for storing gene expression data. The variables are stored in the phenotype data slot and can be obtained as follows:

```
> pheno = pData(bladderEset)
```

The expression data can be obtained from the expression slot of the expression set.

```
> edata = exprs(bladderEset)
```

2 Applying the `fsva` function to remove batch effects for prediction

The surrogate variable analysis functions have been developed for population-level analyses such as differential expression analysis in microarrays. In some cases, the goal of an analysis is prediction. In this case, data sets are generally composed a training set and a test set. For each sample in the training set, the outcome/class is known, but latent sources of variability are unknown. For the samples in the test set, neither the outcome/class or the latent sources of variability are known.

"Frozen" surrogate variable analysis can be used to remove latent variation in the test data set. To illustrate these functions, the bladder data can be separated into a training and test set.

```
> set.seed(12354)
> trainIndicator = sample(1:57,size=30,replace=F)
> testIndicator = (1:57)[-trainIndicator]
> trainData = edata[,trainIndicator]
> testData = edata[,testIndicator]
> trainPheno = pheno[trainIndicator,]
> testPheno = pheno[testIndicator,]
```

Using these data sets, the pamr package can be used to train a predictive model on the training data, as well as test that prediction on a test data set.

```
> mydata = list(x=trainData,y=trainPheno$cancer)
> mytrain = pamr.train(mydata)
```

```
1234567891011121314151617181920212223242526272829 30
```

```
> table(pamr.predict(mytrain,testData,threshold=2),testPheno$cancer)
```

```
        Biopsy
  Biopsy      3
  Cancer      0
  Normal      0

        Cancer
  Biopsy      1
  Cancer     16
  Normal      2

        Normal
  Biopsy      4
  Cancer      1
  Normal      0
```

Next, the sva function can be used to calculate surrogate variables for the training set.

```
> trainMod = model.matrix(~cancer,data=trainPheno)
> trainMod0 = model.matrix(~1,data=trainPheno)
> trainSv = sva(trainData,trainMod,trainMod0)
```

```
Number of significant surrogate variables is:  6
Iteration (out of 5 ):1  2  3  4  5
```

The fsva function can be used to adjust both the training data and the test data. The training data is adjusted using the calculated surrogate variables. The testing data is adjusted using the "frozen" surrogate variable algorithm (to be submitted). The output of the fsva function is an adjusted training set and an adjusted test set. These can be used to train and test a second, more accurate, prediction function.

```
> fsvaobj = fsva(trainData,trainMod,trainSv,testData)

Correcting sample (out of 27 ):1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17

> mydataSv = list(x=fsvaobj$db,y=trainPheno$cancer)
> mytrainSv = pamr.train(mydataSv)

12345678910111213141516171819202122232425262728293O

> table(pamr.predict(mytrainSv,fsvaobj$new,threshold=1),testPheno$cancer)

        Biopsy
  Biopsy      3
  Cancer      0
  Normal      0

        Cancer
  Biopsy      0
  Cancer     18
  Normal      1

        Normal
  Biopsy      1
  Cancer      0
  Normal      4
```

Furthermore, we can cluster both the corrected and uncorrected training samples. Before `fsva` is applied, the biopsy and normal samples do not all cluster together (Figure 1 left panel). After `fsva` adjustment, the clustering of the test set is improved, with only one cancer sample clustering with the normals and biopsies (Figure 1 right panel).

```
> ## Load helper functions
>
> myplclust <- function( hclust, lab=hclust$labels,
+ lab.col=rep(1,length(hclust$labels)), hang=0.1,...){
+  ## modifiction of plclust for plotting hclust objects *in colour*!
+  ## Copyright Eva KF Chan 2009
```

```
+  ## Arguments:
+  ##    hclust:    hclust object
+  ##    lab:        a character vector of labels of the leaves of the tree
+  ##    lab.col:    colour for the labels; NA=default device foreground colour
+  ##    hang:     as in hclust & plclust
+  ## Side effect:
+  ##    A display of hierarchical cluster with coloured leaf labels.
+  y <- rep(hclust$height,2)
+  x <- as.numeric(hclust$merge)
+  y <- y[which(x<0)]
+  x <- x[which(x<0)]
+  x <- abs(x)
+  y <- y[order(x)]
+  x <- x[order(x)]
+  plot( hclust, labels=FALSE, hang=hang, ... )
+  text( x=x, y=y[hclust$order]-(max(hclust$height)*hang),
+  labels=lab[hclust$order], col=lab.col[hclust$order],
+  srt=90, adj=c(1,0.5), xpd=NA, ... )}
> mypar <- function(a=1,b=1,brewer.n=8,brewer.name="Dark2",...){
+  par(mar=c(2.5,2.5,1.6,1.1),mgp=c(1.5,.5,0))
+  par(mfrow=c(a,b),...)
+  palette(brewer.pal(brewer.n,brewer.name))
+ }
> ## Make plots
>
>
> mypar()
> tmp = apply(testData,1,var)
> keep = which(rank(-tmp) < 2000)
> hBefore = hclust(dist(t(testData[keep,])))
> tmp2  = apply(fsvaobj$new,1,var)
> keep2 = which(rank(-tmp2) < 2000)
> hAfter = hclust(dist(t(fsvaobj$new[keep2,])))
> par(mfrow=c(1,2))
> myplclust(hBefore,lab=testPheno$cancer,
+ lab.col=as.numeric(testPheno$cancer),main="Before fsva",xlab="")
> myplclust(hAfter,lab=testPheno$cancer,
+ lab.col=as.numeric(testPheno$cancer),main="After fsva",xlab="")
```

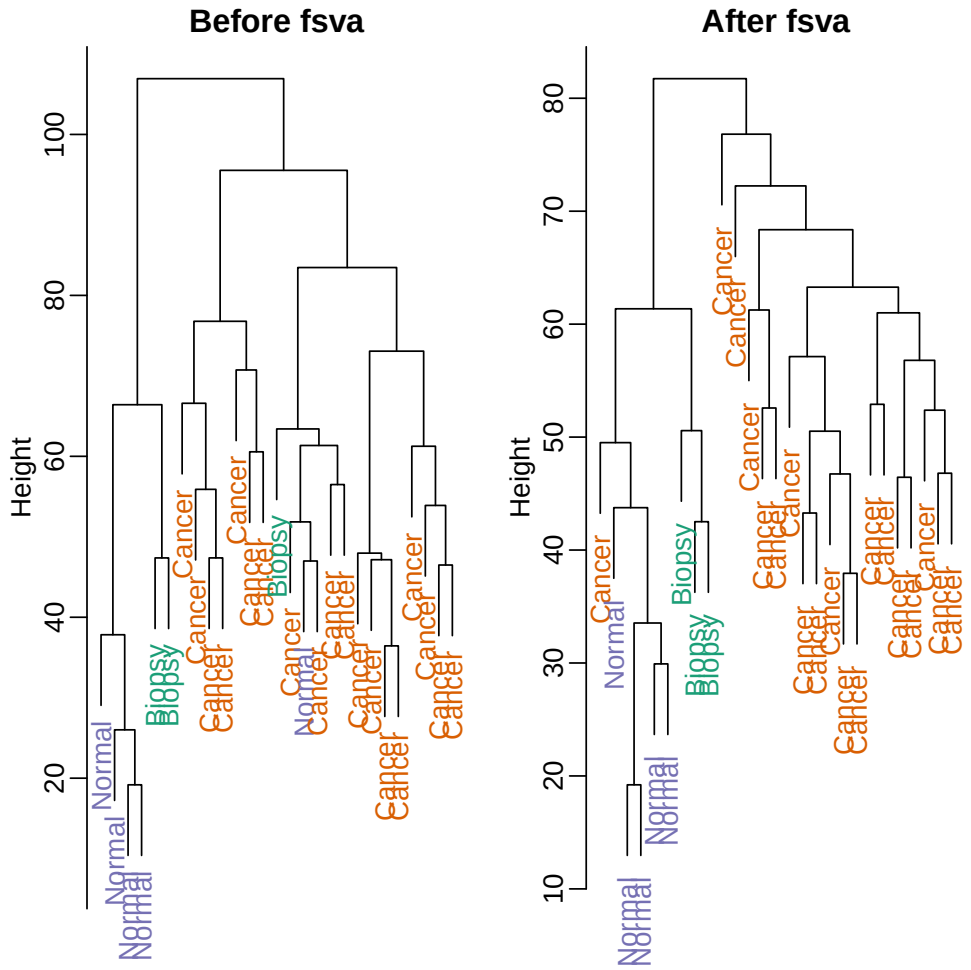

Figure 1: Clustering of the test samples in the bladder cancer study before (left) and after (right) adjustment with the `fsva` function. After using the `fsva` function, the normal and biopsy samples cluster together and only one cancer sample is "misclustered".

References

[1] L. Dyrskjot, M. Kruhoffer, T. Thykjaer, N. Marcussen, J. L. Jensen, K. Moller, and T. F. Orntoft. Gene expression in the urinary bladder: a common carcinoma in situ gene expression signature exists disregarding histopathological classification. *Cancer Res.*, 64:4040–4048, Jun 2004.