BB512/BB612 - Week V

```
suppressPackageStartupMessages(library(Biobase))
suppressPackageStartupMessages(library(bladderbatch))
suppressPackageStartupMessages(library(sva))
```

Data

The analyses are based on gene expression data from a bladder cancer study: Gene expression in the urinary bladder: a common carcinoma in situ gene expression signature exists disregarding histopathological classification. The data can be loaded from the bladderbatch data package.

```
data(bladderdata)

pheno <- pData(bladderEset)
edata <- exprs(bladderEset)
```

```
head(pheno)
```

```
##              sample outcome batch cancer
## GSM71019.CEL      1  Normal     3 Normal
## GSM71020.CEL      2  Normal     2 Normal
## GSM71021.CEL      3  Normal     2 Normal
## GSM71022.CEL      4  Normal     3 Normal
## GSM71023.CEL      5  Normal     3 Normal
## GSM71024.CEL      6  Normal     3 Normal
```

```
table(pheno$outcome)
```

```
##
##   Biopsy     mTCC   Normal sTCC-CIS sTCC+CIS
##        9       12        8       16       12
```

```
table(pheno$cancer)
```

```
##
## Biopsy Cancer Normal
##      9     40      8
```

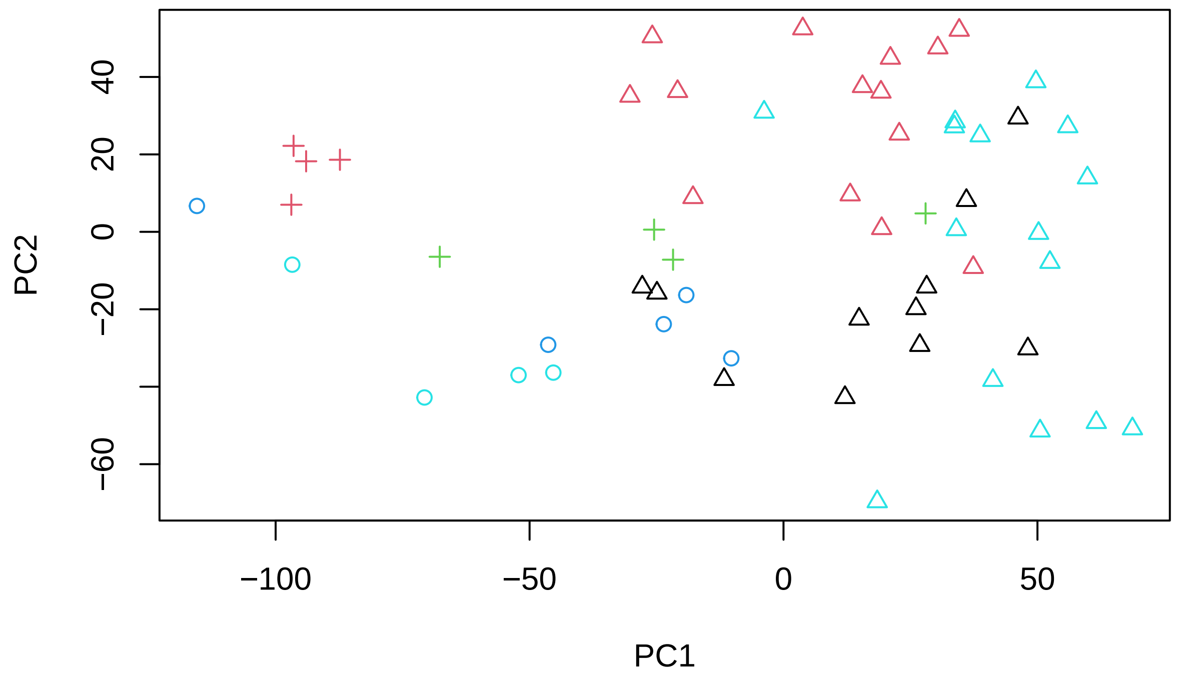
```
table(pheno$batch)
```

```
##
##  1  2  3  4  5
## 11 18  4  5 19
```

Visualizing by batch

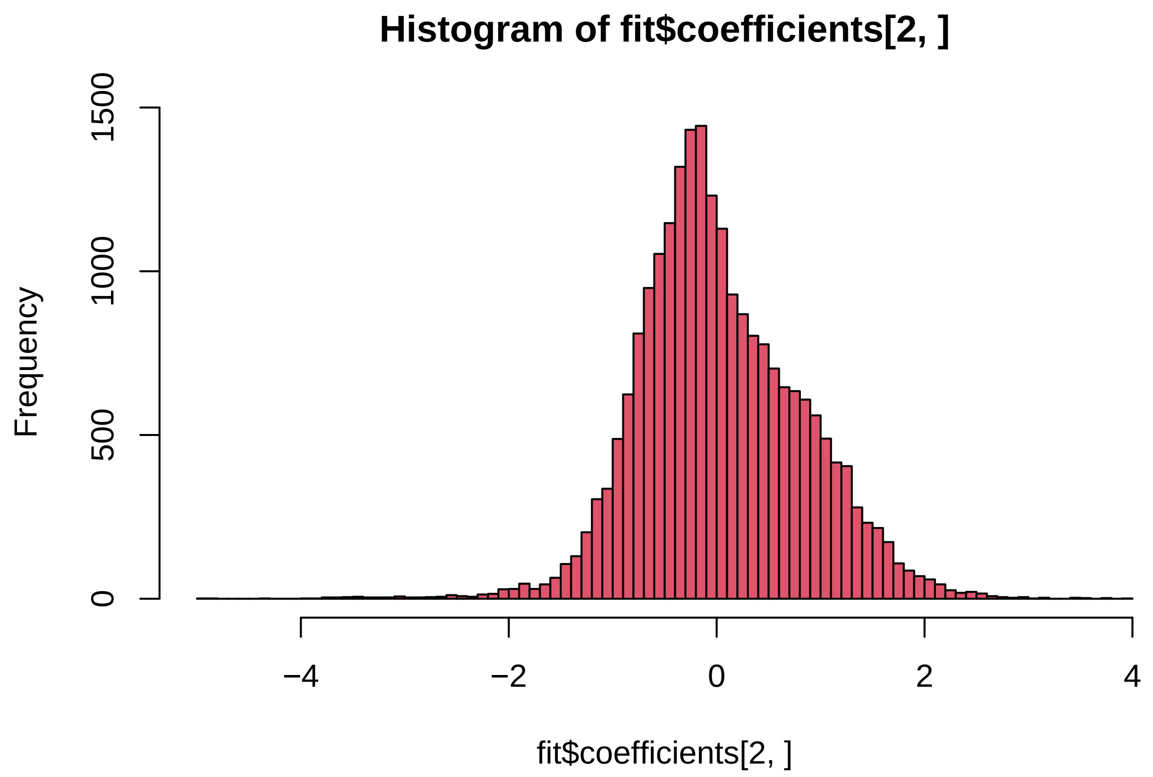
```
pr_res <- prcomp(t(edata))
```

```r
plot(pr_res$x[, 1], pr_res$x[, 2], col = pheno$batch, pch = as.numeric(pheno$cancer),
     xlab = "PC1", ylab = "PC2")
```



Adjusting for batch effects with a linear model

```r
mod <- model.matrix(~as.factor(cancer) + as.factor(batch), data=pheno)
fit <- lm.fit(mod, t(edata))
hist(fit$coefficients[2, ], col = 2, breaks = 100)
```



This will only work if the batch effects aren't too highly correlated with the outcome:

```
table(pheno$cancer, pheno$batch)
```

```
##
##           1  2  3  4  5
##   Biopsy  0  0  0  5  4
##   Cancer 11 14  0  0 15
##   Normal  0  4  4  0  0
```

Adjusting for batch effects with Combat

Another approach is to use Combat. Combat returns a "cleaned" data matrix after batch effects have been removed. Here we pass a model matrix with any known adjustment variables and a second parameter that is the batch variable.

```
batch <- pheno$batch
modcombat <- model.matrix(~1, data=pheno)
modcancer <- model.matrix(~cancer, data=pheno)
combat_edata <- ComBat(dat = edata, batch = batch, mod = modcombat, par.prior = TRUE, prior.plots = FALS
```

```
## Found5batches
```

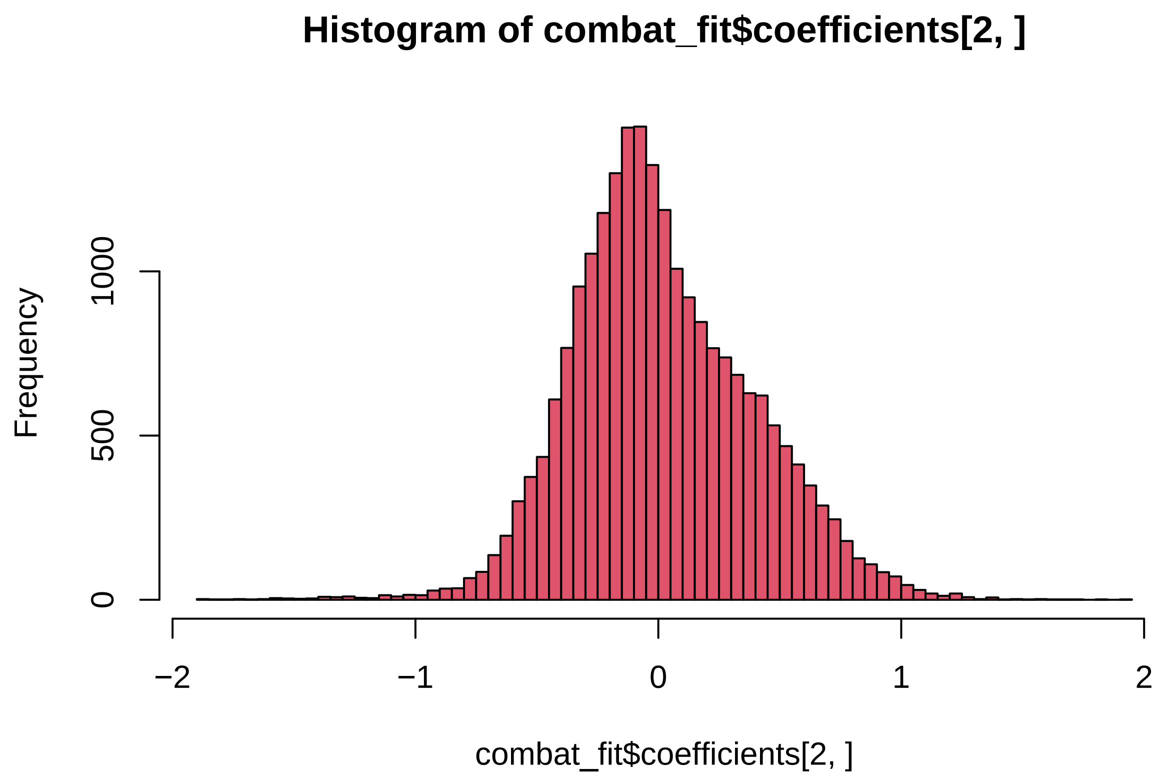
```
## Adjusting for0covariate(s) or covariate level(s)
```

```
## Standardizing Data across genes
```

```
## Fitting L/S model and finding priors
## Finding parametric adjustments

## Adjusting the Data
```

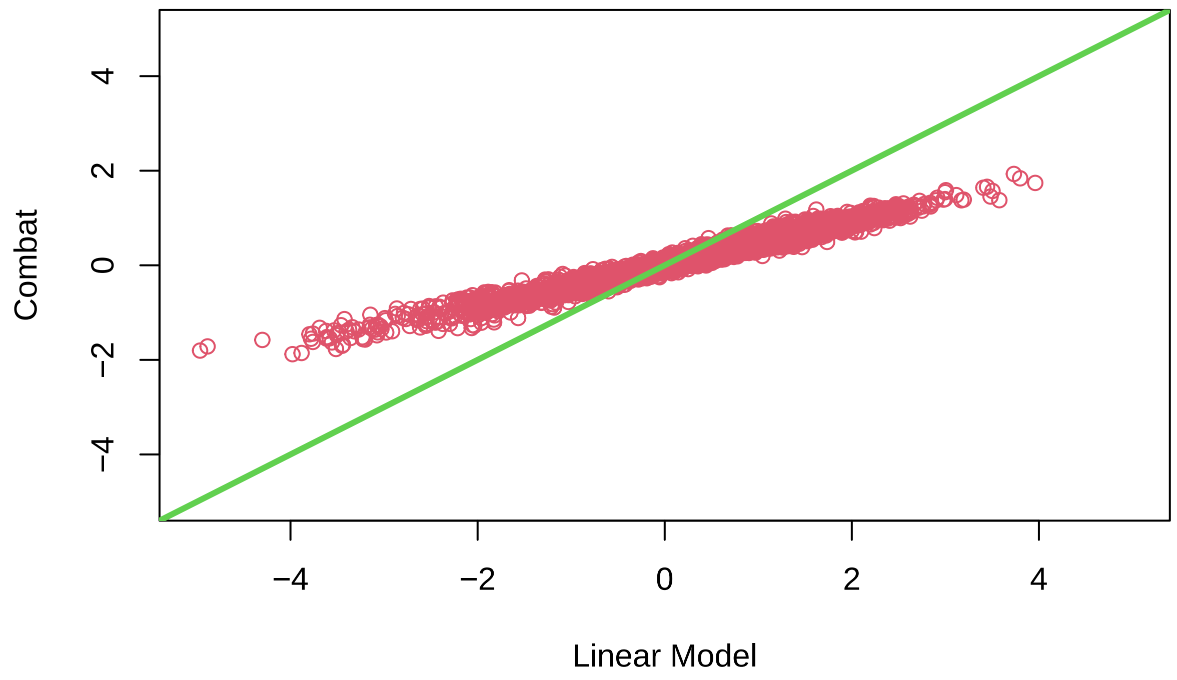
```
combat_fit <- lm.fit(modcancer, t(combat_edata))
hist(combat_fit$coefficients[2,], col = 2, breaks = 100)
```



Comparing Combat and linear adjustment

We can compare the estimated coefficients from Combat and linear adjustment by looking at the right coefficients for each model:

```
plot(fit$coefficients[2, ], combat_fit$coefficients[2, ], col = 2,
      xlab = "Linear Model", ylab = "Combat", xlim = c(-5, 5), ylim = c(-5, 5))
abline(c(0, 1), col = 3, lwd = 3)
```



Adjusting for batch effects with sva

First we need to estimate the surrogate variables. To do this, we need to build a model with any known adjustment variables and the variable we care about `mod` and another model with only the adjustment variables `mod0`. Here, we won't adjust for anything to see if `sva` can "discover" the batch effect.

```
mod <- model.matrix(~cancer, data = pheno)
mod0 <- model.matrix(~1, data = pheno)
sva1 <- sva(edata, mod, mod0, n.sv=2)
```

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

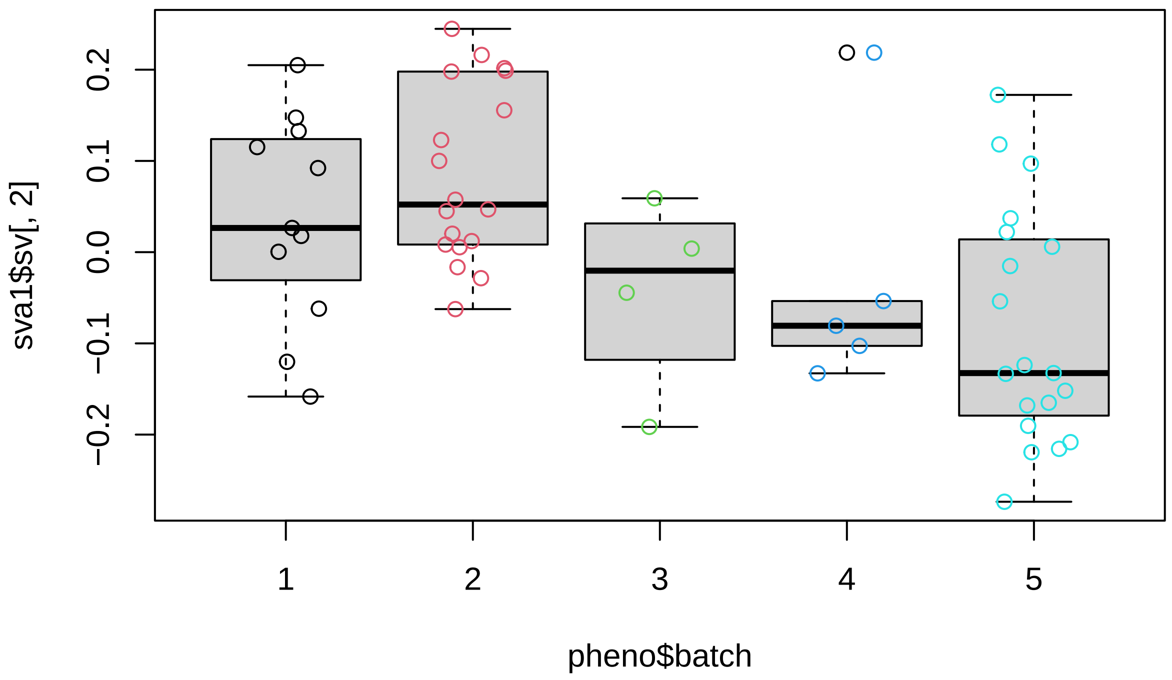
See if any of the variables correlate with batch:

```
summary(lm(sva1$sv ~ pheno$batch))
```

```
## Response Y1 :
##
## Call:
## lm(formula = Y1 ~ pheno$batch)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.26953 -0.11076  0.00787  0.10399  0.19069
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01847    0.03869   -0.48     0.64
## pheno$batch  0.00605    0.01125    0.54     0.59
##
## Residual standard error: 0.134 on 55 degrees of freedom
## Multiple R-squared:  0.00523,    Adjusted R-squared:  -0.0129
## F-statistic: 0.289 on 1 and 55 DF,  p-value: 0.593
##
##
## Response Y2 :
##
## Call:
## lm(formula = Y2 ~ pheno$batch)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.2397 -0.0747 -0.0216  0.0812  0.2563
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.12111    0.03416    3.55  0.00081 ***
## pheno$batch -0.03967    0.00993   -3.99  0.00019 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.119 on 55 degrees of freedom
## Multiple R-squared:  0.225,  Adjusted R-squared:  0.211
## F-statistic:   16 on 1 and 55 DF,  p-value: 0.000194
```

```
boxplot(sva1$sv[,2] ~ pheno$batch)
points(sva1$sv[,2] ~ jitter(as.numeric(pheno$batch)),col=as.numeric(pheno$batch))
```

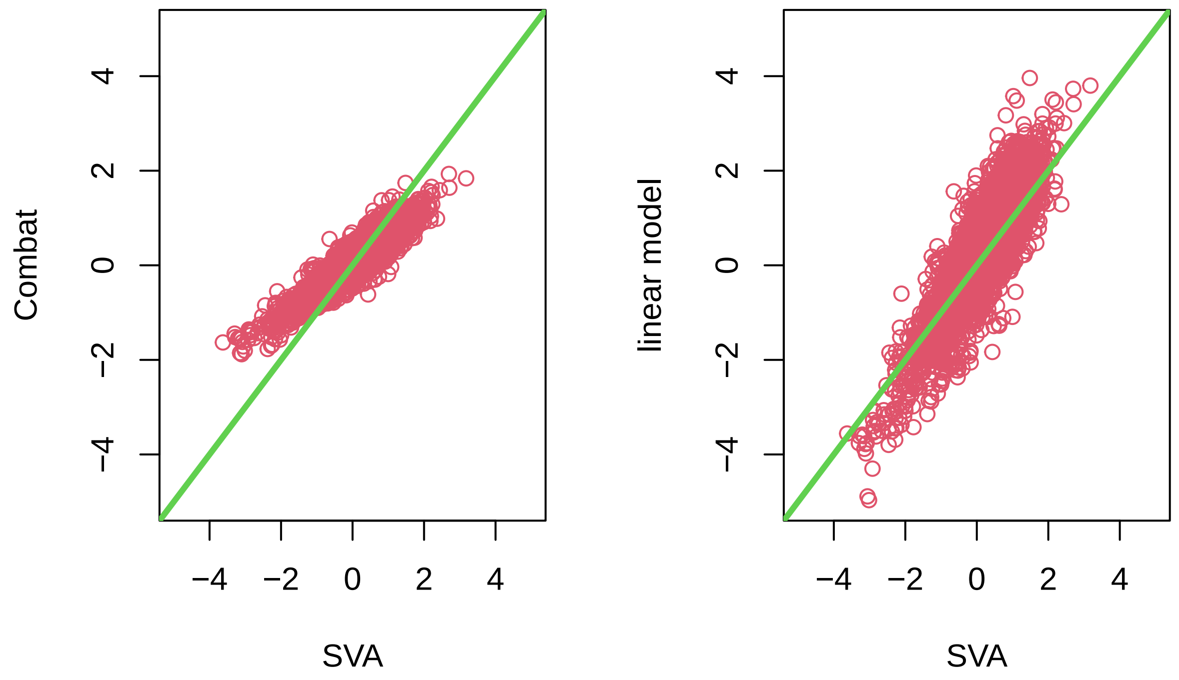


Add the surrogate variables to the model matrix and perform the model fit:

```
modsv <- cbind(mod,sva1$sv)
fitsv <- lm.fit(modsv, t(edata))
```

Compare the fit from surrogate variable analysis to the other two:

```
par(mfrow=c(1, 2))
plot(fitsv$coefficients[2, ], combat_fit$coefficients[2, ], col = 2,
      xlab = "SVA", ylab = "Combat", xlim = c(-5, 5), ylim = c(-5, 5))
abline(c(0, 1), col = 3, lwd = 3)
plot(fitsv$coefficients[2, ], fit$coefficients[2, ], col = 2,
      xlab="SVA", ylab="linear model", xlim = c(-5, 5),ylim = c(-5, 5))
abline(c(0, 1), col = 3, lwd = 3)
```



```
par(mfrow=c(1, 1))
```

Read more about bath effect corrections on the sva vignette