Appendix A, description of the ComBat procedure

Rik de Wijn

9/3/2020

## Introduction

* This is a technical description of the Combat method for batch correction of micro array data by Johnson et al. (Biostatistics (2007) 8 p118-127). The goal is to calculate a batch correction for *N* x *G* data matrix **X** with *N* observations and *G* spots, where the observations are subdivided in multiple batches between which a batch effect occurs (or may occur). **bx** is the batch indicator vector of length *N*.
* Below the original ComBat method estimating an L/S model without designated reference batch is described
* Two optional modifications of the ComBat model have been introduced by Zhang et al. (BMC bioinformatics (2018) 19:226) that are of practical importance: – Estimating a Location-model (mean.only) rather than a Location and Scale (L/S) model. – Use of a designated reference batch that will not be changed by the correction; the other batches will be corrected *towards* the reference batch.

bx = factor(c(1,1,1,1,2,2,2,2,2,3,3,3)) #12 observations divided over 3 batches  
Y = matrix(nrow = length(bx), ncol = 100, data = rnorm(100\*length(bx))) # 100 spots

The goal is to find a location and scale (L/S) adjustment for systematic effects between the batches.

Let represent the signal for spot *g* from observation *j* in batch *i*, the L/S model is

, overall signal for spot *g*

, additive batch effect (location) of batch *i*

, multiplicative batch effect (scale) of batch *i*

, residual error (assumed normal) for spot *g* in observation *j* in batch *i*

Note that here the term for covariates used by Johnson et al. has been dropped.

The batch adjusted data is then given by:

, and we have to find the estimators , , and

## Empirical Bayes Method for finding the batch adjustment

### Step 1. standardize the data

Where the *overall* L estimator is obtained using an ordinary least squares approach, i.e. numerically solve (for ) the system of equations:

, where is the batch design matrix.

represents an estimate of the mean per spot per batch. For each spot is then calculated as a weighted mean over the batches.

nObsPerBatch = summary(bx)  
nObs = sum(nObsPerBatch)  
  
B = model.matrix(~-1 + bx)  
lamb.hat = solve( t(B)%\*% B, t(B) %\*% Y)  
alpha\_g = (nObsPerBatch/nObs) %\*% lamb.hat

*Overall* variance is estimated as a pooled variance over the batches,

and calculated.

siggsq = t((Y - (B %\*% lamb.hat))^2) %\*% rep(1/nObs, nObs)  
Z = scale(Y, center = alpha\_g, scale = sqrt(siggsq))

### Step 2. Empirical Bayes estimate of batch effect parameters using parametric emprirical priors

Scaled data Z is used as input for estimating the EB batch correction. The EB corrected data is given by:

is the adjusted estimator for the batch location effect and the adjusted estimator for the batch scale effect (Adjusted compared the per spot estimators and used above). The adjusted estimators are given by:

and

Where are the parameters of the assumed prior distributions for the location (normal) and scale (inverse gamma) effect,

and

and are estimated from the scaled data Z. With these parameters obtained the equations for and can be solved by a simple iterative procedure.

aprior = function(X) {  
 m <- mean(X)  
 s2 <- var(X)  
 (2\*s2 + m^2) / s2  
}  
  
bprior = function(X){  
 m <- mean(X)  
 s2 <- var(X)  
 (m\*s2 + m^3) / s2  
}  
  
postmean = function(g.hat,g.bar,n,d.star,t2){  
 (t2\*n\*g.hat + d.star\*g.bar) / (t2\*n + d.star)  
}  
  
postvar = function(sum2,n,a,b){  
 (.5\*sum2 + b) / (n/2 + a - 1)  
}  
it.sol = function(params, Z, lambda.hat, sigma.hat, conv = .0001)  
{  
 g.old = lambda.hat  
 d.old = sigma.hat  
 change <- 1  
 count <- 0  
 while(change>conv){  
 g.new <- postmean(lambda.hat, params$lambda.bar, params$n, d.old, params$t2)  
 sum2 = colSums(scale(Z, center = g.new, scale = FALSE)^2)  
 d.new <- postvar(sum2, params$n, params$gamma, params$theta)  
 change <- max(abs(g.new-g.old) / g.old, abs(d.new-d.old) / d.old)  
 g.old <- g.new  
 d.old <- d.new  
 count <- count+1  
 }  
 cat("This batch took", count, "iterations until convergence\n")  
 result = data.table(lambda.star = list(g.new), sigma.star = list(d.new))  
}  
  
  
lambda.hat = solve( t(B)%\*% B, t(B) %\*% Z) # unadjusted location  
sigma.hat = NULL  
for(i in 1:nlevels(bx)){  
 sigma.hat = rbind(sigma.hat, apply(Z[bx == levels(bx)[i],,drop = FALSE],2,var))  
} # unadjusted scale  
  
params = data.frame( bx = levels(bx),  
 bi = 1:nlevels(bx),  
 lambda.bar = rowMeans(lambda.hat),  
 t2 = apply(lambda.hat,1, var),  
 gamma = apply(sigma.hat, 1, aprior),  
 theta = apply(sigma.hat, 1, bprior),  
 n = summary(bx))  
  
# solving for batch effect  
post = params %>% group\_by(bx) %>% do(it.sol(., Z = Z[bx == .$bx,], lambda.hat[.$bi,], sigma.hat[.$bi,]))

## This batch took 7 iterations until convergence  
## This batch took 6 iterations until convergence  
## This batch took 9 iterations until convergence

### Step 3. Return batch corrected data

batchcorrect = function(Z, bx, post, lambda\_g, sigmasq\_g){  
 Ystar = matrix(nrow = dim(Z)[1], ncol = dim(Z)[2])  
 for (i in 1:dim(Z)[1]){  
 bIdx = (1:nlevels(bx))[bx[i] == levels(bx)]  
 zstar= (Z[i,] - post$lambda.star[[bIdx]])/sqrt(post$sigma.star[[bIdx]])  
 Ystar[i,] = sqrt(sigmasq\_g) \* zstar + lambda\_g  
 }  
 return(Ystar)  
}  
  
#Ystar = batchcorrect(Z, bx, post, alpha\_g, siggsq)

In the pgBatch package (<https://rikdewijn@bitbucket.org/bnoperator/pgbatch.git>) the procedure is implemented using a class with a “Fit” and “Apply” method. Fit method returns an object with teh correction parameters, Apply method can be used to apply the correction to a data set. In this way a ComBat model fitted on data set (i.e. REF samples), can be applied to set (DAS samples). A clean version of the code is provided in “combat\_reworked.r”.

### Reference batch

* Data is corrected while keeping a designated reference batch unchanged.
* Data scaling is performed based on the data in the reference set.
* EB parameters found in the same way as with ref batch.
* Location and scale of the ref batch are forced to identical 1 and 0, respectively, before applying the correction

See re-engineered code for implementation.

### Location only model

Only batch differences for Location are estimated, Scale is not adjusted. The model for teh batch effect is slightly changed:

The same approach as above is used for estimating , the are not estimated or used for adjustment.

See re-engineered code for implementation.