BatchQC Advanced Usage

Claire Ruberman, Solaiappan Manimaran

2021-05-19

Simulated Example

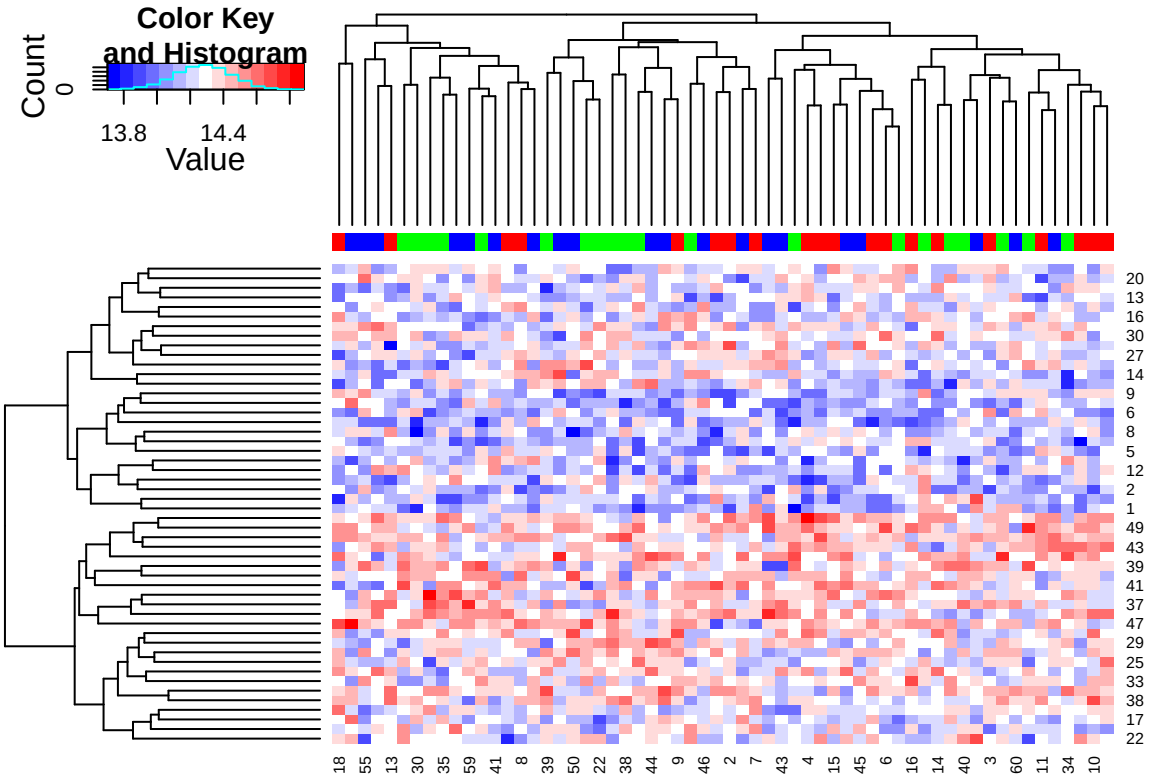
(a) Use the data simulating mechanism from the batchQC package

```
require(MCMCpack)
```

```
## Warning in .recacheSubclasses(def@className, def, env): undefined subclass
## "numericVector" of class "Mnumeric"; definition not updated
```

```
require(sva)
require(BatchQC)

# Simulate Count Data
## output is ngenes by (nbatch x ncond x npercond) matrix
## ggstep: Gene to Gene step variation
## bbstep: Batch to Batch step variation
## ccstep: Condition to Condition step variation
## basedisp: Base Dispersion
## bdispstep: Batch to Batch Dispersion step variation

set.seed(47)

nbatch <- 3
ncond <- 2
npercond <- 10
ngenes <- 50
ggstep <- 50
bbstep <- 2000
ccstep <- 800

basedisp <- 100
bdispstep <- -10
data.matrix <- rnaseq_sim(ngenes=ngenes, nbatch=nbatch, ncond=ncond, npercond=
    npercond, basemean=10000, ggstep=ggstep, bbstep=bbstep, ccstep=ccstep,
    basedisp=basedisp, bdispstep=bdispstep, swvar=1000, seed=1234)

# genes 10 to 25 affected by an independent unobserved factor
unmodeled.factor.indicator=rbinom(60,1,.5)

nsamples=nbatch*ncond*npercond
eh.matrix <- matrix(0, nrow=ngenes, ncol=nsamples)
for(j in 1:nsamples){
    bsize <- seq(basedisp, length.out=nbatch, by=bdispstep)
    size<-rinvgamma(1, shape=mean(bsize), scale=1)
```

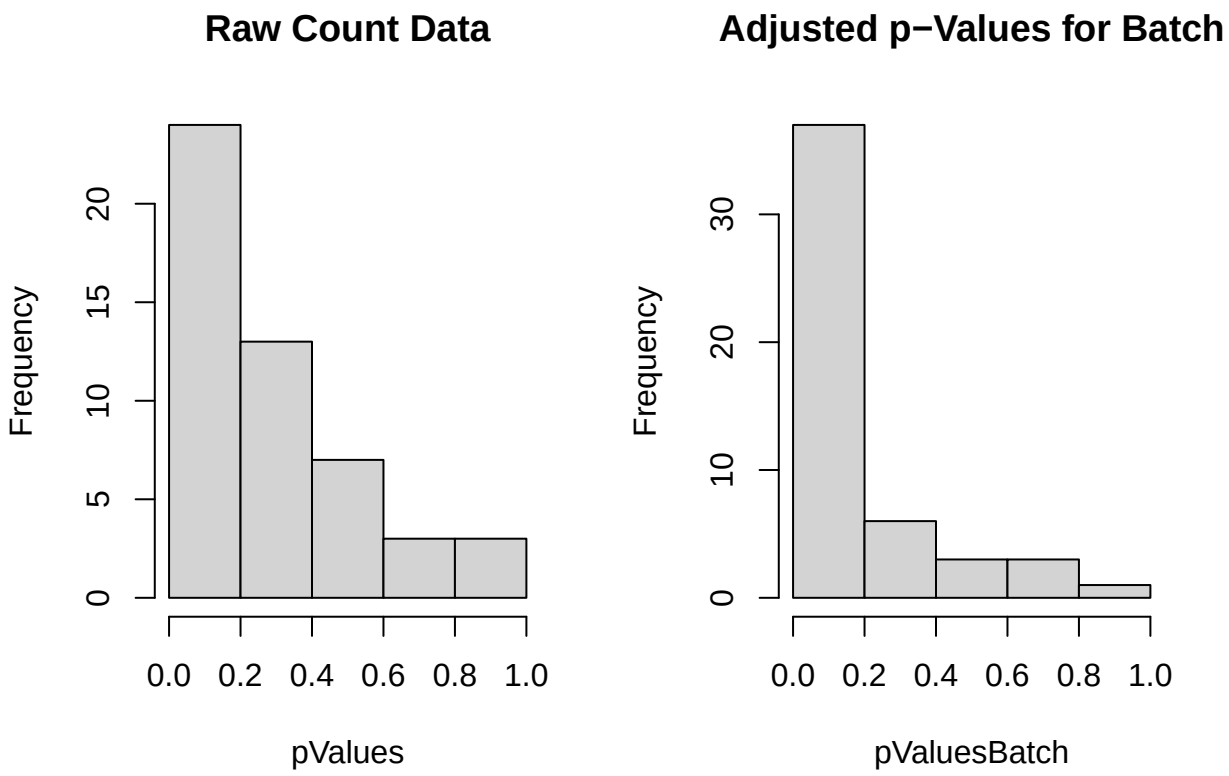
```
    bmu <- seq(bbstep, length.out=nbatch, by=bbstep)
    cmu <- seq(ccstep, length.out=ncond, by=ccstep)
    eh.mu=rnorm(1, mean=mean(bmu), sd=1)
    mu=eh.mu*unmodeled.factor.indicator[j]
    eh.matrix[10:25,j]=rnbinom(16,size=size,mu=mu)
}

data.matrix.eh=data.matrix+eh.matrix

# Apply BatchQC
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
nsample <- nbatch*ncond*npercond
sample <- 1:nsample
pdata <- data.frame(sample, batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
modmatrix.null = model.matrix(~1,data=pdata)
## null model matrix (just intercept)

par(mfrow=c(1,1))
heatmap=batchqc_heatmap(data.matrix, batch, mod=modmatrix)
```




```
#heatmap.eh=batchqc_heatmap(data.matrix.eh, batch, mod=modmatrix)

n.sv=batchQC_num.sv(data.matrix,modmatrix)
#n.sv.eh=batchQC_num.sv(data.matrix.eh,modmatrix)

combat_data.matrix = ComBat(dat=data.matrix, batch=batch, mod=modmatrix)
#combat_data.matrix.eh = ComBat(dat=data.matrix.eh, batch=batch, mod=modmatrix)
```

```
# sva.object=batchQC_sva(data.matrix, modmatrix)
# #sva.object.eh=batchQC_sva(data.matrix.eh, modmatrix)
#
# ## Plot the surrogate variables by batch and the unmodeled factor
#
# par(mfrow=c(1,2))
# if (sva.object$n.sv > 1)  {
#     for(i in 1:sva.object$n.sv){
#         boxplot(sva.object$sv[,i]~unmodeled.factor.indicator,xlab=
#             "Unmodeled Factor ",ylab=paste("Surrogate Variable",i,sep=" "),main=
#             "Simulated Data ")
#         boxplot(sva.object$sv[,i]~batch,xlab="Batch",ylab=paste(
#             "Surrogate Variable",i,sep=" "),main="Simulated Data")
#     }
# } else  {
#     boxplot(sva.object$sv~unmodeled.factor.indicator,xlab=
#         "Unmodeled Factor ",ylab=paste("Surrogate Variable",1,sep=" "),main=
#         "Simulated Data ")
#     boxplot(sva.object$sv~batch,xlab="Batch",ylab=paste(
#         "Surrogate Variable",1,sep=" "),main="Simulated Data")
# }
#
# pprob.gam=sva.object$pprob.gam ##prob each gene is affected by EH
# index.p.batch.8=which(pprob.gam>=.8)
#
# par(mfrow=c(1,1))
# ### Look at heatmap for genes just affected by EH
# heatmap=batchqc_heatmap(data.matrix[index.p.batch.8,], batch, mod=modmatrix)


### Histogram of the p values

#### (a) Raw data
pValues=f.pvalue(data.matrix,modmatrix,modmatrix.null)
qValues = p.adjust(pValues,method="BH")

#### (b) Apply Combat
pValuesComBat=f.pvalue(combat_data.matrix,modmatrix,modmatrix.null)
qValuesComBat = p.adjust(pValuesComBat,method="BH")

#### (c) Include Batch
modBatch = model.matrix(~as.factor(condition) + as.factor(batch),data=pdata)
mod0Batch = model.matrix(~as.factor(batch),data=pdata)

pValuesBatch = f.pvalue(data.matrix,modBatch,mod0Batch)
qValuesBatch = p.adjust(pValuesBatch,method="BH")

#### (d) Include SV's
# modSv = cbind(modmatrix,sva.object$sv)
# ## include surrogate variables in model matrix with condition (cancer status)
# mod0Sv = cbind(modmatrix.null,sva.object$sv)
# ## include surrogate variables in null model matrix
#
```

```r
# pValuesSv = f.pvalue(data.matrix,modSv,modOSv)
# ## pvalues including sv's in model matrix
# qValuesSv = p.adjust(pValuesSv,method="BH")

par(mfrow=c(1,2))
## Original Simulated Data
hist(pValues,main="Raw Count Data")
# hist(pValuesSv,main="Count Data after Combat")
hist(pValuesBatch,main="Adjusted p-Values for Batch")
```




```r
# hist(pValuesSv,main="Adjusted p-Values for SVs")
```

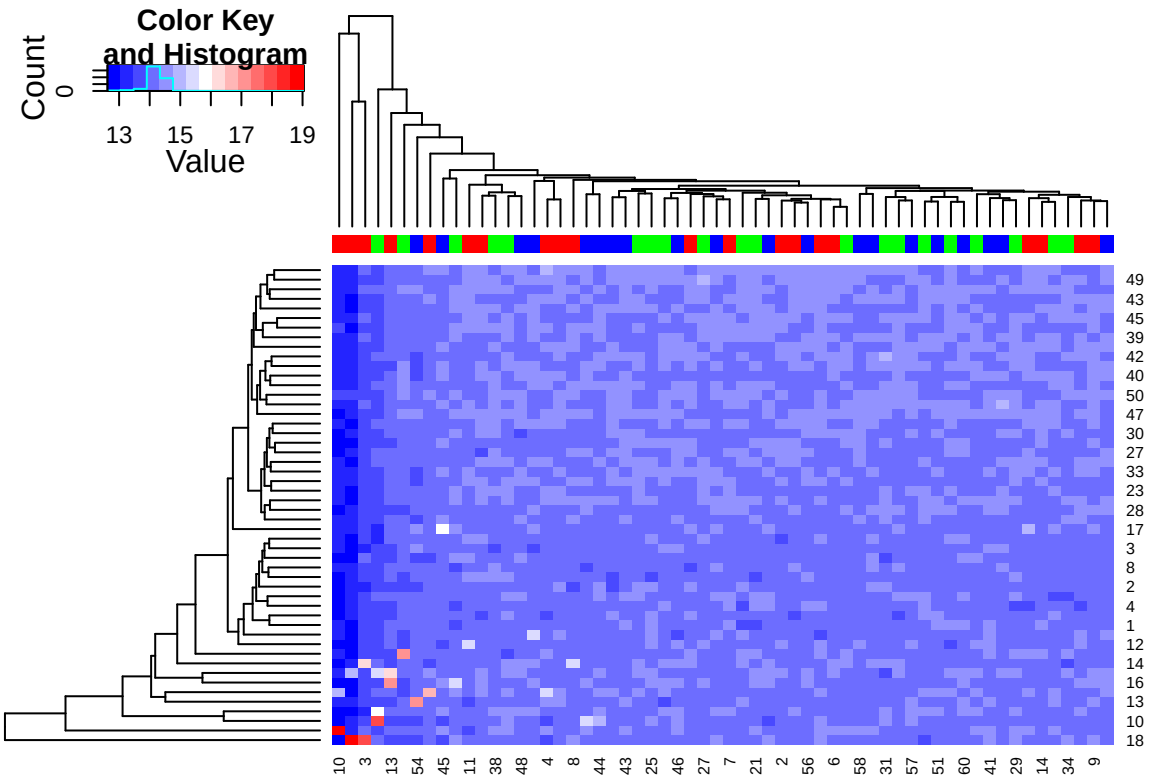
(b) Use the data simulating mechanism from the batchQC package,

adding additional expression heterogeneity from a dichotomous unmodeled

factor, independent of both batch and condition

```r
# Use simulated data as before, except genes 10 to 25 affected by an
# independent unobserved factor
unmodeled.factor.indicator=rbinom(60,1,.5)

nsamples=nbatch*ncond*npercond
eh.matrix <- matrix(0, nrow=ngenes, ncol=nsamples)
for(j in 1:nsamples){
    bsize <- seq(basedisp, length.out=nbatch, by=bdispstep)
    size<-rinvgamma(1, shape=mean(bsize), scale=1)
    bmu <- seq(bbstep, length.out=nbatch, by=bbstep)
    cmu <- seq(ccstep, length.out=ncond, by=ccstep)
    eh.mu=rnorm(1, mean=mean(bmu), sd=1)
    mu=eh.mu*unmodeled.factor.indicator[j]
```

```
    eh.matrix[10:25,j]=rnbinom(16,size=size,mu=mu)
}

data.matrix.eh=data.matrix+eh.matrix

# Apply BatchQC
batch <- rep(1:nbatch, each=ncond*npercond)
condition <- rep(rep(1:ncond, each=npercond), nbatch)
nsample <- nbatch*ncond*npercond
sample <- 1:nsample
pdata <- data.frame(sample, batch, condition)
modmatrix = model.matrix(~as.factor(condition), data=pdata)
modmatrix.null = model.matrix(~1,data=pdata)
## null model matrix (just intercept)

par(mfrow=c(1,1))
heatmap.eh=batchqc_heatmap(data.matrix.eh, batch, mod=modmatrix)
```




```
n.sv.eh=batchQC_num.sv(data.matrix.eh,modmatrix)

combat_data.matrix.eh = ComBat(dat=data.matrix.eh, batch=batch, mod=modmatrix)

# sva.object.eh=batchQC_sva(data.matrix.eh, modmatrix)
#
# par(mfrow=c(1,2))
# if (sva.object.eh$n.sv > 1)  {
#     for(i in 1:sva.object.eh$n.sv){
#         boxplot(sva.object.eh$sv[,i]~unmodeled.factor.indicator,xlab=
#             "Unmodeled Factor ",ylab=paste("Surrogate Variable",i,sep=" "),main=
```

```
#             "Simulated Data with EH")
#
#         boxplot(sva.object.eh$sv[,i]~batch,xlab="Batch",ylab=paste(
#             "Surrogate Variable",i,sep=" "),main="Simulated Data with EH")
#     }
# } else  {
#     boxplot(sva.object.eh$sv~unmodeled.factor.indicator,xlab=
#         "Unmodeled Factor ",ylab=paste("Surrogate Variable",1,sep=" "),main=
#         "Simulated Data with EH")
#
#     boxplot(sva.object.eh$sv~batch,xlab="Batch",ylab=paste(
#         "Surrogate Variable",1,sep=" "),main="Simulated Data with EH")
# }
#
# pprob.gam.eh=sva.object.eh$pprob.gam ##prob each gene is affected by EH
# index.p.batch.8.eh=which(pprob.gam.eh>=.8)

### Look at heatmap for genes just affected by EH
# par(mfrow=c(1,1))
# heatmap.eh=batchqc_heatmap(data.matrix.eh[index.p.batch.8.eh,], batch,
#     mod=modmatrix)

### Histogram of the p values

#### (a) Raw data
pValues=f.pvalue(data.matrix,modmatrix,modmatrix.null)
pValues.eh=f.pvalue(data.matrix.eh,modmatrix,modmatrix.null)

qValues = p.adjust(pValues,method="BH")
qValues.eh = p.adjust(pValues.eh,method="BH")

#### (b) Apply Combat
pValuesComBat=f.pvalue(combat_data.matrix,modmatrix,modmatrix.null)
pValuesComBat.eh=f.pvalue(combat_data.matrix.eh,modmatrix,modmatrix.null)

qValuesComBat = p.adjust(pValuesComBat,method="BH")
qValuesComBat.eh = p.adjust(pValuesComBat.eh,method="BH")

#### (c) Include Batch
modBatch = model.matrix(~as.factor(condition) + as.factor(batch),data=pdata)
mod0Batch = model.matrix(~as.factor(batch),data=pdata)

pValuesBatch = f.pvalue(data.matrix,modBatch,mod0Batch)
qValuesBatch = p.adjust(pValuesBatch,method="BH")

pValuesBatch.eh = f.pvalue(data.matrix.eh,modBatch,mod0Batch)
qValuesBatch.eh = p.adjust(pValuesBatch.eh,method="BH")

#### (d) Include SV's
# modSv = cbind(modmatrix,sva.object$sv)
# ## include surrogate variables in model matrix with condition (cancer status)
# mod0Sv = cbind(modmatrix.null,sva.object$sv)
# ## include surrogate variables in null model matrix
```

```
#
# modSv.eh = cbind(modmatrix,sva.object.eh$sv)
# ## include surrogate variables in model matrix with condition (cancer status)
# modOSv.eh = cbind(modmatrix.null,sva.object.eh$sv)
# ## include surrogate variables in null model matrix
#
# pValuesSv = f.pvalue(data.matrix,modSv,modOSv)
# ## pvalues including sv's in model matrix
# qValuesSv = p.adjust(pValuesSv,method="BH")

# pValuesSv.eh = f.pvalue(data.matrix.eh,modSv.eh,modOSv.eh)
# ## pvalues including sv's in model matrix
# qValuesSv.eh = p.adjust(pValuesSv.eh,method="BH")

par(mfrow=c(2,2))
## Additional source of EH Added
hist(pValues.eh,main="Raw Count Data (Added EH)")
# hist(pValuesSv.eh,main="Count Data after Combat (Added EH)")
hist(pValuesBatch.eh,main="Adjusted p-Values for Batch (Added EH)")
# hist(pValuesSv.eh,main="Adjusted p-Values for SVs (Added EH)")
```