

cbcbSEQ: RNAseq analysis for UMD CBCB collaborators

Kwame Okrah, Hector Bravo

Modified: June 7, 2013. Compiled: May 11, 2016

1 Overview of cbcbSEQ pipeline

The purpose of this pipeline is to streamline the process for analyzing RNA-seq data with potential batch effects. The pipeline includes 1) quantile normalization 2) log-transformation of counts 3) ComBat (location) batch correction 4) voom calculation of weights.

The functions in this package can be grouped into two main categories:

1. The functions used for assessing batch effects.
 - `makeSVD`
 - `pcRes`
 - `plotPC`
2. The functions for removing batch effect and computing weights for limma.
 - `qNorm`
 - `log2CPM`
 - `voomMod`
 - `combatMod`
 - `batchSEQ`

`batchSEQ` is the pipeline function. It combines `qNorm`, `log2CPM`, `voomMod`, and `combatMod` into one step.

Below we will illustrate how to use these functions using the pasilla data set.

note: All the functions in this package have a detailed help file which tells you what kind of objects go in and what kind of objects come out. It is important to look at these help files for each function.

2 Examples of how to use the functions

We will use the `pasilla` dataset found in the `pasilla` package. (This is the same dataset used in the DESeq vignette)

```
> require(pasilla)
> # locate the path of the dataset and read in the dataset
> datafile = system.file("extdata/pasilla_gene_counts.tsv", package="pasilla")
> counts = read.table(datafile, header=TRUE, row.names=1)
> head(counts)
```

	untreated1	untreated2	untreated3	untreated4	treated1	treated2
FBgn0000003	0	0	0	0	0	0
FBgn0000008	92	161	76	70	140	88
FBgn0000014	5	1	0	0	4	0
FBgn0000015	0	2	1	2	1	0
FBgn0000017	4664	8714	3564	3150	6205	3072
FBgn0000018	583	761	245	310	722	299

	treated3
FBgn0000003	1
FBgn0000008	70
FBgn0000014	0
FBgn0000015	0
FBgn0000017	3334
FBgn0000018	308

```
> dim(counts)
```

```
[1] 14599    7
```

```
> counts = counts[rowSums(counts) > ncol(counts),]
> dim(counts)
```

```
[1] 10153      7
```

In this dataset there are two biological conditions: treated (3 samples) and untreated (4 samples). Two samples are single-end and the other 4 are paired-end. We will use single-end and paired-end as illustration of batch effects. Below is the experiment design matrix (pheno data.frame).

```
> design = data.frame(row.names=colnames(counts),
+                      condition=c("untreated","untreated","untreated",
+                                  "untreated","treated","treated","treated"),
+                      libType=c("single-end","single-end","paired-end",
+                                 "paired-end","single-end","paired-end","paired-end"))
> design
```

```
      condition  libType
untreated1 untreated single-end
untreated2 untreated single-end
untreated3 untreated paired-end
untreated4 untreated paired-end
treated1      treated single-end
treated2      treated paired-end
treated3      treated paired-end
```

2.1 Explore data for batch effects

We will begin our analysis by exploring the data for possible/significant batch effects. We implemented here some of the analysis methods outlined in Leek et al. [2].

```
> # load batch package
> require(cbcSEQ)
> #
> # quantile normalize: adjust counts for library size.
> qcounts = qNorm(counts)
> # convert counts to log2 counts per milliom. (voom scale)
> cpm = log2CPM(qcounts)
> names(cpm)
```

```
[1] "y"      "lib.size"
```

```

> libsize = cpm$lib.size
> cpm = cpm$y
> #
> # PCA analysis
> # returns a list with two components v and d.
> res = makeSVD(cpm)

```

We can now call pcRes and plotPC.

- pcRes: computes variance of each principal component and how they "correlate" with batch and condition.

```

> pcRes(res$v,res$d, design$condition, design$libType)

```

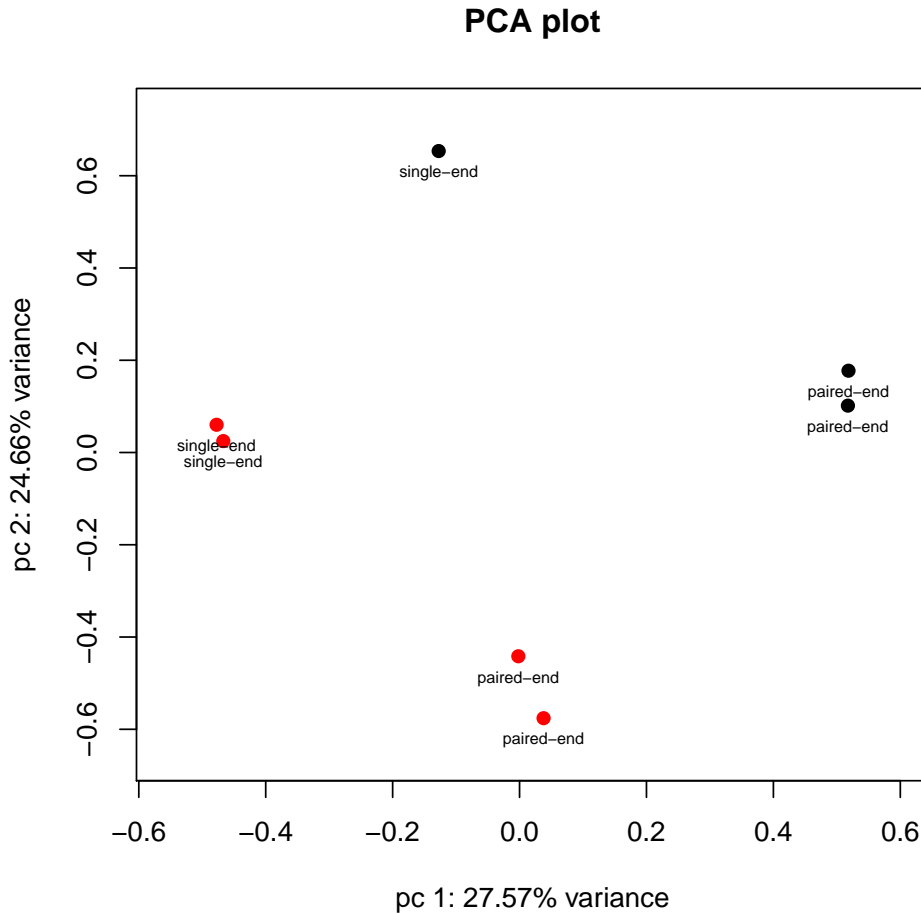
	propVar	cumPropVar	cond.R2	batch.R2
1	27.57	27.57	48.13	67.00
2	24.66	52.23	50.74	31.82
3	15.62	67.85	0.57	0.04
4	12.15	80.00	0.05	0.35
5	10.53	90.53	0.14	0.14
6	9.46	99.99	0.37	0.65

- plotPC: Plot first 2 principal components. This function works like the regular plot function in R. ie. We can add all the options to make the plot sensible and well labelled. Below is an example:

```

> plotPC(res$v,res$d,
+       col=design$condition, # color by batch
+       pch=19, main="PCA plot",
+       xlim=c(min(res$v[,1])-.08,max(res$v[,1])+.08),
+       ylim=c(min(res$v[,2])-.08,max(res$v[,2])+.08))
> text(res$v[,1], res$v[,2], design$libType, pos=1, cex=0.6)

```



We see that there is a batch effect in the data. Both in the PCA "correlation" table and the PCA plot.

2.2 Correct data for batch effects

A standard way of accounting for batch effects in data analysis is to include batch indicators as covariates in a linear model (e.g., in `limma` with weights computed by `voom` to model heteroscedasticity through a mean-variance relationship). However, in some cases we may want to obtain robust estimates of batch effects using a hierarchical model like `ComBat` [1]. However, we made some modifications to `Combat`. The most significant is that we do not estimate or adjust for batch scale effect due to heteroskedasticity. In order to account for scaling we have to take into account the mean var relationship inherent in this kind of data (we're working on it, but it's not done yet). We adjust data by removing the empirical

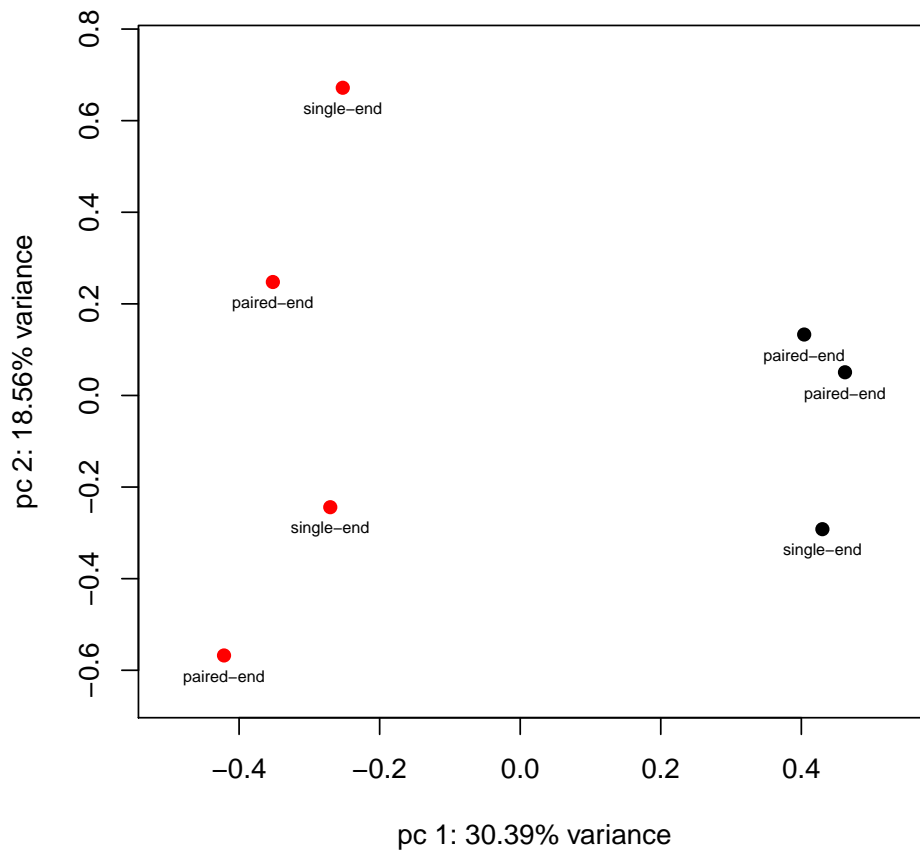
bayesian estimates of batch location effects.

```
> # combatMod function
> # noScale=TRUE option not to scale adjust
> tmp = combatMod(cpm, batch=design$libType, mod=design$condition, noScale=TRUE)
> # look at PCA results again
> res = makeSVD(tmp)
> # batch effect is reduced
> pcRes(res$v,res$d, design$condition, design$libType)
```

	propVar	cumPropVar	cond.R2	batch.R2
1	30.39	30.39	98.00	0.50
2	18.56	48.95	0.68	1.08
3	14.71	63.66	0.23	12.77
4	12.92	76.58	0.33	57.05
5	12.39	88.97	0.03	13.24
6	11.03	100.00	0.74	15.36

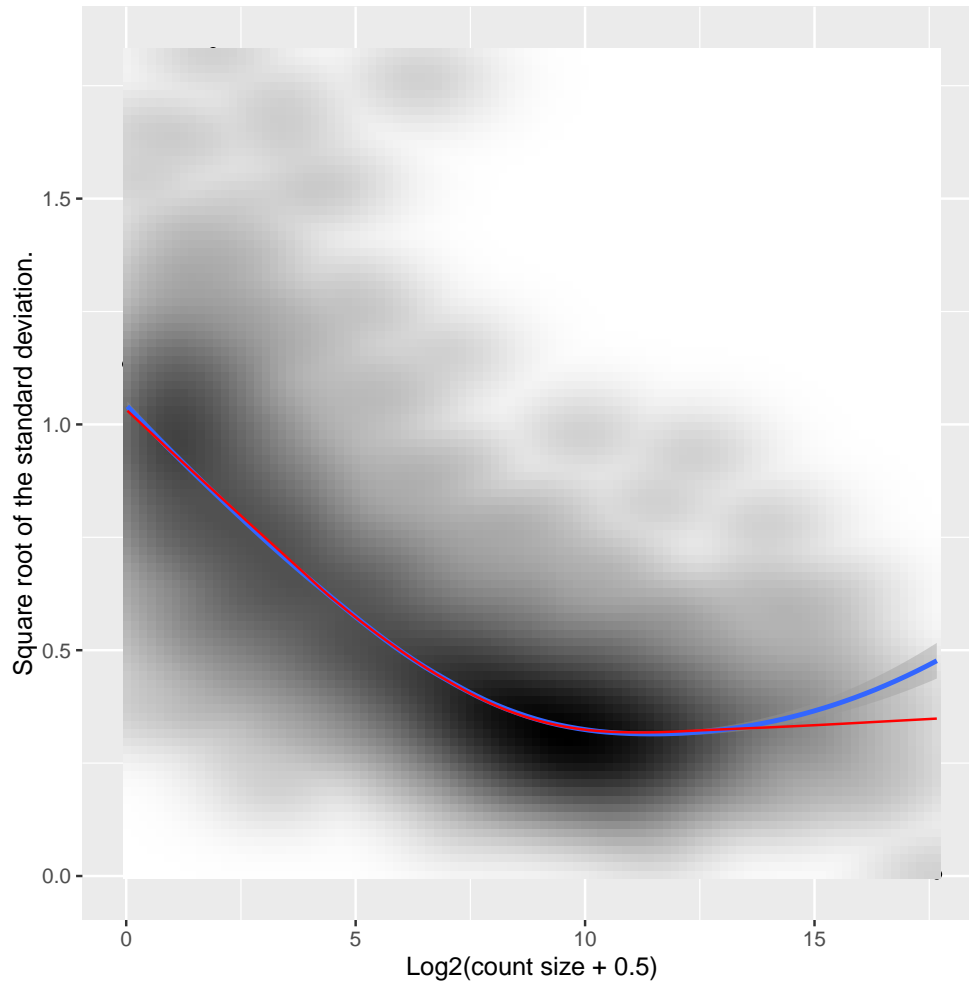
```
> plotPC(res$v,res$d,
+        col=design$condition, # color by batch
+        pch=19, main="PCA plot",
+        xlim=c(min(res$v[,1])-.08,max(res$v[,1])+.08),
+        ylim=c(min(res$v[,2])-.08,max(res$v[,2])+.08))
> text(res$v[,1], res$v[,2], design$libType, pos=1, cex=0.6)
```

PCA plot



We are now ready to use `limma` and `voom`. We also modified the `voom` function so it takes data on log-scale as input.

```
> v = voomMod(tmp, model.matrix(~design$condition), lib.size=libsize)
> v$plot
```



```
> summary(v)
```

	Length	Class	Mode
E	71071	-none-	numeric
weights	71071	-none-	numeric
design	14	-none-	numeric
lib.size	7	-none-	numeric
plot	9	gg	list

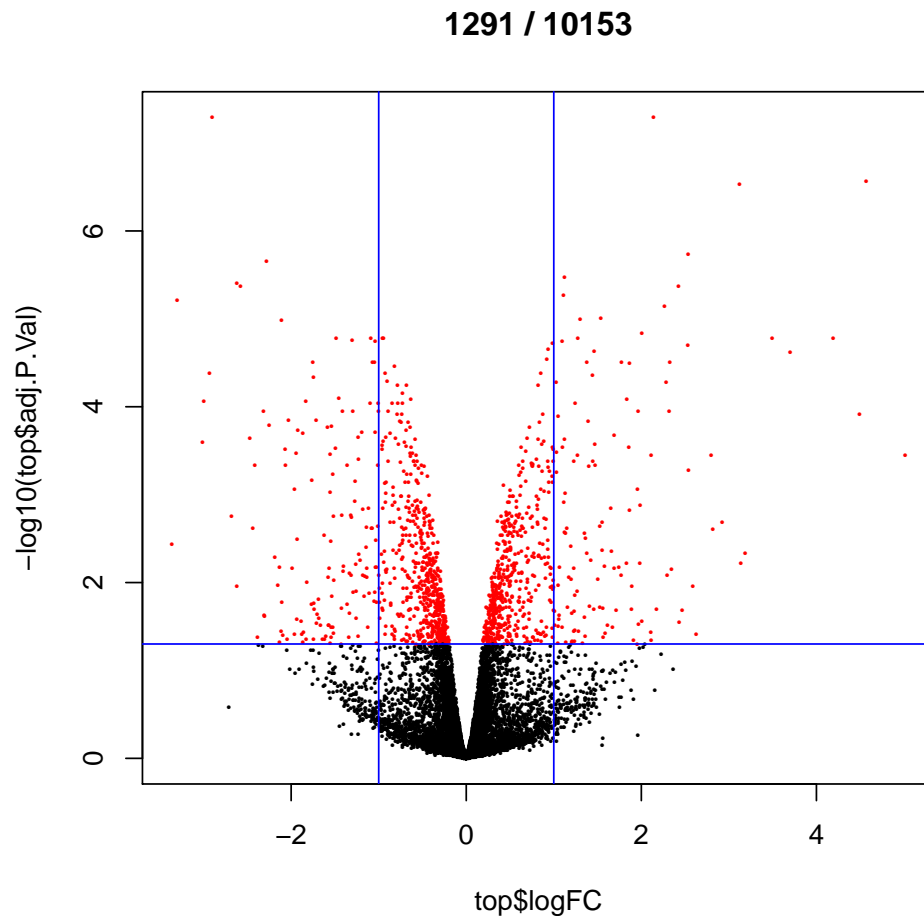
```
> fit = lmFit(v)
> eb = eBayes(fit)
> top = topTable(eb, coef=2, n=nrow(v$E))
```

Plot results as a volcano plot


```

> sel = top$adj.P.Val < 0.05
> plot(top$logFC, -log10(top$adj.P.Val), pch=16, cex=0.3,
+      main=paste(sum(sel), "/", length(sel)), col=ifelse(sel, "red", "black"))
> abline(v=c(-1,1), h=-log10(0.05), col="blue")

```



Let us now compare the results to what we get when we adjust for batch in the model

```

> cond=design$condition
> batch=design$libType
> mod = model.matrix(~cond+batch ,
+                   contrasts.arg=list(cond="contr.treatment", batch="contr.sum"))
> v1 = voom(counts, mod)
> fit1 = lmFit(v1)
> eb1 = eBayes(fit1)
> top1 = topTable(eb1, coef=2, n=nrow(v1$E))

```

Compare results:

```
> top$ID = rownames(top)
> top1$ID = rownames(top1)
> tab = merge(top[,c("ID", "adj.P.Val")], top1[,c("ID", "adj.P.Val")], by="ID")
> as.data.frame(table(combat = tab[,2] < 0.05, model = tab[,3] < 0.05))
```

```
  combat model Freq
1 FALSE FALSE 8676
2  TRUE FALSE  241
3 FALSE  TRUE  186
4  TRUE  TRUE 1050
```

After correction with modified ComBat, there are a few more differentially abundant genes.

References

- [1] W Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics (Oxford, England)*, 8(1):118–127, January 2007.
- [2] Jeffrey T Leek, Robert B Scharpf, Héctor Corrada Bravo, David Simcha, Benjamin Langmead, W Evan Johnson, Donald Geman, Keith Baggerly, and Rafael A Irizarry. Tackling the widespread and critical impact of batch effects in high-throughput data. *Nature reviews Genetics*, 11(10):733–739, October 2010.

SessionInfo

- R version 3.3.0 (2016-05-03), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.utf8, LC_NUMERIC=C, LC_TIME=en_US.utf8, LC_COLLATE=en_US.utf8, LC_MONETARY=en_US.utf8, LC_MESSAGES=en_US.utf8, LC_PAPER=en_US.utf8, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.utf8, LC_IDENTIFICATION=C
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils

- Other packages: cbcSEQ 0.9.1, corpcor 1.6.8, genefilter 1.52.1, limma 3.26.9, mgcV 1.8-12, nlme 3.1-128, pasilla 0.10.0, preprocessCore 1.32.0, sva 3.18.0
- Loaded via a namespace (and not attached): annotate 1.48.0, AnnotationDbi 1.32.3, Biobase 2.30.0, BiocGenerics 0.16.1, bootstrap 2015.2, colorspace 1.2-6, DBI 0.4-1, devtools 1.11.1, digest 0.6.9, ecodist 1.2.9, ggplot2 2.1.0, grid 3.3.0, gtable 0.2.0, IRanges 2.4.8, KernSmooth 2.23-15, labeling 0.3, lattice 0.20-33, lava 1.4.3, MASS 7.3-45, Matrix 1.2-6, memoise 1.0.0, munsell 0.4.3, parallel 3.3.0, plyr 1.8.3, prodlim 1.5.7, Rcpp 0.12.4, rmeta 2.16, RSQLite 1.0.0, S4Vectors 0.8.11, scales 0.4.0, splines 3.3.0, stats4 3.3.0, SuppDists 1.1-9.2, survcomp 1.20.0, survival 2.39-3, survivalROC 1.0.3, survJamda 1.1.4, survJamda.data 1.0.2, tools 3.3.0, withr 1.0.1, XML 3.98-1.4, xtable 1.8-2