

# ABSSeq: a new RNA-Seq analysis method based on modelling absolute expression differences

Wentao Yang

December 20, 2017

## 1 Introduction

This vignette is intended to give a brief introduction of the **ABSSeq** R package by analyzing the simulated data from Sonesson et al. [4]. For details about the approach, consult Yang [3]. Currently, **ABSSeq** integrates two modules: basic model for pairwise comparison and linear model for complex design.

RNA-Seq quantifies gene expression with reads count, which usually consists of conditions (or treatments) and several replicates for each condition. The expected expression of each gene is estimated from number of read count, proportional to the expectation value of the true concentration of count. As a result, a normalization method need to be apply on the original counts. The normalized counts usually have enormous variation across genes and compared conditions. The reliable identification of differential expression (DE) genes from such data requires a probabilistic model to account for ambiguity caused by sample size, biological and technical variations, levels of expression and outliers.

**ABSSeq** infers differential expression directly by the counts difference between conditions. It assumes that the sum counts difference between conditions follow a Negative binomial distribution with mean  $\mu$  (proportional to expression level) and dispersion factor  $r$  (size). The  $\mu$  and  $r$  is determined by variation in the experiment, i.e., biological variation, sequencing and mapping biases. Typically, the number of replicates in a study is small and not enough to reveal all variation. To overcome this problem, a common solution is to borrow information across genes. Here, we use local regression to smooth dispersion across genes. The smoothed dispersions are then used to produce pseudocounts in the  $\mu$  estimation to accounts for dynamic dispersions across expression levels, which in turn moderates the fold-change.

**ABSSeq** tests counts difference directly against a baseline estimated from the data set ( $\mu$ ), and therefore reports p-values related to magnitude of difference (fold-change). In addition, **ABSSeq** moderates the fold-changes by two steps: the expression level and gene-specific dispersion, that might facilitate the gene ranking by fold-change and visualization (Heatmap). New alternative approach (named **aFold**) was introduced, which calls DE genes via log fold-change (see second Section for example). **aFold** uses a polynomial function to model the uncertainty (or variance) of read count, and thus takes into consideration the variance of expression levels across treatments and genes. **aFold** produces accurate estimation of fold changes. In combination with the linear model from

limma [2], aFold is capable to analyze data set with complex experimental design (see last Section for example).

## 2 Pairwise study

We firstly import the ABSSeq package.

```
> library(ABSSeq)
```

Then, we load a simulated data set. It is a list object and contains three elements: the counts matrix, denoted by 'counts', the groups, denoted by 'groups' and differential expression genes, denoted by 'DEs'.

```
> data(simuN5)
> names(simuN5)
```

```
[1] "counts" "groups" "DEs"
```

simuN5 is simulated from Negative binomial distribution with means and variances from Pickrell's data [5] and added outliers randomly [4]. simuN5 includes group information.

```
> simuN5$groups
```

```
[1] 0 0 0 0 0 1 1 1 1 1
```

An ABSDataSet object is required for ABSSeq and could be constructed with the ABSDataSet function by providing counts matrix and defined groups. Here, we can also initiate a paired comparison for specific samples, such as data for cancer and normal tissue from same individuals, by setting the paired parameter in ABSDataSet object.

```
> obj <- ABSDataSet(simuN5$counts, factor(simuN5$groups))
> pairedobj <- ABSDataSet(simuN5$counts, factor(simuN5$groups), paired=TRUE)
```

The ABSDataSet function also includes the parameter for the normalization method, which has a default as qtotal. qtotal assesses the influence of DE on data structure to normalize the data. Additional choices of normalization methods are also provided, that are, total by total reads count, geometric from DESeq [6], quantile by reads count in the first three quartiles from baySeq [?], TMM from edgeR [1] and user through size factors provided by users. The normalization method can be showed and revised by normMethod.

```
> obj1 <- ABSDataSet(simuN5$counts, factor(simuN5$groups),
+                   normMethod="user", sizeFactor=runif(10,1,2))
> normMethod(obj1)
```

```
[1] "user"
```

```
> normMethod(obj1) <- "geometric"
> normMethod(obj1)
```

```
[1] "geometric"
```

The size factors could be estimated from an `ABSDataset` object via the function `normalFactors` and retrieved by the function `sFactors`.

```
> obj <- normalFactors(obj)
> sFactors(obj)
```

```
[1] 1.2701723 1.1188592 0.6950348 1.1697439 1.1474554 0.9574766 0.9201882
[8] 1.1359601 0.8054707 0.7796388
```

Reads count after normalization could be retrieved by the function `counts`.

```
> head(counts(obj,norm=TRUE))
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
1	58.25981	14.300280	48.91842	0.854888	1.742987	51.17618
2	1451.77156	967.950211	1256.05230	2067.119167	2926.475379	6327.04741
3	2662.63092	2245.143980	2474.69640	1502.038204	1938.201680	3469.53640
4	25.19343	9.831443	12.94899	17.952648	18.301365	17.75500
5	1491.13630	3674.278225	3630.03432	2285.970493	5126.125128	12830.60043
6	847.12917	832.097550	546.73525	130.797863	1342.100084	1438.15521

	[,7]	[,8]	[,9]	[,10]
1	6.520405	0.000000	7.449061	35.91407
2	57776.223615	4703.510143	12370.407060	12392.91781
3	3163.483249	3104.862488	30017.232227	43420.10493
4	190.178485	4.401563	24.830203	61.56697
5	8308.082941	18284.092397	15852.843010	17635.08870
6	949.805689	1277.333561	1443.876296	1602.02384

With the size factors, we can calculate the absolute counts difference between conditions, mean ( $\mu$ ), size factor ( $r$ ) and moderate log2 of fold-change for each gene. It can be done by function `callParameter` as

```
> obj=callParameter(obj)
```

If we want to see correlation between the absolute log2 fold-change (with or without moderation) and expression level in same conditions, we can use function `plotDiffToBase`.

```
> obj <- callDEs(obj)
> plotDiffToBase(obj)
```

In the end, we model the counts differences with Negative binomial distribution and calculate the pvalue for each gene. DE calling is performed by the function `callDEs`, which reports pvalues as well as adjusted pvalues, that can be accessed by `results` with names of `pvalue` and `adj.pvalue`. Noticely, this function also provides fold-change moderation according to gene-specific dispersion by utilizing `aFold`, which will report fold-changes closer to gene's dispersion. In the end, `ABSSeq` produces three kinds fold-changes: the original (denoted by 'rawFC'), corrected by expression level (denoted by 'lowFC') and moderated by expression level and gene-specific dispersion (denoted by 'foldChange'), which are stored in the `ABSDataset` object and could be also retrieved by `results`.

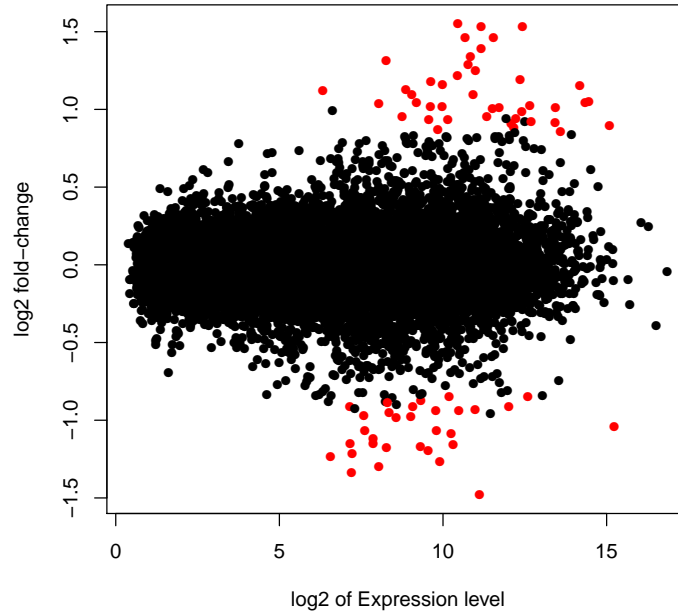


Figure 1: 'Absolute log2 fold-change against expression level'-plot for count data. We show the fitted and raw data with different colors.

```
> obj <- callDEs(obj)
> head(results(obj, c("rawFC", "lowFC", "foldChange", "pvalue", "adj.pvalue")))
```

	rawFC	lowFC	foldChange	pvalue	adj.pvalue
1	-0.1824058	-0.1030012	-0.04125768	7.682173e-01	1.00000000
2	2.9174817	2.6816216	0.36224008	4.230884e-02	1.00000000
3	2.0016381	0.8968301	0.50537531	2.661240e-02	0.86631330
4	0.8868678	0.6140338	0.21658502	1.460386e-01	1.00000000
5	2.2377865	2.1213283	1.02150694	4.989514e-05	0.01516204
6	1.1772042	0.9992554	0.43023062	6.939515e-02	1.00000000

The `results` function can be used to access all information in an `ABSDataset`.

```
> head(results(obj))
```

	Amean	Bmean	baseMean	absD	Variance	rawFC	lowFC
1	3.562610	3.380204	62.16317	23	1.220177e+03	-0.1824058	-0.1030012
2	10.650049	13.567531	23642.68938	84901	4.895532e+08	2.9174817	2.6816216
3	11.052189	13.053827	5514.05983	10509	2.181432e+06	2.0016381	0.8968301
4	4.093067	4.979935	115.68473	215	5.799681e+03	0.8868678	0.6140338
5	11.539225	13.777011	18473.15191	56703	1.873455e+07	2.2377865	2.1213283
6	9.192270	10.369474	1796.40395	3012	2.590491e+05	1.1772042	0.9992554

	foldChange	pvalue	adj.pvalue	trimmed
1	-0.04125768	7.682173e-01	1.00000000	0
2	0.36224008	4.230884e-02	1.00000000	0
3	0.50537531	2.661240e-02	0.86631330	2
4	0.21658502	1.460386e-01	1.00000000	0
5	1.02150694	4.989514e-05	0.01516204	0
6	0.43023062	6.939515e-02	1.00000000	0

In addition to a step-by-step analysis in above, DE calling could be simply performed by the function `ABSSeq`, which runs a default analysis by calling above functions in order and returns a `ABSDataset` object with all information.

```
> data(simuN5)
> obj <- ABSDataset(simuN5$counts, factor(simuN5$groups))
> obj <- ABSSeq(obj)
> res <- results(obj, c("Amean", "Bmean", "foldChange", "pvalue", "adj.pvalue"))
> head(res)
```

	Amean	Bmean	foldChange	pvalue	adj.pvalue
1	3.562610	3.380204	-0.04125768	7.682173e-01	1.00000000
2	10.650049	13.567531	0.36224008	4.230884e-02	1.00000000
3	11.052189	13.053827	0.50537531	2.661240e-02	0.86631330
4	4.093067	4.979935	0.21658502	1.460386e-01	1.00000000
5	11.539225	13.777011	1.02150694	4.989514e-05	0.01516204
6	9.192270	10.369474	0.43023062	6.939515e-02	1.00000000

Moreover, `ABSSeq` also allows testing on user-defined baseline for counts difference by giving a same value to `minRates` and `maxRates` as

```
> data(simuN5)
> obj <- ABSDataset(simuN5$counts, factor(simuN5$groups), minRates=0.2, maxRates=0.2)
> #or by slot functions
> #minRates(obj) <- 0.2
> #maxRates(obj) <- 0.2
> obj <- ABSSeq(obj)
> res <- results(obj, c("Amean", "Bmean", "foldChange", "pvalue", "adj.pvalue"))
> head(res)
```

	Amean	Bmean	foldChange	pvalue	adj.pvalue
1	3.562610	3.380204	-0.04125768	7.038358e-01	1.0000000000
2	10.650049	13.567531	0.36224008	2.192792e-02	0.4006852443
3	11.052189	13.053827	0.50537531	4.337366e-03	0.1594078122
4	4.093067	4.979935	0.21658502	8.852225e-02	0.6238114781
5	11.539225	13.777011	1.02150694	7.752227e-07	0.0002099674
6	9.192270	10.369474	0.43023062	1.754797e-02	0.3604790716

`ABSSeq` penalizes the dispersion estimation by adding a common dispersion value to the observed dispersion for each gene, which is obtained by quantile estimation on observed dispersions. This penalized dispersion could be provided by user as

```

> data(simuN5)
> obj <- ABSDataSet(simuN5$counts, factor(simuN5$groups), minDispersion=0.1)
> #or by slot functions
> #minimalDispersion(obj) <- 0.2
> obj <- ABSSeq(obj)
> res <- results(obj, c("Amean", "Bmean", "foldChange", "pvalue", "adj.pvalue"))
> head(res)

```

	Amean	Bmean	foldChange	pvalue	adj.pvalue
1	3.562610	3.380204	-0.04125768	0.75314495	1.0000000
2	10.650049	13.567531	0.36224008	0.04397104	1.0000000
3	11.052189	13.053827	0.50537531	0.04324281	1.0000000
4	4.093067	4.979935	0.21658502	0.14923425	1.0000000
5	11.539225	13.777011	1.02150694	0.00043097	0.1732082
6	9.192270	10.369474	0.43023062	0.09085401	1.0000000

In addition, ABSSeq provides special parameter estimation for data set without replicates. It firstly treats the two groups as replicates and separates genes into two sets according to fold-change cutoff (depends on expression level). The set with fold-change under cutoff is used to estimate the dispersion for each gene by local regression as well as fold-change moderation. Here is the example, which replaces the `callParameter` by `callParameterwithoutReplicates`.

```

> data(simuN5)
> obj <- ABSDataSet(simuN5$counts[, c(1,2)], factor(c(1,2)))
> obj <- ABSSeq(obj)
> res <- results(obj, c("Amean", "Bmean", "foldChange", "pvalue", "adj.pvalue"))
> head(res)

```

	Amean	Bmean	foldChange	pvalue	adj.pvalue
1	6.141438	4.176475	-1.30085645	0.003116490	0.01707491
2	10.760856	10.176462	-0.56268049	0.153013673	0.35063405
3	11.635516	11.389556	-0.23985467	0.845037624	1.00000000
4	4.958562	3.671725	-0.73879630	0.089056224	0.23482571
5	10.799432	12.100000	1.25952601	0.001112932	0.00740706
6	9.984289	9.958485	-0.02476103	0.999999092	1.00000000

### 3 Detecting DE via aFold

Recently, ABSSeq integrates a new method for DE detection: aFold. aFold utilizes a polynomial function to model the uncertainty of observed reads count and moderate the fold change calculation. aFold takes into account variations among samples and genes and reports DE and fold changes in a reliable way. The fold changes produced by aFold may help the experimentalist to avoid arbitrary choice of cut-off thresholds and may enhance subsequent downstream functional analyses. Here is the example for how to use aFold in ABSSeq.

```

> data(simuN5)
> obj <- ABSDataSet(counts=simuN5$counts, groups=factor(simuN5$groups))
> obj <- ABSSeq(obj, useaFold=TRUE)

```

```
> res <- results(obj,c("Amean","Bmean","foldChange","pvalue","adj.pvalue"))
> head(res)
```

	Amean	Bmean	foldChange	pvalue	adj.pvalue
1	3.562610	3.380204	-0.04125768	7.757687e-01	9.410283e-01
2	10.650049	13.567531	0.36224008	1.238933e-02	1.665142e-01
3	11.052189	13.053827	0.50537531	4.847530e-04	1.531717e-02
4	4.093067	4.979935	0.21658502	1.348430e-01	5.603766e-01
5	11.539225	13.777011	1.02150694	1.759256e-12	5.345993e-10
6	9.192270	10.369474	0.43023062	2.975533e-03	6.158168e-02

## 4 PCA analysis via aFold

aFold model also stabilizes variances across expression levels, which could be used for principal component analysis (PCA). Here is an example. Noticeably, the group information is not necessary for the `ABSDataset` object under PCA analysis.

```
> data(simuN5)
> obj <- ABSDataset(counts=simuN5$counts)
> ##as one group
> cond <- as.factor(rep("hex",ncol(simuN5$counts)))
> ##normalization
> cda <- counts(obj,T)
> ##variance stabilization
> sds <- genAFold(cda,cond)
> ##sds is list vector, which contains variance stabilized read counts in 3rd element
> ##or expression level adjusted counts in 4th element. 3rd element is more sensitive
> ##to difference between samples than the 4th one. Here we use the 4th element for a
> ##PCA analysis.
> ## log transformation
> ldat <- log2(sds[[4]])
> ## PCA analysis
> PCA <- prcomp(t(ldat), scale = F)
> ## Percentage of components
> percentVar <- round(100*PCA$sdev^2/sum(PCA$sdev^2),1)
> ## plotting
> pc1=PCA$x[,1]
> pc2=PCA$x[,2]
> #plot(pc1,pc2,main="",pch=16,col="black",xlab="PC1",ylab="PC2",cex=1.2)
```

## 5 DE analysis with complex design

In combination with linear model from `limma` [2], aFold is capable to analyze data set with complex experimental design, which is performed by the function `ABSSeqlm`. Here is an example.

```
> data(simuN5)
> groups<-factor(simuN5$groups)
```

```
> obj <- ABSDataSet(counts=simuN5$counts)
> design <- model.matrix(~0+groups)
> res <- ABSSeqlm(obj,design,condA=c("groups0"),condB=c("groups1"))
> head(res)
```

	basemean	logFC	pvalue	p.adj
1	4.690153	-0.03870026	8.051044e-01	9.610751e-01
2	14.191909	0.36116377	2.129451e-02	4.435846e-01
3	14.022025	0.35511690	2.356381e-02	4.705514e-01
4	5.924728	0.20662942	1.876926e-01	8.335799e-01
5	13.832014	1.00860220	1.270311e-10	4.796003e-08
6	10.391500	0.41516516	8.120320e-03	2.443746e-01

Noticely, the parameters `condA` and `condB` could contain multiple conditions (factors) to run a comparison between multiple conditions. The function `ABSSeqlm` could be also used for analysis of variance (ANOVA) across conditions. To run ANOVA, all conditions (factors) are imported to the parameter `condA` in the function `ABSSeqlm` (without `condB`).

```
> res <- ABSSeqlm(obj,design,condA=c("groups0","groups1"))
> head(res)
```

	basemean	logFC	pvalue	p.adj
1	4.690153	0.02736522	8.631600e-01	0.99995335
2	14.191909	0.25538135	1.077354e-01	0.99995335
3	14.022025	0.25110556	1.137579e-01	0.99995335
4	5.924728	0.14610906	3.574518e-01	0.99995335
5	13.832014	0.71318946	7.060844e-06	0.00266579
6	10.391500	0.29356610	6.446461e-02	0.99995335

The linear model is performed by `lmFit` from `limma` [2], which could be suppressed via the parameter `lmodel` as

```
> res <- ABSSeqlm(obj,design,condA=c("groups0"),condB=c("groups1"),lmodel=FALSE)
> head(res)
```

	basemean	logFC	pvalue	p.adj
1	4.690153	-0.03870026	8.051044e-01	9.610751e-01
2	14.191909	0.36116377	2.129451e-02	4.435846e-01
3	14.022025	0.35511690	2.356381e-02	4.705514e-01
4	5.924728	0.20662942	1.876926e-01	8.335799e-01
5	13.832014	1.00860220	1.270311e-10	4.796003e-08
6	10.391500	0.41516516	8.120320e-03	2.443746e-01

## References

- [1] Robinson, Mark D., Davis J. McCarthy, and Gordon K. Smyth. *edgeR: a Bioconductor package for differential expression analysis of digital gene expression data*. Bioinformatics 26.1 (2010): 139-140.



- [2] Gordon K. Smyth.. *Linear models and empirical Bayes methods for assessing differential expression in microarray experiments*. Statistical Applications in Genetics and Molecular Biology 3.1 (2004): 1-25.
- [3] Wentao Yang, Philip Rosenstielb and Hinrich Schulenburg. *ABSSeq: a new RNA-Seq analysis method based on modelling absolute expression differences*. BMC Genomics 2016 17:541.
- [4] Soneson C, Delorenzi M *A comparison of methods for differential expression analysis of RNA-seq data*. BMC Bioinformatics 2013, 14(1):91.
- [5] Pickrell JK, Marioni JC, Pai AA, Degner JF, Engelhardt BE, Nkadori E, Veyrieras J-B, Stephens M, Gilad Y, Pritchard JK *Understanding mechanisms underlying human gene expression variation with RNA sequencing* Nature 2010, 464(7289):768-772.
- [6] Anders S, Huber W *Differential expression analysis for sequence count data*. Genome Biol 2010, 11(10):R106.