ALDEx2: ANOVA-Like Differential Expression tool for compositional data

Andrew Fernandes, Jean Macklaim, Greg Gloor

April 9, 2014

Contents

1 Why another package?

High-throughput sequencing approaches generate proportional data, not count data. Fundamentally, many high throughput sequencing approaches generate similar data: reads are mapped to features in each sample, these features are normalized, then statistical difference between features is calculated. The standard statistical tools used to analyze RNA-seq, ChIP-seq, 16S rRNA gene sequencing, metagenomics, etc. are fundamentally different for each approach — despite the underlying similarity in the data structures. ALDEx2 provides a simple consistent framework for data analysis that encompasses all these experimental designs.

2 Introduction

This guide provides an overview of the R package ALDEx version 2 (ALDEx2) for differential abundance analysis of proportional data. The package was developed specifically for multiple-organism RNA-Seq data generated by high-throughput sequencing platforms (meta-RNA-Seq), but testing showed that it performed very well with traditional RNA-Seq datasets, 16S rRNA variable region sequencing and selective growth-type (SELEX) experiments. In principle, the analysis method should be applicable to nearly any type of data that is generated by high-throughput sequencing that generates tables of per-feature counts for each sample: in addition to the examples outlined above this would include ChIP-Seq or metagenome sequencing. See the following examples for application on these types of problems.

The ALDEx2 package is suitable for comparison of samples between two conditions where there are at least three biological replicates per condition. If there are only two replicates in one condition ALDEx2 can give the same information as ALDEx version 1, if the user sets the mc.samples flag to at least 1000 (2000 is recommended), and uses the effect size and overlap values to infer significant differential abundance as for the original version of ALDEx (effect ≥ 2, overlap ≤ 0.01). The p values that are returned under this scenario are invalid.

ALDEx2 estimates per-feature technical variation within each sample using Monte-Carlo instances drawn from the Dirichlet distribution. This distribution maintains the proportional nature of the data. ALDEx2 uses the centred log-ratio (clr) transformation that ensures the data are scale invariant and sub-compositionally coherent[2]. The scale invariance property removes the need for a between sample data normalization step since the data are all placed on a consistent numerical co-ordinate. The sub-compositional coherence property ensures that the answers obtained are consistent when parts of the dataset are removed (e.g., removal of rRNA reads or rare OTU species). All feature abundance values are expressed relative to the geometric mean abundance of all features in a sample. This is conceptually similar to a quantitative PCR where abundances are expressed relative to a standard: in the case of the clr transformation, the standard is the per-sample geometric mean abundance. See Aitchison (1986)[2] for a complete description, or the lecture notes from CoData5[3].

3 Installation

Download the most current non-test version of ALDEx2. Versions that are noted as ALDEx2t are test versions and may not be reliable or documented. Do not unzip or untar the file. Inside an R environment type the following (replacing the /path/to with the path to the file on your computer):

```
install.packages("/path/to/ALDEx2_2.0.6.2.tar.gz", repos=NULL, type="source")
```

At the present, ALDEx2 requires only the base R packages, and has been tested and found to run on R from version 2.12 onward.

4 Quick start with example data and 2 groups:

See section 7 for a full description of what is happening.

```
# load the library
library(ALDEx2)
```

```
# load the included selex dataset
data(selex)

# set comparison groups
conds <- c(rep("NS", 7), rep("S", 7))

# run aldex
x <- aldex(selex, conds, paired.test=TRUE)

# summarize the results in a table
y <- summary.aldex(x)

# plot the results using all three tests
par(mfrow=c(1,3))
plot.aldex(x, test="welches")
plot.aldex(x, test="wilcoxon")
plot.aldex(x, test="effect", cutoff=1.5)
```

ALDEx2 returns expected values for summary statistics. It is also important to note that both versions of ALDEx use Bayesian sampling from a Dirichlet distribution to estimate the underlying technical variation. This is controlled by the mc.samples, in practice we find that setting this to 128 is sufficient for most cases as ALDEx2 is estimating the expected value of the distributions. The user is cautioned that the number of features called as differential will vary somewhat between runs because of the sampling procedure. Only features with values close to the chosen significance cutoff will vary between runs.

5 General information

The underpinnings of ALDEx and ALDEx2 are explained two papers[4, 5]. The purpose of these instructions is an explanation of how to use ALDEx2. There are only three functions in ALDEx2, **aldex**, **summary.aldex**, and **plot.aldex**. How to use these functions is given by typing ?aldex, ?plot.aldex, or ?summary.aldex at the R command prompt. Examples of all three functions, along with some basic R pitfalls in setting up the required input tables are included below. The authors expect that there is a current working installation of the R statistical programming language that is properly installed, and that the user has a basic understanding of it. ALDEx2 and has been tested on R 2.12 and above, including R version 3.

The authors appreciate bug reports or suggestions for improvement to the package, although suggestions should be focussed on improving the performance of the package. We are working on adding in multiple group testing.

5.1 Differences with the version in the 2013 Fernandes et al PLoS ONE paper

- With the release of R version 3, memory management appears to be much better. In addition, we have corrected two bugs in the code. The major bug was identified by Jean Macklaim, and restricted the experimental design in some cases to 2x2 data comparisons. We have successfully run ALDEx version 1.0.4 on an 8x8 meta-RNA-seq comparison composed of 3200 functional groups with mc.samples set to 2048. This consumes slightly over 12 Gb of

Table 1: Time to complete a dataset in seconds between exhaustive (2.0.5) and random (2.0.6) versions. The sample sizes are rows x columns by samples.

version	dataset	DMC	time
2.0.5	selex	128	107.246
2.0.5	selex	128	108.601
2.0.6	selex	128	92.718
2.0.6	selex	128	94.241
2.0.5	bottomly	128	1198.984
2.0.6	bottomly	128	919.952
2.0.5	16S-tag	16	DNF
2.0.6	16S-tag,vag (133x133x7239)	16	542.607
2.0.6	16S-tag,oral (316x312x23393)	16	4571.960

RAM on OS X 10.8. ALDEx2 requires substantially less memory and can accordingly be run over even larger datasets. ALDEx v1.0.4 is available at:

`https://github.com/ggloor/ALDEx2/blob/master/ALDEx_1.0.4.tar.gz`

No further changes other than bug fixes are expected for that version.

- ALDEx2 uses Welch's t-test and Wilcoxon rank test to determine p values and false discovery rates. This is possible because ALDEx2 requires 3 or more samples per condition for these tests, where ALDEx version 1 requires only 2 samples per condition. We strongly encourage experimental designs that use sufficient replicates to gain meaningful insights. ALDEx version 1 is not being developed further. ALDEx2 will give results with only 2 replicates, however, only the effect and overlap statistics are meaningful under these conditions: *p values and Benjamini-Hochberg values are not reliable with only 2 conditions.*

- Note that ALDEx2 removes all features that have 0 reads mapped in all samples.

5.2 New in version 2.0.6 and greater:

The major memory constraint has been largely removed. It was caused by an all vs. all approach for the comparison of vectors used to generate the within and between group difference values, and for the effect size calculation. The exhaustive approach is taken up to the point where the vectors exceed a length of 10000. After that limit, the vector is randomly sampled to 10000 instance. Testing has found that the values are identical to the second decimal place with the original approach for large datasets. The differences in speed are given in Table 1.

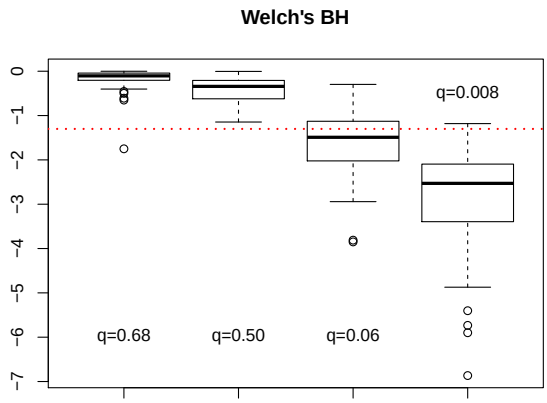

Figure 1: The effect of modelling the data as proportions using Monte-Carlo instances drawn from a Dirichlet distribution and clr transformation on corrected p values. The typical experiment has many replicates and few samples, thus the technical variation in the data is modelled poorly. Shown here are the distribution of Benjamini-Hochberg corrected p values (q value) for 128 Monte-Carlo instances. Only the last

5.3 For versions 2.0.5 and lower:

It is important to note that ALDEx2 can also be somewhat memory intensive. It can require a large amount of memory if there are a large number of features, a large number of replicates, or if there are a large number of Dirichlet samples generated. It is best *not* to try and run ALDEx2 on a machine with limited memory.

6 Modelling the data as proportions rather than counts

For high-throughput sequencing experiments, including RNA-seq, individual sequence reads are assigned to genes or genomic features and the typical output is a table of counts per feature. Reads assigned to these features have several sources of variation: technical variation, biological variation within a condition, biological variation between conditions and unexplained variation. The consensus view in the literature is that the underlying variation is best explained by the modelling the pooled variation of features with a given condition using negative binomial distribution.

We take a different approach with ALDEx and ALDEx2. First, we do not assume the technical variation of the features in a given sample share any underlying distribution. Second, we model the reads as proportions of the data available rather than as counts. The proportional nature of the data is a result of the large but finite number of reads available from a next-generation sequencing run. See Fernandes et al.[4] for a discussion of the advantages of modelling the data in this way.

In brief, we we use Baysian techniques to infer the underlying technical variation in the read count proportions in a way that preserves the proportional nature of the data. This is done by sampling from a Dirichlet distribution, and results in a transformation of the observed read counts into a distribution of read counts. These distributions are centre log-ratio transformed following the advice of Aitchison[2] for dealing with proportional data. This has three effects. First, the abundance values for each sample are centred on their geometric mean abundance levels. Second, the data now expose the same relationships regardless of how the input data is altered by subsetting. Third, the values become largely independent and can be dealt with as statistically independent features when there are large numbers of features.

These clr transformed values are used for standard between-group statistical tests followed by Benjamini-Hochberg corrections for false discovery rate. The expected value of the p and q scores is returned to the user. This ensures that features that appear significant only because of random sampling error are weeded out, and that features that have significance that is robust to random sampling error are retained. An example for four features that have various significance scores is shown in Figure 1.

7 Case study a growth selection type experiment:

This section contains an analysis of a dataset collected where a single gene library was made that contained 1600 sequence variants at 4 codons in the sequence. These variants were cloned into an

expression vector at equimolar amounts. The wild-type version of the gene conferred resistance to a topoisomerase toxin. Seven independent growths of the gene library were conducted under selective and non-selective conditions and the resulting abundances of each variant was read out by sequencing a pooled, barcoded library on an Illumina MiSeq (McMurrough et al., submitted). The data table is included as selex_table.txt in the package. In this data table,there are 1600 features and 14 samples. The analysis takes approximately 2 minutes and memory usage tops out at less than 1Gb of RAM on a mobile i7 class processor. The commands used are presented below:

comments that help understand the code start with a hash symbol

while the actual commands typed into the R console are identified as a teletype-like font as below. Note that a single command input line may wrap onto more than one line if it is very long

```
library(ALDEx2)
#load the selex dataset
data(selex)
conds <- c("NS","NS","NS","NS","NS","NS","NS","S","S","S","S","S","S","S")

#run aldex
x <- aldex(selex, conds, paired.test=TRUE)

#summarize the results in a table
y <- summary.aldex(x) #now defaults to median TRUE

#plot the result to choose cutoff values
pdf("selex_mw.pdf", height=4, width=8)
par(mfrow=c(1,2), mar=c(4,4,2,2))
plot.aldex(x, test="welches")
plot.aldex(x, test="wilcoxon")
dev.off()
```

In this instance, shown in Figure 2, it appears that the Welch's or Wilcoxon tests are similar. This is likely because these data are both very reproducible across all features due to a high number of read counts per feature and low biological variability (these are replicate growths of the same library). Note that a paired t-test was performed.

7.1 Getting and examining the output:

The built-in plot function can be used to rapidly display traditional Bland-Altman or MA style plots of the output as well as the MW plots given in [4]. Additional options can be found in the function.

```
plot.aldex(x, type="MA")
```

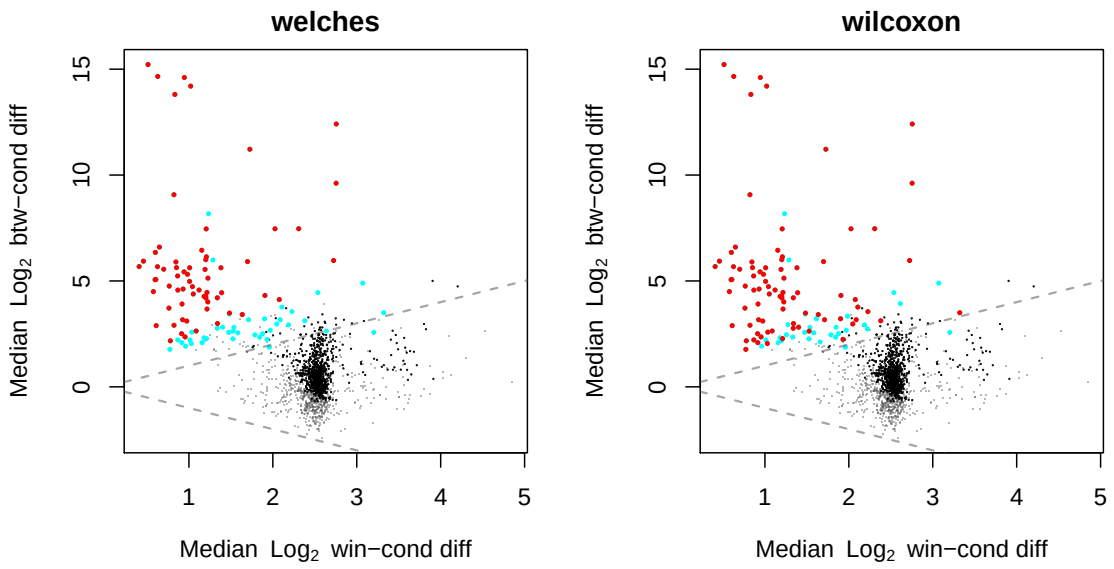

Figure 2: Differential abundance in the selex dataset using either the Welch's t-test or the Wilcoxon rank test. In this case, Welch's t-test with a fdr of 0.05 identified 64, and Wilcoxon with fdr of 0.05 identified 82 differentially abundant variants. There were 63 overlapping calls by both methods. Red dots indicate the significant features, cyan dots indicate features with $p < 0.05$, black dots represent rare features, and grey dots indicate abundant features.

```
plot.aldex(x, type="MW", test="welches", cutoff=0.05)
```

The built-in summary function generates a table of all quantile summary values derived from the analysis. The table is suitable for export externally.

```
y <- summary.aldex(xb, minimal.only=FALSE)
```

for a more compact summary that does not include median relative abundances for all samples:

```
y.min <- summary.aldex(xb, minimal.only=TRUE)
```

finally, you can write the table for import into excel, etc

```
write.table(y, file="selex.txt", sep="\t", quote=FALSE, col.names=NA)
```

This is the dataset that you want to explore. ALDEx2 uses the clr transformation described above that scales all expression values by log2(feature proportion) - log2(mean-feature proportion) (see the paper for details). In this scaling a value of 0 for a feature represents the mean abundance per sample, and features with values less than 0 are less abundant than the mean and vice-versa. The fields give all the summary information output by the program at each step, the field names are":

- `(blank) - this holds the gene identifier information.`

- rab.all.q50 - median relative abundance of the mixture distribution of all samples
- rab.win.N.q50 - median value of the mixture distribution samples in condition N
- rab.win.B.q50 - median value of the mixture distribution samples in condition B
- rab.X1_BNS.q50 - median expression value of features in sample X1_BNS
- etc for each sample
- diff.btw.q50 - median relative abundance difference between and B and N
- diff.win.q50 - maximum median relative abundance difference within B and N
- effect.q50 - effect size, how distinct the between difference is as a function of the within difference.
- overlap - the fraction of the effect size distribution that overlaps 0. If the effect size is 2 and the overlap is less than 1% this is equivalent to the significance test used by ALDEx version 1. This measure is not expected to be reliable if mc.samples is less than 1024
- criteria.we.pval - Expected Welch's t-test p value
- criteria.we.BH - Benjaminin-Hochberg fdr from Welch's t-test
- criteria.wi.pval - Expected Wilcoxon rank test p value
- criteria.wi.BH - Benjaminin-Hochberg fdr from Wilcoxon test

8 Test using the Bottomly dataset[1]

Here we use a dataset collected to identify differentially expressed genes in two different strains of mouse brains. This was used by Soneson et al.. [6] to benchmark a number of different RNA-seq analysis tools. The count table can be obtained from a list of publicly available datasets generated by bowtie at: http://bowtie-bio.sourceforge.net/recount/.

We assume that the counts are stored in a plain text file, with the gene identifier in the first column. An example of the input data format expected for this example is given below. The example assumes a tab or white-space delimited file format. The first column is the feature ID, this must be unique for each feature. Each following column contains the counts per feature.

```
refseq N1 B2 N5 B6 B3 N2 ...
id1 0 100 1 26 50 0 ...
id2 560 400 320 100 350 423 ...
...
```

An example of how to use ALDEx2 is given below. We assume that there are two conditions, N and B and that there are three or more replicates per condition. In the code that follows:
Load the library

```
library(ALDEx2)}
```

8.1 Reading the data and making the input dataframe

#INPUT 1, the data frame of reads

```
db <- read.table("bottomly_count_table.txt", header=T, row.names=1)
```

#INPUT 2, a list of conditions to be compared

```
conds <- c("N","N","N","N","N","N","N","N","N","N",
"B","B","B","B","B","B","B","B","B","B","B")
```

#INPUT 3, the number of Dirichlet Monte-Carlo instances. This is optional, and the user can safely leave at the default number of 128

#This dataset contains 10 replicates of one strain (N) and 11 replicates of a second strain (B).

run aldex. NOTE: this can take a while depending on your machine and the memory available. The test dataset has 21 different samples and 13932 non-zero features per sample. The test set took approximately 20 minutes on a MacBook Pro with an i7 mobile class processor and 16Gb RAM. The amount of RAM topped out at less than 8Gb.

```
xb <- aldex(db, conds, mc.samples=128)
```

8.2 Examine the Bottomly data output

The built-in plot function can be used to rapidly display traditional Bland-Altman or MA style plots of the output as well as the MW plots given in [4]. Additional options can be found in the function

```
plot.aldex(xb, type="MA")
```

```
plot.aldex(xb, type="MW", test="welches", cutoff=0.05)
```

The built-in summary function generates a table of all quantile summary values derived from the analysis. The table is suitable for export externally.

```
yb <- summary.aldex(xb, minimal.only=FALSE)
```

for a more compact summary that does not include median relative abundances for all samples:

```
yb.min <- summary.aldex(xb, minimal.only=TRUE)
```

finally, you can write the table for import into excel, etc

```
write.table(yb, file="bottomly_qval.txt", sep="\t", quote=FALSE, col.names=NA)
```

This is the dataset that you want to explore. ALDEx2 uses the clr transformation described above that scales all expression values by log2(feature proportion) - log2(mean-feature proportion) (see the paper for details). In this scaling a value of 0 for a feature represents the mean abundance per sample, and features with values less than 0 are less abundant than the mean and vice-versa. The fields give all the summary information output by the program at each step, the field names are":

- `(blank) - this holds the gene identifier information.`
- `rab.all.q50 - median relative abundance of the mixture distribution of all samples`
- `rab.win.N.q50 - median value of the mixture distribution samples in condition N`
- `rab.win.B.q50 - median value of the mixture distribution samples in condition B`
- `rab.B6033480 - median expression value of individual sample B6033480`
- `etc for each sample`
- `diff.btw.q50 - median relative abundance difference between and B and N`
- `diff.win.q50 - maximum median relative abundance difference within B and N`
- `effect.q50 - effect size, how distinct the between difference is as a function of the within difference.`
- `overlap - the fraction of the effect size distribution that overlaps 0. If the effect size is 2 and the overlap is less than 1% this is equivalent to the significance test used by ALDEx version 1`
- `criteria.we.pval - Expected Welch's t-test p value`
- `criteria.we.BH - Benjaminin-Hochberg fdr from Welch's t-test`
- `criteria.wi.pval - Expected Wilcoxon rank test p value`
- `criteria.wi.BH - Benjaminin-Hochberg fdr from Wilcoxon test`

It is important to note that the values given in the table are all Expected values of Dirichlet Monte-Carlo instances. Thus, the summary statistics cannot be derived from the abundance statistics in the table. It is also important to remember that the rab values are the log2-based abundance values relative to the geometric mean abundance. Therefore, if feature A has a value of 6 and feature B has a value of 7 then feature B is twice as

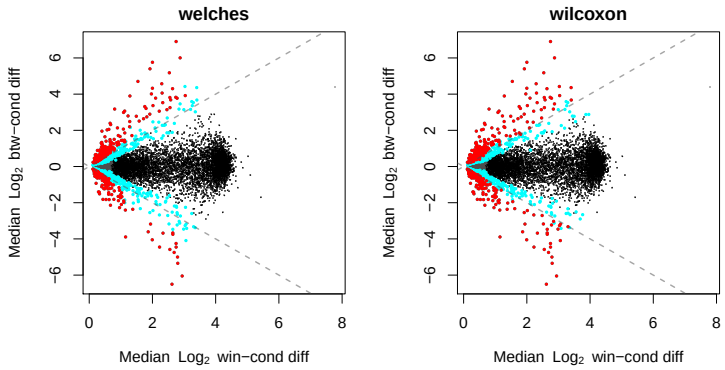

Figure 3: Differential expression in the Bottomly dataset using Welch's t-test and BH false discovery rate set at 0.05. The Welch's test with fdr identified 414 differentially expressed genes, Wilcoxon with fdr 465, and the overlap between the methods was 402.

abundant as feature A relative to the geometric mean abundance.
This makes a plot of the data, either as MA, MW, or hist plots (see Figure 3):

```
plot.aldex(x, type="MW",
  test="Welch's", cutoff=0.05)
```

Generate the plot:

```
pdf("bottomly.pdf",
  height=4, width=8)
par(mfrow=c(1,2),
  mar=c(4,4,2,2))
plot.aldex(xb, test="welches")
plot.aldex(xb, test="wilcoxon")
dev.off()
```

The MW plot is illustrated in Figure 3. Blue are features not called significantly different, cyan have a p value less than 0.05, red is a false discovery rate less than 0.05. The Benjamini-Hochberg method is used for fdr correction[7].

We recommend that the MW plots be inspected by the user to determine appropriate false discovery rate methods and cutoff values. The diagonal lines represent the region of the graph where the difference between the conditions is equal to the largest within-condition difference[4]. Empirical observation suggests the following recommendations. When the number of samples and the number of features are both large, we find that either the Welch's or Wilcoxon test coupled with the local false discovery rate give nearly identical results.

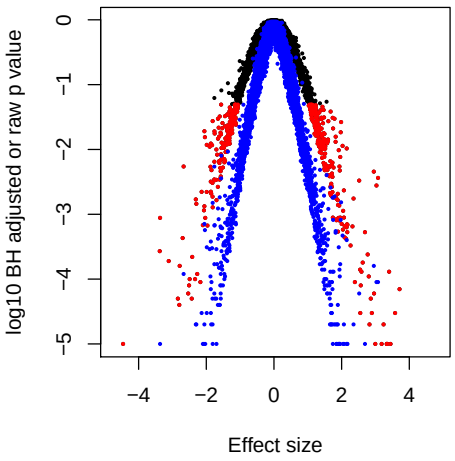

Figure 4: Plot showing correlation between effect size and p values. Black and red, show the plot for the Bottomly dataset, with black showing BH values > 0.05 and red showing BH values $<= 0.05$. The blue dots show the plot for the selex dataset, with no distinction.

9 The Relationship between Effect Size and P values

There is a very strong association between the effect size calculated as in Fernandes et al.[4] and the Expected p value and the associated corrected p value. The effect size is determined by dividing the random vector of between-group differences by both random vectors of within-group differences, taking the larger of the two results, and taking the Expected value of the final vector. Effect size is thus a measure of the mean ratio of the difference between conditions and the maximum difference within conditions. When the sample size is small (say less than 4 in each condition) the estimation of p values by either statistical

test is less robust, it may be more appropriate to use an
effect size cutoff of 1.5-2 and an overlap cutoff of 0.01 to
identify differential features of interest[4]. Conversely, when the number of samples is very large, the adjusted p value cutoffs will identify a large number of differential features because of the well-know relationship between sample size and power. However, it is likely that with extremely large sample sizes that statistical significance will over-estimate biological relevance when the effect size is small and the user is cautioned to use prudent effect size cutoffs[8].

```
pdf("effect.pdf", height=4, width=4)
par(mfrow=c(1,1), mar=c(4,4,2,2))
plot(yb$effect, log10(yb$criteria.we.BH),pch=19,
  cex=0.3, xlab="Effect size", ylab="log10 BH adjusted or raw p value")
points(yb$effect[yb$criteria.we.BH <0.05],
  log10(yb$criteria.we.BH[yb$criteria.we.BH <0.05]),pch=19, cex=0.3, col="red")
points(yb$effect, log10(yb$criteria.we.pval),
  pch=19, cex=0.3,col="blue", xlab="Effect size", ylab="log10 p value")
dev.off()
```

10 Grouping by Gene Function: SumFunctionsAitchison.R

This section describes the approach used by Macklaim et al.[9] to sum read counts across SEED, KEGG or COG functional groups. In brief: read counts are converted to proportions with an uninformative prior using the Aitchison mean[2]; these proportions are divided by the sequence length to normalize the read counts per unit length; the resulting values are divided by the sum of values for each sample to renormalize the sum to 1; each value is then multiplied by the original number of counts per sample; finally, values are summed by the group identifier.

This approach is coded in the R function 'SumFunctionsAitchison.R'. The function can be invoked on the command line as follows:

```
R CMD BATCH '--args in_filename out_filename 3' SumByAitchisonTransform.R log.txt
```

It requires three arguments, the input file name, the output file name, and the index of the first count column. The column structure of the input file should be as in the following example, where the first n columns are ID columns, the next column is the length, followed by count columns. The final column must contain the grouping information.

ID	length	count1	count2	...	countn	Group
RATOTCC	1505	24	19	...	341	K00001
K02100	1463	3	0	...	2	K00001

References

[1] Bottomly D, Walter NAR, Hunter JE, Darakjian P, Kawane S, et al. (2011) Evaluating gene expression in C57BL/6J and DBA/2J mouse striatum using RNA-seq and microarrays. PLoS One 6: e17820.

[2] Aitchison J (1986) The Statistical Analysis of Compositional Data. Chapman & Hall.

[3] Aitchison J (2003). A concise guide to compositional data analysis. URL http://ima.udg.edu/Activitats/CoDaWork05/A_concise_guide_to_compositional_data_analysis.pdf.

[4] Fernandes AD, Macklaim JM, Linn T, Reid G, Gloor GB (2013) ANOVA-like differential expression (ALDEx) analysis for mixed population RNA-seq. PLoS ONE 8: e67019.

[5] Fernandes AD, Reid J, Macklaim JM, McMurrough TA, Edgel DR, et al. (2014) Unifying the analysis of high-throughput sequencing datasets: characterizing rna-seq, 16s rrna gene sequencing and selective growth experiments by compositional data analysis. Microbiome May 25.

[6] Soneson C, Delorenzi M (2013) A comparison of methods for differential expression analysis of RNA-seq data. BMC Bioinformatics 14: 91.

[7] Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society Series B (Methodological) : 289–300.

[8] Nakagawa S, Cuthill IC (2007) Effect size, confidence interval and statistical significance: a practical guide for biologists. Biol Rev Camb Philos Soc 82: 591-605.

[9] Macklaim MJ, Fernandes DA, Di Bella MJ, Hammond JA, Reid G, et al. (2013) Comparative meta-RNA-seq of the vaginal microbiota and differential expression by lactobacillus iners in health and dysbiosis. Microbiome doi: 10.1186/2049-2618-1-12.