ALDEx2: ANOVA-Like Differential Expression tool for compositional data

Greg Gloor

June 27, 2014

Contents

1 Why another package?

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 the features composing each group or condition 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 by modelling the data as proportions rather than counts.

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 and used initially for multiple-organism RNA-Seq data generated by high-throughput sequencing platforms (meta-RNA-Seq)[1], but testing showed that it performed very well with traditional RNA-Seq datasets, 16S rRNA gene variable region sequencing and selective growth-type (SELEX) experiments[2],[3]. 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. We will be including examples for application on these types of problems as we move forward.

Versions of the ALDEx2 package greater than 2.0.7 (version 0.99.0 in Bioconductor) are modular and are suitable for the comparison of many different experimental designs. This is achieved by exposing the underlying values to make it possible for anyone to add the specific R code for their experimental design — a guide to these values is available. If there are only two replicates in one condition ALDEx2 can give the same information as ALDEx version 1.

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[4]. 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 from RNA-seq studies or rare OTU species from 16S rRNA gene amplicon studies). 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) for a complete description.

3 Installation

Download and install the most current of ALDEx2. At the present, ALDEx2 will run with only the base R packages and is capable of running several functions with the 'parallel' package if installed. It has been tested with version R version 3, but should run on version 2.12 onward. If the package `parallel` is present, ALDEx2 will make use of it. Otherwise, ALDEx2 will run in serial mode.

[1]Macklaim et al (2013) Microbiome doi: 10.1186/2049-2618-1-12
[2]Fernandes et al (2013) PLoS ONE http://dx.doi.org/10.1371%2/journal.pone.0067019)
[3]Fernandes et al (2014) Microbiome doi:10.1186/2049-2618-2-15
[4]Aitchison (1986) The statistical analysis of compositional data ISBN:978-930665-78-1

4 Quick example with 'selex' example data and 2 groups:

Case study a growth selection type experiment[5] 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. 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 for modular ALDEx are presented below:

First we load the library and the included selex dataset

```
> library(ALDEx2)
> data(selex)
```

Then we set the comparison groups. This must be a vector of conditions in the same order as the samples in the input counts table.

```
> conds <- c(rep("NS", 7), rep("S", 7))
```

ALDEx2 is now modular, offering the user the ability to build a data analysis pipeline for their experimental design. However, for two sample tests and one-way ANOVA design, the user can run the aldex wrapper. This wrapper will link the modular elements together to emulate ALDEx2 prior to the modular approach. Note that if the test is 'glm', then effect should be FALSE. If the test is 't', then effect should be set to TRUE. The 't' option evaluates the data as a two-factor experiment using both the Welch's t + Wilcoxon rank tests. The 'glm' option evaluates the data as a one-way ANOVA using the glm and Kruskal-Wallace test. All tests include a Benjamini-Hochberg correction of the raw P values. The data can be plotted onto MA or MW plots for for two-way tests using the 'aldex.plot' function. See the end of the modular section for examples of the plots.

```
> x <- aldex(selex, conds, mc.samples=16, test="t", effect=TRUE,
+       include.sample.summary=FALSE, verbose=FALSE)
> aldex.plot(x, type="MA", test="welch")
> aldex.plot(x, type="MW", test="welch")
```

The modular approach exposes the underlying intermediate data so that users can generate their own tests. The simple approach outlined above just calls `aldex.clr, aldex.ttest, aldex.effect` in turn and then merges the data into one object.

The workflow for the modular approach first generates instances of the centred log-ratio transformed values. There are three inputs: counts table, number of Monte-Carlo instances, level of verbosity (TRUE or FALSE). We recommend 128 or more mc.samples for the t-test, 1000 for a rigorous effect size calculation, and at least 16 for ANOVA.

This operation is fast

```
> x <- aldex.clr(selex, mc.samples=16, verbose=TRUE)
```

[5]McMurrough et al (2014) PNAS doi:10.1073/pnas.1322352111

```
> par(mfrow=c(1,2))
> aldex.plot(x.all, type="MA", test="welch")
> aldex.plot(x.all, type="MW", test="welch")
```

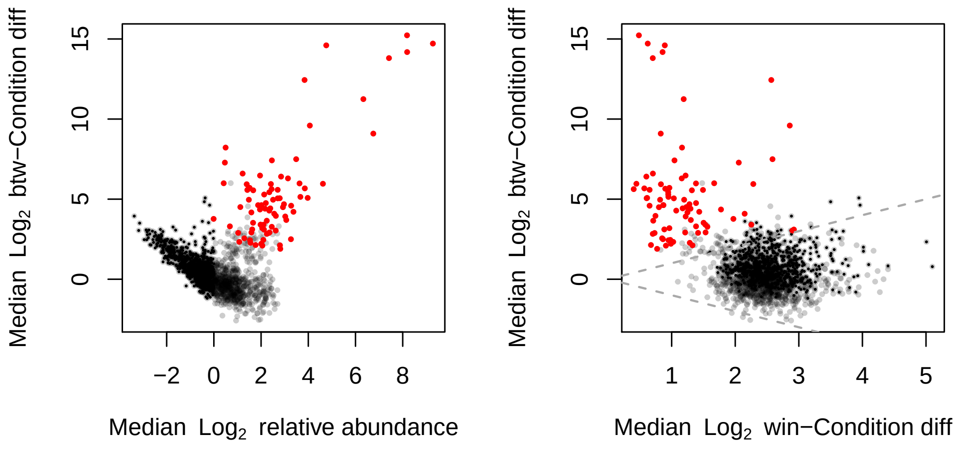

Figure 1: Output from `aldex.plot` function. The left panel is the MA plot, the right is the MW plot. In both plots red represents features called as differentially abundant with $q < 0.1$; grey are abundant, but not non-differentially abundant; black are rare, but not differentially abundant.

The next operation performs the Welch's t and Wilcoxon rank test for the instance when there are only two conditions. There are three inputs: the aldex object from aldex.clr, the vector of conditions, whether a paired test should be conducted or not (TRUE or FALSE).

This operation is reasonably fast.

```
> x.tt <- aldex.ttest(x, conds, paired.test=TRUE)
```

Alternatively, the user can perform the glm and Kruskal Wallace tests for one-way ANOVA of two or more conditions. Here there are only two inputs: the aldex object from aldex.clr, and the vector of conditions. Note that this is slow!

```
> x.glm <- aldex.glm(x, conds)
```

Finally, we estimate effect size and the within and between condition values in the case of two conditions. This step is required for plotting. There are four inputs: the aldex object from aldex.clr, the vector of conditions, and a flag as to whether to include values for all samples or not, and the level of verbosity.

```
> x.effect <- aldex.effect(x, conds, include.sample.summary=FALSE, verbose=TRUE)
```

Finally, all data are merged into one object.

```
> x.all <- data.frame(x.tt, x.glm, x.effect)
```

And the data are plotted

```
# examine the data
head(x.all)
               we.ep      we.eBH        wi.ep      wi.eBH       kw.ep
A:D:A:D 4.030100e-01 0.630807054 0.2393830128 0.437328198 0.215320607
A:D:A:E 1.154636e-01 0.347445969 0.0409018065 0.157258414 0.037453159
A:E:A:D 8.987974e-05 0.003290767 0.0005827506 0.008207592 0.001745119
            kw.eBH       glm.ep      glm.eBH  rab.all rab.win.NS rab.win.S
A:D:A:D 0.39327439 3.610615e-01 5.235822e-01 1.424946   1.308861  2.453840
A:D:A:E 0.14865903 8.122652e-02 1.922925e-01 1.712300   1.497671  4.233156
A:E:A:D 0.02457865 7.736602e-08 3.354920e-06 3.974840   1.411636 11.021544
        diff.btw diff.win    effect      overlap
A:D:A:D 1.122613 1.729108 0.4710433 0.2672607019
A:D:A:E 2.730902 2.381348 1.0348739 0.1358577816
A:E:A:D 9.642872 2.850081 3.4290684 0.0001566327
```

ALDEx2 returns expected values for summary statistics. It is important to note that ALDEx uses Bayesian sampling from a Dirichlet distribution to estimate the underlying technical variation. This is controlled by the number of `mc.samples`, in practice we find that setting this to 16 or 128 is sufficient for most cases as ALDEx2 is estimating the expected value of the distributions[3]. 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.

In the list below, the `aldex.ttest` function returns the values highlighted with $\ast$, the `aldex.glm` function returns the values highlighted with $\circ$, and the `aldex.effect` function returns the values highlighted with $\diamondsuit$.

- $\ast$ we.ep - Expected P value of Welch's t test
- $\ast$ we.eBH - Expected Benjamini-Hochberg corrected P value of Welch's t test
- $\ast$ wi.ep - Expected P value of Wilcoxon rank test
- $\ast$ wi.eBH - Expected Benjamini-Hochberg corrected P value of Wilcoxon test
- $\circ$ kw.ep - Expected P value of Kruskal-Wallace test
- $\circ$ kw.eBH - Expected Benjamini-Hochberg corrected P value of Kruskal-Wallace test
- $\circ$ glm.ep - Expected P value of glm test
- $\circ$ glm.eBH - Expected Benjamini-Hochberg corrected P value of glm test
- $\diamondsuit$ rab.all - median clr value for all samples in the feature
- $\diamondsuit$ rab.win.NS - median clr value for the NS group of samples
- $\diamondsuit$ rab.win.S - median clr value for the S group of samples
- $\diamondsuit$ rab.X1_BNS.q50 - median expression value of features in sample X1_BNS if [include.item.summary=TRUE]
- $\diamondsuit$ dif.btw - median difference in clr values between S and NS groups
- $\diamondsuit$ dif.win - median of the largest difference in clr values within S and NS groups
- $\diamondsuit$ effect - median effect size: diff.btw / max(dif.win) for all instances

[3]Fernandes et al (2014) Microbiome doi:10.1186/2049-2618-2-15

◊ overlap - proportion of effect size that overlaps 0 (i.e. no effect)

The built-in aldex.plot function described above will usually be sufficient, but for more user control the example below shows a plot that incorporates all the significance tests into one plot.

```
> # identify which values are significant in all tests
> found.by.all <- which(x.all$we.eBH < 0.05 &
+  x.all$wi.eBH < 0.05 & x.all$glm.eBH < 0.05 & x.all$kw.eBH < 0.05)
> # identify which values are significant in fewer than all tests
> found.by.one <- which(x.all$we.eBH < 0.05 |
+  x.all$wi.eBH < 0.05 | x.all$glm.eBH < 0.05 | x.all$kw.eBH < 0.05)
> # plot the within and between variation of the data
> plot(x.all$diff.win, x.all$diff.btw, pch=19, cex=0.3, col=rgb(0,0,0,0.3),
+  xlab="Difference within", ylab="Difference betweenn")
> points(x.all$diff.win[found.by.one], x.all$diff.btw[found.by.one], pch=19,
+  cex=0.5, col=rgb(0,0,1,0.5))
> points(x.all$diff.win[found.by.all], x.all$diff.btw[found.by.all], pch=19,
+  cex=0.5, col=rgb(1,0,0,1))
> abline(0,1,lty=2)
> abline(0,-1,lty=2)
```

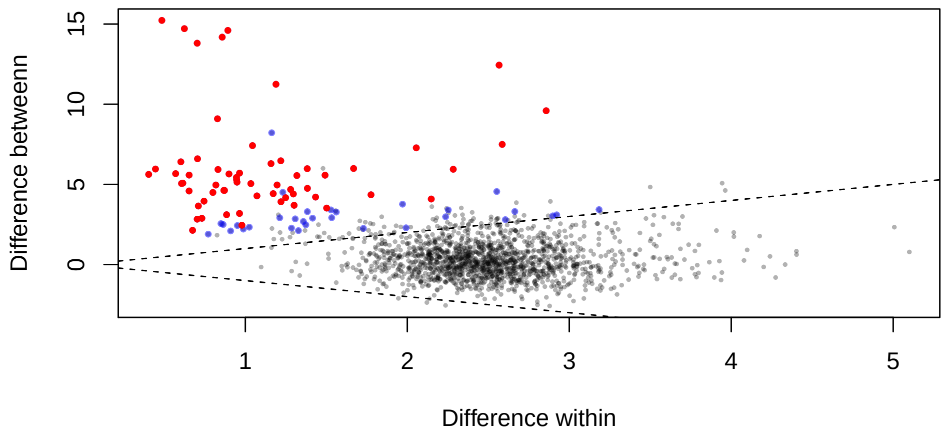

Figure 2: Differential abundance in the selex dataset using the Welch's t-test, Wilcoxon rank test, Kruskall-Wallace test or glm function. Features identified by all tests shown in red. Features identified by three or fewer tests are shown in blue dots. Non-significant features represent rare features if black and abundant features if grey dots.

5 Contributors

Andrew Fernandes wrote the original ALDEx code, and designed ALDEx2. Jean Macklaim found and squished a number of bugs, did testing and did much of the validation. Matt Links incorporated several ALDEx2 functions into a multicore environment. Adrianne Albert wrote the correlation and the one-way ANOVA modules. Andrew Fernandes, Jean Macklaim and Ruth Wong contributed to the Sum-FunctionsAitchison.R code. Greg Gloor is currently maintaining ALDEx2 and played roles in design and implementation.

6 Version information

Version 1.04 of ALDEx was the version used for the analysis in [1],[2]. This version was suitable only for two-sample two-group comparisons, and provided only effect size estimates of difference between groups. ALDEx v1.0.4 is available at:

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

. No further changes are expected for that version since it can be replicated completely within ALDEx2 by using only the `aldex.clr` and `aldex.effect` commands.

Versions 2.0 to 2.05 were development versions that enabled P value calculations. Version 2.06 of ALDEx2 was the version used for the analysis in [3]. This version enabled large sample comparisons by calculating effect size from a random sample of the data rather than from an exhaustive comparison.

Version 2.07 of ALDEx2 is the modular version that exposes the intermediate calculations so that investigators can write functions to analyze different experimental designs. As an example, this version contains an example one-way ANOVA module. This is the version submitted to Bioconductor as 0.99.

Future releases of ALDEx2 will use the Bioconductor versioning numbering.

7 Using ALDEx2 for a one-way ANOVA

ALDEx2 has a module for 1-way ANOVA using the Kruskal-Wallace and glm methods. This module was contributed by Arianne Albert (UBC), and shows that access to the underlying clr-transformed data can be used to build an arbitrary analysis module. All that is required is to identify the analysis to be conducted, and then to collect the values of the statistical test for each Monte-Carlo instance and finally, to take the mean of those test statistics.

As an example, we will use the OTU data table from Hsiao et. al 'Microbiota modulate behavioral and physiological abnormalities associated with neurodevelopmental disorders'[1]. In this table the authors are interested in identifying which OTUs are different between a saline control, and the offspring of polyIC treated mouse dams where the offspring were either treated with *Bacillus fragilis* or not treated with *B. fragilis*. The hypothesis was that *B. fragilus* treatment restored the microbiota composition to the control compositions.

Download dataset from:

`http://www.sciencedirect.com/science/MiamiMultiMediaURL/1-s2.0-`

[1]Macklaim et al (2013) Microbiome doi: 10.1186/2049-2618-1-12

[2]Fernandes et al (2013) PLoS ONE http://dx.doi.org/10.1371%2/journal.pone.0067019)

[3]Fernandes et al (2014) Microbiome doi:10.1186/2049-2618-2-15

```
S0092867413014736/1-s2.0-S0092867413014736-mmc2.zip/272196/FULL/
S0092867413014736/00157c3f7ba58d2d8ff5379d9c2f3c8b/mmc2.zip
```

Load and prepare the data:

```
> d <- read.csv("cell7251mmc2/otu_table_S_P_P+Bf.csv", header=T,
+  row.names=1, comment.char = "", skip=1)
> taxon <- d$taxonomy #save the taxonomy information
> d$taxonomy  <- NULL #empty the taxonomy column
```

First do a two sample test for OTUs that differ between pups of mothers treated with PolyIC+Bfrag or PolyIC. This is the same workflow as when using the selex dataset.

```
> library(ALDEx2)
> dBvsI <- data.frame(d[,1:20])
> condsBvsI <- c(rep("BF", 10), rep("C", 10))
> dBvI.clr <- aldex.clr(dBvsI, mc.samples=128)
> dBvI.tt <- aldex.ttest(dBvI.clr, condsBvsI)
> dBvI.effect <- aldex.effect(dBvI.clr, condsBvsI)
> dBvsI.all <- data.frame(dBvI.effect,dBvI.tt)
```

Determine the OTU numbers where the uncorrected Wilcoxon test P value is less than 0.05:

```
> rownames(dBvsI.all)[which(dSvsI.all$wi.ep < 0.05),]
```

This reveals that only OTUs 135 and 956 are in common with Figure 4 and Supplementary table 2, and that OTUs 53, 638, 837, and 836 are not identified by ALDEx2. It is of note that these latter 4 OTUs had very small read counts (on the order of 1-2 read counts per sample) contrary to the first two. PLEASE ALSO NOTE: these are P values, not corrected for false discovery rates. The authors did not correct for multiple tests in their analysis.

Now conduct a one-way ANOVA for OTUs that differ between pups of mothers treated with PolyIC+Bfrag or PolyIC including the control saline group. There were 10 animals in each replicate.

```
> conds <- c(rep("BF", 10), rep("C", 10), rep("Sal", 10))
```

Calculate MC Dirichlet instances and clr transform the values. Perform the ANOVA and get the OTU numbers for those OTUs where the raw P values from the glm calculation were less than 0.05. Note, that again, no OTUs were significant upon multiple test correction.

```
> d.clr <- aldex.clr(d, mc.samples=4)
> d.glm <- aldex.glm(d.clr, conds)
> d.glm[which(d.glm$kw.ep < 0.05),]
```

The results show that OTU 53 is now significant along with OTUs 135 and 956.

CONCLUSION: ALDEx2 recapitulates the observations that relative abundance changes in the most abundant organisms may correlate with B. fragilis treatment. However, these conclusions are not robust to multiple test correction under either a two-sample test or a one-way ANOVA test. Many other OTUs were too rare for robust conclusions, and were excluded by ALDEx2 this is not surprising, since there were only between 2160 to 2947 reads per sample (mean 2608) scattered amongst 1446 OTU sequences. In other words, the expected OTU read count per sample was about 2, and the sequencing depth was likely not sufficient for robust conclusions.

8 General information

The authors appreciate bug reports or suggestions for improvements to the package, and would encourage users to incorporate additional modular functions for more complex experimental designs.

8.1 New in version 0.99.1 and greater:

The program versions are renumbered to fit the Bioconductor numbering schema. Version 0.99.0 was identical to 2.0.7.2. The current version now compiles on cleanly on all platforms, and multi-processor awareness is now *off* by default. To enable multiprocessor awareness set `useMC=FALSE` to `TRUE` in the `aldex.clr`, `aldex.effect` and `aldex.glm` functions when they are run.

8.2 New in version 2.0.7 (0.99.0) and greater:

The program is now modular and provides access to the underlying instances of the clr values calculated from the Dirichlet Monte-Carlo replicates. Arianne Albert wrote the glm and Kruskal-Wallice module for one-way ANOVA and the correlation module. The program can now use parallel processing through the parallel package if present. Matt Links provided this code. It will run without the package, but somewhat slower.

8.3 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.

Note: All versions of ALDEx2 remove all features that have 0 reads mapped in all samples.

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

- ALDEx2 requires substantially less memory and can accordingly be run over even larger datasets than the original version of ALDEx. 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. The functionality of ALDEx v1.0.4 can be emulated in ALDEx2 by running only `aldex.clr` with mc.samples set to 1028 followed by running `aldex.effect`.

- ALDEx2 uses standard statistical tests to determine p values and false discovery rates. The modular version described here can be easily adapted to any study design. 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.*

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

9 Modelling the data

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 of reads 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 RNA-seq 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[2].

We take a different approach with ALDEx2. First, we do not assume the technical variation of the features in a given sample share any underlying pooled 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. Third, we use the principles of compositional data analysis[3] to normalize

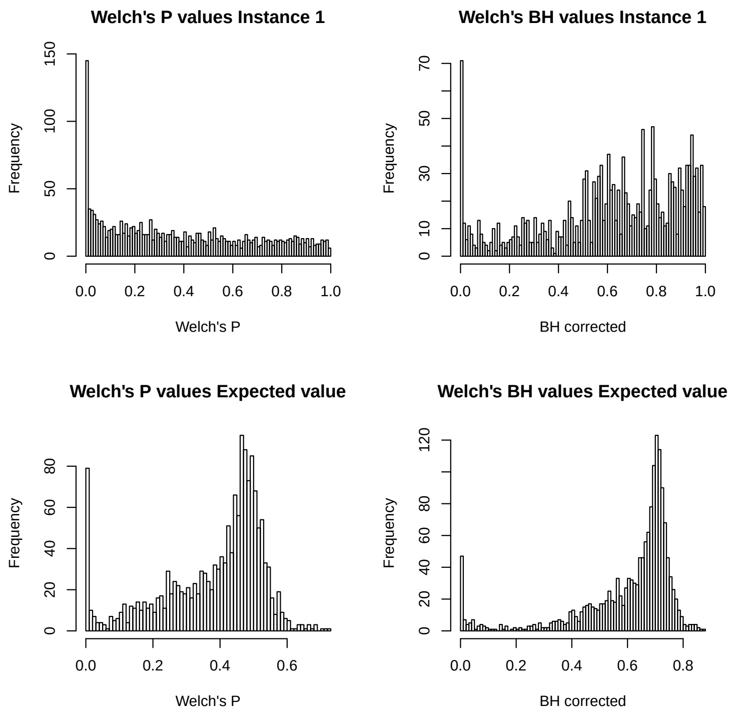

Figure 3: The difference between individual values and the reported expected values. P values and Benjamini Hochberg false positive rates (q values) are determined for each Monte-Carlo instance drawn from the Dirichlet distribution. These values are transformed into centred log-ratios and P and q values are calculated using standard tools. The result for each instance is an estimate of the P value per feature that is consistent with the observed data, and follows a random uniform distribution as expected. However, there are many possible P values that are consistent with the initial data, because each instance of the Monte-Carlo sampling from the Dirichlet distribution returns somewhat different values. Thus, the expected values regress towards the mean, resulting in the plot of the expected P values showing a mode of 0.5.

and scale the data as ratios. See Fernandes et al.[4] for a discussion of the advantages of modelling the data in this way.

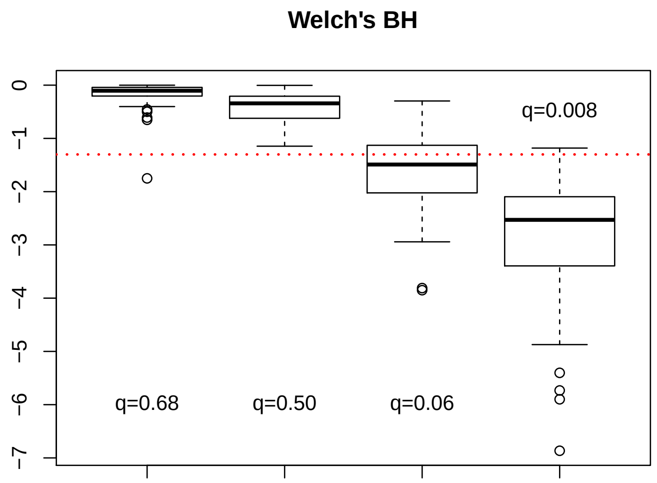

Figure 4: The effect of modelling the data as proportions using Monte-Carlo instances drawn from a Dirichlet distribution and clr transformation on corrected p values. The typical experiment has many replicates and few samples, thus the technical variation in the data is modelled poorly. Extreme p values that occur by chance thus regress to the mean when multiple instances are sampled. Shown here are the distribution of Benjamini-Hochberg corrected p values (q value) for 128 Monte-Carlo instances. Only the last one is reported as significant by ALDEx2 (expected q value indicated above the bars), yet some inferred technical replicates of each feature would give significant q values. Taking the expected value of the distribution rather than a single point estimate ensures that only features that exhibit significance consistently are identified as significantly different. The red line shows the location of q = 0.05.

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 posterior distribution of read proportions. These distributions are centre log-ratio transformed following the advice of Aitchison[3] 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 a Benjamini-Hochberg correction for false discovery rate. The expected value of the P and q score for each feature 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. For the most part, this is restricted only to those features where the read counts are low, and hence unreliable. An example showing this effect is shown in Figures 3 and 4.

10 Access to the underlying clr transformed Dirichlet Monte-Carlo instances

In the example from the quick start, x contains the initial instances of the clr values.

```
# generate instances of the centred log-ratio transformed values
x <- aldex.clr(selex, mc.samples=128, verbose=TRUE)


x is a list, where each list item is a matrix of features (rows) and
```

```
clr-transformed Dirichlet Monte-Carlo instances (columns)

The list contents are named by the sample IDs: names(x)

Features (genes, OTUs, etc) are in rows in the matrix, and the number
of features are the number of rows in any column, eg. number of features:
     length(x[[1]][,1])

Instances are in the columns of the matrix, and the number of instances
are the number of columns in a row, eg. number of Monte-Carlo Dirichlet instances:
     length(x[[1]][1,])

Identifiers for the features are the rownames of a sample: rownames(x[[1]])
```

11 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[5].

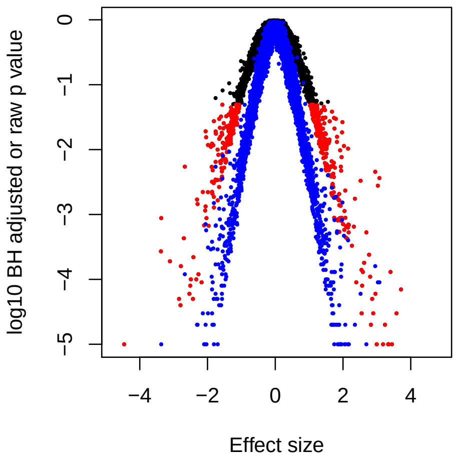

Figure 5: Plot showing correlation between effect size and p values. Black and red, show the plot for the Bottomly dataset, with black showing BH values > 0.05 and red showing BH values <= 0.05. The blue dots show the plot for the selex dataset, with no distinction.

```
pdf("effect.pdf", height=4, width=4)
plot(x.all$effect, log10(x.all$we.eBH),pch=19,
  cex=0.3, xlab="Effect size", ylab="log10 BH adjusted or raw p value")
points(x.all$effect[x.all$we.eBH <0.05],
  log10(x.all$we.BH[x.all$we.eBH <0.05]),pch=19, cex=0.3, col="red")
points(x.all$effect, log10(x.all$we.ep),
  pch=19, cex=0.3,col="blue", xlab="Effect size", ylab="log10 p value")
dev.off()
```

12 Grouping by Gene Function: SumFunctionsAitchison.R

This section describes the approach used by Macklaim et al.[6] 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[3]; 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 re-proportion the values so they 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] Hsiao EY, McBride SW, Hsien S, Sharon G, Hyde ER, et al. (2013) Microbiota modulate behavioral and physiological abnormalities associated with neurodevelopmental disorders. Cell 155: 1451-63.

[2] Anders S, McCarthy DJ, Chen Y, Okoniewski M, Smyth GK, et al. (2013) Count-based differential expression analysis of RNA sequencing data using R and Bioconductor. Nat Protoc 8: 1765-86.

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

[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] 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.

[6] 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 1: 15.