ALDEx2: ANOVA-Like Differential Expression tool for compositional data

Greg Gloor

October 13, 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

[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

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 the Bioconductor release 0.99.1 are modular and are suitable for the comparison of many different experimental designs. This is achieved by exposing the underlying centre log-ratio transformed Dirichlet Monte-Carlo replicate 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.

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)
```

[4]Aitchison (1986) The statistical analysis of compositional data ISBN:978-930665-78-1

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

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)
```

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

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

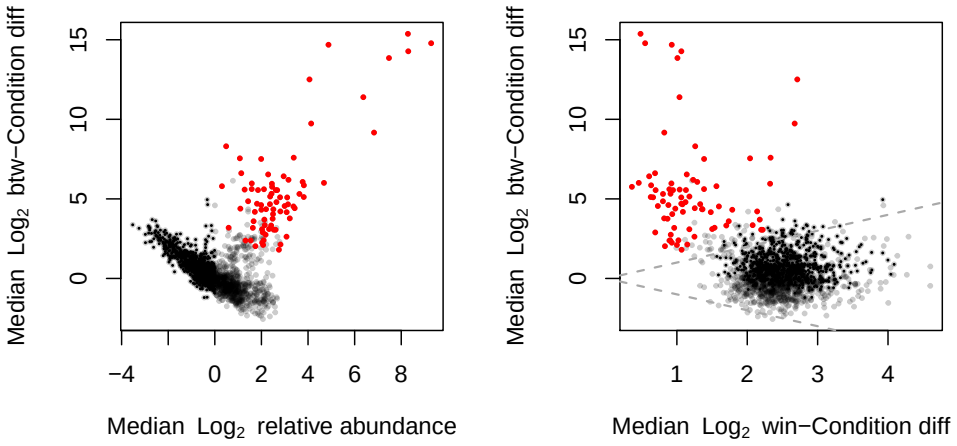

Figure 1: Output from `aldex.plot` function. The left panel is the MA plot, the right is the MW plot. In both plots red represents features called as differentially abundant with q < 0.1; grey are abundant, but not non-differentially abundant; black are rare, but not differentially abundant.

```
# 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.

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

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

- $*$ we.ep - Expected P value of Welch's t test
- $*$ we.eBH - Expected Benjamini-Hochberg corrected P value of Welch's t test
- $*$ wi.ep - Expected P value of Wilcoxon rank test
- $*$ 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
- $\diamondsuit$ 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)
```

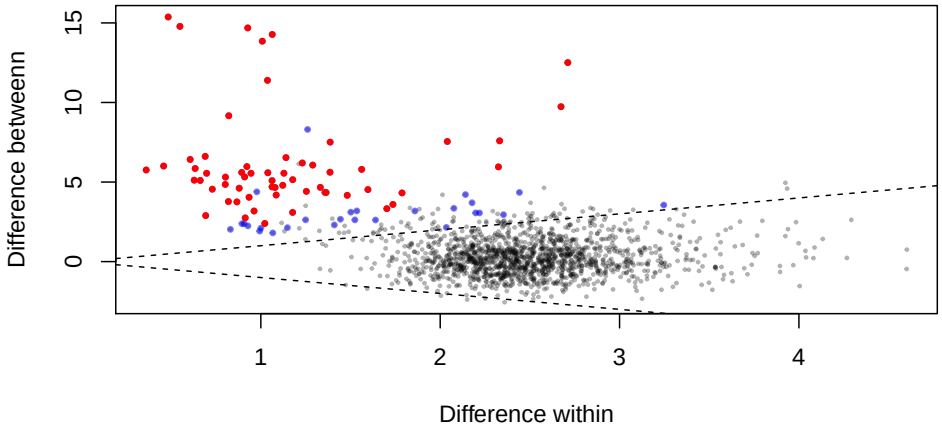

Figure 2: Differential abundance in the selex dataset using the Welch's t-test, Wilcoxon rank test, Kruskall-Wallace test or glm function. Features identified by all tests shown in red. Features identified by three or fewer tests are shown in blue dots. Non-significant features represent rare features 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. Ruth Grace Wong added function definitions and made the parallel code functional with BioConducter. Andrew Fernandes, Jean Macklaim and Ruth Grace 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 was the initial the modular version that exposed the intermediate calculations so that investigators could write functions to analyze different experimental designs. As an example, this version contains an example one-way ANOVA module. This is identical to the version submitted to Bioconductor as 0.99.1.

Future releases of ALDEx2 will use the Bioconductor versioning numbering.

[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