ALDEx2: ANOVA-Like Differential Expression tool for compositional data

Greg Gloor

October 30, 2017

Contents

1 Why the ALDEx2 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 a log-ratio transformed probability distribution 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 [2] and selective growth-type (SELEX) experiments,[3][4]. In principle,

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

[2]Bian et al (2017) mSphere doi:10.1128/mSphere.00327-17

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

[4]McMurrough et al (2015) PNAS June 10, 2014 vol. 111 no. 23 E2376-E2383 http://doi: 10.1073/pnas.1322352111
)

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 and citations 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 and returns a multivariate probability distribution. ALDEx2 uses the centred log-ratio (clr) transformation that ensures the data are scale invariant and sub-compositionally coherent[5]. 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.

In extreme cases we observe that the centre log-ratio can be asymmetric in the data. This occurs when the data are extremely asymmetric, such as when one group is largely composed of features that are absent in the other group. In this case the geometric mean will not accurately represent the appropriate basis of comparison for each group. ALDEx2 incorporates four methods to deal with this footnote[6]Wu et al (in prep). The first is to include as the denominator for the geometric mean those features that are relatively invariant across all samples. This is termed the 'iqlr' method, and takes as the denominator the geometric mean of those features with variance calculated from the clr that are between the first and third quartile. This approach can be used until the asymmetry becomes so severe that more than 25% of the features are asymmetric between the groups. The iqlr approach has the advantage that it gives essentially the same answer as using the entire set of features in symmetric datasets. The second approach, termed the 'zero' approach uses a different denominator for each group. The per-group denominator is the set of non-zero features in the group. This method introduces much stronger assumptions, but is helpful when the two groups under comparison are very different. The third approach, termed the 'lvha' approach identifies those features that are in the bottom quartile in variance across samples and are in the top quartile in relative abundance in every sample. This approach attempts to find 'housekeeping' features akin to those used as internal standards for qPCR. The final approach allows the user to specify a vector of row indices for the input data that are to be used as the denominator. In this case, the user is making an assumption regarding which features are invariant. In the case of RNA-seq, this might include a set of 'housekeeping' genes that are though to be invariant under the perturbation being tested. IMPORTANT: all rows must contain one or more counts when the user defines the row indices to ensure the appropriate rows are chosen.

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

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. ALDEx2 will make use of the BiocParallel package if possible, otherwise, ALDEx2 will run in serial mode.

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

Case study a growth selection type experiment[6] 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)
> #subset for efficiency
> selex <- selex[1201:1600,]
```

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 and the 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 effect (MW) plots[7] 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, denom="iqlr", verbose=FALSE)
> aldex.plot(x, type="MA", test="welch")
> aldex.plot(x, type="MW", test="welch")
```

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

[7] Gloor et al (2016) J. Comp. Graph. Stat. DOI:10.1080/10618600.2015.1131161

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, a vector of conditions, the number of Monte-Carlo instances, a string indicating if iqlr, zero or all feature are used as the denominator is required, and 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, conds, mc.samples=16, denom="iqlr", 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! and is not evaluated for the documentation

```
> 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 are used as the denominator, and the level of verbosity.

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

Finally, the t-test and effect data are merged into one object.

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

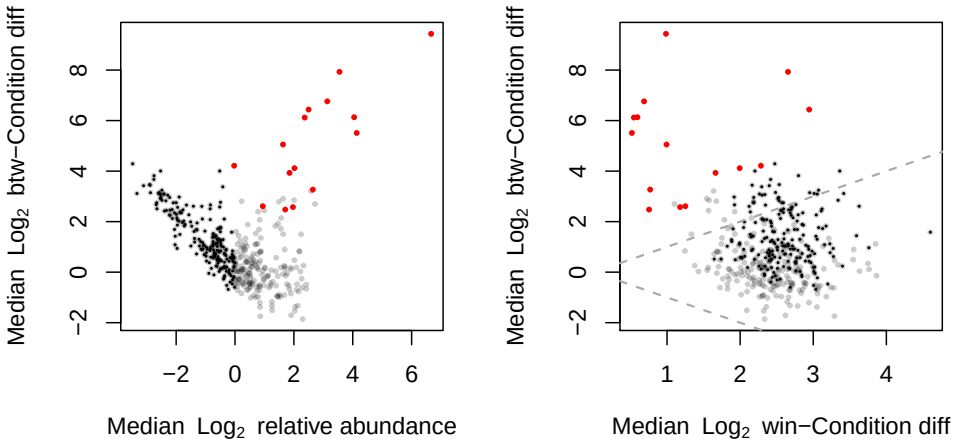

Figure 1: Output from `aldex.plot` function. The left panel is the MA plot, the right is the MW (effect) plot. In both plots red represents features called as differentially abundant with q < 0.1; grey are abundant, but not non-differentially abundant; black are rare, but not differentially abundant.

```
# 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 ∘, and the `aldex.effect` function returns the values highlighted with ♢.

∗ 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

∘ kw.ep - Expected P value of Kruskal-Wallace test

∘ kw.eBH - Expected Benjamini-Hochberg corrected P value of Kruskal-Wallace test

∘ glm.ep - Expected P value of glm test

∘ glm.eBH - Expected Benjamini-Hochberg corrected P value of glm test

♢ rab.all - median clr value for all samples in the feature

♢ rab.win.NS - median clr value for the NS group of samples

♢ rab.win.S - median clr value for the S group of samples

♢ rab.X1_BNS.q50 - median expression value of features in sample X1_BNS if [include.item.summary=TRUE]

♢ dif.btw - median difference in clr values between S and NS groups

♢ dif.win - median of the largest difference in clr values within S and NS groups

♢ effect - median effect size: diff.btw / max(dif.win) for all instances

♢ 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 shows which features are found by the Welchs' or Wilcoxon test individually (blue) or by both (red).

```
> # identify which values are significant in both the t-test and glm tests
> found.by.all <- which(x.all$we.eBH < 0.05 & x.all$wi.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)
> # 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 between")
> 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)
```

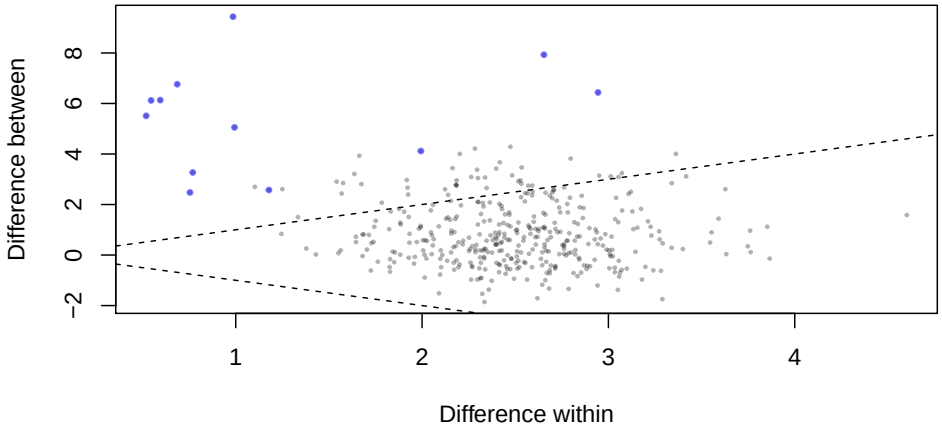

Figure 2: Differential abundance in the selex dataset using the Welch's t-test or Wilcoxon rank test. Features identified by both tests shown in red. Features identified by only one test are shown in blue dots. Non-significant features represent rare features if black and abundant features if grey dots.

4.1 Correcting for asymmetric datasets

ALDEx2 now includes methods to centre the data properly when the dataset contains an asymmetry between the groups. An asymmetry can arise for many reasons: in RNA-seq it could arise because samples in one group contain a plasmid and the samples in the other group do not; in metagenomics or 16S rRNA gene sequencing it can arise when the samples in the two groups are taken from different environments; in a selex experiment it can arise because the two groups are under different selective constraints. The asymmetry can manifest either as a difference in sparsity (i.e., one group contains more 0 value features than the other) or as a systematic difference in abundance. When this occurs the geometric mean of each group can be markedly different, and thus an inherent skew in the dataset can occur that leads to false positive and false negative feature calls.

ALDEx2 now incorporates four approaches to deal with asymmetric datasets:

1. *all*: The default is to calculate the geometric mean of all features. This is the approach used

in previous versions of ALDEx2, and is the usual method for the compositional data analysis approach.

2. *iqlr*: The first new approach is to identify those features that exhibit reproducible variance in the entire dataset. This is called the inter-quartile log-ratio or *iqlr* approach. For this, a uniform prior of 0.5 is applied to the dataset, the clr transformation is applied, and the variance of each feature is calculated. Those features that have variance values that fall between the first and third quartiles of variance for all features are retained. When aldex.clr is called, the geometric mean abundance of only the retained features is calculated and used as the denominator for log-ratio calculations. Modelling shows that this approach is effective in dealing with datasets with minor amounts of asymmetry, and begins to fail when more than 25% of the features are asymmetric. The approach has the advantage it has little or no effect on symmetric datasets and so is a safe approach if the user is unsure if the data is mildly asymmetric.

3. *zero*: This approach identifies those features that are non-zero in each group. In this approach the per-group non-zero features are used aldex.clr calculates the geometric mean in the clr transformation. This method is appropriate when the groups are very asymmetric, but the experimentalist must ask whether the comparison is valid in these extreme cases.

4. *lvha*: This method identifies those features that in the bottom quartile for variance in each group and the top quartile for relative abundance for each sample. This method is appropriate when the groups are very asymmetric, but there are some features that are expected to be relatively constant. Experience suggests that meta-genomic and meta-transcriptomic datasets benefit from this method of choosing the denominator. This method does not work with the selex dataset, since no features fit the criteria.

5. *user*: The last new approach is to let the user define the set of 'invariant' features. In the case of meta-rna-seq, it could be argued that the levels of housekeeping genes should be standard for all samples. In this case the user could define the row indices that correspond to the particular set of housekeeping genes to use as the standard. *It is important that no row contain all 0 values for any feature when this method is used.*

Figure **??** shows the effect of the iqlr correction on the example dataset. When the denominator is all, we see that the bulk of the points fall above the midpoint (dotted line), but that the bulk of the points are centered around 0 for the iqlr dataset. The mean difference between groups when the denominator is all is 0.6, and 0.05 when the denominator is iqlr; thus, we have a demonstrably better centring of the data in the latter. Practically speaking, we alter the p values and effect sizes of features near the margin of significance following iqlr transformation. The effect is largest for those features that are close to the bulk datapoints..

First the code:

```
> # this is the function to retrieve the denominator
> # running it as illustration only
> iqlr <- aldex.set.mode(selex, conds, denom="iqlr")
> # show the first 5 rows that will be used for the clr denominator
> iqlr[[1]][1:5]
> # use the iqlr features for the analysis
> y <- aldex.clr(selex, conds, denom="iqlr")
> y.e <- aldex.effect(y, conds)
```

```
> y.t <- aldex.ttest(y, conds)
> y.all <- data.frame(y.e, y.t)
> # find significant features in each
> sig.x <- x.all$wi.eBH < 0.05
> sig.y <- y.all$wi.eBH < 0.05
>
```

```
> # plot it
> par(mfrow=c(1,2))
> plot(x.all$diff.win,x.all$diff.btw, pch=19, col=rgb(0,0,0,.3), cex=0.3, main="denom=all")
> points(x.all$diff.win[sig.x],x.all$diff.btw[sig.x], pch=19, col="red", cex=0.4)
> points(x.all$diff.win[iqlr[[1]]],x.all$diff.btw[iqlr[[1]]], col=rgb(0,0,1,.7), cex=0.5)
> abline(0,0,lty=3, col="grey")
> abline(0,1,lty=2, col="grey")
> abline(0,-1,lty=2, col="grey")
> abline(0,2,lty=2)
> abline(0,-2,lty=2)
> plot(y.all$diff.win,y.all$diff.btw, pch=19, col=rgb(0,0,0,.3), cex=0.3, main="denom=iqlr")
> points(y.all$diff.win[sig.y],y.all$diff.btw[sig.y], pch=19, col="red", cex=0.4)
> points(y.all$diff.win[iqlr[[1]]],y.all$diff.btw[iqlr[[1]]], col=rgb(0,0,1,.7), cex=0.5)
> abline(0,0,lty=3, col="grey")
> abline(0,1,lty=2, col="grey")
> abline(0,-1,lty=2, col="grey")
> abline(0,2,lty=2)
> abline(0,-2,lty=2)
>
```

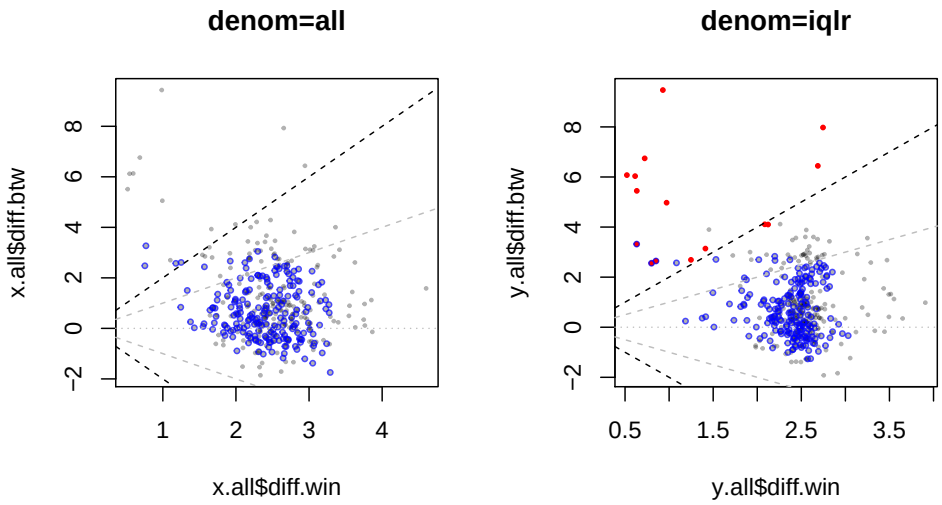

Figure 3: Differential abundance in the selex dataset using different denominators for the clr calculation. Features with a corrected p value less than 0.05 are shown in red. Features that are non-significant are in grey. Features that have variance between the first and third quartiles are outlined in blue. The geometric mean of these features is used as the denominator when calculating the clr transformation. Lines of constant effect are drawn at 0, and ± 1 and 2.

5 Contributors

Andrew Fernandes wrote the original ALDEx code, and designed ALDEx2. Jean Macklaim found and squished a number of bugs, performed testing, 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. Jia Rong Wu developed and implemented the alternate denominator method to correct for asymmetric datasets. Andrew Fernandes, Jean Macklaim and Ruth Grace Wong contributed to the Sum-FunctionsAitchison.R code. Tom Quinn rewrote the t-test and Wilcoxon functions to make them substantially faster. His propr R package is able to use the output from aldex.clr. Greg Gloor is currently maintaining ALDEx2 and played roles in design, testing 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