ALDEx: ANOVA-Like Differential Gene Expression Analysis of Single-Organism and Meta-RNA-Seq

Andrew Fernandes, Gregory Gloor, Jean Macklaim

July 18, 2012

1 Introduction

This guide provides an overview of the R package ALDEx for differential expression 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. In principle, the analysis method should be applicable to any type of count data that is generated by high-throughput sequencing such as ChIP-Seq or sequencing of metagenomes, although ALDEx has not been tested on these types of problems as of yet.

The ALDEx package is suitable for comparison of samples between two conditions where there are at least two biological replicates per condition. One unique feature of ALDEx is the explicit estimation of the within-sample biological variation using the Dirichlet distribution. This distribution maintains the proportional nature of the data, which is not the case for Poisson or negative-binomial based methods. The second unique feature of ALDEx is that it introduces a mathematical transform to normalize the data by removing the uninformative subspace introduced by logarithmic transformation.

2 How to get help

Questions about how ALDEx works are explained in the accompanying paper, under consideration at PLoS ONE. The purpose of these instructions is an explanation of how to use ALDEx. There are only three functions in ALDEx, **aldex**, **summary.aldex**, and **plot.aldex**. How to use these functions is given by typing ?aldex or help(aldex) at the R command prompt. All three functions, along with some basic R pitfalls in setting up the required input tables are included in the quick start and vignette sections of this document. 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.

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 do not expect to introduce a large number of new features in the near term.

2.1 A brief discussion of memory

It is important to note that ALDEx is somewhat memory intensive. It can require a large amount of memory if there are a large number of genes, a large number of replicates, or if there are a large number of Dirichlet samples generated. It is best *not* to try and run ALDEx on a laptop with limited memory. The critical limiting factor is the number of values in the distribution used to calculate ΔQ_A^0 statistic given in Fernandes et al. This number of values in the statistic scales with the number of replicate samples in each group. As one example, the meta-RNA-Seq dataset used in the Liver vs. Kidney vignette requires a minimum of 10GB of RAM, and takes approximately 1 hour to run on a server class machine. Some rules of thumb will be helpful: for a 2x2 design, mc.samples=2048 would be more than adequate, for a 3x3 design mc.samples=1024 and for a 4x4 design mc.samples=512 would be adequate. Larger designs can be scaled accordingly, with the minimum number of samples being approximately 512.

3 Quick start

A simple example of how to use ALDEx is given below. We assume that there are two conditions, N and T and that there are two replicates per condition. In the code that follows:

# 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

```
being in a sans-seriff font
```

# Install ALDEx from the downloaded package
```
install.packages(”ALDEx_1.0.2.tar.gz”)
```

# load ALDEx library for use by R
```
library(ALDEx)
```

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.

```
refseq N1 T2 N5 T6
id1 0 100 1 26
id2 560 400 320 100
...
```

# get the data in the right format as a dataframe with rownames
```
aldex.in <- read.table(”reads.txt”, header=T, stringsAsFactors=T, row.names=1)
```

# examine the input with head(aldex.in). Note that we will ignore the P9 column in the analysis

```
                  N1    T2    N5    T6    P9
id1            12826  8825 11322  8332 10774
id2            17262 11981 15622 11082 12883
...
```

# run the aldex function and place the output into a variable, x
# aldex takes 5 inputs:

# 1) the dataframe containing the counts. **aldex.in**

# 2) a list that outlines how to order the conditions based on the columns in the dataframe **c(”N”, ”T”, ”N”, ”T”)**

# 3) the number of Dirichlet samples. We use a minimum of 2048 for production purposes, and 256 for testing **mc.samples**

# 4) the likelihood that the effect size is zero or worse **t.significant**

# 5) the median relative difference between conditions **t.meaningful**. A value of 1.5 ensures that the between condition difference is always greater than the within-condition difference. A value of 2.0 is conservative and ensures that the between condition difference is always more than twice the within condition difference.

# run aldex. NOTE: this can take a while depending on your machine and the memory available
```
x <- aldex(aldex.in, c(”N”, ”T”, ”N”, ”T”), mc.samples=2048, t.significant=0.01,
t.meaningful=1.5)
```

# examine the output file.

# The built-in plot function can be used to rapidly display traditional MA style plots of the output as well as the MW plots given in [**?**]. Additional options can be found in the function
```
plot.aldex(x, type=”MA”)
plot.aldex(x, type=”MW”)
```

# 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(x)
```

# This returns only the medians of the summary values
```
y <- summary.aldex(x, median.only=T, sort=T
```

4 Modelling the data as proportions rather than counts

Individual sequence reads are assigned to genes or genomic features in an RNA-seq experiment. Reads assigned to these features have several sources of variation: technical variation (sometimes called shot noise), biological variation within a condition, biological variation between conditions and unexplained variation. While these sources of variation are being recognized by other widely-used RNA-seq analysis tools, the consensus view in the literature is that the underlying variation is best explained by the negative binomial distribution.

We take a different approach. First, we assume no underlying distribution. Second, we model the reads as proportions of the data available rather than as counts. The proportional nature of the data stems from a two major causes: 1) the cell has finite resources, and resources devoted to one function cannot be devoted to another, and 2) there are a large but finite number of reads available from a next-generation sequencing run. See the **submitted manuscript** 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 isometric log-transformated following the advice of Egozcue for dealing with proporitonal data. This has two effects. First, the data for each sample is centred on its mean expression level and second, the values become largely independent and can be dealt with as statistically independent components.

5 Case study: The Liver vs. Kidney dataset from Marioni et al[1]

5.1 Introduction

This section contains a detailed analysis of the Liver vs. Kidney technical replicate dataset of Marioni et al[1] that is often used to benchmark RNA-Seq methodologies. This dataset is a very good test of ALDEx because most genes have very few reads mapping to them.

5.2 Reading the data and making the input dataframe

```
d <- read.table(”tech_rep_read_counts.txt”, header=T, stringsAsFactors=T)
```

This dataset contains many different sample replicates. There were two libraries made # make a dataframe with the replicates we want

```
id <- as.character(d[,1])
K11 <- d$R1L1Kidney
L12 <- d$R1L2Liver
L21 <- d$R2L1Liver
K22 <- d$R2L2Kidney


aldex.in <- data.frame(K11,L12,L21,K22, row.names=id)

library(ALDEx)
```

5.3 Generating and examining the dataset

# run it and put the values in x

```
x <- aldex(aldex.in, c(”K”, ”L”, ”L”, ”K”), mc.samples=512,t.significant=0.01,
t.meaningful=1.5)
```

# This extracts out a summary of the data useful for plotting or saving:

```
y <- summary.aldex(x, median.only=T)
```

```
write.table(y, file="aldex_512.txt", sep="\t", quote=F)
```

This is the dataset that you want to explore. ALDEx scales all expression values by log2(expression) - log2(mean-expression) (see the paper for details), so values less than 0 are expressed at levels lower 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`
- `expression.among.q50 - median value of the mixture distribution of all samples`
- `expression.within.K.q50 - median value of the mixture distribution samples in K`
- `expression.within.L.q50 - median value of the mixture distribution samples in L`
- `expression.sample.K11.q50 - median expression value of individual sample K11`
- `expression.sample.L12.q50 - etc.`
- `expression.sample.L21.q50 - etc.`
- `expression.sample.K22.q50 - etc.`
- `difference.between.q50 - median expression difference between K and L`
- `difference.within.q50 - maximum median difference within K or L`
- `effect.q50 - effect size, how distinct the between difference is as a function of the within difference`
- `criteria.significance - the likelihood that effect.q50 is less than 0`
- `criteria.significant - T if criteria.significance < t.significant`
- `criteria.meaning - absolute value of effect.q50`
- `criteria.meaningful - T if criteria.significance > t.meaningful`

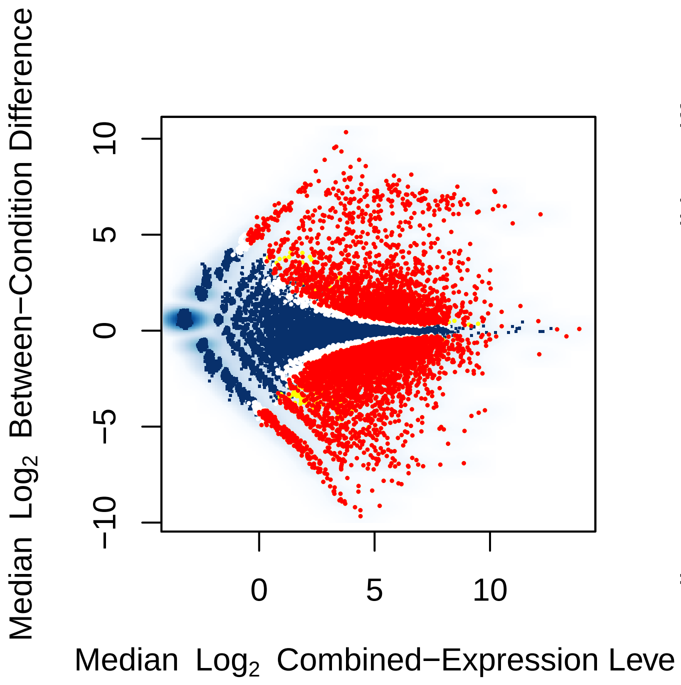
Both criteria must be TRUE for a gene to be called as differentially expressed.

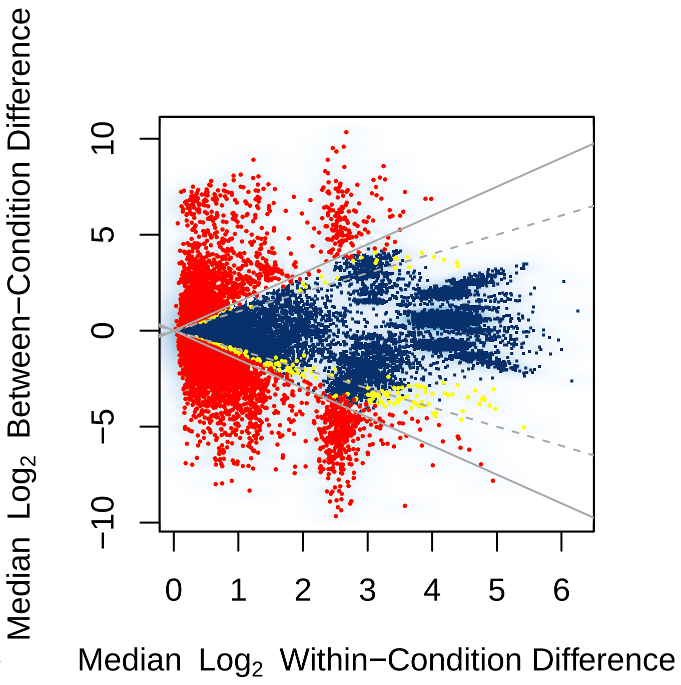
(id)	expr .amg .q50	expr .wthn .K .q50	expr .wthn .L .q50	expr .smp .K11 .q50	expr .smp .L12 .q50	expr .smp .L21 .q50	expr .smp .K22 .q50	diff .be-tween .q50	diff .wthn .q50	effect .q50	crit .sig-nance	crit .sig	crit .mn-ing	crit .mn-ingful
9349	-3.19	-3.56	-2.77	-3.64	-2.73	-2.82	-3.46	0.71	4.16	0.17	0.42	F	0.173	F
9350	2.18	0.99	3.74	0.72	4.29	2.88	1.19	2.67	1.49	1.84	0.001	T	1.84	T
9351	-3.17	-3.35	-2.98	-3.32	-3.08	-2.89	-3.37	0.45	4.05	0.12	0.44	F	0.12	F
9352	-1.50	-2.05	-0.79	-3.34	0.27	-2.82	-1.20	1.18	4.40	0.28	0.35	F	0.286	F
2679	5.41	5.41	5.90	5.43	6.61	4.95	5.40	0.48	1.66	0.35	0.48	F	0.358	F
2678	12.27	12.27	12.23	12.28	12.32	12.15	12.27	-0.04	0.17	-0.29	0.481	F	0.294	F

This makes a plot of the data, either as MA or MW plots:

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

The MA and MW plots are illustrated. Blue are genes not called significantly different, yellow are significant but not meaningful, white are meaningful but not significant, and red passes both filters.





References

[1] Marioni JC, Mason CE, Mane SM, Stephens M, Gilad Y (2008) Rna-seq: an assessment of technical reproducibility and comparison with gene expression arrays. Genome Res 18: 1509-17.