Tutorial: Testing for Differential Abundance

Michael B. Waak

24 November 2018

Contents

1 Description

This tutorial discusses methods of testing feature frequency data—e.g., amplicon sequence variants (ASVs) or operational taxonomic units (OTUs)—for differential abundance (DA). DA should be used with sample sets that come from similar/identical environments either before/after an experimental treatment (versus a control group) or sequentially in space/time (e.g., before and after water treatment). Testing for DA between unrelated environments (e.g., different geographic sites without direct or even indirect interaction) is not advisable, as most features will probably be differentially abundant (and that's not surprising). To assess such environments for differences, look at their alpha and beta diversities instead—not differential abundance. As with any statistical test, ask yourself what you want to achieve with the test and whether the test is appropriate for your scientific question. The null hypothesis for DA is that a feature has similar abundance among all samples in two or more groupings of the samples. If a feature is differentially abundant, it could be interpretted that this difference in abundance is a result of the treatment, for example.

Due to the large variance of library sizes in high-throughput sequencing datasets, it is common practice to rarefy feature frequencies, such that all samples have equal sample sizes. Accounting for this variance (or sampling bias) is critical for analysis of diversity (alpha and beta) as well as differential abundance. It has been argued, however, that rarefying is statistically inadmissable, as one loses statistical power and may introduce artifacts of non-biological origin [1]. This is especially true for differential abundance.

Assuming you have specific goals and have given careful thought to your data, this tutorial provides guidance for

testing differential abundance of features using two methods: `DESeq2` [2] and `metagenomeSeq` [3]. You should consider filtering your data prior to assessing differential abundance, by removing low-abundance samples (e.g., <1000 sequences) and/or removing rare/low-abundance features [4].

The following summaries come from McMurdie and Holmes [1].

- **DESeq2** A Negative Binomial Wald Test using standard maximum likelihood estimates for generalized linear model coefficients assuming a zero-mean normal prior distribution, implemented in the `nbinomWaldTest` method of the `DESeq2` package.
- **metagenomeSeq** An Expectation-Maximization estimate of the posterior probabilities of differential abundance based on a Zero Inflated Gaussian model, implemented in the `fitZig` method of the `metagenomeSeq` package

2 Installing required packages

Installation only needs to be done once, unless you are updating to a new version. Load installed packages when starting a new session of R using `library("[package name]")`.

2.1 GitHub packages

This tutorial imports Qiime 2 artifacts directly into R using the `qiime2R` package. This package is available on GitHub and cannot be installed using the `install.packages()` command in base R.

To install `qiime2R`, the `devtools` package must also be installed.

```
install.packages("devtools") # If you don't have devtools installed
devtools::install_github("jbisanz/qiime2R")
```

2.2 Bioconductor packages

Several packages must be installed using Bioconductor, meaning they cannot be installed using the `install.packages()` command in base R, which installs packages either from a CRAN repository or a local package archive file.

To install these packages:

DESeq2

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("DESeq2")
```

metagenomeSeq

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("metagenomeSeq")
```

NOTE: *You likely need to restart R (i.e., closing and then re-opening RStudio) to see and access these packages*

3 Getting Started

3.1 Import data into R

You will need your *un-rarefied* feature table. A feature table may contain either ASVs or OTUs. These should be absolute frequencies (i.e., integers)—not relative frequencies (fractions/decimals).

```
library(qiime2R)
```

```
feature_table <- read_qza("feature_table.qza")
```

To get the data from the artifact into a data frame, do this:

```
feature_table.df <- as.data.frame(feature_table$data)
```

We can take a look at what this data frame looks like. Features should be listed by row and samples by column.

```
head(feature_table.df)
```

```
##                                  M01 M02 M04 M05 M06 M07 M08 M11 M13 M14
## 1c4729e8356ac59b7860570975cff890   0   0   0   0   0   0   0   0   0   0
## bc4e67db39d75a263a1f898c48d72caa   0   0   0   0   0   0   0   0   0   0
## e5c7d83f882731e36dcd929f1f313401   0   0   0   0   0   0   0   0   0   0
## 48a7f675e277fca203661d12aef5c839   0   0   0   0   0   0   0   0   0   0
## cc4f6a6a0b3f786fd6fa66ba07c1cc36   0   0   0   0   0   0   0   0   0   0
## 11058fd13ce009a6ab5def20dbfd6a2a   0   0   0   0   0   0   0   0   0   0
```

Feature tables are typically very sparse (i.e., containing mostly zeroes, as you see above).

3.2 Subset your data

You will need metadata to test for differential abundance (remember: DA is for comparing sets, e.g., before and after a treatment). You could do this by importing a metadata file (or *mapping file*, in Qiime 1 lingo). This was created manually in MS Excel from a Qiime 2 metadata file (`metadata.txt`), first by deleting the `#q2:types` row (the column headers were kept) and then saving as a new tab-delimited text file (`metadata_R.txt`). Importing the Qiime 2 version first and then removing this row may result in columns being imported into R as factors, rather than as numeric values (which may have undesirable effects you may or may not become aware of). If you want to subset your data to remove certain samples, do it now. Note that you could also do subsetting in Qiime prior to importing.

```
# For a tab-delimited .txt or .tsv file:
metadata <- read.delim("metadata_R.txt", row.names=1)

# For a comma-delimited .csv file:
metadata <- read.delim("metadata_R.csv", row.names=1, sep = ",")

# Subset
metadata.sub <- subset(metadata, City == "City_A", drop=TRUE)
metadata.sub <- subset(metadata.sub, Sample_type == "01_Raw_Water" |
                         Sample_type == "04_Treated_Water", drop=TRUE)

# Make sure to drop levels for factors
metadata.sub$City <- droplevels(metadata.sub$City)
metadata.sub$Sample_type <- droplevels(metadata.sub$Sample_type)

# OR don't subset
metadata.sub <- metadata
```

3.3 Filter your data

At this point, you may want to filter your feature table to remove sequences and/or features that are low in abundance. You could also do this in Qiime prior to importing, but doing so in R is more flexible. Qiime could create many intermediate files (especially if doing multiple tests of DA on different subsets). These files may be undesirable if you wish to keep your working directories tidy and organized, but you lose the provenance associated with Qiime artifacts.

Rows are features. Remove low-abundant features (e.g., <10 observations across all samples).

```
feature_table.df <- feature_table.df[rowSums(feature_table.df[,-1]) < 10,]
```

Columns are samples. Remove low-abundant samples (e.g., <1000 sequences/sample). Because the number of features per sample may have decreased, this should be performed *after* removing low-abundant features.

```
feature_table.df <- feature_table.df[,colSums(feature_table.df[-1,]) < 1000]
```

Disclaimer: *You should report any filtering steps when publishing results derived from this filtered dataset. The above is for demonstration only and is not necessarily an endorsement of any specific filtering threshold. Though filtering is advised, you should have justification for whichever thresholds you choose.*

3.4 Sort and tidy up your data

After subsetting/filtering, make sure everything is accounted for and sorted correctly

```
common.ids <- intersect(row.names(metadata.sub), colnames(feature_table.df))
metadata.sub <- metadata.sub[common.ids,]
feature_table.df <- feature_table.df[,common.ids]
```

If you subset but did not filter, you should make sure any empty rows (i.e., features with no observations in the sample subset) are removed. Even if you did not subset, there is no harm in checking for null features.

```
feature_table.df <- feature_table.df[rowSums(feature_table.df)!=0,]
```

4 Differential abundance methods

4.1 DESeq2

Load the `DESeq2` package.

```
library(DESeq2)
```

Import the feature table as a `DESeqDataSet-class` object. `colData` is for metadata about the columns (each sample is a column in `countData`). Also specify what you are interested in testing (i.e., specify a column that contains sample groupings; I specify `~ Sample_type`. In R, ~ means, "as a function of").

```
da_ds2 <- DESeqDataSetFromMatrix(countData = feature_table.df,
                                 colData = metadata.sub,
                                 ~ Sample_type)
```

Perform the test for DA.

```
# Estimate size factors
da_ds2 <- estimateSizeFactors(da_ds2)

# Estimate dispersions for Negative Binomial distributed data
da_ds2 <- estimateDispersions(da_ds2)
```

```
# Test for significance of coefficients in a Negative Binomial GLM using the Wald test
da_ds2 <- nbinomWaldTest(da_ds2)
```

Now we can view our results. You will need to make some statistical decisions at this point, such as your desired alpha value for significance (default is 0.1, but I chose 0.05). Because you are testing differential abundance for potentially thousands of features, it is wise to control for Type I errors (i.e., false positives). This means *P* values will be adjusted. I recommend using the false discovery rate (FDR) via the procedures of either Benjamini-Hochberg (`"BH"`) or Benjamini-Yekutieli (“`BY`”) (See `?p.adjust` for all options and their descriptions).

```
results <- results(da_ds2,
                   alpha = 0.05,
                   pAdjustMethod = "BH",
                   format = "DataFrame")
```

You should explore `results`, which is `Formal class DESeqResults`, using `View(results)`.Running the following will create a data frame with the relevant information you will need. The `P.adj` column is the final *P* value you want, which has been adjusted for Type I errors, to identify significant differences in feature abundances. Make sure you report this as an **adjusted** *P* value, indicating it was changed using the Benjamini-Hochberg procedure [5].

```
results.df <- data.frame(Feature = results@rownames,
                         baseMean = results@listData[["baseMean"]],
                         log2Change = results@listData[["log2FoldChange"]],
                         std_err = results@listData[["lfcSE"]],
                         Wald_stat = results@listData[["stat"]],
                         P.val = results@listData[["pvalue"]],
                         P.adj = results@listData[["padj"]])

#View the description of column headers
results@elementMetadata@listData[["description"]]
```

```
## [1] "mean of normalized counts for all samples"
## [2] "log2 fold change (MLE): Sample type 04 Treated Water vs 01 Raw Water"
## [3] "standard error: Sample type 04 Treated Water vs 01 Raw Water"
## [4] "Wald statistic: Sample type 04 Treated Water vs 01 Raw Water"
## [5] "Wald test p-value: Sample type 04 Treated Water vs 01 Raw Water"
## [6] "BH adjusted p-values"
```

You can inspect the data frame manually in R or you could export it for general record-keeping and/or viewing elsewhere (e.g., MS Excel). See [Exporting-results].

To inspect the full data frame:

```
View(results.df)
```

To see just the top of the data frame:

```
head(results.df)
```

```
##              Feature baseMean log2Change std_err Wald_stat  P.val P.adj
## 1 7b653832e2b9cae...   0.5876     2.6043  2.9735    0.8758 0.3811    NA
## 2 b13437ac388f1f7...   0.0747     1.0441  2.9907    0.3491 0.7270    NA
## 3 487c01b1e1a2fbb...   0.1120     1.2611  2.9898    0.4218 0.6732    NA
## 4 50c415ffae51559...   0.0777    -0.7094  2.9856   -0.2376 0.8122    NA
## 5 035b0d04317e364...   0.1320    -1.0512  2.9830   -0.3524 0.7245    NA
```

```
## 6 5e9c67e765ca65e...   0.0475    -0.4324  2.9882   -0.1447 0.8849    NA
```

You will likely notice `NA` values under `P.val` and `P.adj`. The process used for replacing a value with `NA` is separate for *P* values and adjusted *P* values.

For *P* values, some results are flagged as outliers by `DESeq2`. This can be turned off by adding `cooksCutoff = FALSE` to `results(da_ds2, ...)`. The `DESeq2` vignette offers the following:

> RNA-seq data sometimes contain isolated instances of very large counts that are apparently unrelated to the experimental or study design, and which may be considered outliers. There are many reasons why outliers can arise, including rare technical or experimental artifacts, read mapping problems in the case of genetically differing samples, and genuine, but rare biological events. In many cases, users appear primarily interested in genes that show a consistent behavior, and this is the reason why by default, genes that are affected by such outliers are set aside by DESeq2, or if there are sufficient samples, outlier counts are replaced for model fitting.

For adjusted *P* values, `NA` means independent filtering has been performed. This can be turned off by adding `independentFiltering = FALSE` to `results(da_ds2, ...)`. The `DESeq2` vignette offers the following:

> The goal of independent filtering is to filter out those tests from the procedure that have no, or little chance of showing significant evidence, without even looking at their test statistic. Typically, this results in increased detection power at the same experiment-wide type I error. Here, we measure experiment-wide type I error in terms of the false discovery rate.

To read more on either filtering method, run `vignette("DESeq2")` and then search, respectively, for either "count outliers" or "independent filtering".

4.2 MetagenomeSeq

[Coming soon…]

4.3 Exporting results

```r
# To create a tab-delimited text file (.txt):
write.table(results.df,
            file = "DESeq2_results.txt",
            quote = FALSE, sep = "\t", col.names = TRUE)

# To create a comma-separated file (.csv):
write.table(results.df,
            file = "DESeq2_results.csv",
            quote = FALSE, sep = ",", col.names = TRUE)
```

License

Software versions

This tutorial was prepared using the following software/versions:

Operating system: macOS 10.14.1

R version 3.5.1 (2018-07-02)

Platform: x86_64-apple-darwin15.6.0 (64-bit)

locale: en_US.UTF-8

attached base packages: *parallel*, *stats4*, *stats*, *graphics*, *grDevices*, *utils*, *datasets*, *methods* and *base*

other attached packages: *DESeq2(v.1.22.1)*, *SummarizedExperiment(v.1.12.0)*, *DelayedArray(v.0.8.0)*, *BiocParallel(v.1.16.0)*, *matrixStats(v.0.54.0)*, *Biobase(v.2.42.0)*, *GenomicRanges(v.1.34.0)*, *GenomeInfoDb(v.1.18.1)*, *IRanges(v.2.16.0)*, *S4Vectors(v.0.20.1)*, *BiocGenerics(v.0.28.0)* and *qiime2R(v.0.99.1)*

References

1. McMurdie PJ, Holmes S. Waste Not, Want Not: Why rarefying microbiome data is inadmissible. *PLoS Comput Biol* 2014; **10**: e1003531.

2. Love MI, Huber W, Anders S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol* 2014; **15**: 31–21.

3. Paulson JN, Stine OC, Bravo HC, Pop M. Differential abundance analysis for microbial marker-gene surveys. *Nature* 2013; **10**: 1200–1202.

4. Bokulich NA, Subramanian S, Faith JJ, Gevers D, Gordon JI, Knight R, et al. Quality-filtering vastly improves diversity estimates from Illumina amplicon sequencing. *Nat Methods* 2013; **10**: 57–59.

5. Benjamini Y, Hochberg Y. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J Roy Stat Soc B Met* 1995; **57**: 289–300.