# `metagenomeSeq`: Statistical analysis for sparse high-throughput sequencing

Joseph Nathaniel Paulson

Applied Mathematics & Statistics, and Scientific Computation
Center for Bioinformatics and Computational Biology
University of Maryland, College Park

jpaulson@umiacs.umd.edu

Modified: April 10, 2014. Compiled: April 11, 2014

## Contents

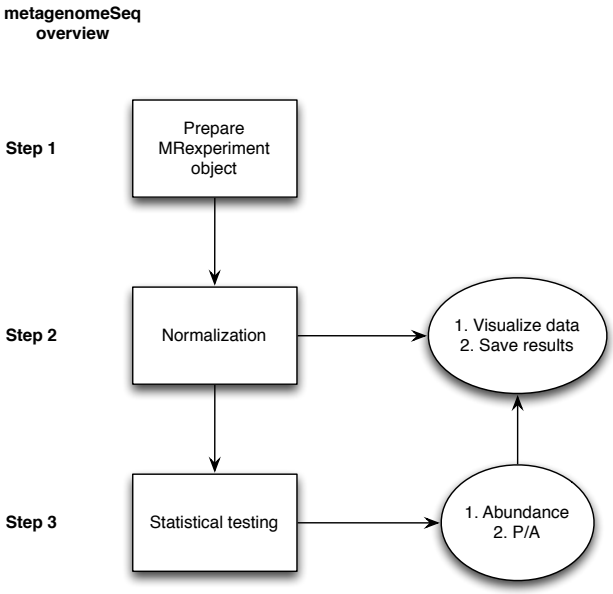


Figure 1: General overview. metagenomeSeq requires the user to convert their data into MRexperiment objects. Using those MRexperiment objects, one can normalize their data, run statistical tests (abundance or presence-absence), and visualize or save results.

# 1 Introduction

**This is a vignette for pieces of an association study pipeline. For a full list of functions available in the package: help(package=metagenomeSeq). For more information about a particular function call: ?function.**

Metagenomics is the study of genetic material targeted directly from an environmental community. Originally focused on exploratory and validation projects, these studies now focus on understanding the differences in microbial communities caused by phenotypic differences. Analyzing high-throughput sequencing data has been a challenge to researchers due to the unique biological and technological biases that are present in marker-gene survey data.

We present a R package, `metagenomeSeq`, that implements methods developed to account for previously unaddressed biases specific to high-throughput sequencing microbial marker-gene survey data. Our method implements a novel normalization technique and method to account for sparsity due to undersampling. Other methods include White *et al.*'s Metastats and Segata *et al.*'s LEfSe. The first is a non-parametric permutation test on $t$-statistics and the second is a non-parametric Kruskal-Wallis test followed by subsequent wilcox rank-sum tests on subgroups to guard against positive discoveries of differential abundance driven by potential confounders - neither address normalization nor sparsity.

This vignette describes the basic protocol when using `metagenomeSeq`. A normalization method able to control for biases in measurements across taxanomic features and a mixture model that implements a zero-inflated Gaussian distribution to account for varying depths of coverage are implemented. Using a linear model methodology, it is easy to include confounding sources of variability and interpret results. Additionally, visualization functions are provided to examine discoveries.

The software was designed to determine features (be it Operational Taxanomic Unit (OTU), species, etc.) that are differentially abundant between two or more groups of multiple samples. The software was also designed to address the effects of both normalization and undersampling of microbial communities on disease association detection and testing of feature correlations.

## 2 Data preparation

Microbial marker gene sequence data is preprocessed and counts are algorithmically defined from project-specific sequence data by clustering reads according to read similarity. Given $m$ features and $n$ samples, the elements in a count matrix $\mathbf{C}$ $(m, n)$, $c_{ij}$, are the number of reads annotated for a particular feature $i$ (whether it be OTU, species, genus, etc.) in sample $j$.

$$\begin{array}{c} \\ feature_1 \\ feature_2 \\ \vdots \\ feature_m \end{array} \begin{array}{c} \begin{array}{cccc} sample_1 & sample_2 & \dots & sample_n \end{array} \\ \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \dots & c_{mn} \end{pmatrix} \end{array}$$

Count data should be stored in a delimited (tab by default) file with sample names along the first row and feature names along the first column.

Data is prepared and formatted as a MRexperiment object. For an overview of the internal structure please see Appendix A.

### 2.1 Example datasets

There are two datasets included as examples in the `metagenomeSeq` package. Data needs to be in a `MRexperiment` object format to normalize, run statistical tests, and visualize. As an example, throughout the vignette we'll use the following datasets. To understand a function's usage or included data simply enter ?functionName.

```
library(metagenomeSeq)
```

1. Human lung microbiome [1]: The lung microbiome consists of respiratory flora sampled from six healthy individuals. Three healthy nonsmokers and three healthy smokers. The upper lung tracts were sampled by oral wash and oro-/nasopharyngeal swabs. Samples were taken using two bronchoscopes, serial bronchoalveolar lavage and lower airway protected brushes.

```
data(lungData)
lungData

## MRexperiment (storageMode: environment)
## assayData: 51891 features, 78 samples
##   element names: counts
## protocolData: none
## phenoData
##   sampleNames: CHK_6467_E3B11_BRONCH2_PREWASH_V1V2
##     CHK_6467_E3B11_OW_V1V2 ... CHK_6467_E3B09_BAL_A_V1V2
##     (78 total)
##   varLabels: SampleType SiteSampled SmokingStatus
##   varMetadata: labelDescription
## featureData
##   featureNames: 1 2 ... 51891 (51891 total)
##   fvarLabels: taxa
```

```
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
##   pubMedIds: 21680950
## Annotation:
```

2. Humanized gnotobiotic mouse gut [2]: Twelve germ-free adult male C57BL/6J mice were fed a low-fat, plant polysaccharide-rich diet. Each mouse was gavaged with healthy adult human fecal material. Following the fecal transplant, mice remained on the low-fat, plant polysacchaaride-rich diet for four weeks, following which a subset of 6 were switched to a high-fat and high-sugar diet for eight weeks. Fecal samples for each mouse went through PCR amplification of the bacterial 16S rRNA gene V2 region weekly. Details of experimental protocols and further details of the data can be found in Turnbaugh et. al. Sequences and further information can be found at: `http://gordonlab.wustl.edu/TurnbaughSE_10_09/STM_2009.html`

```
data(mouseData)
mouseData

## MRexperiment (storageMode: environment)
## assayData: 10172 features, 139 samples
##   element names: counts
## protocolData: none
## phenoData
##   sampleNames: PM1:20080107 PM1:20080108 ... PM9:20080303
##     (139 total)
##   varLabels: mouseID date diet
##   varMetadata: labelDescription
## featureData
##   featureNames: Prevotellaceae:1 Lachnospiraceae:1 ...
##     Parabacteroides:956 (10172 total)
##   fvarLabels: fdata
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
##   pubMedIds: 20368178
## Annotation:
```

## 2.2 Loading count data

Following preprocessing and annotation of sequencing data `metagenomeSeq` requires a count matrix with features along rows and samples along the columns. `metagenomeSeq` includes functions for loading delimited files of counts `load_meta` and phenodata `load_phenoData`.

As an example, a portion of the lung microbiome [1] OTU matrix is provided in `metagenomeSeq`'s library "extdata" folder. The OTU matrix is stored as a tab delimited file. `load_meta` loads the taxa and counts into a list.

```
dataDirectory <- system.file("extdata", package = "metagenomeSeq")
lung = load_meta(file.path(dataDirectory, "CHK_NAME.otus.count.csv"))
dim(lung$counts)

## [1] 1000   78
```

## 2.3 Loading taxonomy

Next we want to load the annotated taxonomy. Check to make sure that your taxa annotations and OTUs are in the same order as your matrix rows.

```
taxa = read.delim(file.path(dataDirectory, "CHK_otus.taxonomy.csv"),
    stringsAsFactors = F)[, 2]
otu = read.delim(file.path(dataDirectory, "CHK_otus.taxonomy.csv"),
    stringsAsFactors = F)[, 1]
```

As our OTUs appear to be in order with the count matrix we loaded earlier, the next step is to load phenodata.

**Warning**: features need to have the same names as the rows of the count matrix when we create the MRexperiment object for provenance purposes.

## 2.4 Loading metadata

Phenotype data can be optionally loaded into `R` with `load_phenoData`. This function loads the data as a list.

```
clin = load_phenoData(file.path(dataDirectory, "CHK_clinical.csv"),
    tran = TRUE)
ord = match(colnames(lung$counts), rownames(clin))
clin = clin[ord, ]
head(clin[1:2, ])

##                                            SampleType
## CHK_6467_E3B11_BRONCH2_PREWASH_V1V2 Bronch2.PreWash
## CHK_6467_E3B11_OW_V1V2                             OW
##                                              SiteSampled
## CHK_6467_E3B11_BRONCH2_PREWASH_V1V2 Bronchoscope.Channel
## CHK_6467_E3B11_OW_V1V2                       OralCavity
##                                     SmokingStatus
## CHK_6467_E3B11_BRONCH2_PREWASH_V1V2        Smoker
## CHK_6467_E3B11_OW_V1V2                     Smoker
```

**Warning**: phenotypes must have the same names as the columns on the count matrix when we create the MRexperiment object for provenance purposes.

## 2.5 Creating a `MRexperiment` object

Function `newMRexperiment` takes a count matrix, phenoData (annotated data frame), and featureData (annotated data frame) as input. `Biobase` provides functions to create annotated data frames. Library sizes (depths of coverage) and normalization factors are also optional inputs.

```
phenotypeData = as(clin, "AnnotatedDataFrame")
phenotypeData

## An object of class 'AnnotatedDataFrame'
##   rowNames: CHK_6467_E3B11_BRONCH2_PREWASH_V1V2
##     CHK_6467_E3B11_OW_V1V2 ... CHK_6467_E3B09_BAL_A_V1V2
##     (78 total)
```

```
##   varLabels: SampleType SiteSampled SmokingStatus
##   varMetadata: labelDescription
```

A feature annotated data frame. In this example it is simply the OTU numbers, but it can as easily be the annotated taxonomy at multiple levels.

```
OTUdata = as(lung$taxa, "AnnotatedDataFrame")
varLabels(OTUdata) = "taxa"
OTUdata

## An object of class 'AnnotatedDataFrame'
##   rowNames: 1 2 ... 1000 (1000 total)
##   varLabels: taxa
##   varMetadata: labelDescription
```

```
obj = newMRexperiment(lung$counts,phenoData=phenotypeData,featureData=OTUdata)
# Links to a paper providing further details can be included optionally.
# experimentData(obj) = annotate::pmid2MIAME("21680950")
obj

## MRexperiment (storageMode: environment)
## assayData: 1000 features, 78 samples
##   element names: counts
## protocolData: none
## phenoData
##   sampleNames: CHK_6467_E3B11_BRONCH2_PREWASH_V1V2
##     CHK_6467_E3B11_OW_V1V2 ... CHK_6467_E3B09_BAL_A_V1V2
##     (78 total)
##   varLabels: SampleType SiteSampled SmokingStatus
##   varMetadata: labelDescription
## featureData
##   featureNames: 1 2 ... 1000 (1000 total)
##   fvarLabels: taxa
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
```

Alternatively, you can load in Biome-format data (outputs of QIIME and Mothur) using the `load_biom` function. If a biom class object is already loaded into R it can be converted to a MRexperiment-class object using the `biom2MRexperiment` function.

# 3 Normalization

Normalization is required due to varying depths of coverage across samples. `cumNorm` is a normalization method that calculates scaling factors equal to the sum of counts up to a particular quantile.

Denote the $l$th quantile of sample $j$ as $q_j^l$, that is, in sample $j$ there are $l$ taxonomic features with counts smaller than $q_j^l$. For $l = \lfloor .95m \rfloor$ then $q_j^l$ corresponds to the 95th percentile of the count distribution for sample $j$.

Denote $s_j^l = \sum_{(i|c_{ij} \leq q_j^l)} c_{ij}$ as the sum of counts for sample $j$ up to the $l$th quantile. Our normalization chooses a value $\hat{l} \leq m$ to define a normalization scaling factor for each sample to produce normalized counts $\tilde{c_{ij}} = \frac{c_{ij}}{s_j^{\hat{l}}} N$ where $N$ is an appropriately chosen normalization constant. See Appendix C for more information on how our method calculates the proper percentile.

These normalization factors are stored in the experiment summary slot. Functions to determine the proper percentile `cumNormStat`, save normalized counts `exportMat`, or save various sample statistics `exportStats` are also provided. Normalized counts can be called easily by `cumNormMat(MRexperimentObject)` or `MRcounts(MRexperimentObject,norm=TRUE,log=FALSE)`

## 3.1 Calculating normalization factors

After defining a `MRexperiment` object, the first step is to calculate the proper percentile by which to normalize counts. There are several options in calculating and visualizing the relative differences in the reference. Figure 3 is an example from the lung dataset.

```
data(lungData)
p = cumNormStatFast(lungData)
```

To calculate the scaling factors we simply run `cumNorm`

```
lungData = cumNorm(lungData, p = p)
```

The user can alternatively choose different percentiles for the normalization scheme by specifying $p$.

There are other functions, including `normFactors`, `cumNormMat`, that return the normalization factors or a normalized matrix for a specified percentile. To see a full list of functions please refer to the manual and help pages.

## 3.2 Exporting data

To export normalized count matrices:

```
mat = MRcounts(lungData, norm = TRUE, log = TRUE)[1:5, 1:5]
exportMat(mat, file = file.path(dataDirectory, "tmp.tsv"))
```

To save sample statistics (sample scaling factor, quantile value, number of identified features and library size):

```
exportStats(lungData[, 1:5], file = file.path(dataDirectory, "tmp.tsv"))
head(read.csv(file = file.path(dataDirectory, "tmp.tsv"), sep = "\t"))
```

```
##                                Subject Scaling.factor
## 1 CHK_6467_E3B11_BRONCH2_PREWASH_V1V2             67
## 2              CHK_6467_E3B11_OW_V1V2           2475
## 3              CHK_6467_E3B08_OW_V1V2           2198
## 4           CHK_6467_E3B07_BAL_A_V1V2            836
## 5           CHK_6467_E3B11_BAL_A_V1V2           1008
##   Quantile.value Number.of.identified.features Library.size
## 1              5                            60          271
## 2              1                          3299         7863
## 3              2                          2994         8360
## 4              2                          1188         5249
## 5              2                          1098         3383
```

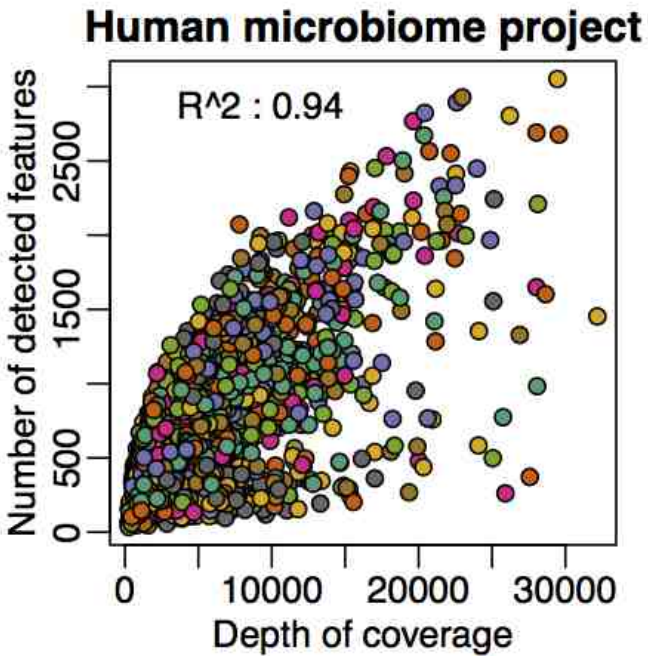


Figure 2: The number of unique features is plotted against depth of coverage for samples from the Human Microbiome Project [3]. Including the depth of coverage and the interaction of body site and sequencing site we are able to acheive an adjusted $R^2$ of .94. The zero-inflated Gaussian mixture was developed to account for missing features.

# 4 Statistical testing

Now that we have taken care of normalization we can address the effects of under sampling on the detecting differentially abundant features (OTUs, genes, etc).

## 4.1 Zero-inflated Gaussian mixture model

The depth of coverage in a sample is directly related to how many features are detected in a sample motivating our zero-inflated Gaussian (ZIG) mixture model. Figure 2 is representative of the linear relationship between depth of coverage and OTU identification ubiquitous in marker-gene survey datasets currently available. For a quick overview of the mathematical model see Appendix B.

Function `fitZig` performs a complex mathematical optimization routine to estimate probabilities that a zero for a particular feature in a sample is a technical zero or not. The function relies heavily on the `limma` package [4]. Design matrices can be created in R by using the `model.matrix` function and are inputs for `fitZig`.

For large survey studies it is often pertinent to include phenotype information or confounders into a design matrix when testing the association between the abundance of taxonomic features and a phenotype phenotype of interest (disease, for instance). Our linear model methodology can easily incorporate these confounding covariates in a straightforward manner. `fitZig` output includes weighted fits for each of the $m$ features. Results can be filtered and saved using `MRcoefs` or `MRtable`.

## 4.2 Example using fitZig for differential abundance testing

**Warning**: The user should restrict significant features to those with a minimum number of positive samples. What this means is that one should not claim features are significant unless the effective number of samples is above a particular percentage. For example, fold-change

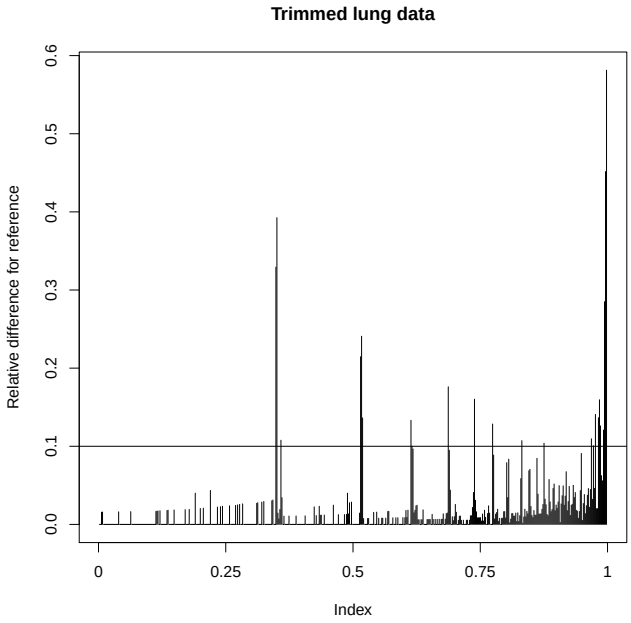


Figure 3: Relative difference for the median difference in counts from the reference.

estimates might be unreliable if an entire group does not have a positive count for the feature in question.

We recommend the user remove features based on the number of estimated effective samples, please see `calculateEffectiveSamples`. We recommend removing features with less than the average number of effective samples in all features. In essence, setting eff = .5 when using `MRcoefs`, `MRfulltable`, or `MRtable`. To find features absent from a group the function `uniqueFeatures` provides a table of the feature ids, the number of positive features and reads for each group.

In our analysis of the lung microbiome data, we can remove features that are not present in many samples, controls, and calculate the normalization factors. The user needs to decide which metadata should be included in the linear model.

```
controls = grep("Extraction.Control", pData(lungData)$SampleType)
lungTrim = lungData[, -controls]
sparseFeatures = which(rowSums(MRcounts(lungTrim) > 0) < 10)
lungTrim = lungTrim[-sparseFeatures, ]
lungp = cumNormStat(lungTrim, pFlag = TRUE, main = "Trimmed lung data")
```

```
lungTrim = cumNorm(lungTrim, p = lungp)
```

After the user defines an appropriate model matrix for hypothesis testing there are optional inputs to `fitZig`, including settings determined by `zigControl`. We ask the user to review the help files for both `fitZig` and `zigControl`. For this example we include body site as covariates and want to test for the bacteria differentially abundant between smokers and non-smokers.

```
smokingStatus = pData(lungTrim)$SmokingStatus
bodySite = pData(lungTrim)$SampleType
normFactor = normFactors(lungTrim)
```

```
normFactor = log2(normFactor/median(normFactor) + 1)
mod = model.matrix(~smokingStatus + bodySite + normFactor)
settings = zigControl(maxit = 10, verbose = TRUE)
fit = fitZig(obj = lungTrim, mod = mod, useCSSoffset = FALSE, control = settings
```

```
## it= 0, nll=88.49, log10(eps+1)=Inf, stillActive=1029
## it= 1, nll=93.19, log10(eps+1)=0.07, stillActive=299
## it= 2, nll=93.29, log10(eps+1)=0.05, stillActive=130
## it= 3, nll=93.76, log10(eps+1)=0.07, stillActive=20
## it= 4, nll=93.87, log10(eps+1)=0.02, stillActive=4
## it= 5, nll=93.87, log10(eps+1)=0.00, stillActive=1
## it= 6, nll=93.86, log10(eps+1)=0.00, stillActive=1
## it= 7, nll=93.85, log10(eps+1)=0.00, stillActive=1
## it= 8, nll=93.85, log10(eps+1)=0.00, stillActive=1
## it= 9, nll=93.84, log10(eps+1)=0.00, stillActive=1
```

```
# The default, useCSSoffset = TRUE, automatically includes the CSS
# scaling normalization factor.
```

The result, `fit`, is a list providing detailed estimates of the fits including a `limma` fit in `fit$fit` and an `ebayes` statistical fit in `fit$eb`. This data can be analyzed like any `limma` fit and in this example, the column of the fitted coefficientsrepresents the fold-change for our "smoker" vs. "nonsmoker" analysis.

Looking at the particular analysis just performed, there appears to be OTUs representing two *Prevotella*, two *Neisseria*, a *Porphyromonas* and a *Leptotrichia* that are differentially abundant. One should check that similarly annotated OTUs are not equally differentially abundant in controls.

Alternatively, the user can input a model with their own normalization factors including them directly in the model matrix and specifying the option `useCSSoffset = FALSE` in fitZig.

### 4.3 Exporting fits

Currently functions are being developed to wrap and output results more neatly, but `MRcoefs`, `MRtable`, `MRfulltable` can be used to view coefficient fits and related statistics and export the data with optional output values - see help files to learn how they differ.

To only consider features that are found in a large percentage of effectively positive (positive samples + the weight of zero counts included in the Gaussian mixture) use the **eff** option in the `MRtables`.

```
taxa = sapply(strsplit(as.character(fData(lungTrim)$taxa), split = ";"),
    function(i) {
        i[length(i)]
    })
head(MRcoefs(fit, taxa = taxa, coef = 2))
```

```
##                             smokingStatusSmoker    pValue
## Neisseria polysaccharea                  -4.115 2.555e-15
## Neisseria meningitidis                   -3.982 2.278e-14
## Leptotrichia genomosp. C1                -2.977 8.361e-31
## Prevotella intermedia                    -2.887 4.666e-12
## Porphyromonas sp. UQD 414                -2.688 9.442e-10
## Prevotella paludivivens                   2.661 8.370e-09
```

```
##                             adjPvalue
## Neisseria polysaccharea   1.143e-13
## Neisseria meningitidis    7.562e-13
## Leptotrichia genomosp. C1 1.721e-28
## Prevotella intermedia     1.117e-10
## Porphyromonas sp. UQD 414 1.117e-08
## Prevotella paludivivens   7.489e-08
```

### 4.4 Permutation test

Included is a standard permutation test similar to what was used in Metastats. Below we show the fit for the same model as above using 100 permutations providing p-value resolution to the hundredth. The `coef` parameter refers to the coefficient of interest to test. Below we first generate the list of significant features. We observe two *Prevotella*, two *Veillonella*, a *Streptococcus* and a *Neisseria*.

```
coeffOfInterest = 2
res = fitMeta(obj = lungTrim, mod = mod, useCSSoffset = FALSE, B = 100,
    coef = coeffOfInterest)

# extract p.values and adjust for multiple testing res$p are the
# p-values calculated through permutation
adjustedPvalues = p.adjust(res$p, method = "fdr")

# extract the absolute fold-change estimates
foldChange = abs(res$fit$coef[, coeffOfInterest])

# determine features still significant and order by the
sigList = which(adjustedPvalues <= 0.05)
sigList = sigList[order(foldChange[sigList])]

# view the top taxa associated with the coefficient of interest.
head(taxa[sigList])

## [1] "Streptococcus sanguinis"      "Veillonella montpellierensis"
## [3] "Prevotella pallens"           "Veillonella montpellierensis"
## [5] "Solobacterium moorei"         "Prevotella pallens"
```

### 4.5 Presence-absence testing

The hypothesis for the implemented presence-absence test is that the proportion/odds of a given feature present is higher/lower among one group of individuals compared to another, and we want to test whether any difference in the proportions observed is significant. We use Fisher's exact test to create a 2x2 contingency table and calculate p-values, odd's ratios, and confidence intervals. `fitPA` calculates the presence-absence for each organism and returns a table of p-values, odd's ratios, and confidence intervals. The function will accept either a `MRexperiment` object or matrix. `MRfulltable` when sent a result of fitZig will also include the results of `fitPA`. If there is a desire for a more detailed description, please email me.

```
data(mouseData)
classes = pData(mouseData)$diet
res = fitPA(mouseData[1:5, ], cl = classes)
# Warning - the p-value is calculating 1 despite a high odd's
# ratio.
head(res)
```

```
##                           pvalues oddsRatio  lower upper
## Prevotellaceae:1           1.0000       Inf 0.0163   Inf
## Lachnospiraceae:1          1.0000       Inf 0.0163   Inf
## Unclassified-Screened:1    1.0000       Inf 0.0163   Inf
## Clostridiales:1            0.3885         0 0.0000 24.78
## Clostridiales:2            1.0000       Inf 0.0163   Inf
```

### 4.6 Discovery odds ratio testing

The hypothesis for the implemented discovery test is that the proportion of observed counts for a feature of all counts are comparable between groups. We use Fisher's exact test to create a 2x2 contingency table and calculate p-values, odd's ratios, and confidence intervals. `fitDO` calculates the proportion of counts for each organism and returns a table of p-values, odd's ratios, and confidence intervals. The function will accept either a `MRexperiment` object or matrix.

```
data(mouseData)
classes = pData(mouseData)$diet
res = fitDO(mouseData[1:100, ], cl = classes, norm = FALSE, log = FALSE)
head(res)
```

```
##                           pvalues oddsRatio  lower upper
## Prevotellaceae:1           1.0000       Inf 0.0163   Inf
## Lachnospiraceae:1          1.0000       Inf 0.0163   Inf
## Unclassified-Screened:1    1.0000       Inf 0.0163   Inf
## Clostridiales:1            0.3885         0 0.0000 24.78
## Clostridiales:2            1.0000       Inf 0.0163   Inf
## Firmicutes:1               0.3885         0 0.0000 24.78
```

### 4.7 Feature correlations

To test the correlations of abundance features, or samples, in a pairwise fashion we have implemented `correlationTest` and `correctIndices`. The `correlationTest` function will calculate basic pearson, spearman, kendall correlation statistics for the rows of the input and report the associated p-values.

```
data(mouseData)
cors = correlationTest(mouseData[55:60, ], norm = FALSE, log = FALSE)
head(cors)
```

```
##                                      correlation        p
## Clostridiales:11-Lachnospiraceae:35    -0.02206 7.966e-01
## Clostridiales:11-Coprobacillus:3       -0.01701 8.424e-01
## Clostridiales:11-Lactobacillales:3     -0.01264 8.826e-01
```

```
## Clostridiales:11-Enterococcaceae:3      0.57315 1.663e-13
## Clostridiales:11-Enterococcaceae:4     -0.01264 8.826e-01
## Lachnospiraceae:35-Coprobacillus:3      0.24573 3.548e-03
```

**Caution:** http://www.ncbi.nlm.nih.gov/pubmed/23028285

# 5 Aggregating features

Normalization is recommended at the OTU level. However, functions are in place to aggregate the count matrix (normalized or not), based on a particular user defined level. Using the featureData information in the MRexperiment object, calling `aggregateByTaxonomy` or `aggTax` on a MRexperiment object and declaring particular featureData column name (i.e. 'genus') will aggregate counts to the desired level with the aggfun function (default colSums). Possible aggfun alternatives include colMeans and colMedians.

```
data(mouseData)

# first featureData column as 'chr' vector
taxa = as.character(fData(mouseData)[, 1])

# 2nd item of each taxonomy if delimited by ';'
phylum = sapply(strsplit(taxa, split = ";"), function(i) {
    i[2]
})

obj = aggTax(MRcounts(mouseData), lvl = phylum, out = "matrix")
head(obj[1:5, 1:5])

##                 PM1:20080107 PM1:20080108 PM1:20080114
##  Actinobacteria            0            3            2
##  Bacteroidetes           486          921         1103
##  Cyanobacteria             0            0            0
##  Firmicutes              455          922         1637
##  Proteobacteria           29           14           30
##                 PM1:20071211 PM1:20080121
##  Actinobacteria           37            0
##  Bacteroidetes           607          818
##  Cyanobacteria             0            0
##  Firmicutes              772         1254
##  Proteobacteria           38           23
```

The `aggregateByTaxonomy` and `aggTax` functions are flexible enough to put in either 1) a matrix with a vector of labels or 2) a MRexperiment object with a vector of labels or featureData column name. The function can also output either a matrix or MRexperiment object.

# 6 Visualization of features

metagenomeSeq has several plotting functions to visualize and gain insight into the overall structural composition of the data. Heatmaps of feature counts: plotMRheatmap. Basic feature correlation structures: plotCorr. PCA/MDS coordinates of samples or features: plotOrd. And rarefaction effects: plotRare.

Other plotting functions include plotting the abundance differences for a single feature, plotOTU, plotFeature or multiple features plotGenus. Plotting multiple OTUs with similar annotations allows for additional control of false discoveries.

## 6.1 Structural overview

Many studies begin by comparing the abundance composition across sample or feature phenotypes. Often a first step of data analysis is a heatmap, correlation or co-occurence plot or some other data exploratory method. The following functions have been implemented to provide a first step overview of the data:

1. plotMRheatmap - heatmap of abundance estimates (Fig. 4 right)
2. plotCorr - heatmap of pairwise correlations (Fig. 4 left)
3. plotOrd - PCA/CMDS components (Fig. 5 left)
4. plotRare - rarefaction effect (Fig. 5 right)

Each of the above can include phenotypic information in helping to explore the data.

Below we show an example of how to create a heatmap and hierarchical clustering of $\log_2$ transformed counts for the 200 OTUs with the largest overall variance. Red values indicate counts close to zero. Row color labels indicate OTU taxonomic class; column color labels indicate diet (green = high fat, yellow = low fat). Notice the samples cluster by diet in these cases and there are obvious clusters. We then plot a correlation matrix for the same features.

```
data(mouseData)
trials = pData(mouseData)$diet
heatmapColColors = brewer.pal(12, "Set3")[as.integer(factor(trials))]
heatmapCols = colorRampPalette(brewer.pal(9, "RdBu"))(50)

# plotMRheatmap
plotMRheatmap(obj = mouseData, n = 200, cexRow = 0.4, cexCol = 0.4,
    trace = "none", col = heatmapCols, ColSideColors = heatmapColColors)

# plotCorr
plotCorr(obj = mouseData, n = 200, cexRow = 0.25, cexCol = 0.25, trace = "none",
    dendrogram = "none", col = heatmapCols)
```

Below is an example of plotting CMDS plots of the data and the rarefaction effect at the OTU level. None of the data is removed (we recommend removing outliers typically).

```
data(mouseData)
cl = factor(pData(mouseData)$diet)

# plotOrd - can load vegan and set distfun = vegdist and use
# dist.method='bray'
```

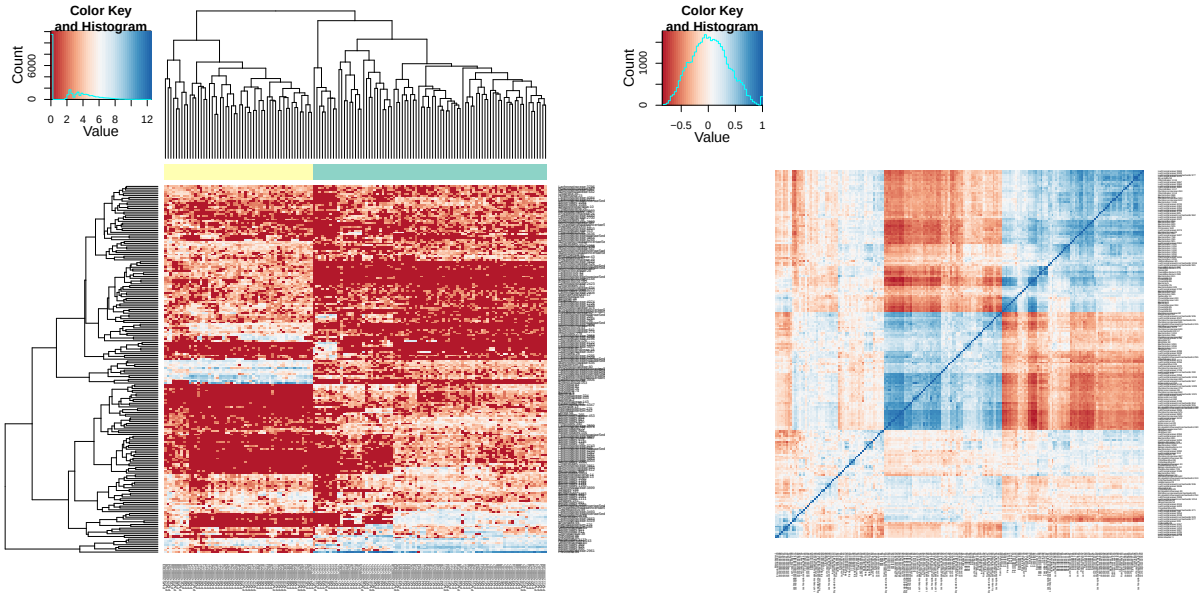


Figure 4: Left) Abundance heatmap (plotMRheatmap). Right) Correlation heatmap (plotCorr).

```
plotOrd(mouseData, tran = TRUE, usePCA = FALSE, useDist = TRUE, bg = cl,
    pch = 21)

# plotRare
res = plotRare(mouseData, cl = cl, ret = TRUE, pch = 21, bg = cl)

# Linear fits for plotRare / legend
tmp = lapply(levels(cl), function(lv) lm(res[, "ident"] ~ res[, "libSize"] -
    1, subset = cl == lv))
for (i in 1:length(levels(cl))) {
    abline(tmp[[i]], col = i)
}
legend("topleft", c("Diet 1", "Diet 2"), text.col = c(1, 2), box.col = NA)
```

### 6.2 Feature specific

Reads clustered with high similarity represent functional or taxonomic units. However, it is possible that reads from the same organism get clustered into multiple OTUs. Following differential abundance analysis. It is important to confirm differential abundance. One way to limit false positives is ensure that the feature is actually abundant (enough positive samples). Another way is to plot the abundances of features similarly annotated.

1. `plotOTU` - abundances of a particular feature by group (Fig. 6 left)
2. `plotGenus` - abundances for several features similarly annotated by group (Fig. 6 right)
3. `plotFeature` - abundances of a particular feature by group (similar to plotOTU, Fig. 7)

Below we use `plotOTU` to plot the normalized log(cpt) of a specific OTU annotated as *Neisseria meningitidis*, in particular the 779th row of lungTrim's count matrix. Using `plotGenus` we plot the normalized log(cpt) of all OTUs annotated as *Neisseria meningitidis*.

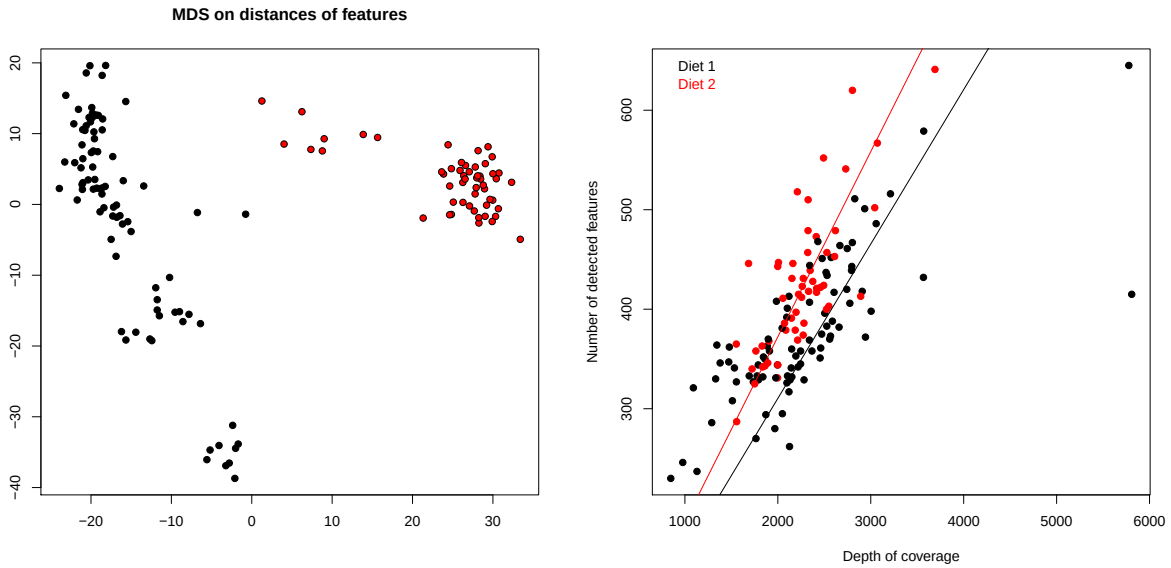


Figure 5: Left) CMDS of features (plotOrd). Right) Rarefaction effect (plotRare).

It would appear that *Neisseria meningitidis* is differentially more abundant in nonsmokers.

```
head(MRtable(fit, coef = 2, taxa = 1:length(fData(lungTrim)$taxa)))

##     +samples in group 1 +samples in group 0 counts in group 1
## 63                    6                  24                11
## 779                   7                  23                22
## 910                   3                   8                 4
## 358                   1                  24                 1
## 499                   2                  21                 2
## 25                   26                  15              1893
##     counts in group 0 smokingStatusSmoker    pValue adjPvalue
## 63               1538              -4.115 2.555e-15 1.143e-13
## 779              1512              -3.982 2.278e-14 7.562e-13
## 910               172              -2.977 8.361e-31 1.721e-28
## 358               390              -2.887 4.666e-12 1.117e-10
## 499               326              -2.688 9.442e-10 1.117e-08
## 25                162               2.661 8.370e-09 7.489e-08

patients = sapply(strsplit(rownames(pData(lungTrim)), split = "_"),
    function(i) {
        i[3]
    })
pData(lungTrim)$patients = patients
classIndex = list(smoker = which(pData(lungTrim)$SmokingStatus ==
    "Smoker"))
classIndex$nonsmoker = which(pData(lungTrim)$SmokingStatus == "NonSmoker")
otu = 779

# plotOTU
plotOTU(lungTrim, otu = otu, classIndex, main = "Neisseria meningitidis")
```

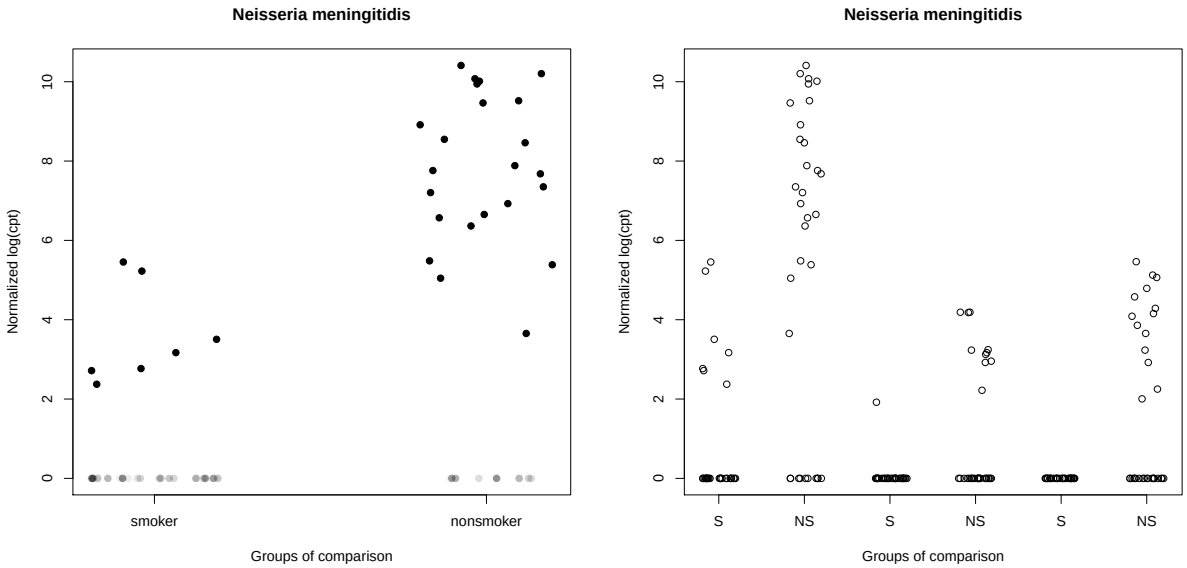


Figure 6: Left) Abundance plot (plotOTU). Right) Multiple OTU abundances (plotGenus).

```
# Now multiple OTUs annotated similarly
x = fData(lungTrim)$taxa[otu]
otulist = grep(x, fData(lungTrim)$taxa)

# plotGenus
plotGenus(lungTrim, otulist, classIndex, labs = FALSE, main = "Neisseria meningi

lablist <- c("S", "NS")
axis(1, at = seq(1, 6, by = 1), labels = rep(lablist, times = 3))
```

```
classIndex = list(Western = which(pData(mouseData)$diet == "Western"))
classIndex$BK = which(pData(mouseData)$diet == "BK")
otuIndex = 8770

# par(mfrow=c(1,2))
dates = pData(mouseData)$date
plotFeature(mouseData, norm = FALSE, log = FALSE, otuIndex, classIndex,
    col = dates, sortby = dates, ylab = "Raw reads")
```

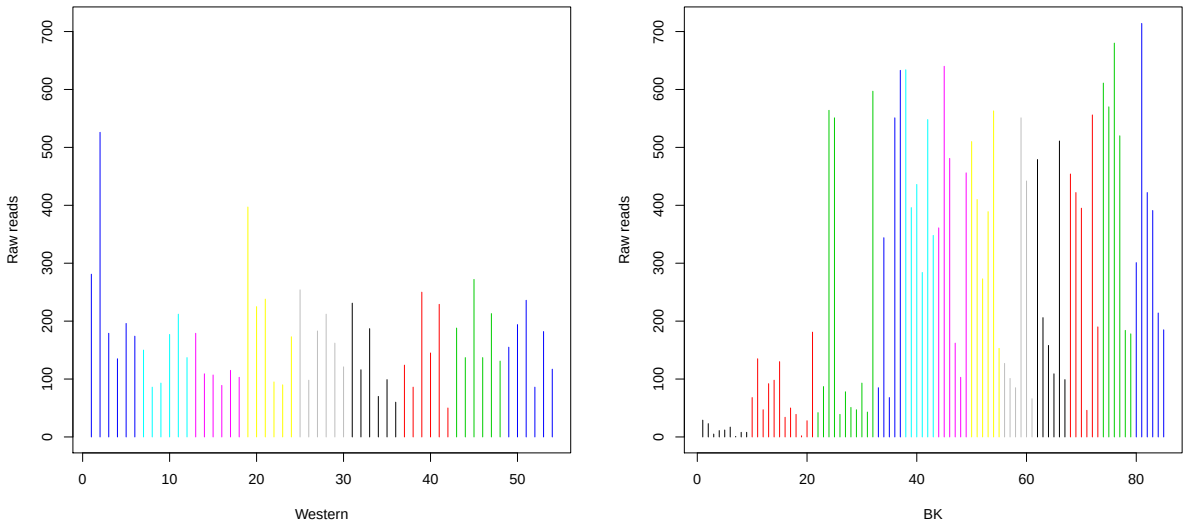


Figure 7: Plot of raw abundances

# 7 Summary

metagenomeSeq is specifically designed for sparse high-throughput sequencing experiments that addresses the analysis of differential abundance for marker gene survey data. The package, while designed for marker-gene survey datasets, may be appropriate for other sparse data sets for which the zero-inflated Gaussian mixture model may apply. If you make use of the statistical method please cite our paper. If you made use of the manual/software, please cite the manual/software!

## 7.1 Citing metagenomeSeq

```
citation("metagenomeSeq")

##
## Please cite the top for the original statistical method
## and normalization method implemented in metagenomeSeq and
## the bottom for the software/vignette guide.
##
##   JN Paulson, OC Stine, HC Bravo, M Pop.  Differential
##   abundance analysis for microbial marker-gene surveys.
##   Nat Meth Accepted
##
##   JN Paulson, M Pop, HC Bravo. metagenomeSeq: Statistical
##   analysis for sparse high-throughput sequncing.
##   Bioconductor package: 1.6.0.
##   http://cbcb.umd.edu/software/metagenomeSeq
```

## 7.2 Session Info

```
sessionInfo()

## R version 3.1.0 RC (2014-04-02 r65358)
## Platform: x86_64-apple-darwin10.8.0 (64-bit)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets
## [7] methods   base
##
## other attached packages:
## [1] metagenomeSeq_1.6.0 gplots_2.13.0       RColorBrewer_1.0-5
## [4] matrixStats_0.8.14  limma_3.20.0        Biobase_2.24.0
## [7] BiocGenerics_0.10.0 knitr_1.5
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-6       caTools_1.16       evaluate_0.5.3
##  [4] formatR_0.10       gdata_2.13.3       gtools_3.3.1
##  [7] highr_0.3          KernSmooth_2.23-12 R.methodsS3_1.6.1
## [10] stringr_0.6.2      tools_3.1.0
```

# 8 Appendix

## 8.1 Appendix A: MRexperiment internals

The S4 class system in R allows for object oriented definitions. `metagenomeSeq` makes use of the `Biobase` package in Bioconductor and their virtual-class, `eSet`. Building off of `eSet`, the main S4 class in `metagenomeSeq` is termed `MRexperiment`. `MRexperiment` is a simple extension of `eSet`, adding a single slot, `expSummary`.

The experiment summary slot is a data frame that includes the depth of coverage and the normalization factors for each sample. Future datasets can be formated as MRexperiment objects and analyzed with relative ease. A `MRexperiment` object is created by calling `newMRexperiment`, passing the counts, phenotype and feature data as parameters.

We do not include normalization factors or library size in the currently available slot specified for the sample specific phenotype data. All matrices are organized in the `assayData` slot. All phenotype data (disease status, age, etc.) is stored in `phenoData` and feature data (OTUs, taxanomic assignment to varying levels, etc.) in `featureData`. Additional slots are available for reproducibility and annotation.

## 8.2 Appendix B: Mathematical model

Defining the class comparison of interest as $k(j) = I\{j \in \text{group}A\}$. The zero-inflated model is defined for the continuity-corrected $\log_2$ of the count data $y_{ij} = \log_2(c_{ij} + 1)$ as a mixture of a point mass at zero $I_{\{0\}}(y_{ij})$ and a count distribution $f_{count}(y_{ij}; \mu_i, \sigma_i^2) \sim N(\mu_i, \sigma_i^2)$. Given mixture parameters $\pi_j$, we have that the density of the zero-inflated Gaussian distribution for feature $i$, in sample $j$ with $S_j$ total counts is:

$$f_{zig}(y_{ij}; \theta) = \pi_j(S_j) \cdot I_{\{0\}}(y_{ij}) + (1 - \pi_j(S_j)) \cdot f_{count}(y_{ij}; \theta) \quad (1)$$

Maximum-likelihood estimates are approximated using an EM algorithm, where we treat mixture membership $\Delta_{ij} = 1$ if $y_{ij}$ is generated from the zero point mass as latent indicator variables [5]. We make use of an EM algorithm to account for the linear relationship between sparsity and depth of coverage. The user can specify within the `fitZig` function a non-default zero model that accounts for more than simply the depth of coverage (e.g. country, age, any metadata associated with sparsity, etc.). See Figure 7 for the graphical model.

More information will be included later. For now, please see the online methods in:
http://www.nature.com/nmeth/journal/vaop/ncurrent/full/nmeth.2658.html

## 8.3 Appendix C: Calculating the proper percentile

To be included: an overview of the two methods implemented for the data driven percentile calculation and more description below.

The choice of the appropriate quantile given is crucial for ensuring that the normalization approach does not introduce normalization-related artifacts in the data. At a high level, the count distribution of samples should all be roughly equivalent and independent of each other up to this quantile under the assumption that, at this range, counts are derived from a common distribution.

More information will be included later. For now, please see the online methods in:
http://www.nature.com/nmeth/journal/vaop/ncurrent/full/nmeth.2658.html

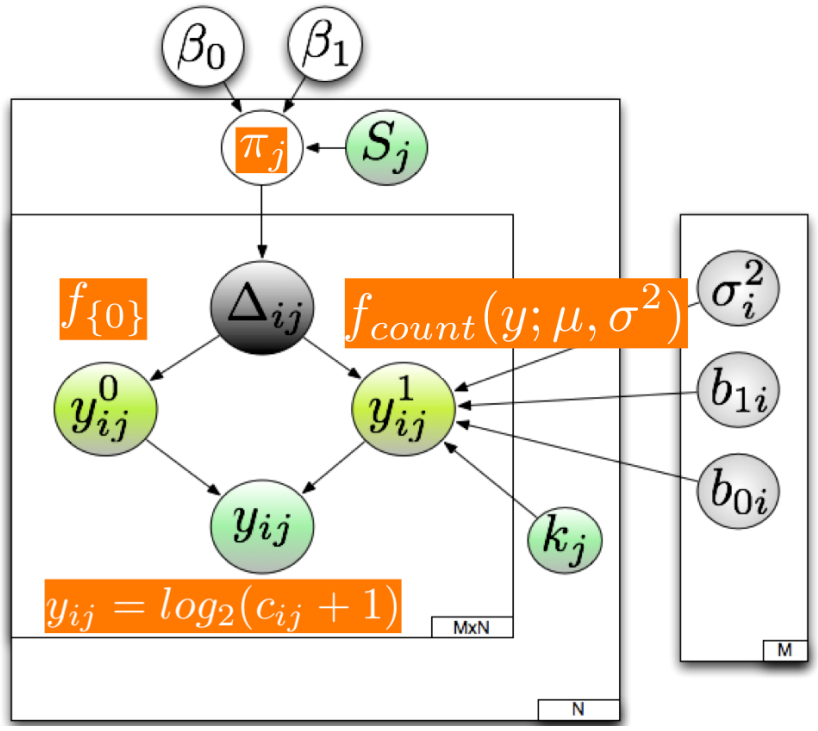


Figure 8: Graphical model. Green nodes represent observed variables: $S_j$ is the total number of reads in sample $j$; $k_j$ the case-control status of sample $j$; and $y_{ij}$ the logged normalized counts for feature $i$ in sample $j$. Yellow nodes represent counts obtained from each mixture component: counts come from either a spike-mass at zero, $y_{ij}^0$, or the "count" distribution, $y_{ij}^1$. Grey nodes $b_{0i}$, $b_{1i}$ and $\sigma_i^2$ represent the estimated overall mean, fold-change and variance of the count distribution component for feature $i$. $\pi_j$, is the mixture proportion for sample $j$ which depends on sequencing depth via a linear model defined by parameters $\beta_0$ and $\beta_1$. The expected value of latent indicator variables $\Delta_{ij}$ give the posterior probability of a count being generated from a spike-mass at zero, i.e. $y_{ij}^0$. We assume $M$ features and $N$ samples.

## References


[1] Emily S Charlson, Kyle Bittinger, Andrew R Haas, Ayannah S Fitzgerald, Ian Frank, Anjana Yadav, Frederic D Bushman, and Ronald G Collman. Topographical continuity of bacterial populations in the healthy human respiratory tract. *American Journal of Respiratory and Critical Care Medicine*, 184, 2011.

[2] Peter J Turnbaugh, Vanessa K Ridaura, Jeremiah J Faith, Federico E Rey, Rob Knight, and Jeffrey I Gordon. The effect of diet on the human gut microbiome: a metagenomic analysis in humanized gnotobiotic mice. *Science translational medicine*, 1(6):6ra14, 2009.

[3] Consortium HMP. A framework for human microbiome research. *Nature*, 486(7402), 2012.

[4] Gordon K Smyth. *Limma: linear models for microarray data.* Number October. Springer, 2005.

[5] A P Dempster, N M Laird, and D B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Series B Methodological*, 39(1):1–38, 1977.