Analyzing microbial gene survey data with `metaR`

Joseph Nathaniel Paulson

Center for Bioinformatics and Computational Biology
University of Maryland, College Park

jpaulson@umiacs.umd.edu

February 27, 2013

Contents

1 Introduction

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 phenotype 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.

Metastats and Lefse are two statistical methods addressing differential abundance detection in clinical metagenomic datasets. White *et al.*'s Metastats used a non-parametric permutation test on *t*-statistics and Segata *et al.*'s Lefse used 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. We present a R package, `metaR` (metagenomic analysis in R), that implements methods developed to account for previously unaddressed biases specific to high-throughput sequencing marker-gene survey data.

This vignette is meant to describe the software package `metaR`. A normalization method is implemented to control for biases in measurements across taxanomic features. We use a mixture model that implements a zero-inflated Gaussian distribution to account for varying depths of coverage. 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 under-sampling of microbial communities on disease association detection and testing of feature correlations.

2 Data structure

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.

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

2.1 Loading count and meta data

Following preprocessing and annotation of sequencing data the easiest way to format the count matrix is to have features (be it OTU, species, genus, etc.) along rows and samples along the columns. `metaR` includes functions for loading delimited files of counts `load_meta` and phenodata `load_phenoData`.

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

```
R> library(metaR)
R> dataDirectory <- system.file("doc", package="metaR")
R> lung = load_meta(file.path(dataDirectory,"CHK_NAME.otus.count.csv"))
R> dim(lung$counts)

[1] 1000   78
```

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.

```
R> taxa = read.csv(file.path(dataDirectory,"CHK_otus.taxonomy.csv"),
                   sep="\t",header=T,stringsAsFactors=F)[,2]
R> otu  = read.csv(file.path(dataDirectory,"CHK_otus.taxonomy.csv"),
                   sep="\t",header=T,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. Phenotype data can be optionally loaded into `R` with `load_phenoData`. This function loads data as a list. It is important to properly order data.

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

2.2 Creating a `MRexperiment` object

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

```
R> phenotypeData = as(clin,"AnnotatedDataFrame")
R> 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 taxa 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.

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

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

Links to a paper providing further details can be included optionally.

```
R> counts = lung$counts
R> obj = newMRexperiment(counts,phenoData=phenotypeData,featureData=OTUdata)
R> experimentData(obj) = annotate::pmid2MIAME("21680950")
R> 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)'
  pubMedIds: 21680950
Annotation:
```

3 Normalization

Normalization is required due to variable depths of coverage. We have implemented in `cumNorm` a normalization method that calculates normalization factors automatically calculated as the sum of counts less than the percentile that the majority of sample counts deviate from a reference. 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)`.

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 1 is an example for the longitudinal gnotobiotic mouse dataset.

```
R> data(lungData)
R> p=cumNormStat(lungData)
```

To calculate the scaling factors we simply run `cumNorm`

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

```
[1] TRUE
```

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

3.2 Exporting data

Functions are provided to easily retrieve or save normalized count matrices or basic sample characteristic statistics.

```
R> mat = MRcounts(lungData,norm=TRUE)[1:5,1:5]
R> exportMat(mat,output=file.path(dataDirectory,"temp.tsv"))

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

```
                              Subject
1 CHK_6467_E3B11_BRONCH2_PREWASH_V1V2
2               CHK_6467_E3B11_OW_V1V2
3               CHK_6467_E3B08_OW_V1V2
4            CHK_6467_E3B07_BAL_A_V1V2
5            CHK_6467_E3B11_BAL_A_V1V2
  Scaling.factor Quantile.value
1             36              4
2           2681              1
3           2390              1
4            897              1
```

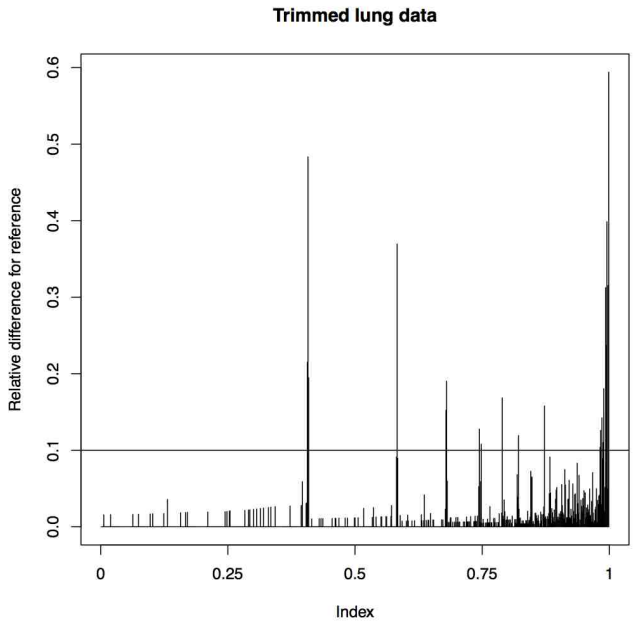

Figure 1: Relative difference for the median difference in counts from the reference. Samples came from the lung dataset (and all were used except the extraction.controls `pData(lungData)$SampleType`).

```
5            802              2
  Number.of.features
1                 60
2               3299
3               2994
4               1188
5               1098
```

4 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 sparsity ubiquitous in marker-gene survey datasets currently available.

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.

4.1 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 π_j, we have that the density of the zero-inflated Gaussian distribution for feature i, in sample j with s_j total counts is:

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

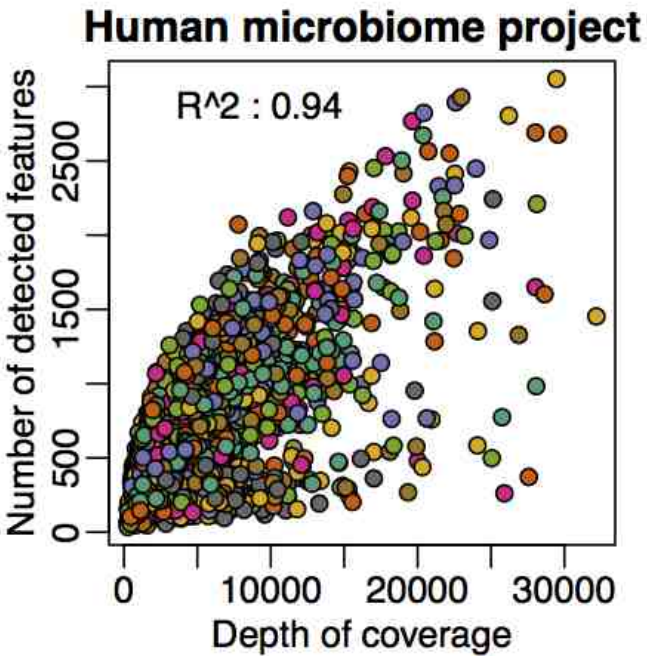

Figure 2: The number of unique features is plotted against depth of coverage for samples from the Human Microbiome Project**hmp**. Including the depth of coverage and the interaction of body site and sequencing site we are able to acheive an adjusted R^2 of .94. The zero-inflated Gaussian mixture was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

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.

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`. `MRfisher` tests assumptions different from the difference in abundance of a feature. Instead, it tests the odds of a feature being present or absent between two groups.

4.2 Running the statistical analysis

In our analysis of the lung microbiome data, we can remove features that are not present in many samples and samples that are controls and calculate the normalization factors. Following the calculation of our normalization factors the user needs to smartly decide/include pertinent phenotype parameters in a model matrix.

```
R> data(lungData)
R> k = grep("Extraction.Control",pData(lungData)$SampleType)
R> lungTrim = lungData[,-k]
R> k = which(rowSums(MRcounts(lungTrim)>0)<10)
```

```
R> lungTrim = lungTrim[-k,]
R> cumNorm(lungTrim)

[1] TRUE
```

Following the preparation, the user must define a model matrix for the data. There are optional inputs to `fitZig`, including the settings found in `zigControl` and 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.

```
R> smokingStatus = pData(lungTrim)$SmokingStatus
R> bodySite = pData(lungTrim)$SampleType
R> mod = model.matrix(~smokingStatus+bodySite)
R> settings = zigControl(maxit=10,verbose=TRUE)
R> fit = fitZig(obj = lungTrim,mod=mod,control=settings)

it= 0, nll=88.55, log10(eps+1)=Inf, stillActive=1029
it= 1, nll=93.10, log10(eps+1)=0.07, stillActive=299
it= 2, nll=93.24, log10(eps+1)=0.05, stillActive=130
it= 3, nll=93.46, log10(eps+1)=0.07, stillActive=31
it= 4, nll=93.50, log10(eps+1)=0.05, stillActive=16
it= 5, nll=93.50, log10(eps+1)=0.02, stillActive=10
it= 6, nll=93.57, log10(eps+1)=0.01, stillActive=5
it= 7, nll=93.57, log10(eps+1)=0.00, stillActive=2
it= 8, nll=93.57, log10(eps+1)=0.00, stillActive=1
it= 9, nll=93.57, log10(eps+1)=0.00, stillActive=0
```

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 coefficients represents the fold-change for our "smoker" vs. "nonsmoker" analysis. Often we're interested in in the fold change of various features with respect to the a condition. Koch's postulates demand an increase in presence and abundance of a particular bacteria. As mentioned above, Fisher's test is implemented in `MRfisher` as a test for presence / absence test. This tests different assumptions of the 16S count data.

Looking at the particular analysis just performed, there appears to be OTUs representing three *Prevotella*, two *Neisseria*, and a *Porphyromonas* that are differentially abundant.

It is important to check that similarly annotated OTUs are not equally differentially abundant in controls.

Currently functions are being developed to wrap and output results more neatly, but `MRcoefs` and `MRfit` can be used to view coefficient fits and related statistics.

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

                            smokingStatusSmoker
Neisseria polysaccharea:1             -4.130323
```

```
Neisseria meningitidis:1              -3.991910
Prevotella intermedia:2               -2.858724
Prevotella paludivivens:2              2.697783
Porphyromonas sp. UQD 414:2           -2.695710
Prevotella sp. DJF_B116:1              2.645772
                                  pValue
Neisseria polysaccharea:1   2.137891e-15
Neisseria meningitidis:1    2.055895e-14
Prevotella intermedia:2     9.189296e-12
Prevotella paludivivens:2   6.764197e-09
Porphyromonas sp. UQD 414:2 9.323449e-10
Prevotella sp. DJF_B116:1   1.296822e-10
                               adjPvalue
Neisseria polysaccharea:1   1.466593e-13
Neisseria meningitidis:1    1.113430e-12
Prevotella intermedia:2     2.781113e-10
Prevotella paludivivens:2   6.609208e-08
Porphyromonas sp. UQD 414:2 1.410857e-08
Prevotella sp. DJF_B116:1   2.696391e-09
```

5 Visualization of features

Reads clustered with high similarity do not represent functional or taxonomic units. It is possible that reads from the same organism get clustered into multiple OTUs. `metaR` has several plotting functions to visualize abundance differences for a single feature, `plotOTU`, and multiple features `plotGenus`. Plotting multiple OTUs with similar annotations allows for additional control of false discoveries. Other functions allow for an understanding of structural composition with heatmaps of feature counts `plotMRheatmap` and basic feature correlation structures `plotCorr`.

Many studies hope to compare biological compositions at a community level. Often a first step of data analysis is a heatmap or some other data exploratory tool, followed by a correlation or co-occurence heatmap.

```
R> data(mouseData)
R> trials = pData(mouseData)$diet
R> plotMRheatmap(obj=mouseData,n=200,trials=trials,
                 cexRow = 0.4,cexCol = 0.4,trace="none")
R> plotCorr(obj=mouseData,n=200,trials=trials,
                 cexRow = 0.25,cexCol = 0.25,trace="none",dendrogram="none")
```

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. `plotOTU` allows the user to plot the abundances of one feature separately for one group versus another.

```
R> #head(MRtable(fit,coef=2,taxa=1:length(fData(lungTrim)$taxa)))
R> patients=sapply(strsplit(rownames(pData(lungTrim)),split="_"),function(i){i[3
R> pData(lungTrim)$patients=patients
R> classIndex=list(controls=which(pData(lungTrim)$SmokingStatus=="Smoker"))
R> classIndex$cases=which(pData(lungTrim)$SmokingStatus=="NonSmoker")
```

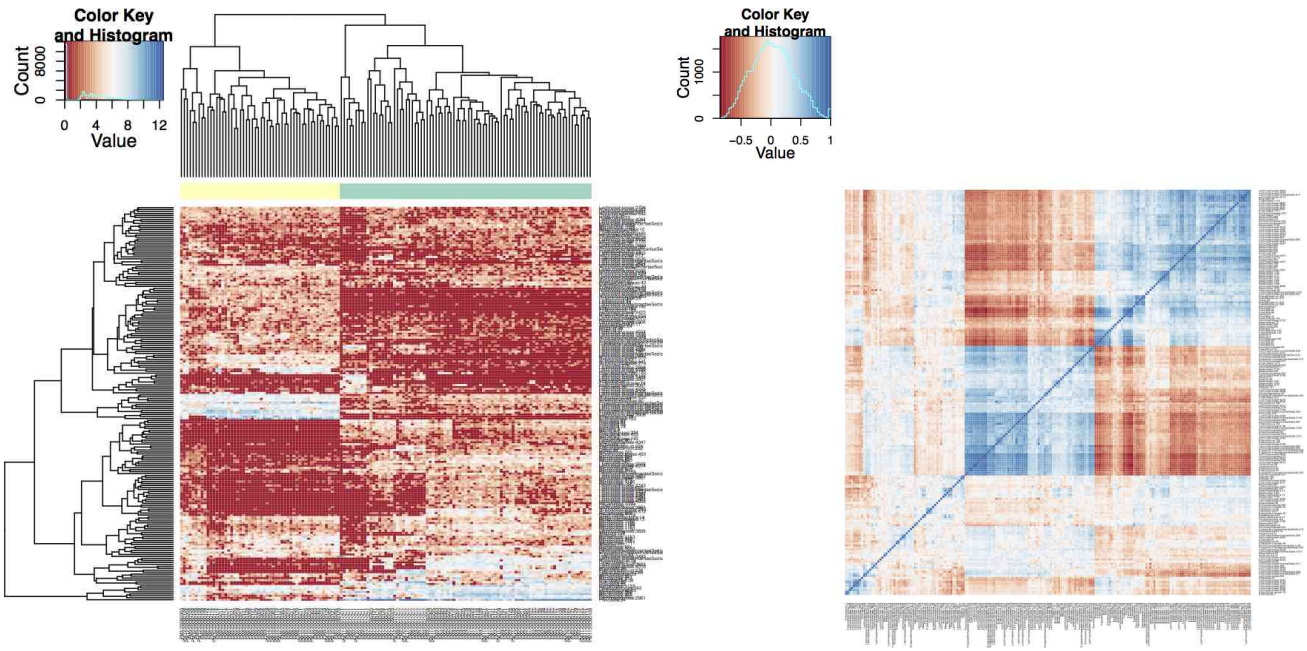

Figure 3: Left) Heatmaps and hierarchical clustering of log2 transformed counts for the 200 OTUs with the largest overall variance. Red values indicate counts close to zero. Row color labels indicate OTU taxonomic class; column color labels indicate diet (green = high fat, yellow = low fat). Notice the samples cluster by diet in these cases and there are obvious clusters. Right) Correlation matrix for the same features.

```
R> otu = 779
R> x = fData(lungTrim)$taxa[otu]
R> plotOTU(lungTrim,otu=otu,classIndex,xaxt="n",
         ylab="Normalized log(cpt)",main="Neisseria meningitidis")
R> lablist<- c("Smoker","NonSmoker")
R> axis(1, at=seq(1,2,by=1), labels = lablist)
```

`plotGenus` allows the user to plot abundances for several features similarly annotated for the various groups. At the OTU level reads are potentially clustered to multiple cluster centers that represent the same organism. It is possible that reads from one group get assigned to one OTU, and reads from another group are assigned to a different center that represents the same organism simply due to read similarity. Using `plotGenus` is one basic method to ensure that a feature is more likely to be differentially abundant.

```
R> otulist = grep(x,fData(lungTrim)$taxa)
R> plotGenus(lungTrim,otulist,classIndex,xaxt="n",
           ylab="Normalized log(cpt)",main="Neisseria meningitidis")
R> lablist<- c("S","NS")
R> axis(1, at=seq(1,6,by=1), labels = rep(lablist,times=3))
```

6 Summary

`metaR` is specifically designed for sparse high-throughput sequencing experiments that addresses the analysis of differential abundance for marker gene survey data. The package,

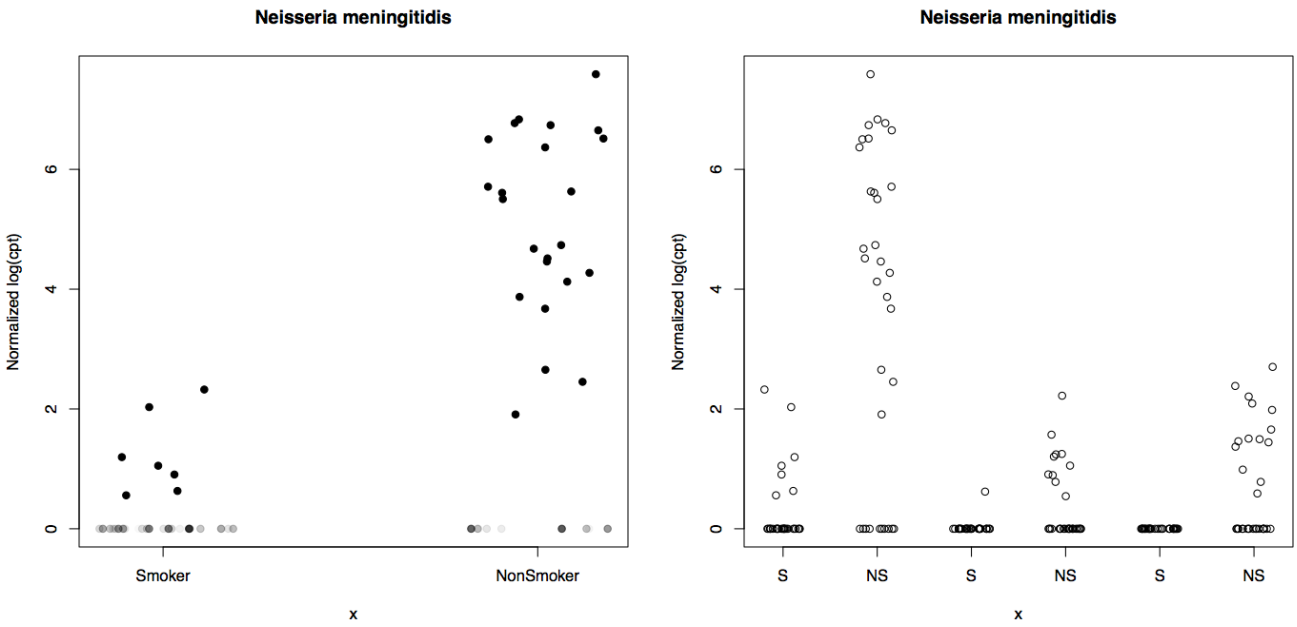

Figure 4: Left) Plot of the normalized log(cpt) of *Neisseria meningitidis*, in particular the 779th row of lungTrim's count matrix. Right) Plot of the normalized log(cpt) of all OTUs annotated as *Neisseria meningitidis*. According to this it would appear that *Neisseria meningitidis* is differentially abundant and more abundant in nonsmokers.

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.