Tutorial on Microbiome Normalization

Emily Palmer

2021-08-17

Contents

1 Introduction 5

1.1 The importance of normalization 5

1.2 The compositional nature of microbiome data 7

1.3 Zero-inflation of microbiome data 10

2 Importing Data 11

2.1 Global Patterns 11

2.2 Pre-procesing Quality Control and Filtering 12

3 Total Sum scaling (TSS) 15

3.1 About TSS 15

3.2 TSS Implementation 16

3.3 TSS on Global Patterns 16

4 Rarefying 19

4.1 Rarefying on Global Patterns 19

5 DESeq 25

5.1 About DESeq 25

5.2 DESeq Implementation 26

5.3 DESeq on Global Patterns 27

6 edgeR TMM 31

6.1 EdgeR TMM implementatin 31

6.2 TMM on Global Patterns 32

7 Cumulative sum scaling (CSS) 35

7.1 CSS on Global Patterns 36

8 GMPR 37

8.1 Global Patterns GMPR 38

9 Wrench 41

9.1 Wrench Implementation 41

9.2 Wrench on Global Patterns 42

10 Log-Ratio Approaches **43**

11 Comparisions **45**

Chapter 1

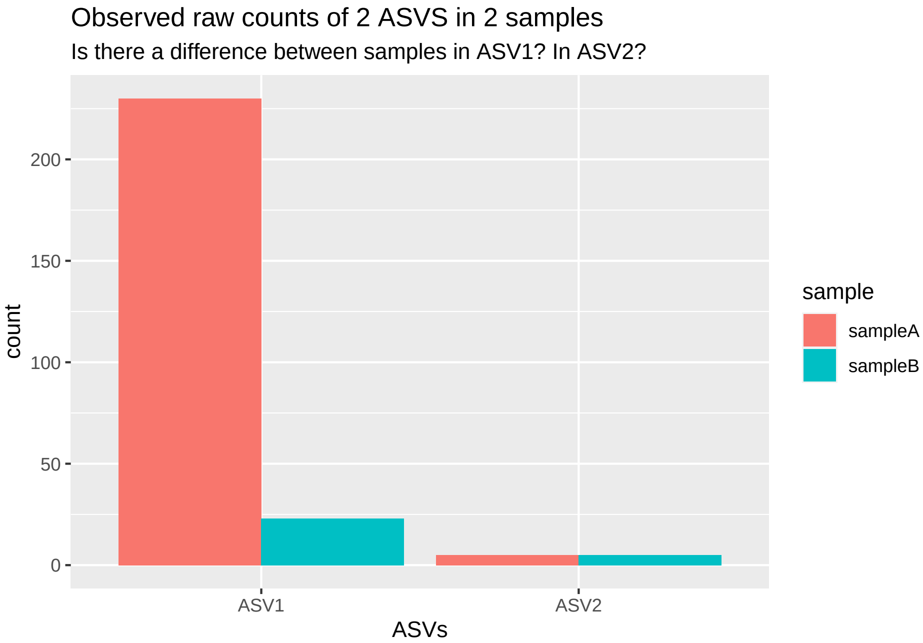
Introduction

1.1 The importance of normalization

Microbiome data must be normalized before any analysis can be performed. Following sequencing and assigning raw reads into counts per observed and classified identified taxa classes/OTUs/ASVs, the data is in the form of a matrix of counts of the observed abundance in each sample. Statistical analysis on this count matrix is then performed depending on the goal of the experiment. Common analysis goals include community-level analysis (alpha/beta diversity), differential abundance testing (the parallel of differential expression testing in gene expression studies), and network analysis.

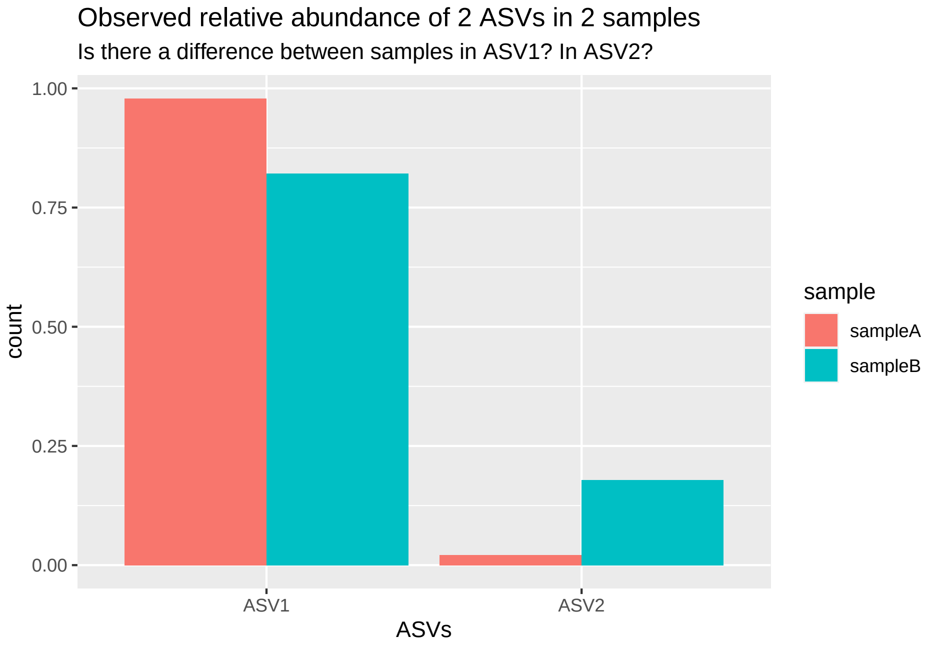
Ideally, any analysis of composition, differences, connections, etc. should be done based only on true biological aspects. However, technical variation in counts across samples is a given hurdle that must be accounted for. Biases can arise in the sequencing process, sample preparation, contamination, preferential amplification, and can manifest in differences in sparsity and unequal library sizes (Salter et al., 2014). An effective normalization strategy should put all samples on equal footing so interpretations are on biological signals, not technical signals. Currently, there is no known 'best' normalization method that removes all technical artifacts leaving only biological signals.

Due to the sequencing technology, samples will have different library sizes (also known as sequencing depth), so directly comparing counts between samples is not possible. To illustrate this, consider the counts of one taxon across two samples. In Sample A, this taxon has a count of 230, and in Sample B, this taxon has a count of 23. Is this taxon differentially abundant between samples?



Normalization in the context of microbiome data often refers to standardizing sequencing depth across samples. One common approach to this is a scaling-based approach, where a scaling factor is calculated for every sample and then the counts for each taxon in a given sample are divided by its scaling factor. Figure B shows the same data as figure A, but where each sample has been transformed into proportions by dividing by the total counts for each sample. The difference in `ASV1` between samples appears much smaller. However, now there appears to be a difference in `ASV2`, even though the counts were originally the same. This is because in sample B `ASV1` consists of a higher proportion of the total count than in sample A.

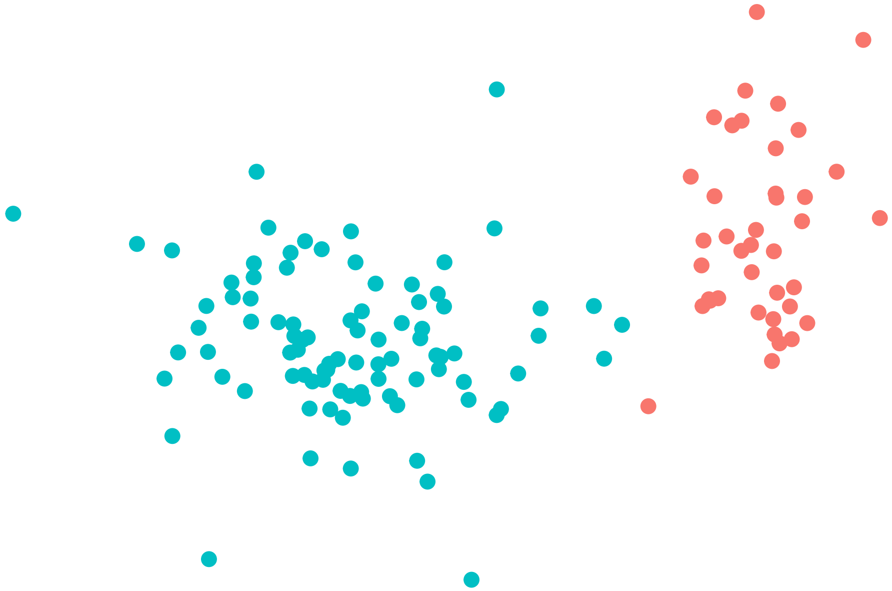
This demonstrates the importance of normalization, but also the artifacts that can occur depending on the method.



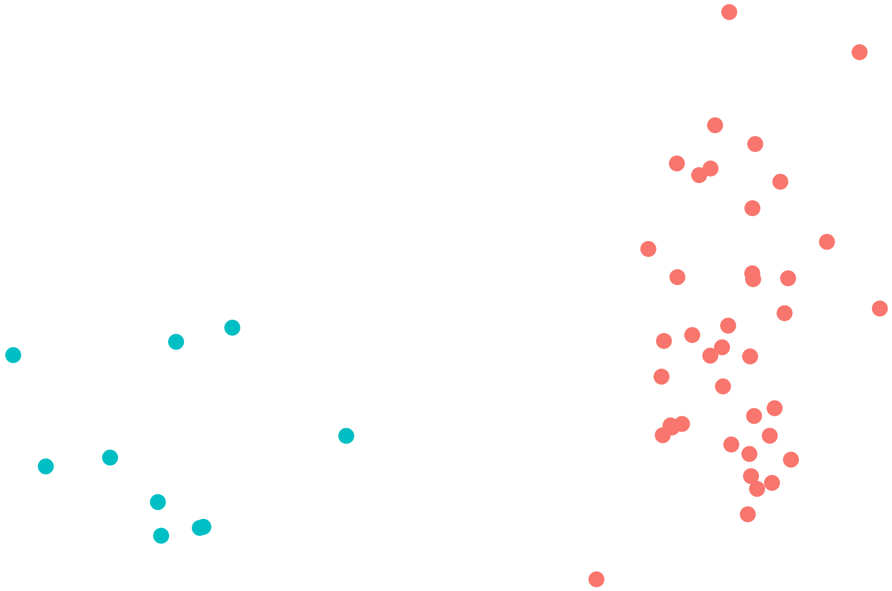
1.2 The compositional nature of microbiome data

Microbiome data are inherently compositional. This means that the count of each sampled taxa is a portion of a larger whole. Each observed taxa is not independent. As we saw in the above example, before normalization, `ASV2` was equal between samples. After converting to proportions, `ASV2` was no longer equal. With compositional data, each sample has a total sum constraint. Numerous traditional statistical methods rely on an independence assumption, which is not met with microbiome data. This can lead to spurious correlations that exist only because of the compositional nature and not any true signal.

With the library size as the sum constraint for each sample, if we know in a biological system that after an event occurs (treatment), ASV1 decreases, this will change the composition of the observed ASV2 regardless of what.



Population of two ASVs before treatment



Population of two ASVs after treatment

Consider again two samples consisting of red and blue points. We can think of the samples as before and after treatment. In the second plot, the number of blue dots has decreased, but the red remains the same. Consider the following potential observation from this population:

Sample	blue	red
Before	90	10
After	40	80

In the after sample, the blue counts have decreased, as we would expect from our knowledge of the population. However, the red counts have increased, even though there is no change in the red population. This observed increase in red counts is due to the compositional nature of the sampled points, and not any true difference in the red population.

1.2.1 Log ratio

The methodology developed by Aitchison in the 1980s is useful for analyzing compositional data. The theory developed there, however, is for data with different characteristics than microbiome data.

Microbiome data is high dimensional, and.

Taking the logarithm of ratios can be an appropriate transformation for compositional data, so standard statistical tests can be appropriate again. This transformation removes the issue of standardizing/normalizing different library sizes. The library size for a given sample will not distort the biological covariance or correlation structure.

This log-ratio method has a drawback, which is the decision of how to define the denominator. Two approaches to this problem can be to use one sample as the reference. This sample should be ‘representative’. The log-ratio transformation is then the ratio of every other taxon to that representative sample. Of course, knowledge of what makes a sample representative is hard to come by and often unknown, and subsequent results can be affected by this choice. This method is frequently called the additive log-ratio approach (alr). The alternative method is using the data to create a pseudo-reference sample. This pseudo-reference sample is the geometric mean of the counts of all taxa. This is called the centered log-ratio method.

While promising, these log-ratio methods have drawbacks in current metagenomic literature. Microbiome data are often incredibly sparse, with up to 80-90% of count matrices containing zero counts. For ratio transformations, if we have sparse data, the geometric mean can be zero. Then the ratio is undefined, and further, so is the log.

One solution to this is adding a small pseudo count to every element in the data. This removes problems occurring from having zero counts in the data, but there is not a clear best choice of what pseudo count to use, and it the choice can impact downstream results.

1.3 Zero-inflation of microbiome data

Include section on considerations that are necessary to account for zero-inflation.

Chapter 2

Importing Data

Luckily there are multiple publicly available pre-compiled microbiome data sets. These data sets begin after the bioinformatics pipeline and are matrices of counts of OTUs per sample.

These data sets can exist as `phyloseq` objects or as separate tables of counts and metadata.

2.1 Global Patterns

The Global Patterns dataset is a publicly available dataset in the `phyloseq` package. These data contain samples from 25 different environmental samples and mock communities. The sampling depth of these samples averages 3.1 million samples.

The following lines load the relevant packages and data.

```r
library(tidyverse)
library(phyloseq)
```

```
##
## Attaching package: 'phyloseq'

## The following object is masked from 'package:SummarizedExperiment':
##
##     distance

## The following object is masked from 'package:Biobase':
##
##     sampleNames

## The following object is masked from 'package:GenomicRanges':
```

```
##
##     distance
```

```
## The following object is masked from 'package:IRanges':
##
##     distance
```

```
data("GlobalPatterns")
# examine object
GlobalPatterns
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 19216 taxa and 26 samples ]
## sample_data() Sample Data:       [ 26 samples by 7 sample variables ]
## tax_table()   Taxonomy Table:    [ 19216 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 19216 tips and 19215 internal nodes ]
```

2.2 Pre-procesing Quality Control and Filtering

In addition to normalization, there are some steps we can perform that ideally remove technical artifacts from the sequencing process that only introduce noise.

These filtering steps commonly consist of filtering out samples with a low total read depth and filtering out taxa that are rarely abundant.

Let's create a filtered version of the Global Patterns dataset. Note that there are only 26 samples, and all have a large library size, so we will not filter out any samples here.

For taxa filtering, we will remove OTUs/ASVs that appear fewer than 5 times in more than half the samples.

```
# Determine which taxa to remove
filter_taxa <- genefilter_sample(GlobalPatterns,
                                  filterfun_sample(function(x) x > 5),
                                  A=0.5*nsamples(GlobalPatterns))
# Remove those taxa from the GlobalPatterns dataset
gp_raw <- prune_taxa(filter_taxa, GlobalPatterns)
gp_raw
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 219 taxa and 26 samples ]
## sample_data() Sample Data:       [ 26 samples by 7 sample variables ]
## tax_table()   Taxonomy Table:    [ 219 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 219 tips and 218 internal nodes ]
```

This decreases the number of taxa from 19216 to 219. This is not surprising, because this dataset contains samples from widely different locations (gut, soil, etc), and few taxa are shared among all samples and locations. One potential problem with this approach is the widely different locations, so it is possible that the remaining taxa could be some technical artifact, or could be a general ‘core’ set of taxa shared across the disparate environments.

Additionally, let us save the total sampling depth as the variable `depth` in the metadata for the GlobalPatterns dataset.

```
gp_raw@sam_data$depth <- sample_sums(gp_raw)
```

Chapter 3

Total Sum scaling (TSS)

3.1 About TSS

The first method described is Total Sum Scaling (TSS). This method is also referred to as Total Count (TC), converting into proportions, or relative abundance. This is a scaling method to normalize the different library sizes across samples. For every entry in the count matrix, we scale by the total read depth of that sample. This converts the counts into the proportion of abundance present in each given sample.

Though a more straightforward method, TSS normalization is not without its drawbacks. In microbiome data, it is common to have numerous low or zero-count observations, and that only a few most common OTUs contribute to the majority of the total sum of the library size. These high-count, frequent, taxa could be an artifact of the sequencing step, where high abundance observations are preferentially sampled. Using these large counts can dominate the scaling factor for each sample. As seen below, we see that the scaling factor for each sample is completely dominated by `ASV1`.

Sample	ASV1	ASV2	ASV3	ASV4	TSS Scaling Factor	Scaling factor w/o ASV1
Sample A	10314	34	8	12	10368	54
Sample B	824	23	13	20	880	56

Because this method does not account for the preferential sequencing over-abundance of `ASV1` it is possible to see an increase in false positives. However, this is a widely used method, and one of the few normalization methods that completely accounts for differing library sizes, which can be an important consideration depending on the analysis goal. Community level-analysis for example can be library-size dependent (ordination, some dissimiliarity measures).

3.2 TSS Implementation

Here, we provide a function that will normalize a `phyloseq` object by Total Sum Scaling. We have the option of keeping the result as proportions (having values 0-1), or transforming to an equal sequencing depth so the results are counts per million.

```
norm_TSS <- function(ps, keep_prop = F){
    # keep as proportions or convert to counts per million?
    scale <- ifelse(keep_prop, 1, 1e6)
    # TSS function
    ps_normed <- phyloseq::transform_sample_counts(ps, function(x) x * scale / sum(x))
    return(ps_normed)
}
```

3.3 TSS on Global Patterns

Now lets use this method on the Global Patterns data.

```
gp_tss <- norm_TSS(gp_raw)
# rename the depth as the scaling factor
sample_data(gp_tss)$scaling_factor <- sample_data(gp_tss)$depth
```

Now lets see the differences between the un-normalized, raw data, and the TSS transformed normalized data.

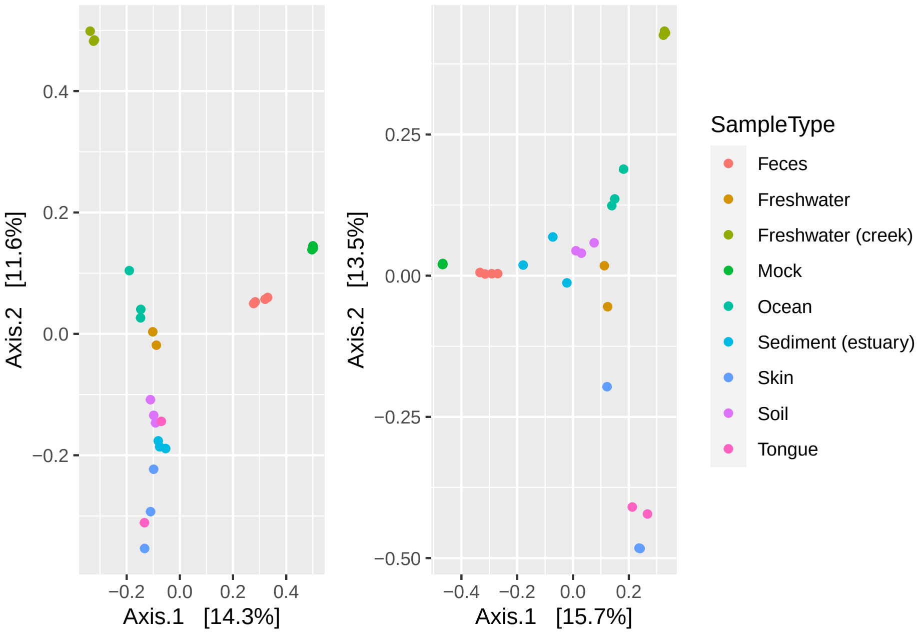
One possible way to do this is by looking at ordination plots. Microbiome data are high dimensional, so visualization directly of the data is difficult. Here, let us examine the principal coordinates plot using the Bray-Curtis dissimilarity.

First calculate the distance matrices, using the `phyloseq` function `ordinate()`

```
gp_raw_dist <- phyloseq::ordinate(gp_raw, "PCoA", "bray")
gp_tss_dist <- phyloseq::ordinate(gp_tss, "PCoA", "bray")
```

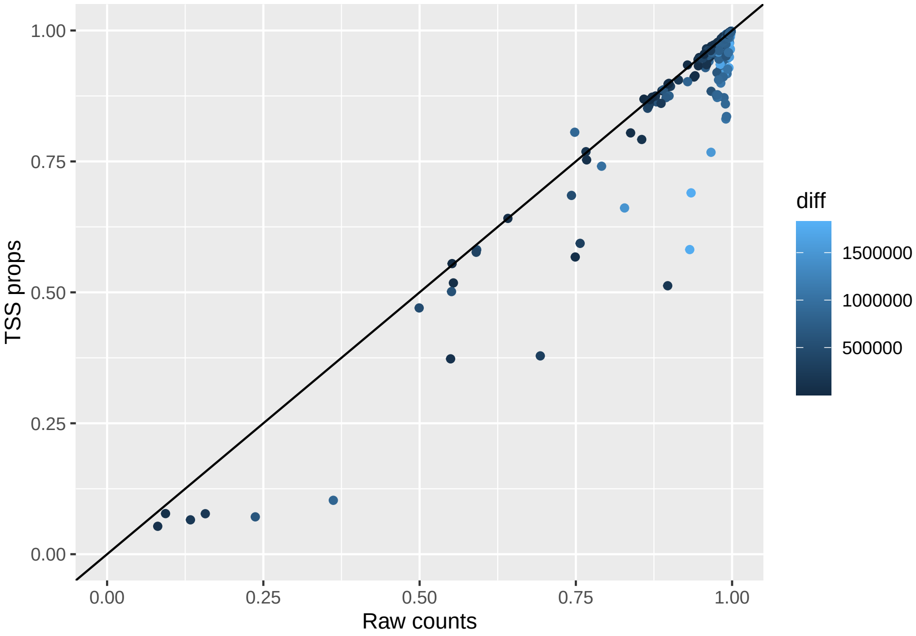
Now plot the two ordinations. Even before normalization, the different communities are clearly clustered.

```
plot_ordination(gp_raw, gp_raw_dist, color = "SampleType") +
plot_ordination(gp_tss, gp_tss_dist, color = "SampleType") +
  plot_layout(guides = 'collect')
```



We can also compare the values of the distance matrices before and after normalization to see how normalization is impacting different types of points.

```r
plot_norm_changes <- function(data_norm, data_raw, dist_method = "bray", x_lab = "raw", y_lab = "
    # calculate the Bray-Cutris distance matrix for the raw data, the normalized data,
    # and calculate the pairwise difference between the original library sizes between samples
    plot <- data.frame(raw = as.numeric(phyloseq::distance(data_raw, dist_method)),
                       norm = as.numeric(phyloseq::distance(data_norm, dist_method)),
                       diff = as.numeric(dist(get_variable(data_raw, "depth")))) %>%
    ggplot(aes(x = raw, y = norm, color = diff)) +
        geom_point() +
        geom_abline() +
        xlab(x_lab) + ylab(y_lab) + xlim(c(0,1)) + ylim(c(0,1))
    return(plot)
}

plot_norm_changes(gp_tss, gp_raw, x_lab = "Raw counts", y_lab = "TSS props")
```



Points below the line are pairs of samples that are marked as more similar after normalization. Points above the line are marked as more different after normalization. Values closer to 1 are 'more different'. Unsurprisingly, pairs that had larger original differences in sampling depth were marked as more different on the raw, unnormalized data, and became marked as more similar after TSS normalization.

Chapter 4

Rarefying

Another common technique that standardizes the library size across samples is to perform rarefying. Rarefying takes subsamples of the counts from the original sample of size of the new minimum library size.

One very common method is Rarefying, a method originally used in ecology. This method standardizes the read depth across all samples. To perform this method we first choose a minimum library size. Looking at rarefaction/collectors curves, or using a certain percentile can guide choosing this cutoff. Then all samples that have a read depth below this cutoff are discarded. Thus this method has a built-in filtering step. Next, we sample without replacement of the size of the chosen cutoff. It can be a standalone method or combined with other methods and transformations.

This is a very commonly used method, but it has also been criticized. First of all, it throws away valid data, and this results in a loss of power and an increase in false positives. Rare taxa can be removed in this approach too. It is however encouraged when we have widely different library sizes as it can lower the false discovery rate, and has also been shown to perform well in community-level analysis, as it completely standardizes the read count depth, and some methods are sensitive to differences in read count. Has been shown to separate by biological signal in ordination methods based on presence/absence

4.1 Rarefying on Global Patterns

The first difficulty is choosing a minimum sampling depth. The Global Patterns dataset already has a very high sampling depth for all samples, so we will chose the lowest as the minimum depth to rarefy to.

The following function returns a rarefied `phyloseq` object. Since we chose the minimum sampling depth, no samples have been dropped. In data sets where we have low sampling depth there is a balance between how many samples to drop and how low to set the minimum depth to.

```
norm_rarefy <- function(phyloseq, depth = min(sample_sums(phyloseq))){
    return(phyloseq::rarefy_even_depth(phyloseq,sample.size = depth))
}

gp_rare <- norm_rarefy(gp_raw)
```

```
## You set `rngseed` to FALSE. Make sure you've set & recorded
##  the random seed of your session for reproducibility.
## See `?set.seed`
```

```
## ...
```

We can check that indeed all samples now have the same sampling depth, which is 15905. Note that the highest sampling depth in this datset was almost 2 million, so we have discarded a lot of data to reduce to 15905.

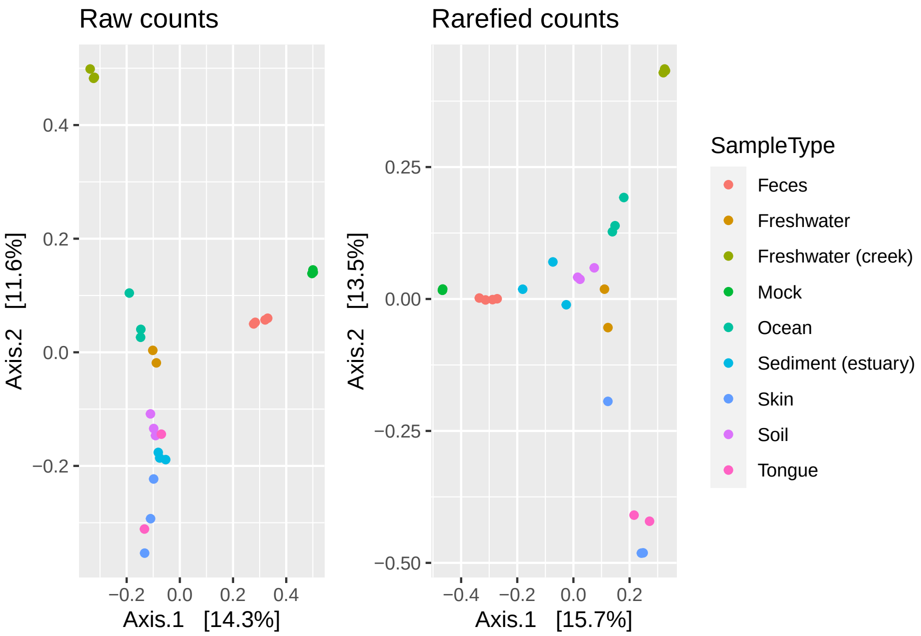
```
max(sample_sums(gp_raw))
```

```
## [1] 1842380
```

```
sample_sums(gp_rare)
```

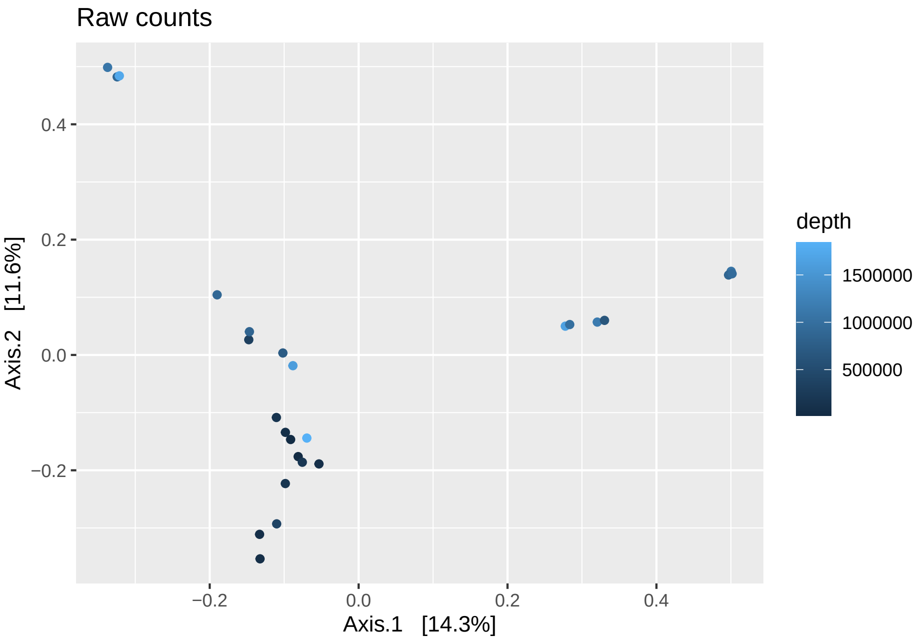
```
##      CL3      CC1      SV1  M31Fcsw  M11Fcsw  M31Plmr  M11Plmr  F21Plmr
##    15905    15905    15905    15905    15905    15905    15905    15905
##  M31Tong  M11Tong LMEpi24M SLEpi20M   AQC1cm   AQC4cm   AQC7cm      NP2
##    15905    15905    15905    15905    15905    15905    15905    15905
##      NP3      NP5  TRRsed1  TRRsed2  TRRsed3     TS28     TS29    Even1
##    15905    15905    15905    15905    15905    15905    15905    15905
##    Even2    Even3
##    15905    15905
```

```
plot_ordination(gp_raw,
                phyloseq::ordinate(gp_raw, "PCoA", "bray"),
                color = "SampleType",
                title = "Raw counts") +
plot_ordination(gp_rare,
                phyloseq::ordinate(gp_rare, "PCoA", "bray"),
                color = "SampleType",
                title = "Rarefied counts")+
  plot_layout(guides = 'collect')
```



Now see the comparison using the sampling depth as the color

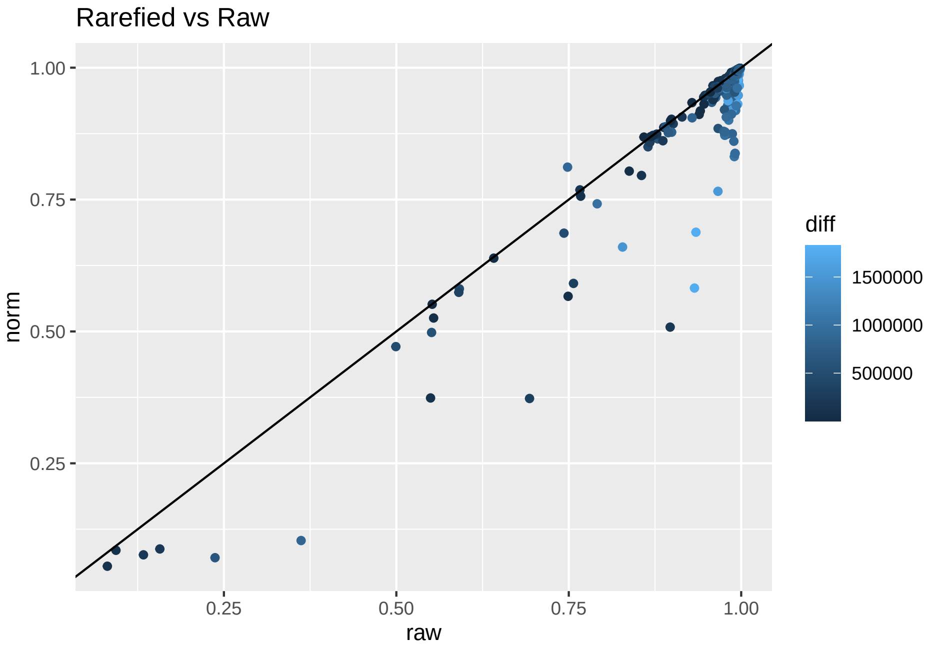
```r
plot_ordination(gp_raw,
                gp_raw_dist,
                color = "depth",
                title = "Raw counts")
```



We dont see any extreme patterns with sampling depth, but this might additionally be due to the differences in different locations might have different sampling depths. This comparison might be more interesting when we only have one location we are sampling from.

Now examine how the distance matrices change before/after normalization

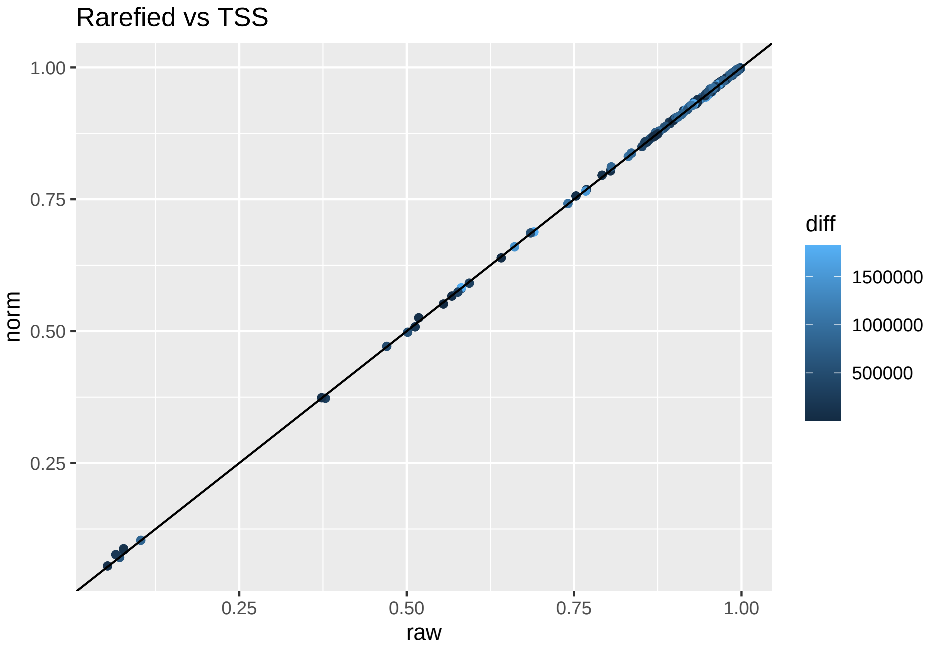
```
rare_samples <- sample_names(gp_rare)
gp_raw_match <- prune_samples(rare_samples, gp_raw)

data.frame(raw = as.numeric(phyloseq::distance(gp_raw_match, "bray")),
           norm = as.numeric(phyloseq::distance(gp_rare, "bray")),
           # Fix to be the saved sampling depth - needed for later
           diff = as.numeric(dist(get_variable(gp_raw_match, "depth")))) %>%
    ggplot(aes(x = raw, y = norm, color = diff)) +
        geom_point() +
        geom_abline() +
    ggtitle("Rarefied vs Raw")
```



```r
## Compare to tss
gp_tss_match <- norm_TSS(gp_raw_match)
length(as.numeric(dist(get_variable(gp_tss_match, "depth"))))
```

```
## [1] 325
```

```r
data.frame(raw = as.numeric(phyloseq::distance(gp_tss_match, "bray")),
           norm = as.numeric(phyloseq::distance(gp_rare, "bray")),
           # Fix to be the saved sampling depth - needed for later
           diff = as.numeric(dist(get_variable(gp_raw_match, "depth")))) %>%
    ggplot(aes(x = raw, y = norm, color = diff)) +
        geom_point() +
        geom_abline() +
    ggtitle("Rarefied vs TSS")
```



```
# Doesnt seem to be much difference between rarefied and tss
```

Chapter 5

DESeq

5.1 About DESeq

DESeq is a method for adjusting for differing library sizes. This method also can account for differences in library composition. This method has been called (MED) or (RLE) in the literature.

DESeq2 first takes the log (natural logarithm) of every entry in the count matrix. Due to this, all entries with zero will be set to negative infinity. Next, the row average is calculated (geometric average), so we have a vector of average counts for each taxon. Taking the log first should avoid undue influence by extreme outliers. All taxa with an average of infinity are removed. This step will remove all taxa with zero read count in one or more samples. This can be a problem in microbiome data. Next, we subtract the average log value from the log(counts), this gives a log ratio. This is equivalent to the ratio of the reads in each sample to the average across all samples. Next, we calculate the median of the log-ratios for each sample. These medians are converted to scaling factors for each sample by exponentiation. An extension of this method, denoted 'poscounts', has been suggested, which instead of taking the geometric mean of the logged counts for each taxon, we take the n-th root of the product of the non-zero counts.

This method assumes that the taxon of median absolute abundance is not differentially abundant, which is more likely true for the RNA-Seq it was developed for, but may not be true for microbiome studies, especially when there are more study groups, or we are analyzing higher taxonomic levels.

An additional option can be used to perform a variance stabilizing transformation on the count matrix before normalizing with the above size factors. This method calculates a dispersion-mean relationship and then transforms the data. The result ideally is an abundance matrix that is approximately homoskedastic or constant variance across the range of mean values. The package also includes an option for a ‘rlog’ transform, which they recommend over the variance stabilizing method in the case when there is a large difference in library sizes.

If differential abundance is of interest to calculate, DESeq uses a negative binomial distribution to model differential abundances. It is possible to provide the size factors calculated by another method to DESeq to perform differential analysis.

5.2 DESeq Implementation

```r
norm_DESeq_RLE_poscounts <- function(ps, group = 1){
  require(DESeq2, quietly = T)
    # keep arbitrary design for normalization
    # Convert to DESeq object
    ps_dds <- phyloseq_to_deseq2(ps, ~1)
    # Calculate the size factors (scaling)
    ps_dds <- estimateSizeFactors(ps_dds, type = "poscounts")
    # Extract counts
    counts <- DESeq2::counts(ps_dds, normalized = T)
    # Convert back to phyloseq
    otu <- otu_table(counts, taxa_are_rows = T)
    sam <- access(ps, "sam_data")
    sam$scaling_factor <- sizeFactors(ps_dds)
    tax <- access(ps, "tax_table")
    phy <- access(ps, "phy_tree")
    ps_DESeq <- phyloseq(otu,sam,tax,phy)
    return(ps_DESeq)
}


norm_DESeq_vs <- function(ps, group = 1){
  require(DESeq2, quietly = T)
  ps_dds <- phyloseq_to_deseq2(ps, ~ 1)
  ps_dds <- estimateSizeFactors(ps_dds, type = "poscounts")
  # Variance transformation
  ps_dds <- estimateDispersions(ps_dds)
  abund <- getVarianceStabilizedData(ps_dds)
  # don't allow deseq to return negative counts
  # add the minimum count to all values
  # another option is to replace negative counts with 0
  abund <- abund + abs(min(abund))
  otu <- otu_table(abund, taxa_are_rows = T)
  sam <- access(ps, "sam_data")
  tax <- access(ps, "tax_table")
  phy <- access(ps, "phy_tree")
  ps_DESeq <- phyloseq(otu,sam,tax,phy)
  return(ps_DESeq)
}
```

5.3 DESeq on Global Patterns

Perform DESeq RLE normalization as well as DESeq variance stabalised transformation.

```
gp_deseq_rle <- norm_DESeq_RLE_poscounts(gp_raw)
```

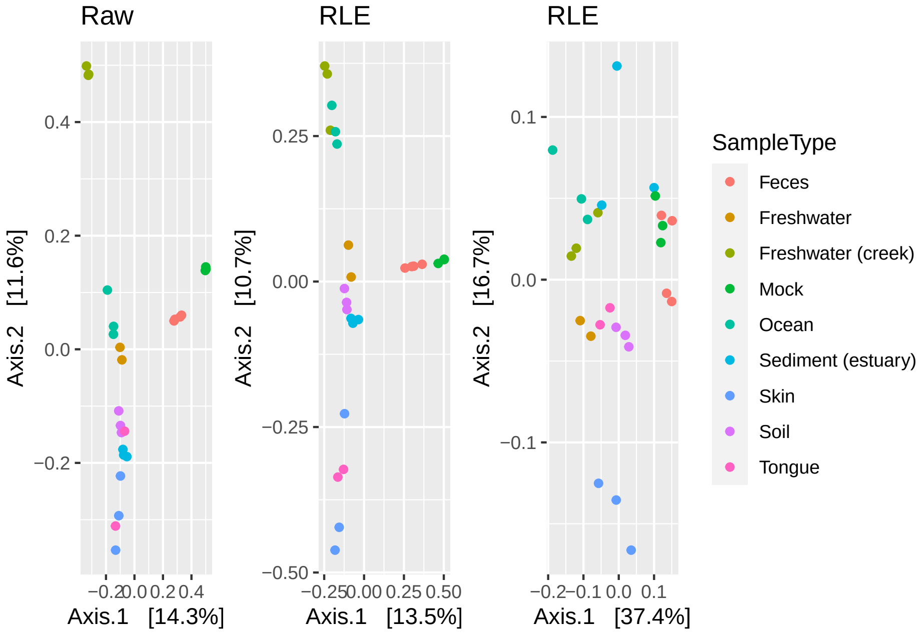
```
## converting counts to integer mode
```

```
gp_deseq_vs <- norm_DESeq_vs(gp_raw)
```

```
## converting counts to integer mode

## gene-wise dispersion estimates

## mean-dispersion relationship

## -- note: fitType='parametric', but the dispersion trend was not well captured by the
##    function: y = a/x + b, and a local regression fit was automatically substituted.
##    specify fitType='local' or 'mean' to avoid this message next time.

## final dispersion estimates
```

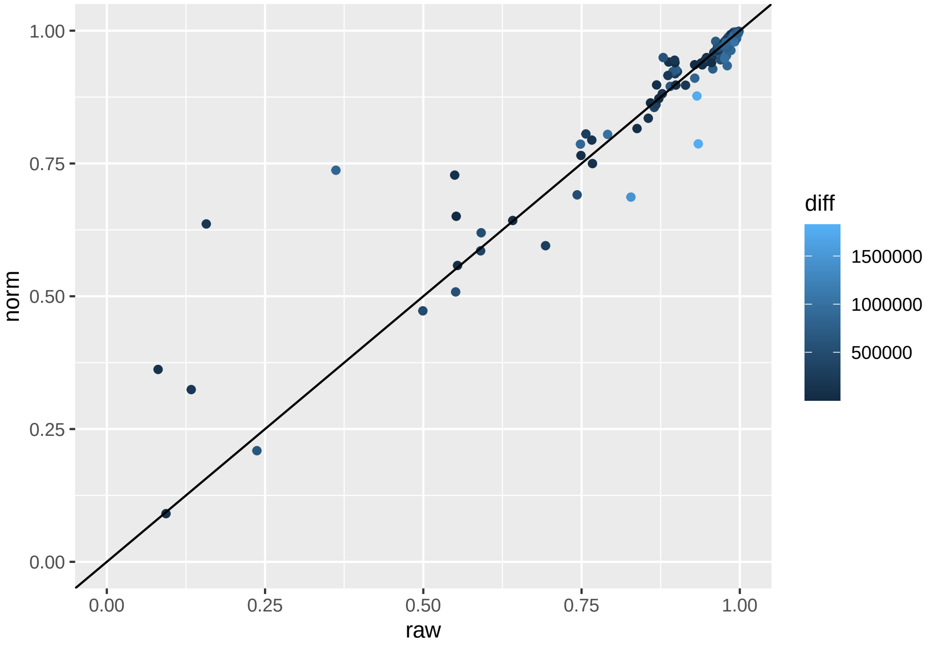
Examine principal coordinate plots between raw data and DESeq normalized data.

```
gp_raw_dist <- phyloseq::ordinate(gp_raw, "PCoA", "bray")
gp_rle_dist <- phyloseq::ordinate(gp_deseq_rle, "PCoA", "bray")
gp_vs_dist <- phyloseq::ordinate(gp_deseq_vs, "PCoA", "bray")

plot_ordination(gp_raw, gp_raw_dist, color = "SampleType", title = "Raw") +
plot_ordination(gp_deseq_rle, gp_rle_dist, color = "SampleType", title = "RLE") +
plot_ordination(gp_deseq_vs, gp_vs_dist, color = "SampleType", title = "RLE") +
  plot_layout(guides = 'collect')
```

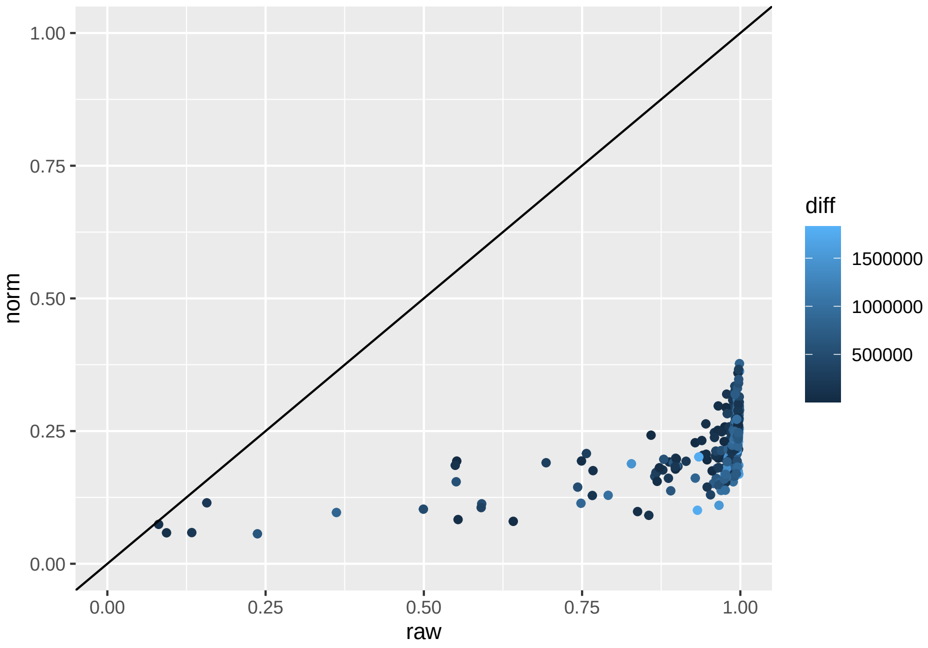


See how dissimilarity matrices differ from raw count dissimilarity matrices.

```r
plot_norm_changes(gp_deseq_rle, gp_raw)
```



```
plot_norm_changes(gp_deseq_vs, gp_raw)
```



Chapter 6

edgeR TMM

TMM (Trimmed median of m-values) is another method borrowed from RNA-Seq analysis. This method uses, or calculates a reference sample, and compares all other samples to the reference sample. The size factor is the mean of the log-ratios after excluding the highest count taxa and taxa with the largest fold change. As taxa with zero counts are excluded, a pseudocount is needed. Additionally, there is the TMMwsp option which is encouraged as it is more robust to zero counts. Positive counts are reused to increase the number of features when we compared. The singleton positive counts are paired up in decreasing order of size and then a modified TMM method is applied to the re-ordered libraries.

6.1 EdgeR TMM implementatin

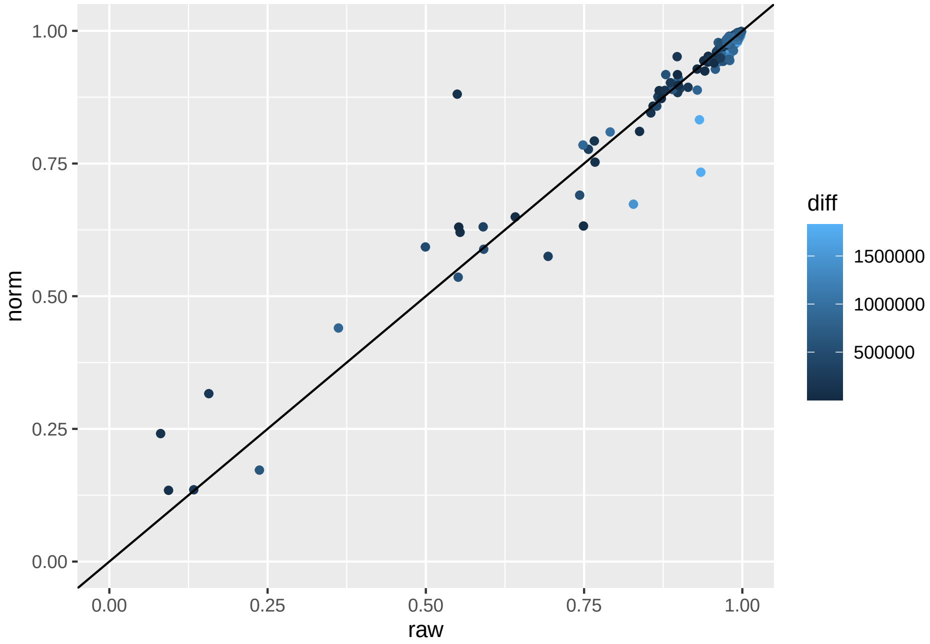
```
norm_TMM <- function(physeq, group = 1, method="TMM", pseudocount = 1, ...){
    require("edgeR", quietly = T)
    require("phyloseq", quietly = T)
    # Enforce orientation.
    if( !taxa_are_rows(physeq) ){ physeq <- t(physeq) }
    x = as(otu_table(physeq), "matrix")
    # Add one to protect against overflow, log(0) issues.
    x = x + pseudocount
    # Check `group` argument
    if( identical(all.equal(length(group), 1), TRUE) & nsamples(physeq) > 1 ){
        # Assume that group was a sample variable name (must be categorical)
        group = get_variable(physeq, group)
    }
    # Define gene annotations (`genes`) as tax_table
    taxonomy = tax_table(physeq, errorIfNULL=FALSE)
    if( !is.null(taxonomy) ){
        taxonomy = data.frame(as(taxonomy, "matrix"))
    }
    # Now turn into a DGEList
    y = DGEList(counts=x, group=group, genes=taxonomy, remove.zeros = TRUE)
    # Calculate the normalization factors
    d = edgeR::calcNormFactors(y, method=method)
    # Check for division by zero inside `calcNormFactors`
    if( !all(is.finite(d$samples$norm.factors)) ){
        stop("Something wrong with edgeR::calcNormFactors on this data,
         non-finite $norm.factors, consider changing `method` argument")
    }
    scalingFactor <- d$samples$norm.factors * d$samples$lib.size / 1e6
    dataNormalized <- t(t(otu_table(physeq)) / scalingFactor)
    #dataNormalized <- cpm(d)
    otu <- otu_table(dataNormalized, taxa_are_rows = T)
    sam <- access(physeq, "sam_data")
    sam$scaling_factor <- scalingFactor
    tax <- access(physeq, "tax_table")
    phy <- access(physeq, "phy_tree")
    ps_edgeR <- phyloseq(otu,sam,tax,phy)

    return(ps_edgeR)

}
```

6.2 TMM on Global Patterns

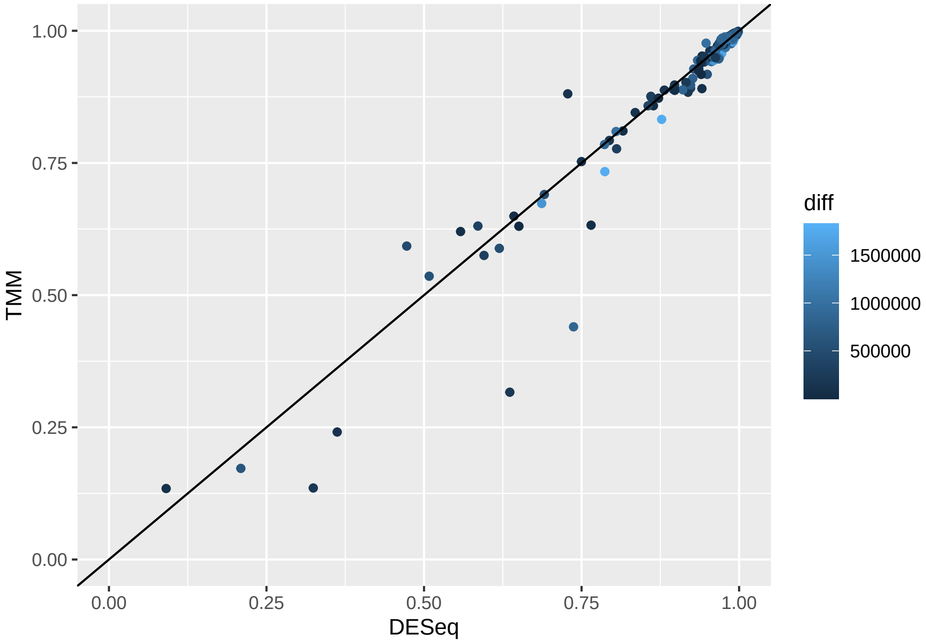
Perform normalization

```
# Use the normalization function
gp_tmm <- norm_TMM(gp_raw)
```

```
plot_norm_changes(gp_tmm, gp_raw)
```



```r
plot_norm_changes(gp_tmm, gp_deseq_rle, x_lab = "DESeq", y_lab = "TMM")
```



Chapter 7

Cumulative sum scaling (CSS)

Cumulative sum scaling is, another scaling method, developed for marker gene sequencing, It is intended to account for undersampling and correct biases from preferentially amplified features in a sample-specific manner (Paulson et al., 2013). This method assumes that count distributions should be roughly equivalent and independent to each other up to the given quantile which is chosen as the smallest value at which instability is found, so this is an extension to UQ scaling where a quantile is specified. If there is high count variability the assumption may not be met. This is not a method that accounts for compositionality. It initially showed improvements in separating samples biologically in ordination, it was shown to be an artifact of unequal application of log transformation across methods (Costea et al., 2014).

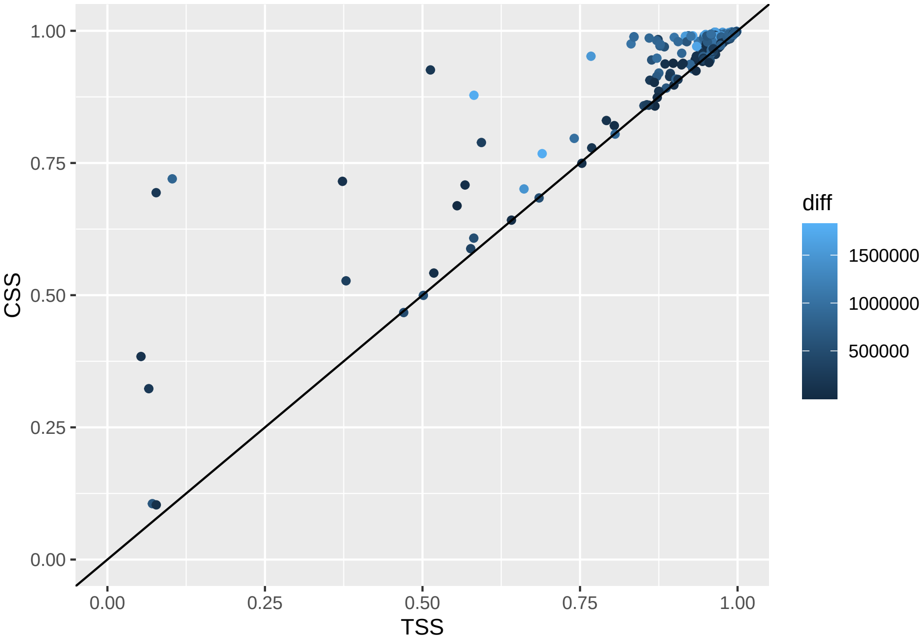
```r
norm_CSS <- function(ps){
  require(metagenomeSeq, quietly = T)
  ps.metaG<-phyloseq_to_metagenomeSeq(ps)
  p_stat = cumNormStatFast(ps.metaG)
  ps.metaG = cumNorm(ps.metaG, p = p_stat)
  ps.metaG.norm <- MRcounts(ps.metaG, norm = T)

  otu <- otu_table(ps.metaG.norm, taxa_are_rows = T)
  sam <- access(ps, "sam_data")
  sam$scaling_factor <- normFactors(ps.metaG)/1e6
  tax <- access(ps, "tax_table")
  phy <- access(ps, "phy_tree")
  ps_CSS <- phyloseq(otu,sam,tax,phy)
  return(ps_CSS)
}
```

7.1 CSS on Global Patterns

```
gp_css <- norm_CSS(gp_raw)
```

```
## Default value being used.
```

```
plot_norm_changes(gp_css, gp_tss,
                  x_lab = "TSS", y_lab = "CSS")
```



CSS normalization appears to consider pairs as more different than TSS normalization, and pairs with high sequencing depth differences even more so.

Chapter 8

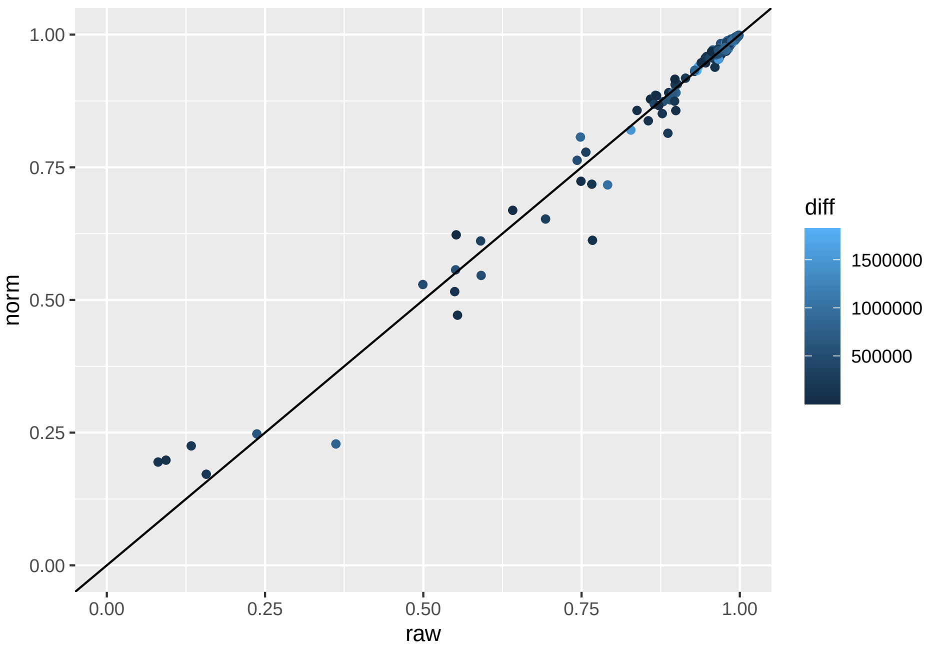
GMPR

A recent extension of the RLE DESeq method is the Geometric mean of Pairwise ratios (GMPR) approach. This method reverses the steps of RLE, and instead calculates the median count ratio of the non-zero counts between pairs of samples as although only a small number of taxa are likely to be shared for every sample, it is more likely that there are many shared taxa between pairs. Use pairwise results to calculate the size factor for each sample. This method has slow computation, but is robust to differential and outlier taxas. It addresses sparsity, but not composition. The size factors can be inputted to DESeq and a VST transformation applied additionally. It is a newer method, and has unfortunately not been included in many benchmarking studies, although initial results show it to be more powerful than DESeq, not surprisingly, as it uses more data, as zero counts do not need to be discarded. It assumes there is a large invariant portion of the count data, similar to other methods.

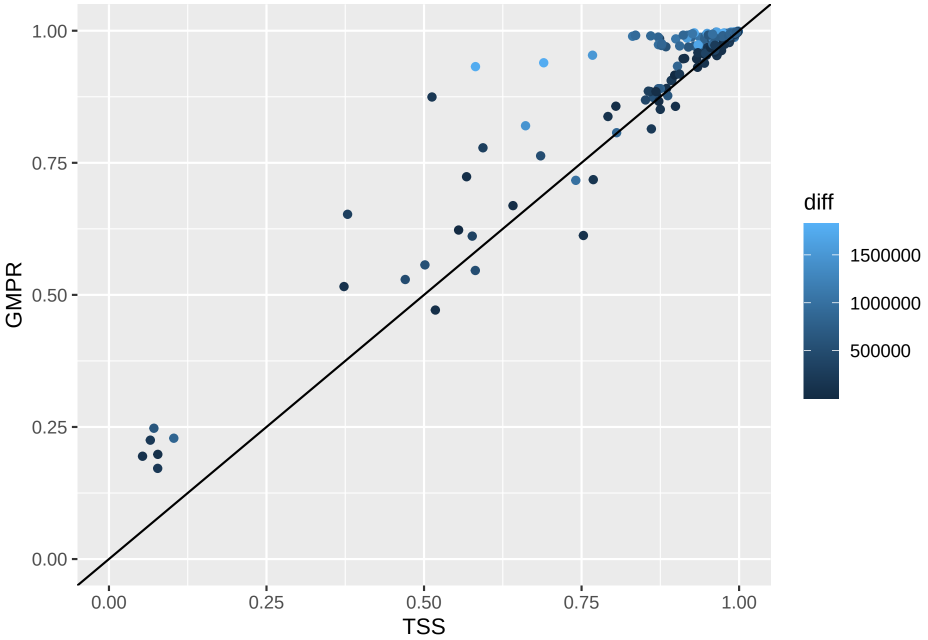
```
norm_GMPR <- function(ps, scale = 1e6){
  require(GMPR, quietly = T)
  #browser()
    # Calculate GMPR size factor
    # Row - features, column - samples
    otu <- as(otu_table(ps), "matrix")
    if(taxa_are_rows(ps)){otu <- t(otu)}
    otu_df = as.data.frame(otu)
    otu.tab <- matrix(otu, ncol = ncol(otu))
    gmpr.size.factor <- GMPR::GMPR(otu_df, intersect_no = 4, min_ct = 2)

    # normalize
    otu.tab.norm <- t(t(otu) / (gmpr.size.factor/scale))

    # convert back to PS
    sam <- access(ps, "sam_data")
    sam$scaling_factor <- gmpr.size.factor
    tax <- access(ps, "tax_table")
    phy <- access(ps, "phy_tree")
    ps_GMPR <- phyloseq(otu_table(otu.tab.norm, taxa_are_rows = F),sam,tax,phy)
    return(ps_GMPR)
}
```

8.1 Global Patterns GMPR

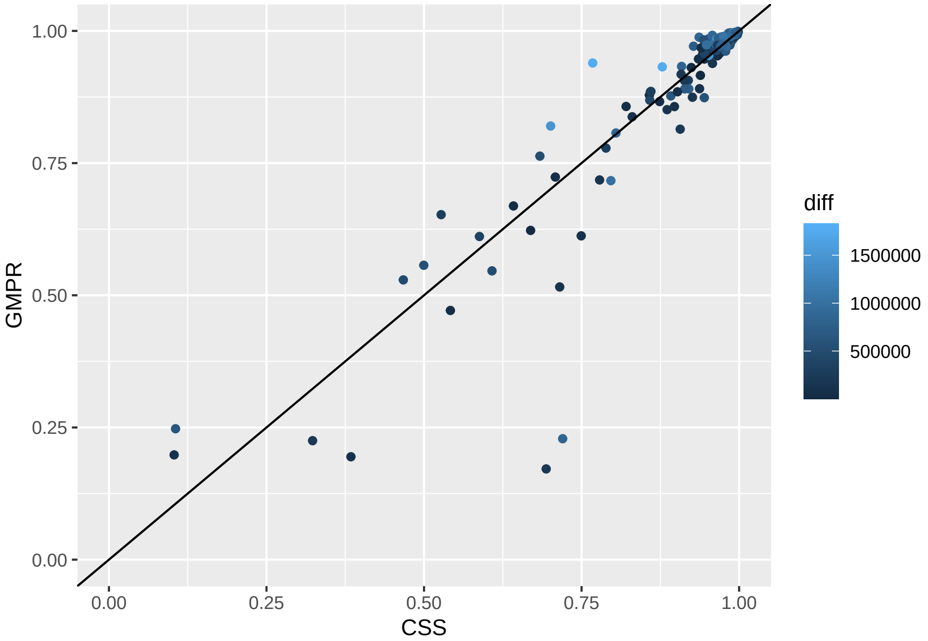
```
gp_gmpr <- norm_GMPR(gp_raw)
plot_norm_changes(gp_gmpr, gp_raw)
```



```
plot_norm_changes(gp_gmpr, gp_tss, x_lab = "TSS", y_lab = "GMPR")
```



```
plot_norm_changes(gp_gmpr, gp_css, x_lab = "CSS", y_lab = "GMPR")
```



Chapter 9

Wrench

Wrench is a recent normalization method developed for microbiome data. This method includes compositional bias correction for sparse datasets. This method uses a hurdle log-normal distribution to estimate the normalization factors (the location estimate for the group). For this method, we assume abundances are drawn from a hurdle Log-Gaussian distribution, and the scaling factor used is essentially the location estimate for the group.

9.1 Wrench Implementation

```
norm_wrench <- function(ps, group_col){
    require(Wrench, quietly = T)
    if( identical(all.equal(length(group_col), 1), TRUE) & nsamples(ps) > 1 ){
        # Assume that group was a sample variable name (must be categorical)
        group = get_variable(ps, group_col)
    }
    otu_tab <- otu_table(ps)
    W <- wrench(otu_tab, group)

    compositionalFactors <- W$ccf
    normalizationFactors <- W$nf

    normed_otu <- otu_tab/(normalizationFactors/1e6)
    otu <- otu_table(normed_otu, taxa_are_rows = T)
    sam <- access(ps, "sam_data")
    sam$scaling_factor <- normalizationFactors
    tax <- access(ps, "tax_table")
    phy <- access(ps, "phy_tree")
    ps_wrench <- phyloseq(otu,sam,tax,phy)

    return(ps_wrench)
}
```

9.2 Wrench on Global Patterns

```
gp_wrench <- norm_wrench(gp_raw, group_col = "SampleType")
```

Chapter 10

Log-Ratio Approaches

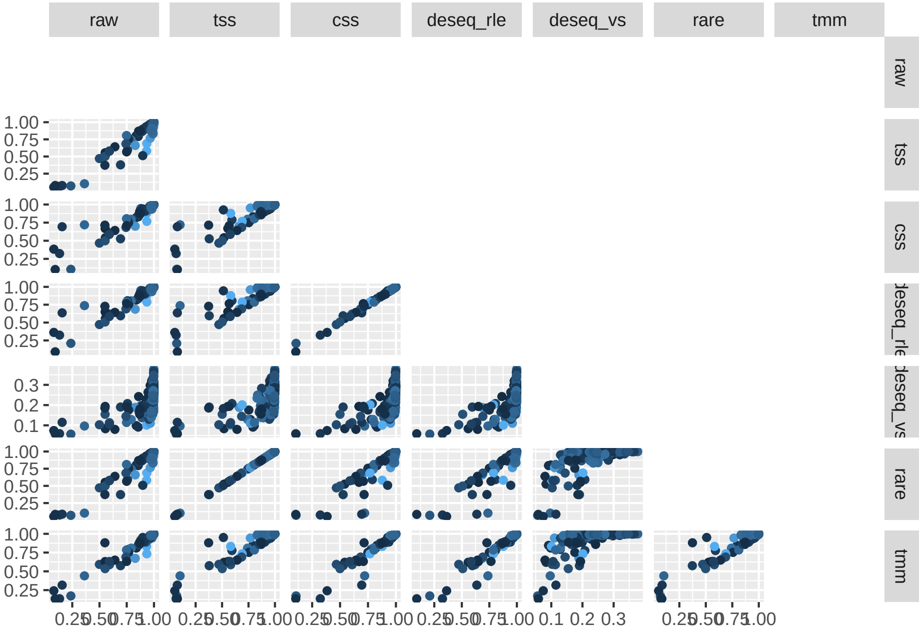
Chapter 11

Comparisions

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
data.frame(raw = as.numeric(phyloseq::distance(gp_raw, "bray")),
           tss = as.numeric(phyloseq::distance(gp_tss, "bray")),
           css = as.numeric(phyloseq::distance(gp_css, "bray")),
           deseq_rle = as.numeric(phyloseq::distance(gp_deseq_rle, "bray")),
           deseq_vs = as.numeric(phyloseq::distance(gp_deseq_vs, "bray")),
           rare = as.numeric(phyloseq::distance(gp_rare, "bray")),
           tmm = as.numeric(phyloseq::distance(gp_tmm, "bray")),
           diff = as.numeric(dist(get_variable(gp_raw, "depth")))) %>%
    ggpairs(columns = 1:7, upper = "blank",
            diag = "blank",ggplot2::aes(colour=diff))
```


raw
tss
css
deseq_rle
deseq_vs
rare
tmm
1.00
0.75
0.50
0.25
0.3
0.2
0.1
0.25
0.50
0.75
1.00
0.1
0.2
0.3

Bibliography

Costea, P., Zeller, G., Sunagawa, S., and Bork, P. (2014). A fair comparison. *Nature Methods*, 11(4):359–359. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 4 Primary_atype: Correspondence Publisher: Nature Publishing Group Subject_term: Data mining;Metagenomics;Statistical methods Subject_term_id: data-mining;metagenomics;statistical-methods.

Paulson, J., Stine, O., Bravo, H., and Pop, M. (2013). Differential abundance analysis for microbial marker-gene surveys. *Nature Methods*, 10(12):1200–1202. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 12 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Data mining;Metagenomics;Statistical methods Subject_term_id: data-mining;metagenomics;statistical-methods.

Salter, S., Cox, M., Turek, E., Calus, S., Cookson, W., Moffatt, M., Turner, P., Parkhill, J., Loman, N., and Walker, A. (2014). Reagent and laboratory contamination can critically impact sequence-based microbiome analyses. *BMC Biology*, 12(1):87.