# Tutorial on Normalization of Microbiome Data

# Contents

# Chapter 1

# Introduction

## 1.1 The importance of normalization

Microbiome data must be normalized before any statistical analysis can be performed. Following the process of sequencing and assigning raw reads into counts per observed and classified identified taxa classes/OTUs/ASVs, microbiome data consist of a matrix of read counts, referred to as a feature table of raw counts. Normalization is the process of transforming raw read count data into data that can be compared between samples. 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.

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 sequencing depths (Salter et al., 2014). An effective normalization strategy should put all samples on equal footing so interpretations are on biological signals, not technical signals such as sequencing depth. 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 sequencing depths, or the sum of all the counts in a sample. Directly comparing raw counts between samples is not possible. To illustrate this, consider the counts of Taxon 1, across two samples shown below. 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? As we see below, the way we normalize the data can change

how we would answer this question.

Table 1.1: The raw counts of taxon 1 are different, and the raw counts of taxon 2 are equal. Sampling depth is not accounted for.

| Taxa | Sample A | Sample B |
|---|---|---|
| Taxon 1 | 230 | 23 |
| Taxon 2 | 5 | 5 |

Figure A

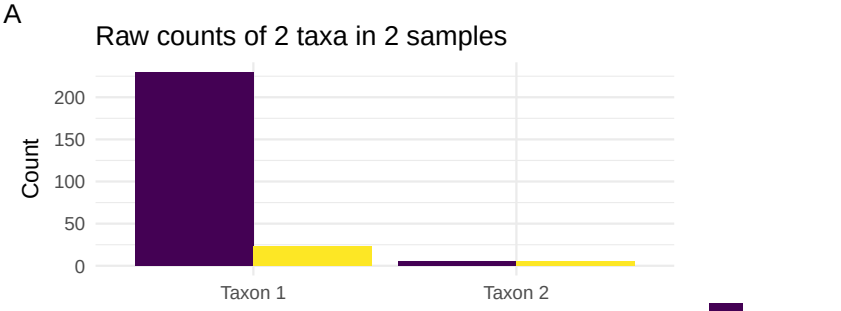
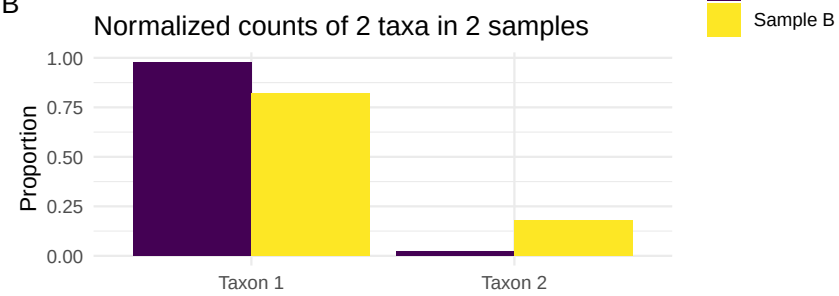



Figure B



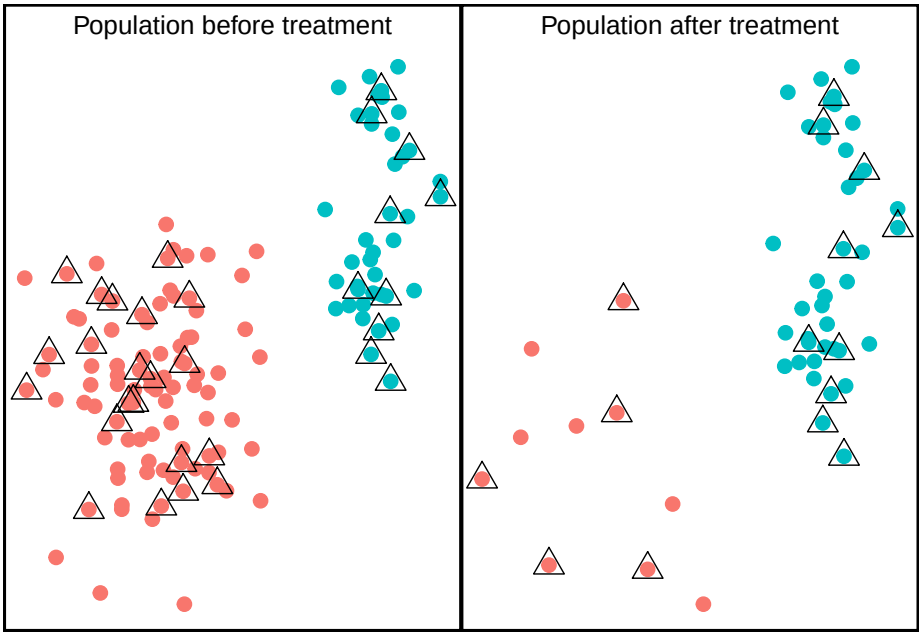

One goal for normalization of microbiome data is the standardization of sequencing depth across samples. One common approach to this is a scaling-based approach, where a scaling factor is calculated for every sample and the counts for each taxon are divided by the scaling factor for that sample. Figure 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 Taxon 1 between samples appears much smaller. However, now there appears to be a difference in Taxon 2 between samples, even though the counts were originally the same. This is because in sample B, Taxon 2 consists of a higher proportion of the total count than it does 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. The counts of the collection of taxa that make up each sample are constrained by the total sum, or sequencing depth for that sample. This means that the count of each sampled taxon is a portion of a larger whole. Each observed taxon is not independent. As we saw in the above example, before normalization, Taxon 2 was equal between samples. After converting to proportions, the values for Taxon 2 no longer appear equal between samples. If there is a difference between two samples it is unclear if that difference is because of a true difference in that taxon or if that taxon is changing because of differences in another taxon. 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 library size as the sum constraint for each sample, if we know in a biological system that after an event occurs (treatment), the red taxon decreases, this will change the composition of the sampled blue taxon regardless of its change or lack thereof in the underlying population.




Consider again two samples consisting of red and blue points shown above. The points encompassed by triangles represent the sampled points from the population. We can think of the samples as before and after treatment. In the second plot, the number of red dots in the population and in the observed

Table 1.2: Counts of sampled red and blue taxa before and after

| Sample | blue | red |
|---|---|---|
| Before | 10 | 20 |
| After | 10 | 4 |

Table 1.3: Proportions of sampled red and blue taxa before and after

| Sample | blue | red |
|---|---|---|
| Before | 0.29 | 0.67 |
| After | 0.71 | 0.33 |

sample has decreased, but the blue remains the same.

This observed increase in the proportion of blue is due to the compositional nature of the sampled points, and not any true difference in the blue population.

# Chapter 2

# Importing Data

There are multiple publicly available pre-compiled microbiome data sets of feature tables. 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, a popular R package for microbiome analysis (McMurdie and Holmes, 2013), or as separate tables of feature counts and metadata. For this tutorial, we will provide code that assumes a `phyloseq` object as input, and outputs a normalized `phyloseq` object. Converting between separate tables and `phyloseq` objects is straightforward using the `otu_table()`, `sample_data()`, `taxa_table()`, and `phyloseq()` functions. For more information on the `phyloseq` package, see https://joey711.github.io/phyloseq/. In this tutorial we will use two data sets that are included in the `phyloseq` package.

## 2.1 Global Patterns

The Global Patterns dataset (Caporaso et al., 2011) is a dataset available 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 total counts. We will use this dataset to work through the different normalization methods.

The following lines load the relevant packages and data.

```
library(tidyverse)
library(phyloseq)

data("GlobalPatterns")
# examine phyloseq 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 Colorectal cancer

Another publicly available dataset is from a study on colorectal cancer (Kostic et al., 2012). See the abstract of this study below:

> The tumor microenvironment of colorectal carcinoma is a complex community of genomically altered cancer cells, nonneoplastic cells, and a diverse collection of microorganisms. Each of these components may contribute to carcinogenesis; however, the role of the microbiota is the least well understood. We have characterized the composition of the microbiota in colorectal carcinoma using whole genome sequences from nine tumor/normal pairs. Fusobacterium sequences were enriched in carcinomas, confirmed by quantitative PCR and 16S rDNA sequence analysis of 95 carcinoma/normal DNA pairs, while the Bacteroidetes and Firmicutes phyla were depleted in tumors. Fusobacteria were also visualized within colorectal tumors using FISH. These findings reveal alterations in the colorectal cancer microbiota; however, the precise role of Fusobacteria in colorectal carcinoma pathogenesis requires further investigation.

This dataset is downloaded with the `phyloseq` package as a .biom file and can be loaded in using the following code, showing how to load qiime files as `phyloseq` objects. These data contain observations from 2505 taxa across 185 samples, which are categorized as 'Tumor' or 'Healthy'. We remove samples that are not categorized into these two categories.

```
# Load the data from a system file that is downloaded if the phyloseq package is insta
filepath <-  system.file("extdata", "study_1457_split_library_seqs_and_mapping.zip", pa
kostic <-  microbio_me_qiime(filepath)
```

```
## Found biom-format file, now parsing it...
## Done parsing biom...
## Importing Sample Metdadata from mapping file...
## Merging the imported objects...
## Successfully merged, phyloseq-class created.
##  Returning...
```

```
# Some samples are labeled as "none" as a dignosis; remove these samples.
# Save as an object with the un-normalized counts
k_raw <- subset_samples(kostic, DIAGNOSIS != "None")
```

```
k_raw
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 2505 taxa and 185 samples ]
## sample_data() Sample Data:       [ 185 samples by 71 sample variables ]
## tax_table()   Taxonomy Table:    [ 2505 taxa by 7 taxonomic ranks ]
```

## 2.3 Pre-processing 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 taxa 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
# Save as an object with the un-normalized counts
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 ]
```

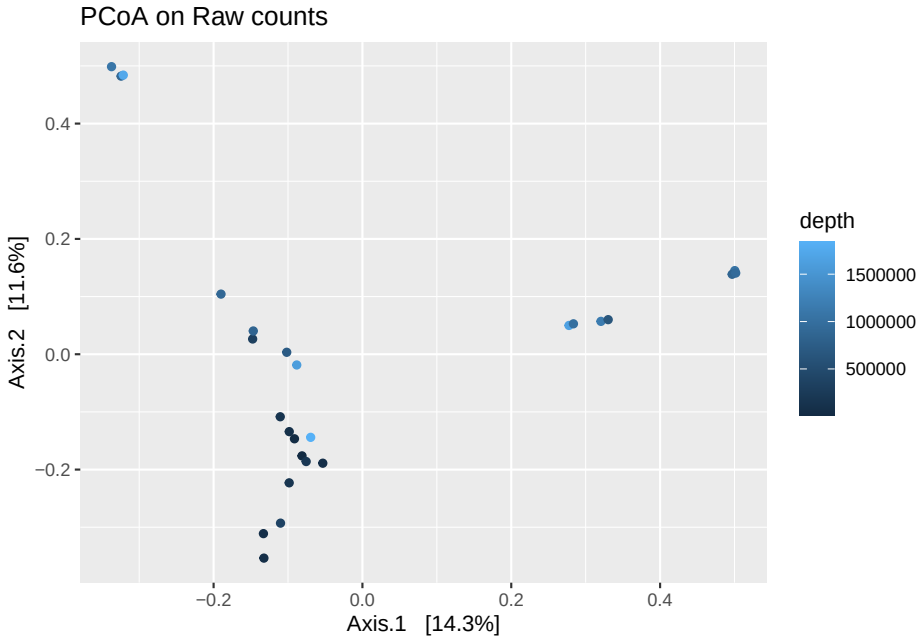
This decreases the number of taxa from 19216 to 219. This is not surprising, because this dataset contains samples from widely different locations (gut, soil, etc), and few taxa are shared among all samples and locations. One potential problem with this approach is the widely different locations, so 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 Global Patterns dataset.

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

We can visualize technical artifacts of sapling depth is by looking at principal coordinates plots using the Bray-Curtis dissimilarity, coloring by sampling depth too see how much variation can be explained by the original sampling depth.

```
gp_raw_dist <- phyloseq::ordinate(gp_raw, "PCoA", "bray")
plot_ordination(gp_raw,
                gp_raw_dist,
                color = "depth",
                title = "PCoA on Raw counts")
```



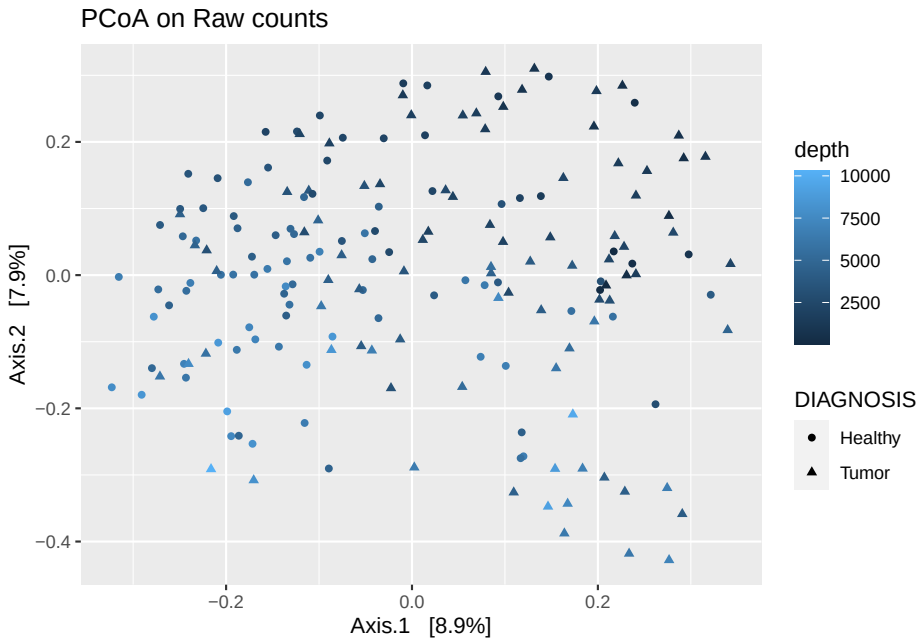

We don't see any extreme patterns with sampling depth, but this might additionally be due to the differences in different locations might have different sampling depths. This comparison might be more interesting when we only have one location we are sampling from.

Repeat this same process for the `kostic` data. Use a prevalence filter to only include taxa that have nonzero counts in over 10% of samples. This results in 478 taxa across 185 samples.

```
# Prevalence filter
prevalenceThreshold <- 0.10 * nsamples(k_raw)
toKeep <- apply(data.frame(otu_table(k_raw)), 1, function(taxa)
    return(sum(taxa > 0) > prevalenceThreshold))
k_raw <- prune_taxa(toKeep, k_raw)
k_raw
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 478 taxa and 185 samples ]
## sample_data() Sample Data:       [ 185 samples by 71 sample variables ]
## tax_table()   Taxonomy Table:    [ 478 taxa by 7 taxonomic ranks ]
```

```
# Save sampling depth as a sample variable
k_raw@sam_data$depth <- sample_sums(k_raw)

k_raw_dist <- phyloseq::ordinate(k_raw, "PCoA", "bray")
plot_ordination(k_raw,
                k_raw_dist,
                color = "depth",
                shape = "DIAGNOSIS",
                title = "PCoA on Raw counts")
```




When considering a normalization method, it is important to know if between sample (DA methods) or within sample (community) analysis is of interest.

# Chapter 3

# Count Scaling Normalization methods

## 3.1 Total Sum scaling (TSS)

### 3.1.1 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 sampling depth. 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`, if that one taxon were not included in the sample, the scaling factor would be widely different.

| 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).

### 3.1.2 TSS R code

#### 3.1.2.1 Function

Here, we provide a wrapper 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)
}
```

#### 3.1.2.2 TSS implementation on Global Patterns

Using the above function, we apply this normalization to 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
```

To see the differences between the un-normalized, raw data, and the TSS transformed normalized data, one possible way is to look 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()`

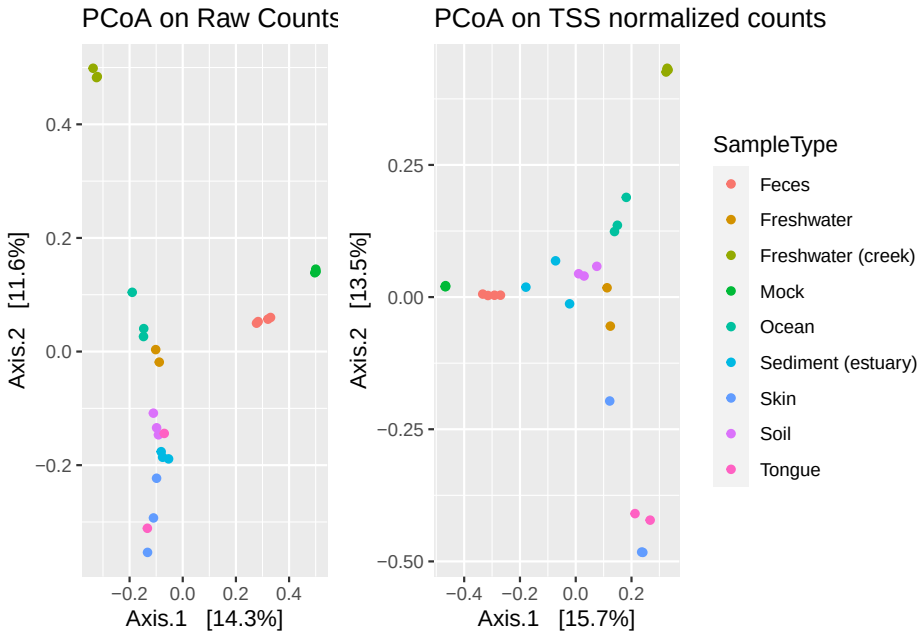
```
gp_raw_dist <- phyloseq::ordinate(gp_raw, "PCoA", "bray")
gp_tss_dist <- phyloseq::ordinate(gp_tss, "PCoA", "bray")
```

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

(Note to self: perhaps choose a different dataset to use for walk-through, currently using Global Patterns since it is small and quick for computations, but

harder to see differences.)

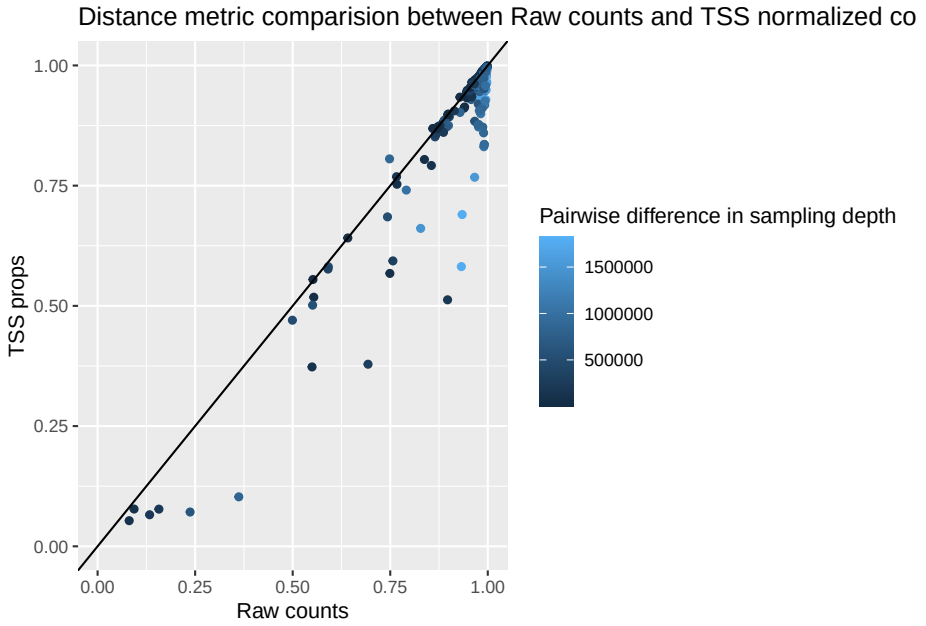
```
plot_ordination(gp_raw, gp_raw_dist, color = "SampleType",
                title = "PCoA on Raw Counts") +
plot_ordination(gp_tss, gp_tss_dist, color = "SampleType",
                title = "PCoA on TSS normalized counts") +
  plot_layout(guides = 'collect')
```




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

```
# Function to visualize potential differences and changes after normalization methods
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, na.rm = T)),
                       norm = as.numeric(phyloseq::distance(data_norm, dist_method, na.rm = T)),
                       diff = as.numeric(dist(get_variable(data_raw, "depth")))) %>%
    ggplot(aes(x = raw, y = norm, color = diff)) +
        geom_point() +
        geom_abline() +
        ggtitle(title) +
        xlab(x_lab) + ylab(y_lab) +
        labs(color = "Pairwise difference in sampling depth")+
      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",
                  title = "Distance metric comparision between Raw counts and TSS norma
```

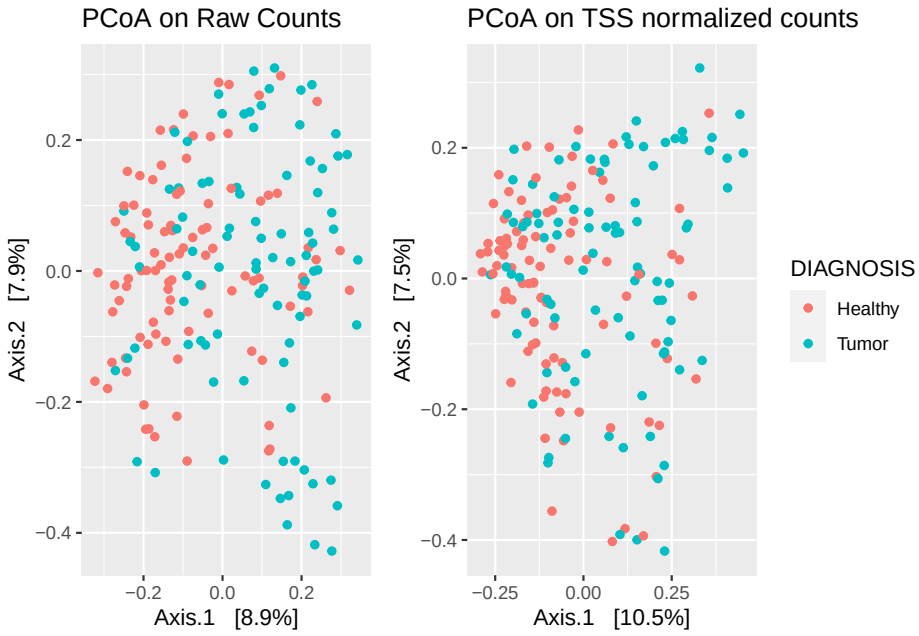



Points below the line are pairs of samples that are marked as more similar after normalization. Points above the line are marked as more different after normalization. Values closer to 1 are 'more different'. Unsurprisingly, pairs that had larger original differences in sampling depth were marked as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

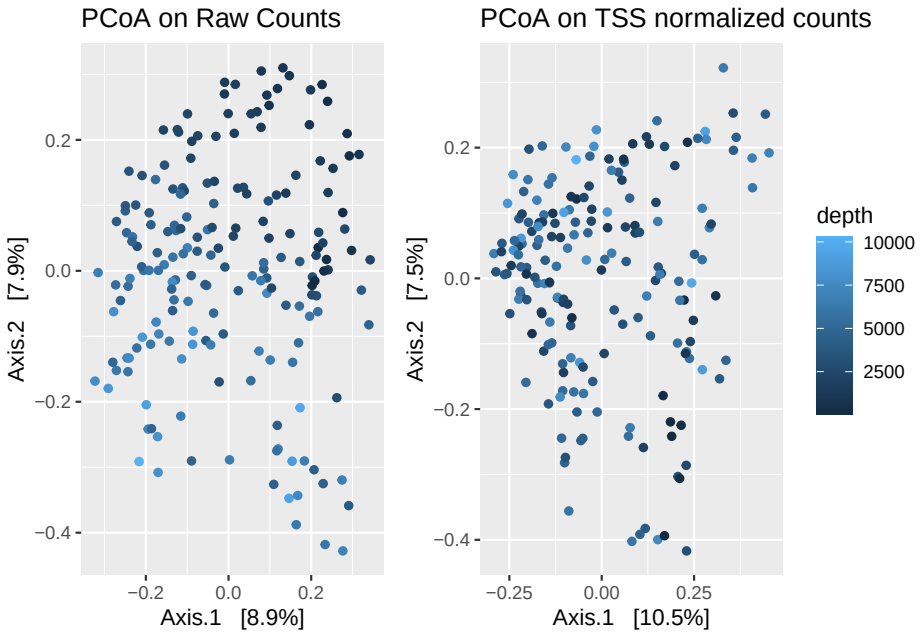
#### 3.1.2.3 TSS on Kostic data

```
k_tss <- norm_TSS(k_raw)
k_raw_dist <- phyloseq::ordinate(k_raw, "PCoA", "bray")
k_tss_dist <- phyloseq::ordinate(k_tss, "PCoA", "bray")

# Ordination with Diagnosis as color
plot_ordination(k_raw, k_raw_dist, color = "DIAGNOSIS",
                title = "PCoA on Raw Counts") +
plot_ordination(k_tss, k_tss_dist, color = "DIAGNOSIS",
```

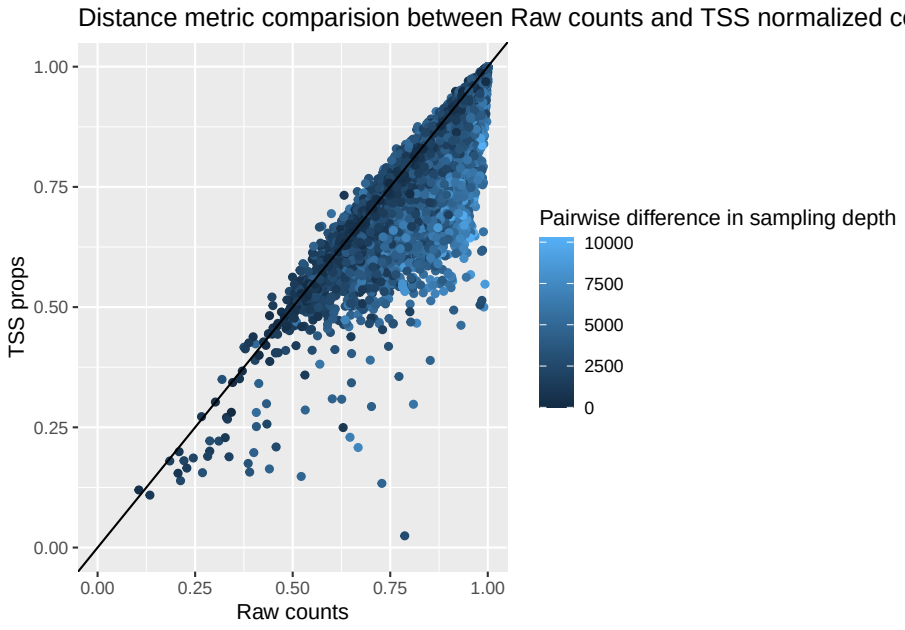
```
                title = "PCoA on TSS normalized counts") +
  plot_layout(guides = 'collect')
```




```
# Ordination with original sampling depth as color
plot_ordination(k_raw, k_raw_dist, color = "depth",
                title = "PCoA on Raw Counts") +
plot_ordination(k_tss, k_tss_dist, color = "depth",
                title = "PCoA on TSS normalized counts") +
  plot_layout(guides = 'collect')
```




```
plot_norm_changes(k_tss, k_raw,
                  x_lab = "Raw counts", y_lab = "TSS props",
                  title = "Distance metric comparision between Raw counts and TSS norma
```



## 3.2 Cumulative sum scaling (CSS)

### 3.2.1 About CSS

Cumulative sum scaling is a scaling normalization 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. If there is high count variability the assumption may not be met. This method is an extension to UQ scaling normalization where one quantile is specified for all samples. In contrast to Total Sum scaling where the total sum is used as the scaling factor, the scaling factor for CSS is the sum of the counts up to the chosen quantile. This is not a method that accounts for compositionality. CSS normalization 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).

### 3.2.2 CSS R code

#### 3.2.2.1 Function

```
norm_CSS <- function(ps){
  require(metagenomeSeq)
  # Convert to metagenomeSeq data type
  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)
  # Convert back to phyloseq with normalized counts
  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)
}
```

#### 3.2.2.2 CSS on Global Patterns

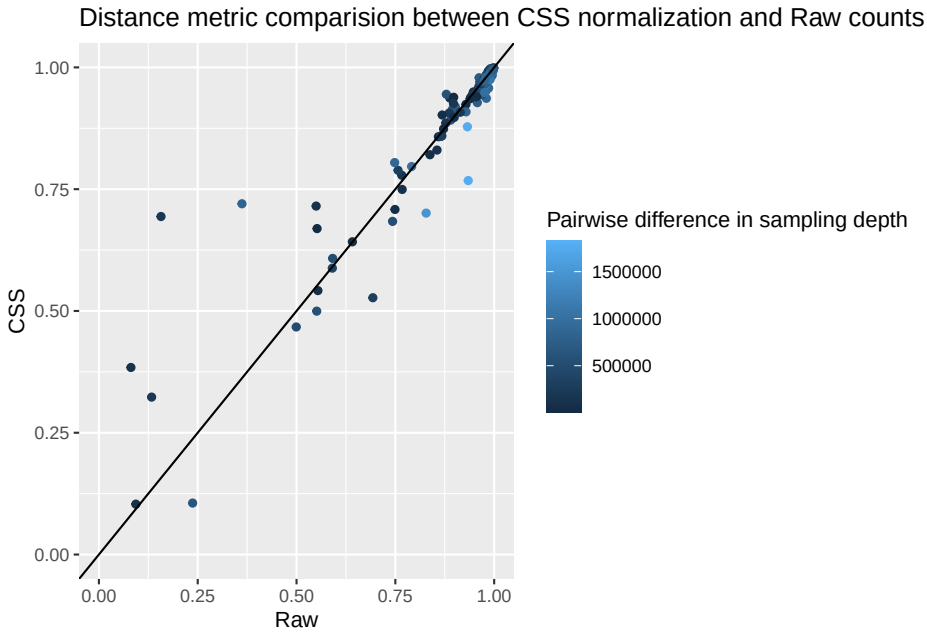
Perform CSS normalization:

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

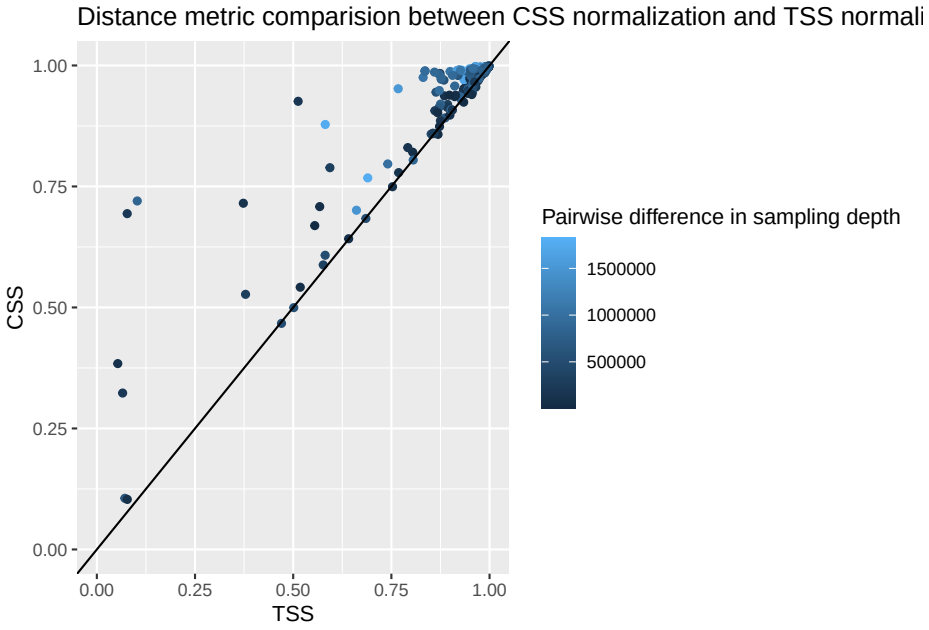
```
## Default value being used.
```

View how TMM normalization changes distance metrics differently than raw counts.

```
plot_norm_changes(gp_css, gp_raw,
                  x_lab = "Raw", y_lab = "CSS",
                  title = "Distance metric comparision between CSS normalization and Ra
```




```
plot_norm_changes(gp_css, gp_tss,
                  x_lab = "TSS", y_lab = "CSS",
                  title = "Distance metric comparision between CSS normalization and TS
```



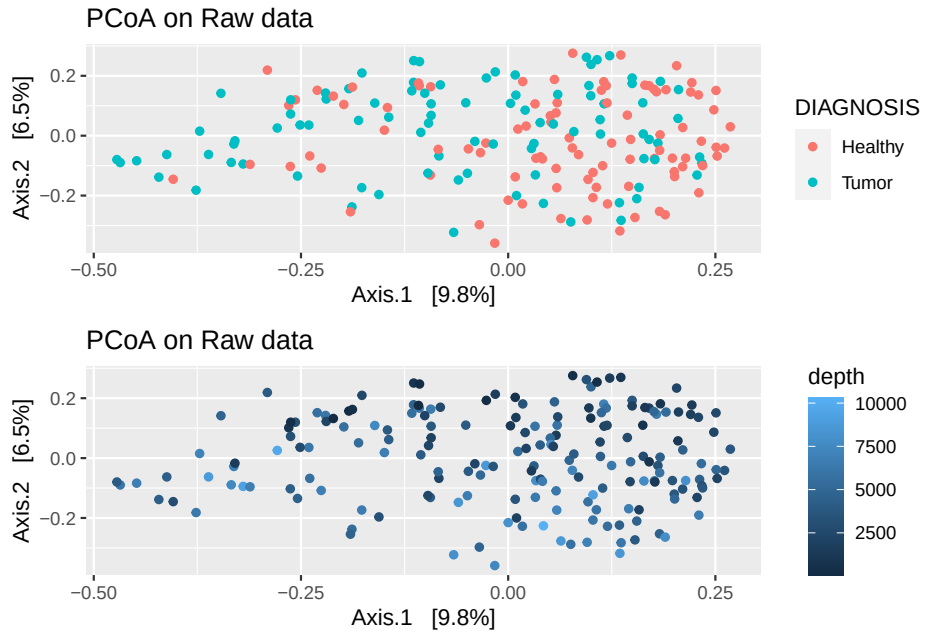

CSS normalization appears to consider pairs as more different than TSS normalization, and pairs with high sequencing depth differences even more so.

#### 3.2.2.3 CSS on Kostic data

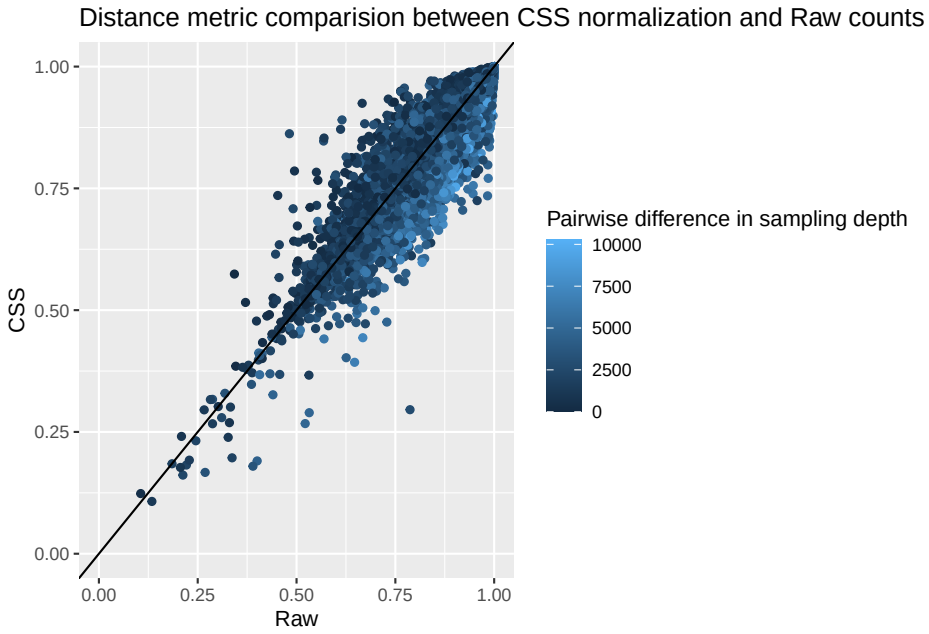
```
k_css <- norm_CSS(k_raw)
```

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

```
plot_ordination(
    k_css,
    phyloseq::ordinate(k_css, "PCoA", "bray") ,
    color = "DIAGNOSIS",
    title = "PCoA on Raw data"
) /
    plot_ordination(
        k_css,
        phyloseq::ordinate(k_css, "PCoA", "bray") ,
        color = "depth",
        title = "PCoA on Raw data"
    )
```




```
plot_norm_changes(k_css, k_raw,
                  x_lab = "Raw", y_lab = "CSS",
                  title = "Distance metric comparision between CSS normalization and Ra
```



# Chapter 4

# Subsampling Methods

## 4.1 Rarefying

### 4.1.1 About Rarefying

Rarefying is another common normalization technique that standardizes the library size across samples that was 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 (McMurdie and Holmes, 2014). 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 (Weiss et al., 2017), and has also been shown to perform well in community-level analysis (McKnight et al., 2019), as it completely standardizes the read count depth, and some methods are sensitive to differences in read count. Rarefying has been shown to separate by biological signal in ordination methods based on presence/absence.

### 4.1.2 Rarefying walkthrough

Consider the following example dataset.

Example dataset - Raw Counts

Taxa

Sample 1

Sample2

Sample 3

Taxon 1

132

103

11

Taxon 2

7

48

3

Taxon 3

0

2

1

Taxon 4

23

15

2

Taxon 5

71

80

5

To normalize this data by rarefying, we first choose a minimum sampling depth, or in other words, the minimum column sum. After normalizing, the column sum of all samples will be this size. Any samples that are below this minimum will be dropped from analysis. We choose 200 as our minimum here. In practice the counts will normally be much higher. In this example, our sampling depths are 223, 248, and 20. Thus the third sample will be dropped in this procedure.

Since we are randomly sampling, we need to keep track of the seed so this process is reproducible.

Table 4.1: Rarefied counts

| Taxa | Sample 1 | Sample2 |
|---|---|---|
| Taxon 1 | 115 | 90 |
| Taxon 2 | 6 | 39 |
| Taxon 3 | 0 | 0 |
| Taxon 4 | 18 | 11 |
| Taxon 5 | 61 | 60 |

```
set.seed(525)
```

We then randomly sample taxa from each sample/column in the proportions corresponding to the raw counts. The below table shows the table from above after we rarefy. Notice that the rare Taxon 3, which originally was present in Sample 1 and 3 is now completely removed from the analysis.

### 4.1.3 Rarefying R code

We now show the process of rarefying a real dataset. The following function returns a rarefied `phyloseq` object. We can either pass in the minimum sampling depth as a second argument, or use the default minimum depth of the samples.

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

#### 4.1.3.1 Rarefying on Global Patterns

We use the above function to rarefy the Global Patterns data. 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.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.

```
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 dataset was almost 2 million, so we have discarded a lot of data to reduce to 15905.

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

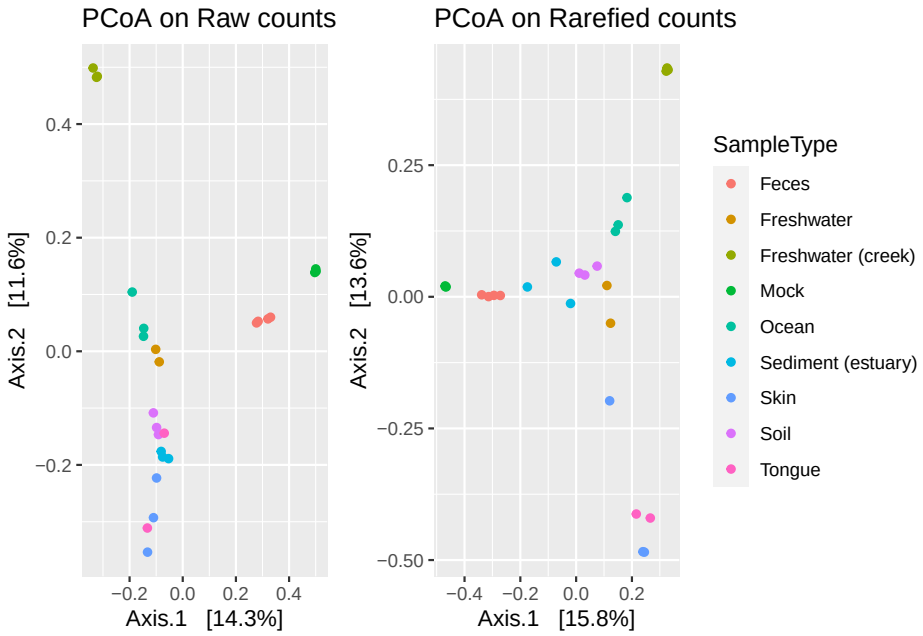
```
## [1] 1842380
```

```
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
```

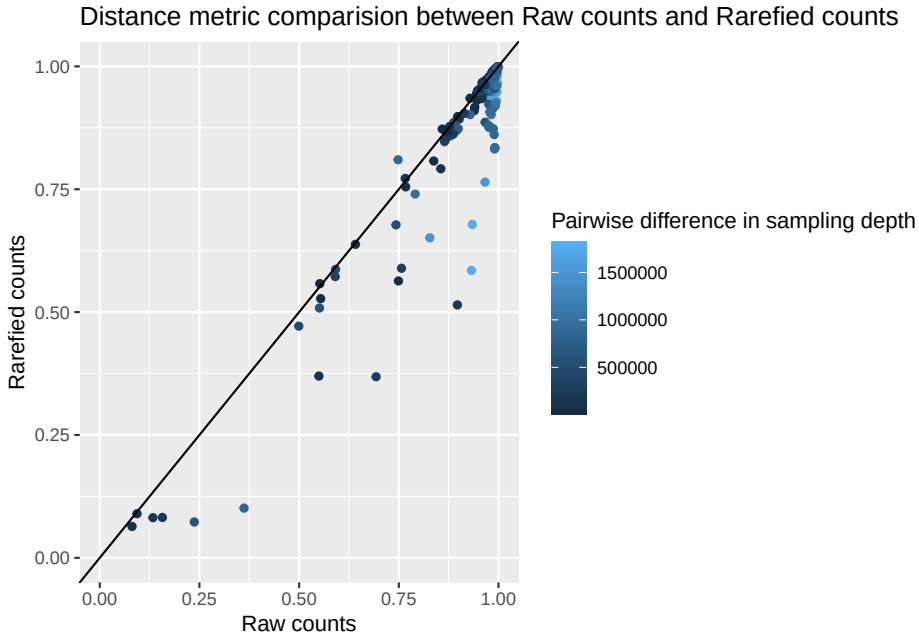
We can again compare the PCoA plots between rarefied and raw counts, coloring by sample type to view clusters.

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

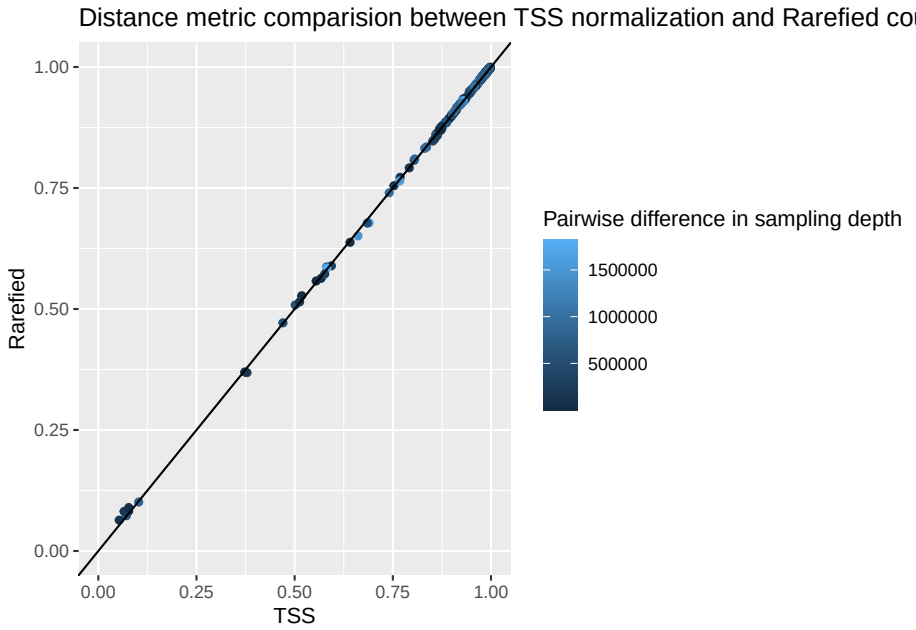



Now examine how the distance matrices change before/after normalization. We see a similar pattern to TSS when distance matrices calculated from rarefied counts are compared to those from the raw counts.

```
# Identify any samples filtered in rarefying process
rare_samples <- sample_names(gp_rare)
gp_raw_match <- prune_samples(rare_samples, gp_raw)
plot_norm_changes(gp_rare, gp_raw_match,
                  x_lab = "Raw counts", y_lab = "Rarefied counts",
                  title = "Distance metric comparision between Raw counts and Rarefied counts ")
```




```
## Compare to tss
gp_tss_match <- norm_TSS(gp_raw_match)
plot_norm_changes(gp_rare, gp_tss,
                  x_lab = "TSS", y_lab = "Rarefied",
                  title = "Distance metric comparision between TSS normalization and Ra
```



### 4.1.4 Rarefying on Kostic Data

```
k_rare <- norm_rarefy(k_raw)
```

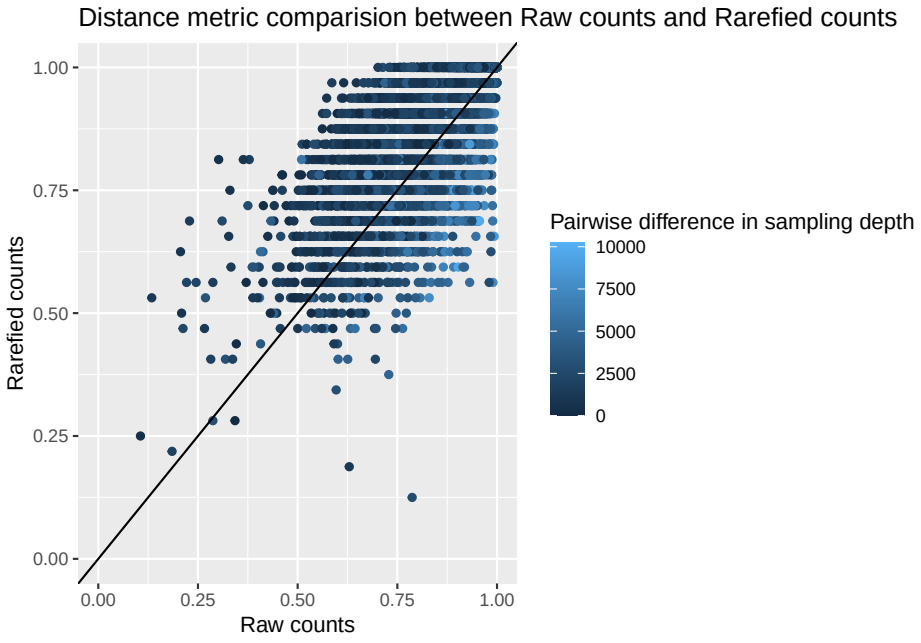
```
## You set `rngseed` to FALSE. Make sure you've set & recorded
##  the random seed of your session for reproducibility.
## See `?set.seed`
```

```
## ...
```

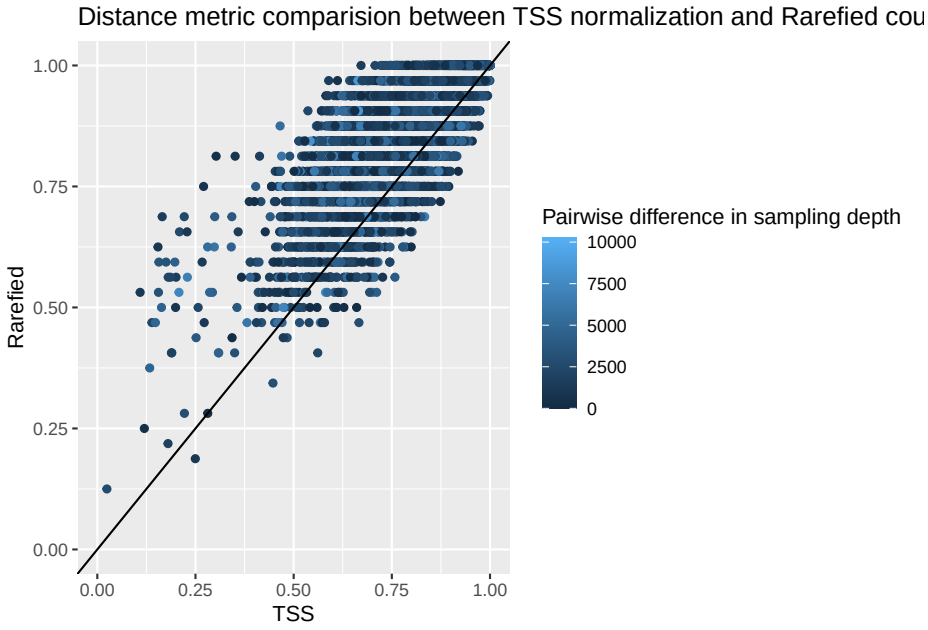
```
## 1280TUs were removed because they are no longer
## present in any sample after random subsampling
```

```
## ...
```

```
plot_norm_changes(k_rare, k_raw,
                  x_lab = "Raw counts", y_lab = "Rarefied counts",
                  title = "Distance metric comparision between Raw counts and Rarefied counts ")
```




```
plot_norm_changes(k_rare, k_tss,
                  x_lab = "TSS", y_lab = "Rarefied",
                  title = "Distance metric comparision between TSS normalization and Rarefied cou
```


Distance metric comparision between TSS normalization and Rarefied cou
Rarefied
TSS
0.00
0.25
0.50
0.75
1.00
Pairwise difference in sampling depth
10000
7500
5000
2500
0

# Chapter 5

# Library Size and Composition Scaling Methods

## 5.1 DESeq

### 5.1.1 About DESeq

The DESeq2 package includes a normalization method for adjusting for differing library sizes across samples (Anders and Huber, 2010). This method also can account for differences in library composition. A scaling factor to normalize each sample takes into account both library size and library composition. This method has also been called MED, RLE, or DESeq in the literature.

DESeq2 first takes the 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.

#### 5.1.1.1 DESeq walkthrough

Consider again this example dataset.

Example dataset - Raw Counts

Taxa

Sample 1

Sample2

Sample 3

Taxon 1

132

103

11

Taxon 2

7

48

3

Taxon 3

0

2

1

Taxon 4

23

15

2

Taxon 5

71

80

5

The first step in DESeq normalization is to take the natural log of each entry in the count matrix.

Step 1: Logged Raw Counts

Taxa

Sample 1

Sample2

Sample 3

Taxon 1

4.88

4.63

2.40

Taxon 2

1.95

3.87

1.10

Taxon 3

-Inf

0.69

0.00

Taxon 4

3.14

2.71

Table 5.1: Step 2: Average Logged Counts

| Taxa | Average Log Counts |
|---|---|
| Taxon 1 | 3.971809 |
| Taxon 2 | 2.305241 |
| Taxon 3 | -Inf |
| Taxon 4 | 2.178897 |
| Taxon 5 | 3.418048 |

Table 5.2: Step 3: Log Ratio of reads in each sample to average across all samples

| Taxa | Sample 1 | Sample 2 | Sample 3 |
|---|---|---|---|
| Taxon 1 | 4.88 - 3.97 = 0.91 | 4.63 - 3.97 = 0.66 | 2.40 - 3.97 = -1.57 |
| Taxon 2 | 1.95 - 2.31 = -0.36 | 3.97 - 2.31 = 1.66 | 1.10 - 2.31 = -1.31 |
| Taxon 4 | 3.14 - 2.18 = 0.96 | 2.71 - 2.18 = 0.53 | 0.69 - 2.18 = -1.49 |
| Taxon 5 | 4.26 - 3.42 = 0.84 | 4.38 - 3.42 = 0.96 | 1.61- 3.42 = -1.81 |

0.69

Taxon 5

4.26

4.38

1.61

Notice the one entry with zero counts is marked as negative infinity.

Next, the average of the logged values is taken across the samples. This results in an average log value for each taxa. This averaged log value is helpful for normalizing because it is not overwhelmed by outliers.

The row for the taxa that contained a zero on one of the samples is negative infinity. This taxon is now excluded from the following steps. This step results in removing any taxa that have zero counts from being considered to contribute to calculating the scaling factors.

The next step is to subtract the average log values (step 2) from the log of the raw counts (step 1), only including rows that were not filtered. This step shows which samples have counts in a sample higher or lower than the average.

Next, to calculate the scaling factors for each sample, we take the median of the log ratios calculated in the above step. Like using logs, calculating medians avoids the influence of outlier taxa that put undue influence on the scaling factor.

Table 5.3: Step 4: Calculating medians per sample

| Taxa | Sample 1 | Sample 2 | Sample 3 |
|---|---|---|---|
| Taxon 1 | 0.91 | 0.66 | -1.57 |
| Taxon 2 | -0.36 | 1.66 | -1.31 |
| Taxon 4 | 0.96 | 0.53 | -1.49 |
| Taxon 5 | 0.84 | 0.96 | -1.81 |
| Median | 0.88 | 0.81 | -1.53 |
| Exponentiated Median | 2.40 | 2.25 | 0.22 |

Finally, we normalize the data, by exponentiating the median log ratios for each sample, which are the final scaling factors. We then divide all the raw counts in a sample by the sample's scaling factor.

Step 5: DESeq Normalized Counts

Taxa

Sample 1

Sample2

Sample 3

Taxon 1

55.03

45.82

50.80

Taxon 2

2.92

21.35

13.85

Taxon 3

0.00

0.89

4.62

Taxon 4

9.59

6.67

9.24

Taxon 5

29.60

35.59

23.09

Since Taxon 3 had a zero count in sample 1, it was excluded from the calculation of scale factors. The above example dataset may not be characteristic of microbiome datasets. Microbiome datasets are zero-inflated, meaning that there are numerous zero counts in the raw count matrix. Even up to 80-90% of the counts in a microbiome datset can be zero. Because of this, if the zero-inflation in the datseta is not accounted for, very few, or even perhaps none of the taxa will contribute to calculating the scaling factor. One option so that all of the taxa are included in the calculation is to add a pseudocount so none of the counts are zero. Another option is the `poscounts` option, which is encouraged for microbiome data. Instead of taking the average of the logged counts, it takes the $n$th root of the non-zero counts. This replaces step 2 in this example.

### 5.1.2 DESeq R Code

#### 5.1.2.1 Function

Here we provide two normalization functions implemented in R using DESeq methods. The first calculates the RLE normalization using the `poscounts` option for microbiome data. The second calculates the variance stabilizing transformation.

```
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)
}
```

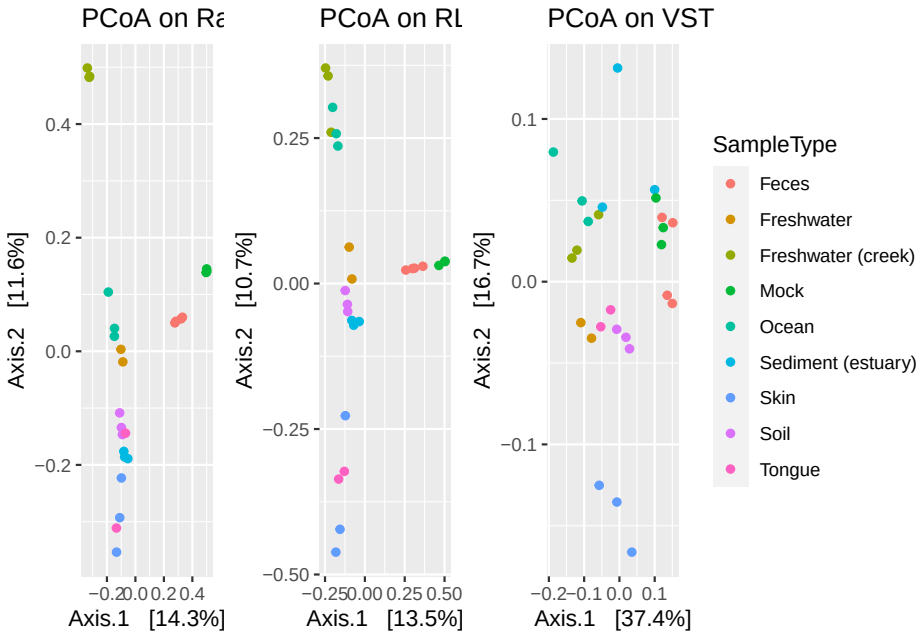
#### 5.1.2.2 DESeq implemented on Global Patterns

Perform DESeq RLE normalization as well as DESeq variance stabilized transformation on Global Patterns:

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

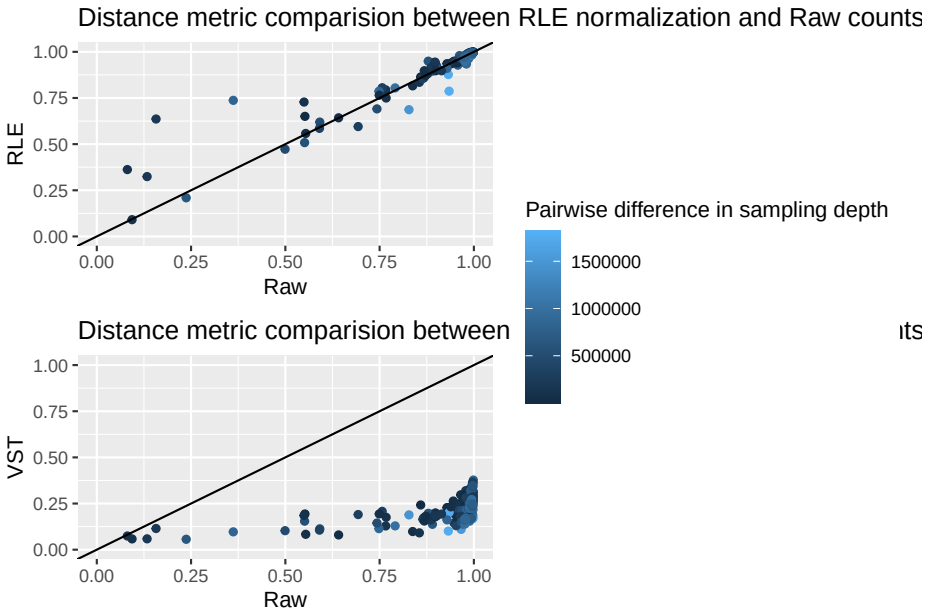
Examine principal coordinate plots between raw data and both DESeq normalized data.

```
# First calculate distance matrices
gp_rle_dist <- phyloseq::ordinate(gp_deseq_rle, "PCoA", "bray")
gp_vs_dist <- phyloseq::ordinate(gp_deseq_vs, "PCoA", "bray")

# Plot ordinations
plot_ordination(gp_raw, gp_raw_dist, color = "SampleType", title = "PCoA on Raw data") +
plot_ordination(gp_deseq_rle, gp_rle_dist, color = "SampleType", title = "PCoA on RLE") +
plot_ordination(gp_deseq_vs, gp_vs_dist, color = "SampleType", title = "PCoA on VST") +
  plot_layout(guides = 'collect')
```



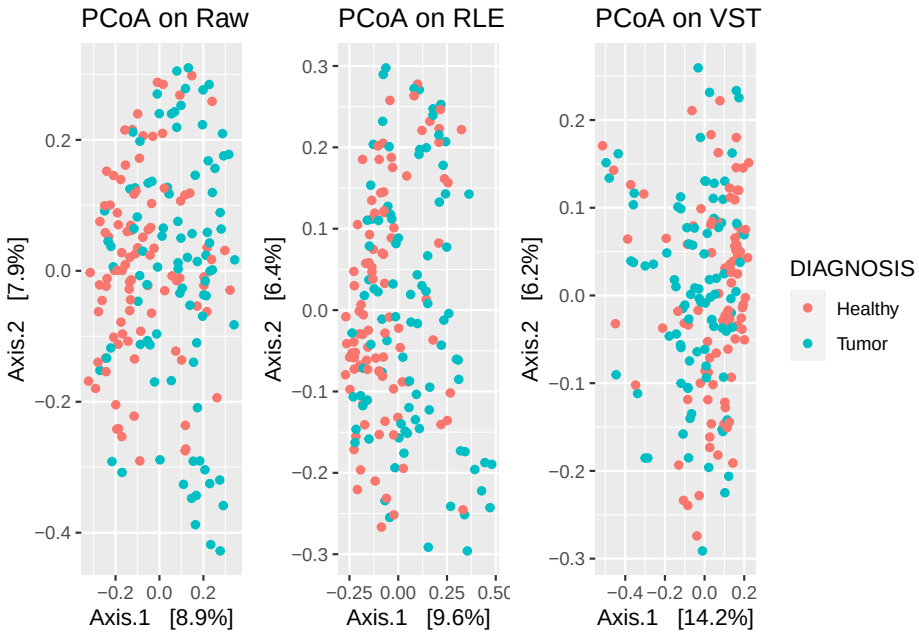

See how dissimilarity matrices differ from raw counts and each DESeq transformation.

```
plot_norm_changes(gp_deseq_rle, gp_raw,
                  x_lab = "Raw", y_lab = "RLE",
                  title = "Distance metric comparision between RLE normalization and Ra
plot_norm_changes(gp_deseq_vs, gp_raw,
                  x_lab = "Raw", y_lab = "VST",
                  title = "Distance metric comparision between VST normalization and Ra
  plot_layout(guides = 'collect')
```

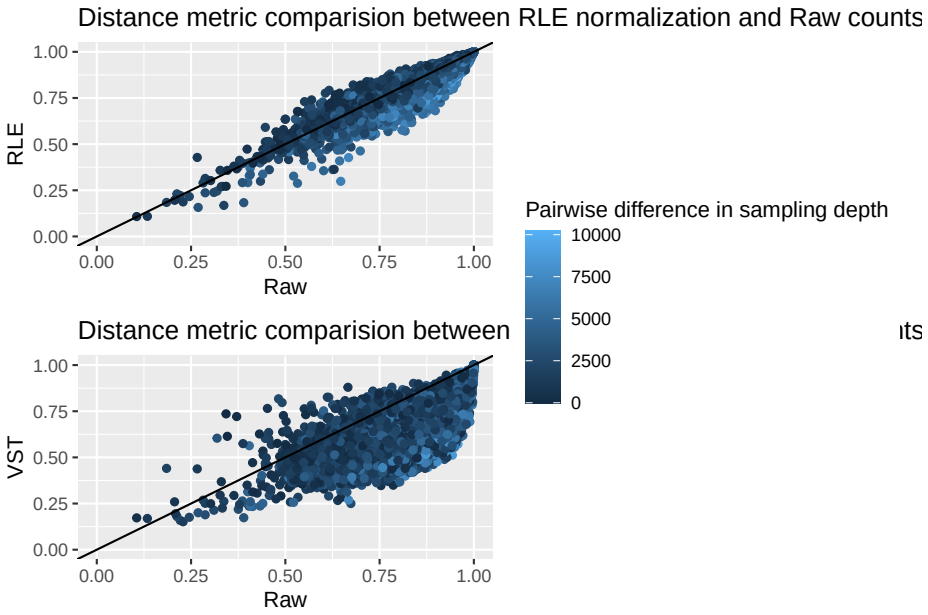



#### 5.1.2.3 DESeq implemented on Kostic dataset

```
k_deseq_rle <- norm_DESeq_RLE_poscounts(k_raw)
k_deseq_vs <- norm_DESeq_vs(k_raw)

# First calculate distance matrices
k_rle_dist <- phyloseq::ordinate(k_deseq_rle, "PCoA", "bray")
k_vs_dist <- phyloseq::ordinate(k_deseq_vs, "PCoA", "bray")

# Plot ordinations
plot_ordination(k_raw, k_raw_dist, color = "DIAGNOSIS", title = "PCoA on Raw data") +
plot_ordination(k_deseq_rle, k_rle_dist, color = "DIAGNOSIS", title = "PCoA on RLE") +
plot_ordination(k_deseq_vs, k_vs_dist, color = "DIAGNOSIS", title = "PCoA on VST") +
  plot_layout(guides = 'collect')
```




```
plot_norm_changes(k_deseq_rle, k_raw,
                  x_lab = "Raw", y_lab = "RLE",
                  title = "Distance metric comparision between RLE normalization and Ra
plot_norm_changes(k_deseq_vs, k_raw,
                  x_lab = "Raw", y_lab = "VST",
                  title = "Distance metric comparision between VST normalization and Ra
  plot_layout(guides = 'collect')
```




## 5.2 GMPR

### 5.2.1 About GMPR

A recent extension of the RLE DESeq method is the Geometric mean of Pairwise ratios (GMPR) approach (Chen et al., 2018). 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. It then uses the pairwise results to calculate the size factor for each sample. This method has slow computation, but is robust to differential and outlier taxa. 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.

### 5.2.2 GMPR Walkthough

Consider the following zero-inflated dataset. Notice that there are no taxa that are present in every sample.

Example dataset - Raw Counts

Taxa

Sample 1

Sample2

Sample 3

Taxon 1

132

103

0

Taxon 2

0

48

74

Taxon 3

0

2

0

Taxon 4

23

0

35

Taxon 5

71

80

0

Taxon 6

0

96

82

The first step in GMPR normalization is to calculate the pairwise median count ratio between samples. We first calculate the scaling factor for sample 1. The

pairwise comparisons we need to make are between sample 1 and sample 2 as well as sample 1 and sample 3. For both pairs, we calculate the ratio of the counts of taxa that the pair shares. Between sample 1 and sample 2, the shared taxa are 1 and 5. Between sample 1 and sample 3, taxon 4 is the only one shared.

Example dataset - Raw Counts

Taxa

Sample 1

Sample2

Sample 3

Count ratio between 1 and 2

Count ratio between 1 and 3

Taxon 1

132

103

0

132/103

Taxon 2

0

48

74

Taxon 3

0

2

0

Taxon 4

23

0

35

23/35

Taxon 5

71

80

0

71/80

Taxon 6

0

96

82

Then we calculate the median of the ratios between each pair. Between sample 1 and 2, the two ratios between shared taxa are (132/103, 71/80). Then the median of those is 1.085. Between sample 1 and 3, there is only one shared taxa, so the median is 0.657.

Finally to find the scaling factor for sample 1, we calculate the geometric mean of the two medians of the pairwise shared taxa ratios. In this case it is the geometric mean of 1.085 and 0.657, which equals 0.844. This is the scaling factor for sample 1. We now repeat this process to find the scaling factor for sample 1 and sample 3.

### 5.2.3 GMPR R Code

#### 5.2.3.1 Function

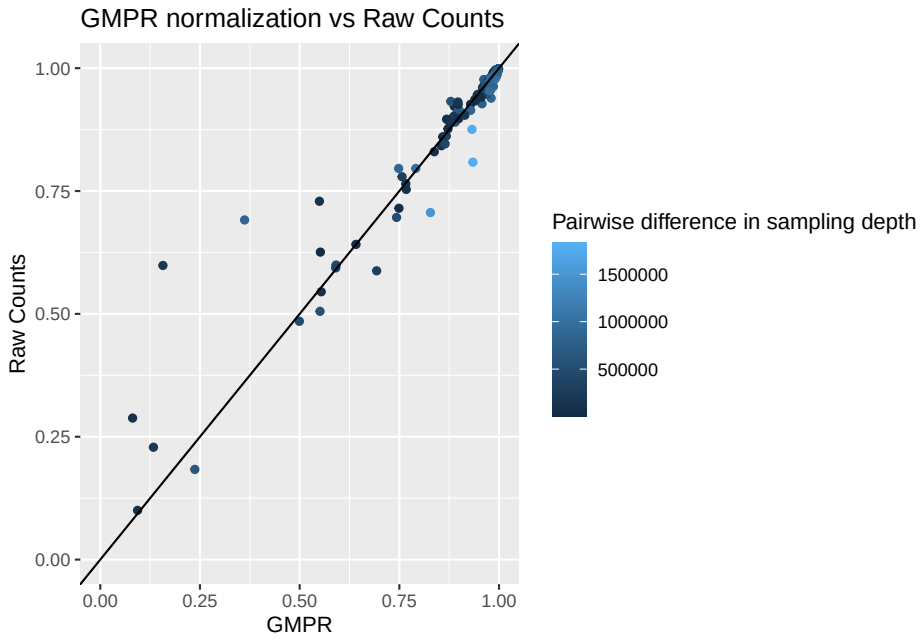
The following provides the R code to implement GMPR normalization in R. We can additionally specify the number of taxa that should be shared between paired samples as well as the minimum count labeled as nonzero. The default values are 4 and 2 respectively.

```
norm_GMPR <- function(ps, intersect_no = 4, min_ct = 2){
  require(GMPR, quietly = T)
    # Convert data to correct format for GMPR function
    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))
    # calculate scaling factor
    # OTU matrix must be a data frame where OTUs are arranged in columns and samples a
    gmpr.size.factor <- GMPR::GMPR(OTUmatrix = otu_df,
                                   intersect_no = intersect_no,
                                   min_ct = min_ct)
    # normalize
    otu.tab.norm <- t(otu / (gmpr.size.factor))
    # convert back to PS
```

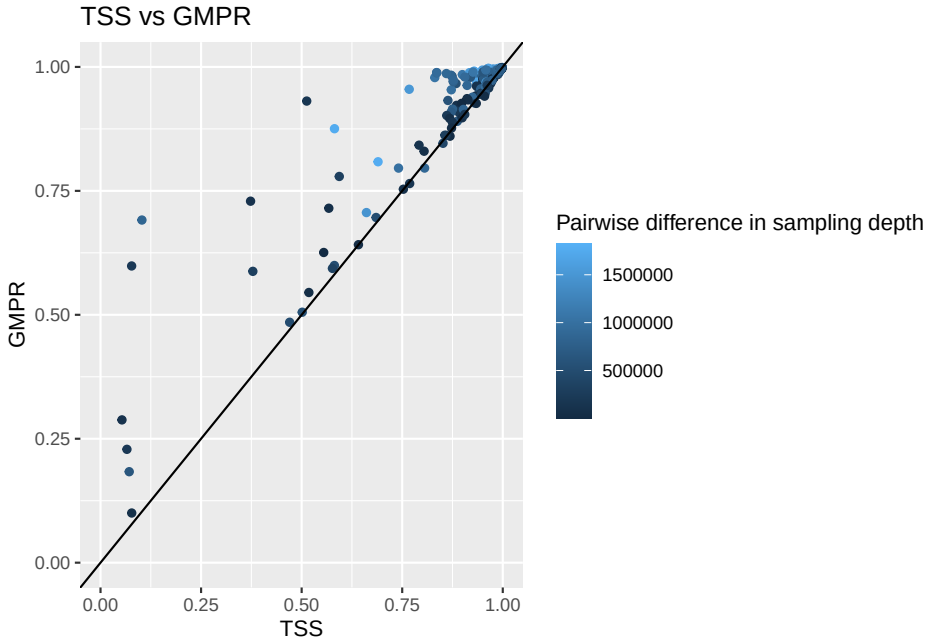
```
    otu_ps <- otu_table(otu.tab.norm, taxa_are_rows = T)
    sam <- access(ps, "sam_data")
    sam$scaling_factor <- gmpr.size.factor
    tax <- access(ps, "tax_table")
    phy <- access(ps, "phy_tree")
    ps_GMPR <- phyloseq(otu_ps,sam,tax,phy)
    return(ps_GMPR)
}
```

#### 5.2.3.2 GMPR implementation on Global Patterns

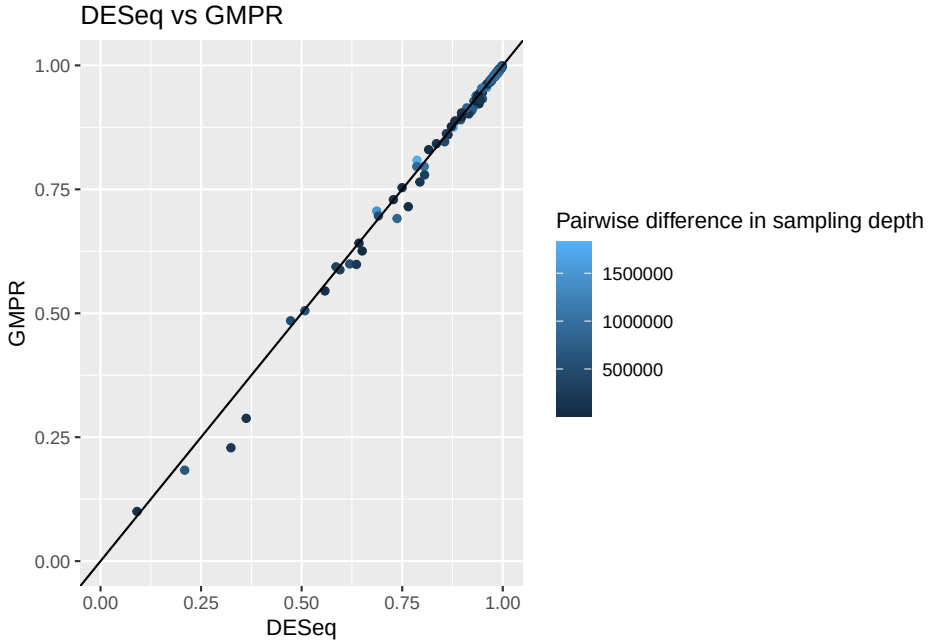
```
# Normalize
gp_gmpr <- norm_GMPR(gp_raw)

# Compare to other methods like before
plot_norm_changes(gp_gmpr, gp_raw,
                  x_lab = "GMPR", y_lab = "Raw Counts",
                  title = "GMPR normalization vs Raw Counts")
```




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




```
plot_norm_changes(gp_gmpr, gp_deseq_rle,
                  x_lab = "DESeq", y_lab = "GMPR",
                  title = "DESeq vs GMPR" )
```



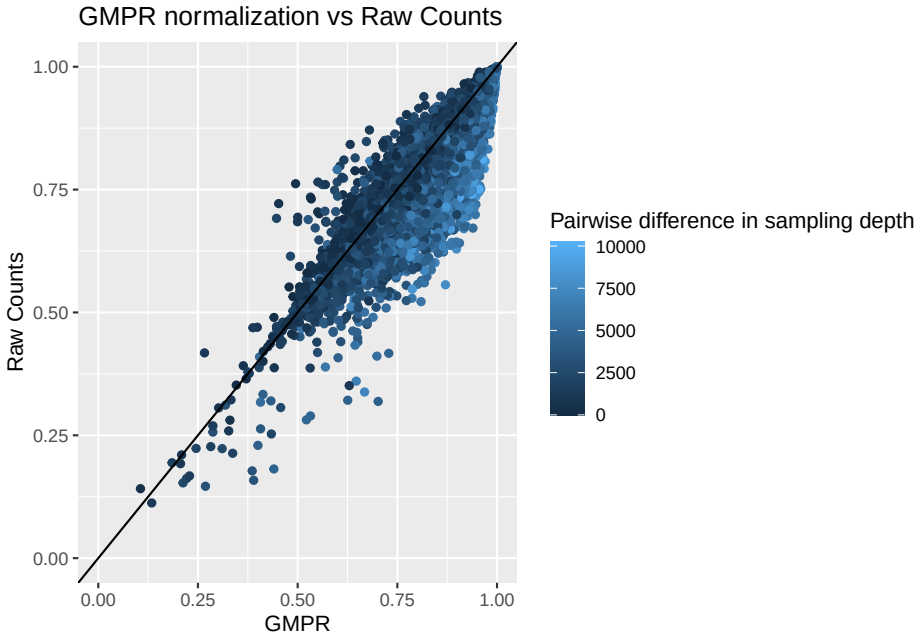
#### 5.2.3.3 GMPR implementation on Global Patterns

```
k_gmpr <- norm_GMPR(k_raw)

plot_norm_changes(k_gmpr, k_raw,
                  x_lab = "GMPR", y_lab = "Raw Counts",
                  title = "GMPR normalization vs Raw Counts")
```

```
## Warning in vegdist(structure(c(14.7364903183976, 5.44797838541712,
## 0.281000656690941, : you have empty rows: their dissimilarities may be
## meaningless in method "bray"
```

```
## Warning in vegdist(structure(c(14.7364903183976, 5.44797838541712,
## 0.281000656690941, : missing values in results
```

```
## Warning: Removed 367 rows containing missing values (geom_point).
```




## 5.3 TMM (edgeR)

### 5.3.1 About TMM

TMM (Trimmed median of m-values) is another method borrowed from RNA-Seq analysis, and implemented in `edgeR` (Robinson and Oshlack, 2010). 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 pseudo count 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.

### 5.3.2 EdgeR TMM T code

#### 5.3.2.1 Function

```
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)

}
```

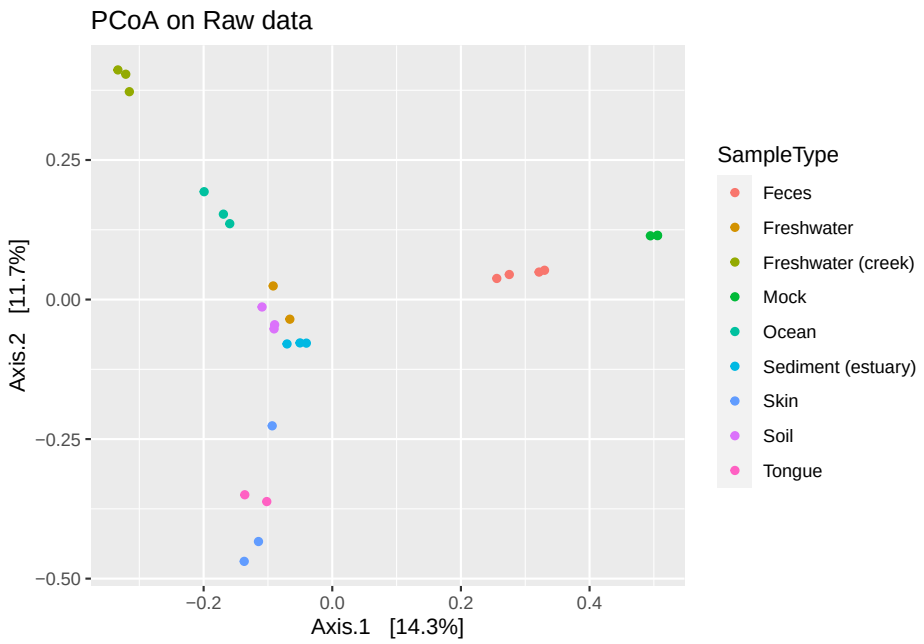
#### 5.3.2.2 TMM implementation on Global Patterns

Perform normalization:

```
gp_tmm <- norm_TMM(gp_raw)
```

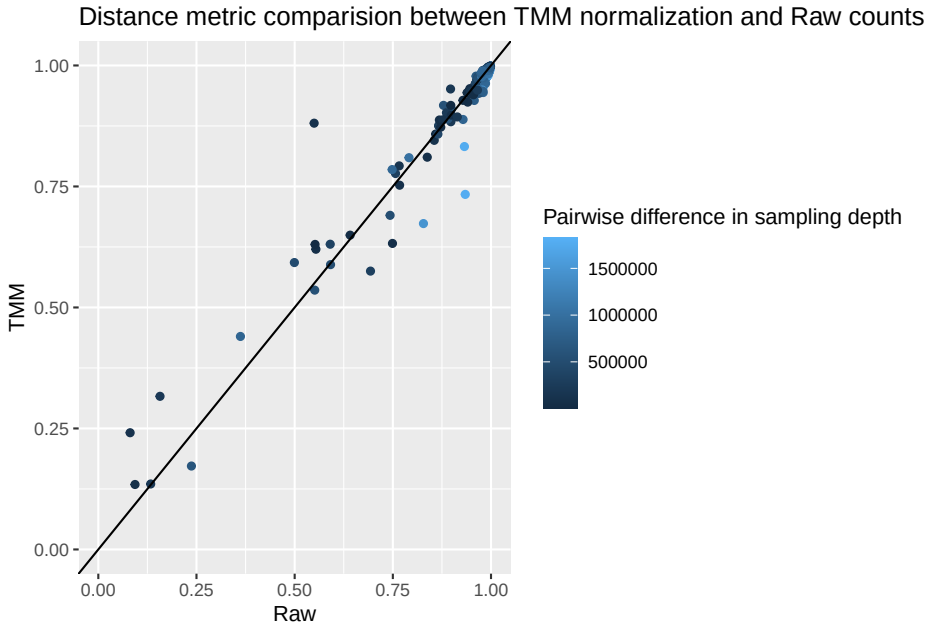
View PCoA plots

```
plot_ordination(gp_tmm, phyloseq::ordinate(gp_tmm, "PCoA", "bray") , color = "SampleType", title
```



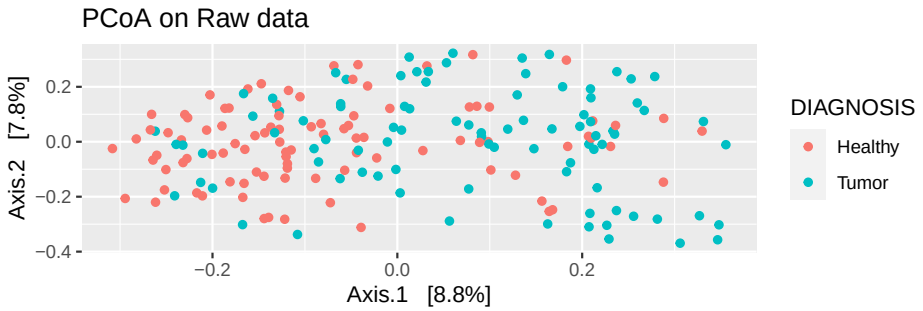

View how TMM normalization changes distance metrics differently than raw counts.

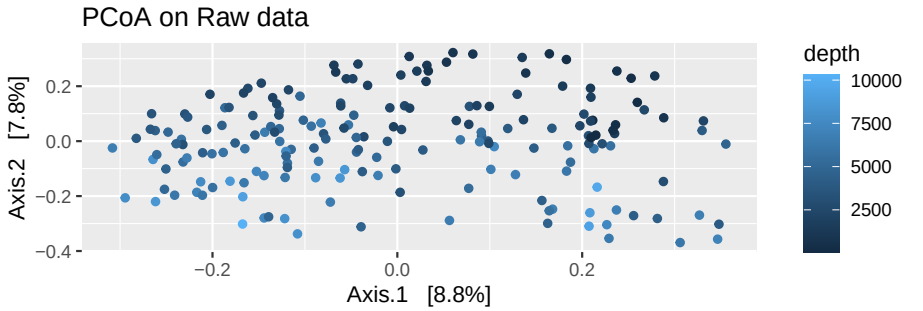
```
plot_norm_changes(gp_tmm, gp_raw,
                  x_lab = "Raw", y_lab = "TMM",
                  title = "Distance metric comparision between TMM normalization and Raw counts '
```



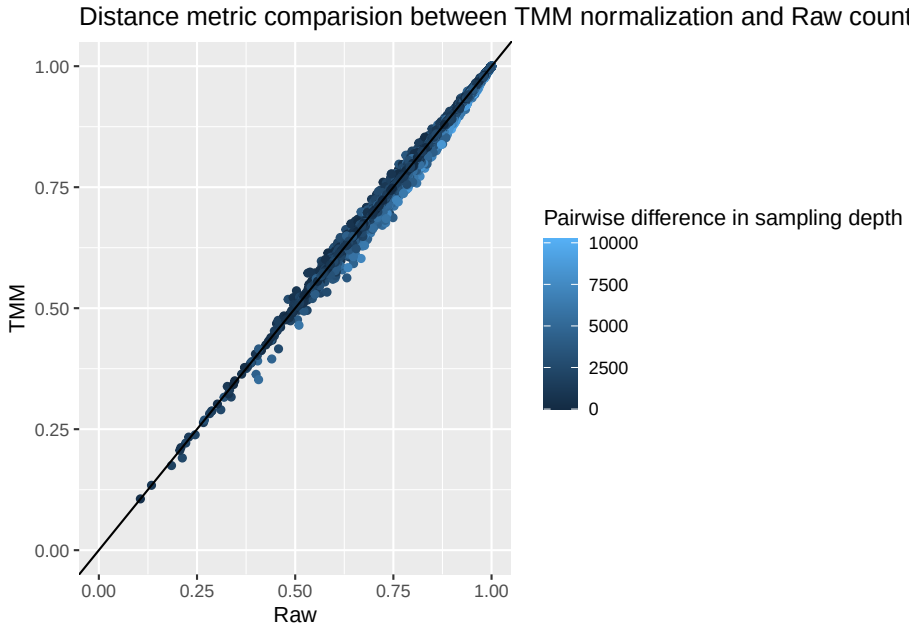

#### 5.3.2.3 TMM implementation on Kostic data

```
k_tmm <- norm_TMM(k_raw)
plot_ordination(
    k_tmm,
    phyloseq::ordinate(k_tmm, "PCoA", "bray") ,
    color = "DIAGNOSIS",
    title = "PCoA on Raw data"
) /
    plot_ordination(
        k_tmm,
        phyloseq::ordinate(k_tmm, "PCoA", "bray") ,
        color = "depth",
        title = "PCoA on Raw data"
    )
```




```
plot_norm_changes(k_tmm, k_raw,
                  x_lab = "Raw", y_lab = "TMM",
                  title = "Distance metric comparision between TMM normalization and Raw counts "
```



# Chapter 6

# Wrench

Wrench is a recent normalization method developed for microbiome data (Kumar et al., 2018) that uses an empirical Bayes Normalization approach. 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.

```
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)
}
```

# Chapter 7

# Log Ratio Transformation Methods

The log-ratio based methodology developed by Aitchison in the 1980s is useful for analyzing compositional data (Aitchison, 1982). 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 sampling depths. The sampling depth 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. One approach to this problem is 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 approach is to use 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 practice. 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 logarithm of zero count taxa.

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.

## 7.1 ANCOM II

ANCOM-II (Kaul et al., 2017) is a log-ratio approach for accounting for zero-inflation in log-ratio transformations. It uses a log-ratio transformation instead of library size normalization. If there are $m$ total taxa, this method performs $m-1$ differential abundance tests, each one using a different taxa as the reference taxa. Before the alr log-ratio transformation is performed, this method identifies and adjusts for three types of zeros that can exist. These three types of zeros are outlier zeros, structural zeros, and sampling zeros. Outlier zeros are zeros caused from some extraneous reasons separate from below detection limits due to sampling depth. To account for these zeros, they are replaced by NA in the data. Secondly, structural zeros are zeros which are zero due to the nature of the groups. If a taxa is structurally zero in a group, it is marked as automatically differentially abundant, and removed from further analysis. Finally, sampling zeros are other zeros that are zero perhaps caused by sampling depth. These are imputed by a small pseudocount. This method thus takes into account zero-inflation and compositionality.

After this correction of zeros, the data are log ratio transformed using each possible taxa as the reference sample. Tests for differential abundance are performed on each of iterations of the log-ratio transformed data, and the count of times the resulting test is significant is used to determining overall significance.

It is possible that no outlier or structural zeros are detected, so as a normalization/transformation technique, this method can be equivalent to alr or clr transoformation on raw data with a pseudocount.

```
## Load the Ancom function (it's not a package)
devtools::source_url("https://raw.githubusercontent.com/FrederickHuangLin/ANCOM/master/

norm_ANCOM2 <- function(ps, group_var, outlier.replace = TRUE) {
    feature_table <- as.data.frame(otu_table(ps))
    groups <- as.factor(get_variable(ps, group_var))
    meta_data <-
        data.frame(sample_id = paste0("sample", seq(dim(raw)[2])), groups = groups)
    #rownames(meta_data) <- meta_data$sample_id
    colnames(feature_table) <- meta_data$sample_id

    # The main ANCOM2 zero-classification function
    prepro <-
        feature_table_pre_process(
            feature_table,
            meta_data,
            sample_var = "sample_id",
            group_var = "groups",
            out_cut = 0.05,
            zero_cut = 0.9,
```

```
        lib_cut = 0,
        neg_lb = FALSE
    )
feature_table <- prepro$feature_table
meta_data <- prepro$meta_data
struc_zero <- prepro$structure_zeros

# OTU table transformation:
# (1) Discard taxa with structural zeros (if any); (2) Add pseudocount (1) and take logarithm
if (!is.null(struc_zero)) {
    num_struc_zero = apply(struc_zero, 1, sum)
    comp_table = feature_table[num_struc_zero == 0,]
} else{
    comp_table = feature_table
}
comp_table <- log(as.matrix(comp_table) + 1)

# Replace outlier zeros with the mean of the rest genes in each
if (outlier.replace) {
    for (col in 1:ncol(comp_table)) {
        comp_table[is.na(comp_table[, col]), col] <-
            mean(comp_table[, col], na.rm = T)
    }
}
# Calculate ratio
n_taxa <- dim(comp_table)[1]
taxa_id <- rownames(comp_table)
n_samp <- dim(comp_table)[2]
log_geo_mean <-
    apply(comp_table, 2, function(x) {
        mean(x, na.rm = T)
    })
mu_group_1 <- mean(log_geo_mean[groups == levels(groups)[1]])
mu_group_2 <- mean(log_geo_mean[groups == levels(groups)[2]])
mu <- rep(mu_group_1, n_samp)
mu[groups == levels(groups)[2]] <- mu_group_2
bias <- log_geo_mean - mu
dat.normed <-
    comp_table - matrix(rep(bias, n_taxa), nrow = n_taxa, byrow = T)
colnames(dat.normed) <- colnames(raw)


# Convert to phyloseq
otu <- otu_table(dat.normed, taxa_are_rows = T)
sam <- access(ps, "sam_data")
```

```
    sam$scaling_factor <- exp(bias)
    tax <- access(ps, "tax_table")
    phy <- access(ps, "phy_tree")
    ps_ancom2 <- phyloseq(otu, sam, tax, phy)
    return(ps_ancom2)
}


k_ancom2 <- norm_ANCOM2(k_raw, "DIAGNOSIS")
k_ancom2
```

## 7.2 ANCOM BC

ANCOM BC (Lin and Peddada, 2020) includes a normalization method that accounts for zero inflation and compositionality.

This method corrects bias by estimating a sampling fraction, which is…. It accounts for sampling fraction by introducing a sample-specific offset in a linear regression framework, estimated form observed data. The offset term is the bias correction, and the linear regression in log scale is analogous to the log ratio transformation to deal with compositionality.

This method also identifies and accounts for outlier zeros, structural zeros, and sampling zeros, using the same methodology as ANCOM2.

```
library(ANCOMBC)
norm_ANCOMBC <- function(ps, group_var) {
    out = ancombc(ps,
                  formula = group_var,
                  group = group_var,
                  struc_zero = T,
                  neg_lb = T,
                  zero_cut = .9)
    #it is recommended to set neg_lb = TRUE when the sample size per group is relative
    sampling.fraction <- exp(out$samp_frac)
    # Normalize counts
    # feature_table <- otu_table(ps, taxa_are_rows = T)
    feature_table <- out$feature_table
    # otu_norm <- t(t(otu) / sampling.fraction)
    otu_norm <- t(t(feature_table) / sampling.fraction)
    otu <- otu_table(otu_norm, T)
    sam <- access(ps, "sam_data")
    sam$scaling_factor <- sampling.fraction
    tax <- access(ps, "tax_table")
    phy <- access(ps, "phy_tree")
```

```
    ps_ancombc <- phyloseq(otu, sam, tax, phy)
    return(ps_ancombc)
}

k_ancombc <- norm_ANCOMBC(k_raw, group_var = "DIAGNOSIS")
```

```
## Warning in data_prep(phyloseq, group, zero_cut, lib_cut, global = global):
## The multi-group comparison will be deactivated as the group variable has < 3
## categories.
```

## 7.3 ALDEx2

ALDEx2 (Fernandes et al., 2014) is a method for differential abundance analysis developed for use with microbiome or any other sequencing setting. It uses a clr log-ratio transformation instead of library size normalization. To deal with sparsity, it creates randomized instances via Monte Carlo draws based on Dirichlet-Multinomial distribution with probabilities from the raw count data, These are zero-free, so log-ratio transformations are possible.

The idea is to estimate biological and sampling variation to calculate expected false discovery rate given variation. The method estimates technical variation within each sample using MC draws from Dirichlet distribution. For differential abundance, statistical tests are performed on each instance, p-values averaged, then corrected for multiple comparisons This method has been shown to be conservative compared to many of the library size normalization methods.

```
library(ALDEx2)
norm_ALDEx2 <- function(ps, group_var) {
    raw <- as.data.frame(otu_table(ps))
    groups <- get_variable(ps, group_var)
    # Run ALDEx2 function
    aldex.clr.res <-
        aldex.clr(
            reads = raw,
            conds = groups,
            mc.samples = 128
        )
    # Extract each clr MC draw
    mc_draws <- getMonteCarloInstances(aldex.clr.res)

    # Average each draw for the 'normalized' clr read table
    dat.normed <-
        matrix(0,
               nrow = nrow(mc_draws[[1]]) ,
               ncol = length(mc_draws))
```

```
    for (col in 1:ncol(dat.normed)) {
        dat.normed[, col] <- apply(mc_draws[[col]], 1, mean)
    }
    rownames(dat.normed) <- rownames(mc_draws[[1]])
    colnames(dat.normed) <- names(mc_draws)

    otu <- otu_table(dat.normed, T)
    sam <- access(ps, "sam_data")
    tax <- access(ps, "tax_table")
    phy <- access(ps, "phy_tree")
    ps_aldex2 <- phyloseq(otu, sam, tax, phy)
    return(ps_aldex2)
}

k_aldex2 <- norm_ALDEx2(k_raw, group_var = "DIAGNOSIS")
```

```
## operating in serial mode
```

```
## computing center with all features
```

# Chapter 8

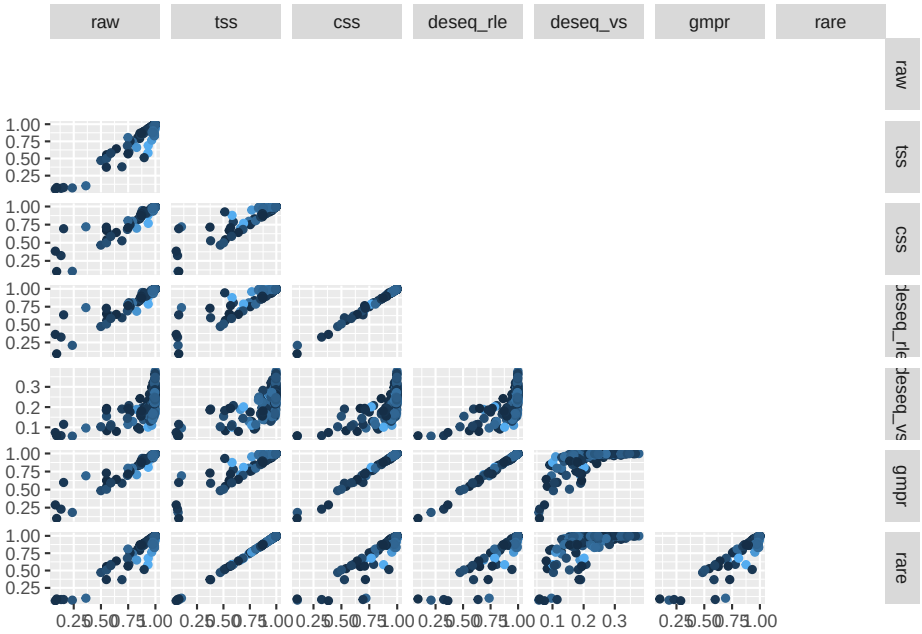
# Comparisions

See all distance comparisions for Global patterns data

```
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")),
           gmpr = as.numeric(phyloseq::distance(gp_gmpr, "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))
```




See all distance comparisons for Kostic data

```
data.frame(raw = as.numeric(phyloseq::distance(k_raw, "bray")),
           tss = as.numeric(phyloseq::distance(k_tss, "bray")),
           css = as.numeric(phyloseq::distance(k_css, "bray")),
           deseq_rle = as.numeric(phyloseq::distance(k_deseq_rle, "bray")),
           deseq_vs = as.numeric(phyloseq::distance(k_deseq_vs, "bray")),
           gmpr = as.numeric(phyloseq::distance(k_gmpr, "bray", na.rm = T)),
           rare = as.numeric(phyloseq::distance(k_rare, "bray")),
           tmm = as.numeric(phyloseq::distance(k_tmm, "bray")),
           diff = as.numeric(dist(get_variable(k_raw, "depth")))) %>%
    ggpairs(columns = 1:7, upper = "blank",
            diag = "blank",ggplot2::aes(colour=diff))
```

```
## Warning in vegdist(structure(c(14.7364903183976, 5.44797838541712,
## 0.281000656690941, : you have empty rows: their dissimilarities may be
## meaningless in method "bray"
```

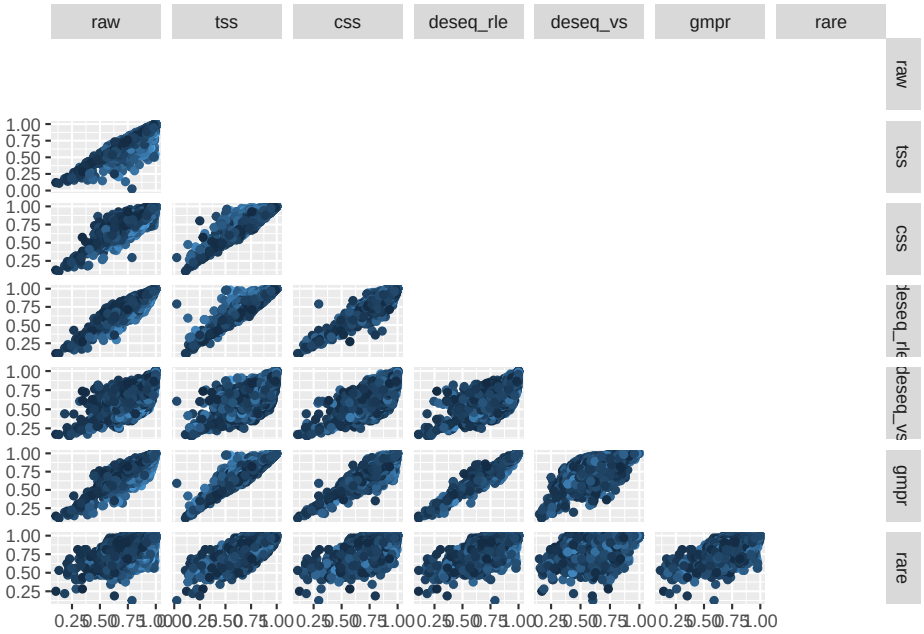
```
## Warning in vegdist(structure(c(14.7364903183976, 5.44797838541712,
## 0.281000656690941, : missing values in results
```

```
## Warning: Removed 367 rows containing missing values (geom_point).
```

```
## Warning: Removed 367 rows containing missing values (geom_point).
```

```
## Warning: Removed 367 rows containing missing values (geom_point).
```

```
## Warning: Removed 367 rows containing missing values (geom_point).

## Warning: Removed 367 rows containing missing values (geom_point).

## Warning: Removed 367 rows containing missing values (geom_point).
```



# Bibliography


Aitchison, J. (1982). The statistical analysis of compositional data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(2):139–160. _eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1982.tb01195.x.

Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106.

Caporaso, J., Lauber, C., Walters, W., Berg-Lyons, D., Lozupone, C., Turnbaugh, P., Fierer, N., and Knight, R. (2011). Global patterns of 16s rrna diversity at a depth of millions of sequences per sample. *Proceedings of the National Academy of Sciences*, 108(Supplement 1):4516–4522. Publisher: National Academy of Sciences Section: Colloquium Paper PMID: 20534432.

Chen, L., Reeve, J., Zhang, L., Huang, S., Wang, X., and Chen, J. (2018). Gmpr: A robust normalization method for zero-inflated count data with application to microbiome sequencing data. *PeerJ*, 6:e4600. Publisher: PeerJ Inc.

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.

Fernandes, A., Reid, J., Macklaim, J., McMurrough, T., Edgell, D., and Gloor, G. (2014). Unifying the analysis of high-throughput sequencing datasets: characterizing rna-seq, 16s rrna gene sequencing and selective growth experiments by compositional data analysis. *Microbiome*, 2:15. PMID: 24910773 PMCID: PMC4030730.

Kaul, A., Mandal, S., Davidov, O., and Peddada, S. (2017). Analysis of microbiome data in the presence of excess zeros. *Frontiers in Microbiology*, 0. Publisher: Frontiers.

Kostic, A., Gevers, D., Pedamallu, C., Michaud, M., Duke, F., Earl, A., Ojesina, A., Jung, J., Bass, A., Tabernero, J., Baselga, J., Liu, C., Shivdasani, R., Ogino, S., Birren, B., Huttenhower, C., Garrett, W., and Meyerson, M.

(2012). Genomic analysis identifies association of fusobacterium with colorectal carcinoma. *Genome Research*, 22(2):292–298. PMID: 22009990 PMCID: PMC3266036.

Kumar, M., Slud, E., Okrah, K., Hicks, S., Hannenhalli, S., and Corrada Bravo, H. (2018). Analysis and correction of compositional bias in sparse sequencing count data. *BMC Genomics*, 19(1):799.

Lin, H. and Peddada, S. (2020). Analysis of compositions of microbiomes with bias correction. *Nature Communications*, 11(1):3514. Bandiera_abtest: a Cc_license_type: cc_by Cg_type: Nature Research Journals Number: 1 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Computational biology and bioinformatics;Ecology;Microbiology Subject_term_id: computational-biology-and-bioinformatics;ecology;microbiology.

McKnight, D., Huerlimann, R., Bower, D., Schwarzkopf, L., Alford, R., and Zenger, K. (2019). Methods for normalizing microbiome data: An ecological perspective. *Methods in Ecology and Evolution*, 10(3):389–400. _eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13115.

McMurdie, P. and Holmes, S. (2013). phyloseq: An r package for reproducible interactive analysis and graphics of microbiome census data. *PLOS ONE*, 8(4):e61217. Publisher: Public Library of Science.

McMurdie, P. and Holmes, S. (2014). Waste not, want not: Why rarefying microbiome data is inadmissible. *PLOS Computational Biology*, 10(4):e1003531. Publisher: Public Library of Science.

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.

Robinson, M. and Oshlack, A. (2010). A scaling normalization method for differential expression analysis of rna-seq data. *Genome Biology*, 11(3):R25.

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.

Weiss, S., Xu, Z., Peddada, S., Amir, A., Bittinger, K., Gonzalez, A., Lozupone, C., Zaneveld, J., Vázquez-Baeza, Y., Birmingham, A., Hyde, E., and Knight, R. (2017). Normalization and microbial differential abundance strategies depend upon data characteristics. *Microbiome*, 5(1):27.