Microbiome analysis using nanopore reads

Marko Suokas

Preprocessing reads

Dorado does not support demultiplexing custom dual indexes located on both the 5' and 3' ends. In ligated libraries, these sequences can appear in either orientation. To address this, we use cutadapt for demultiplexing. Index pairs are identified using the linked adapters approach in both forward and reverse orientations. Afterward, a script is applied to reverse complement sequences in the reverse orientation. Finally, the forward and reverse reads are merged for each sample and primer sequences are trimmed.

Extract Forward Reads

Demultiplexing of forward reads can be performed with following command:

```
cutadapt -e 0 -O 12 -g file:~/scripts/barcodes.fasta --trimmed-only \
-m 1200 -o "fdemuxed/{name}.fastq.gz" reads.fastq.gz
```

The command extracts barcodes defined in the `barcodes.fasta` file and outputs matching reads into individual files within the `fdemuxed` subdirectory. In this example, the minimum read length is set to 1200 bp for 16S amplicons.

Extract Reverse Reads

Reverse reads can be demultiplexed by using using reverse-complemented barcode file:

```
cutadapt -e 0 -O 12 -g file:~/scripts/rev_barcodes.fasta \
--trimmed-only -m 1200 -o "rdemuxed/{name}.fastq.gz" \
reads.fastq.gz
```

The reads are demultiplexed into a rdemuxed directory.

Tip: Parameters `-O`, `-e`, `-m`, and `-M` can help reduce the chances of mismatched alignments.

Reverse Complement Reverse Reads

Next, we use a bash script to reverse complement each reverse orientation read file using the following command:

```
seqkit seq -rp --seq-type DNA -o reverse_comp.fastq.gz \
reverse_out.fastq.gz
```

Merging Forward and Reverse Reads

Next, forward and reverse reads with the same base name are merged from two directories. Here's a simple command for that:

```
cat forward_out.fastq.gz reverse_comp.fastq.gz >merged_reads.fastq.gz
```

Trimming Primers

Finally, `cutadapt` and bash script can be employed to trim forward and reverse PCR primers from the sequence reads.

Read quality

Nanopore sequencing Phred scores should be interpreted with caution, as per-base accuracy cannot be directly determined from the electrical signal. Instead, scores are estimates based on the confidence of the basecalling model rather than a direct probability of sequencing error. As a result, traditional quality filtering approaches may not always be appropriate.

ONT instead calculates an estimated cumulative error rate and converts it into an average quality score per read. The following code computes this metric and generates a violin plot showing the distribution of average sequence quality per read for each sample.

```
# Define path
source_dir <- "data/reads/set2"
# A function to calculate so called AvgQual
extract_nano_qscores <- function(file) {
    # Read fastq
    fq <- readFastq(file)
    # Extract q-values to a matrix
    qmat <- as(quality(fq), "matrix")
    # Convert values to error probabilities
    error_probs <- 10^(-qmat/10)
    # Compute number of expected errors per read
    total_errors <- rowSums(error_probs, na.rm = T)
    # Length per read
```

```r
    read_lenghts <- rowSums(!is.na(qmat))
    # Compute "ONT-style" Q-score
    ont_qscores <- -10*log10(total_errors/read_lenghts)
    # Trim file names
    sampleid <- sub("_trimmed\\.fastq\\.gz$", "", basename(file))

    # Create a dataframe
    qdata <- tibble(
        nanopore_qscore = ont_qscores,
        sampleid = sampleid)
    return(qdata)
}

# Create list of files
files <- list.files(source_dir, pattern = "\\.fastq\\.gz$",
                    full.names = T)
# Apply function and use map_dfr to combine dataframes together
qscore_data <- map_dfr(files, extract_nano_qscores)
# Save result to a rds file
saveRDS(qscore_data, "set2/nanopore_quality.rds")
```

Violin plot

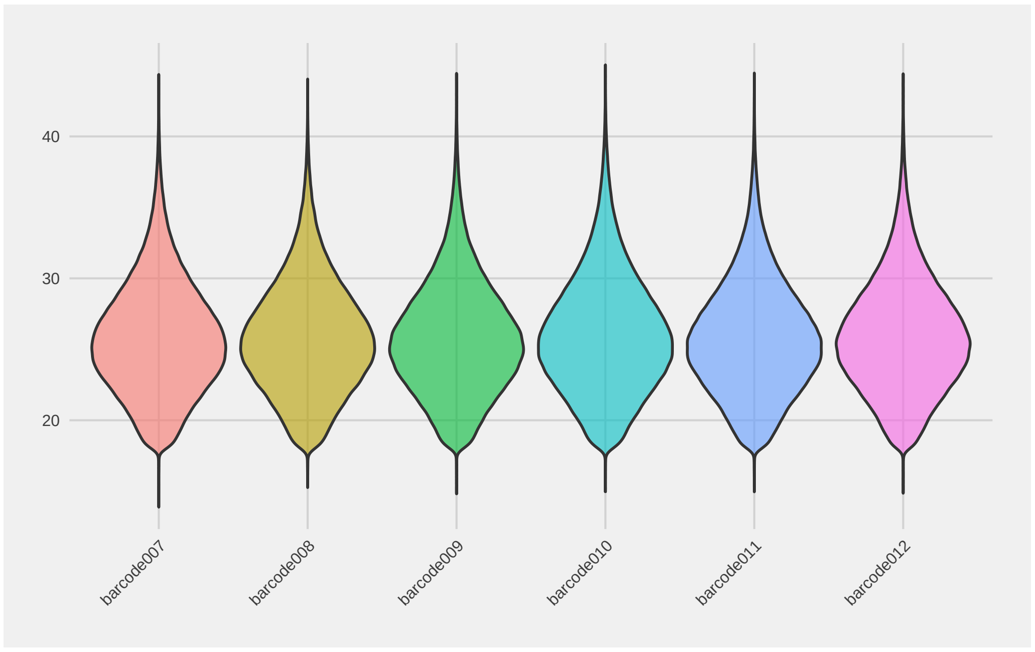
```r
# Read data
qdata <- readRDS("set2/nanopore_quality.rds")

# Divide samples to groups of 12
sample_list <- unique(qdata$sampleid)
sample_groups <- split(sample_list, ceiling(seq_along(sample_list) / 12))

# Generate a separate plot for each group
plots <- map(sample_groups, function(group_samples) {
  ggplot(filter(qdata, sampleid %in% group_samples),
         aes(x = sampleid, y = nanopore_qscore, fill = sampleid)) +
    geom_violin(scale = "width", alpha = 0.6) +
    theme_fivethirtyeight(base_size=8) +
    labs(x = "Sample", y = "Nanopore Q-score") +
    theme(legend.position = "none", axis.text.x = element_text(angle = 45, hjust = 1))
})

# Combine all plots into a single vertical layout
final_plot <- wrap_plots(plots, ncol = 1)

# Show the plot
print(final_plot)
```



Import sequence data to Qiime 2

QIIME2 import requires manifest.csv that defines names and paths of sequence files.

```
# Activate qiime environment
export PATH="/homedir04/msuokas/miniconda3/bin:$PATH"
source activate qiime-2024.10

# Import sequence data
qiime tools import \
  --type 'SampleData[SequencesWithQuality]' \
  --input-path data/reads/set2/manifest.csv \
  --output-path data/set2/demux.qza \
  --input-format SingleEndFastqManifestPhred33
```

Dereplicate sequences

Dereplication removes unnecessary redundancy from sequence files

```
# Activate qiime environment
export PATH="/homedir04/msuokas/miniconda3/bin:$PATH"
source activate qiime-2024.10
# Dereplicate sequences with vsearch plugin
```

```
qiime vsearch dereplicate-sequences \
    --p-min-seq-length 1200 \
    --o-dereplicated-table data/set2/derep_table.qza \
    --o-dereplicated-sequences data/set2/derep_seq.qza \
    --i-sequences data/set2/demux.qza
```

Pick de-novo features

Otu clustering is performed at 97 % identity level.

```
# Activate qiime environment
export PATH="/homedir04/msuokas/miniconda3/bin:$PATH"
source activate qiime-2024.10

# Run denovo-clustering
qiime vsearch cluster-features-de-novo \
   --i-sequences data/set2/derep_seq.qza \
   --i-table data/set2/derep_table.qza \
   --p-strand plus --p-threads 10 --p-perc-identity 0.97 \
   --o-clustered-table data/set2/clustered_table \
   --o-clustered-sequences data/set2/clustered_seq.qza
```

Remove rare features

Rare otus are removed from results before chimera detection

```
# Activate qiime environment
export PATH="/homedir04/msuokas/miniconda3/bin:$PATH"
source activate qiime-2024.10

# QIIME 2 command to filter rare features
qiime feature-table filter-features \
    --i-table data/set2/clustered_table.qza \
    --p-min-frequency 10 \
    --o-filtered-table data/set2/n10_table.qza
qiime feature-table filter-seqs \
    --i-data data/set2/clustered_seq.qza \
    --i-table data/set2/n10_table.qza \
    --o-filtered-data data/set2/n10_seq.qza
```

Detect ja remove chimeric features

```
# Activate qiime environment
export PATH="/homedir04/msuokas/miniconda3/bin:$PATH"
source activate qiime-2024.10

qiime vsearch uchime-denovo --i-sequences data/set2/n10_seq.qza \
    --i-table data/set2/n10_table.qza  \
    --o-chimeras data/set2/chimeras.qza --o-stats data/set2/stats.qza \
    --o-nonchimeras data/set2/nonchimeras.qza

# Keep nonchimeric features
 qiime feature-table filter-features \
    --i-table data/set2/n10_table.qza \
    --m-metadata-file data/set2/nonchimeras.qza \
    --o-filtered-table data/set2/otu_table.qza

# Keep nonchimeric sequences
qiime feature-table filter-seqs \
   --i-data data/set2/n10_seq.qza \
   --i-table data/set2/otu_table.qza \
   --o-filtered-data data/set2/otu_seq.qza
```

Import qiime otu table and associated metadata

```
#sequence file
tse <- importQIIME2(featureTableFile = "data/set2/otu_table.qza")
tse <- tse[, sort(colnames(tse))]
#add metadata
metadata <- data.frame(read_tsv("data/set2_meta.tsv",
                                show_col_types = F))
metadata <- column_to_rownames(metadata, "Sampleid")
colData(tse) <- DataFrame(metadata)
tse
```

```
class: TreeSummarizedExperiment
dim: 1469 6
metadata(0):
assays(1): counts
rownames(1469): 780015658d5994b3e7649355028cdb0ede4aa40e
  92901274c2b00cc23f7a06885764bc41e8da85ec ...
  0ba42ae60d68d1bea239c158de0013e7e169b879
  586e676bc2eba711e9befb7156796a22819d5f91
rowData names(0):
colnames(6): barcode007 barcode008 ... barcode011 barcode012
colData names(2): Name Media
```

```
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
```

Qiime arranges sequence file alphabetically, while TSE expects sequences in same order as rownames. Thus we need to rearrange sequences

```r
ref_sequences <- importQZA("data/set2/otu_seq.qza")
ref_ids <- names(ref_sequences)
tse_ids <- rownames(tse)
# Check if all rownames are present in the reference IDs
if (!all(tse_ids %in% ref_ids)) {
  stop("Not all rownames from tse are present in the reference sequences.")
}
# Reorder `ref_sequences` to match the order of `tse` rownames
ref_sequences_ordered <- ref_sequences[match(tse_ids, ref_ids)]
all(names(ref_sequences_ordered) == rownames(tse))
```

```
[1] TRUE
```

```r
referenceSeq(tse) <- ref_sequences_ordered
```

Assign taxonomy

We are using SILVA nr99 database for assignTaxonomy function.

```r
taxa <- assignTaxonomy(referenceSeq(tse), minBoot=90, multithread=10,
        refFasta="~/reference/silva_nr99_v138.1_train_set.fa.gz")
saveRDS(taxa, "data/set2/taxa.rds" )
```

Add taxonomy results to rowData and rename identifiers

```r
taxa <- readRDS("data/set2/taxa.rds")
#Add taxonomy
rownames(taxa) <- NULL
rowData(tse) <- DataFrame(taxa)
#Rename rows (alternative to Silva ID)
rownames(tse) <- paste0("OTU_", seq_len(nrow(tse)))
tse
```

```
class: TreeSummarizedExperiment
dim: 1469 6
metadata(0):
assays(1): counts
rownames(1469): OTU_1 OTU_2 ... OTU_1468 OTU_1469
rowData names(6): Kingdom Phylum ... Family Genus
colnames(6): barcode007 barcode008 ... barcode011 barcode012
colData names(2): Name Media
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
rowLinks: NULL
rowTree: NULL
colLinks: NULL
colTree: NULL
referenceSeq: a DNAStringSet (1469 sequences)
```

Write object data to files

Write RDS. The object can be easily reloaded in R

```
saveRDS(tse, "set2/tse.rds")
```

Write an abundance table

```
#FeatureID will be rowname
abd <- data.frame(FeatureID = rownames(tse),assays(tse)$counts)
#Write
write_tsv(abd, "set2/feature_table.tsv")
```

Write a taxonomy table

```
#FeatureID will be rowname
taxt <- data.frame(FeatureID = rownames(tse), rowData(tse))
#Write
write_tsv(taxt, "set2/taxonomy.tsv")
```

Write variant sequences to fasta file

```
writeXStringSet(referenceSeq(tse), "set2/repseq.fasta",
                                     append = F, compress = F,
                                     format = "fasta")
```

Write a metadata file

```
metadf <- data.frame(colData(tse)) %>% rownames_to_column(var="Sampleid")
#write
write_tsv(metadf, "set2/metadata.tsv")
```

Microbial data analysis

Agglomerate taxonomy to genus rank and count relative abundance

```
altExp(tse, "Genus") <- agglomerateByRank(tse, rank="Genus",
                             onRankOnly=T, na.rm=F)
#relabundance
altExp(tse, "Genus") <- transformAssay(altExp(tse, "Genus"),
                                       assay.type="counts",
                                       method="relabundance")
```

Pick ten most abundant features

```
#top10 features
top10 <- getTop(altExp(tse, "Genus"), top=10,
                        method = "mean",
                        assay.type="relabundance")
#create and filter table
table <- data.frame(assays(altExp(tse, "Genus"))$relabundance)
table <- table %>%
  rownames_to_column(var = "Genus") %>%
  filter(Genus %in% top10) %>% bind_rows(
  summarise(., Genus = "Others", across(where(is.numeric), ~ 1 - sum(.))))
```

Print abundance table

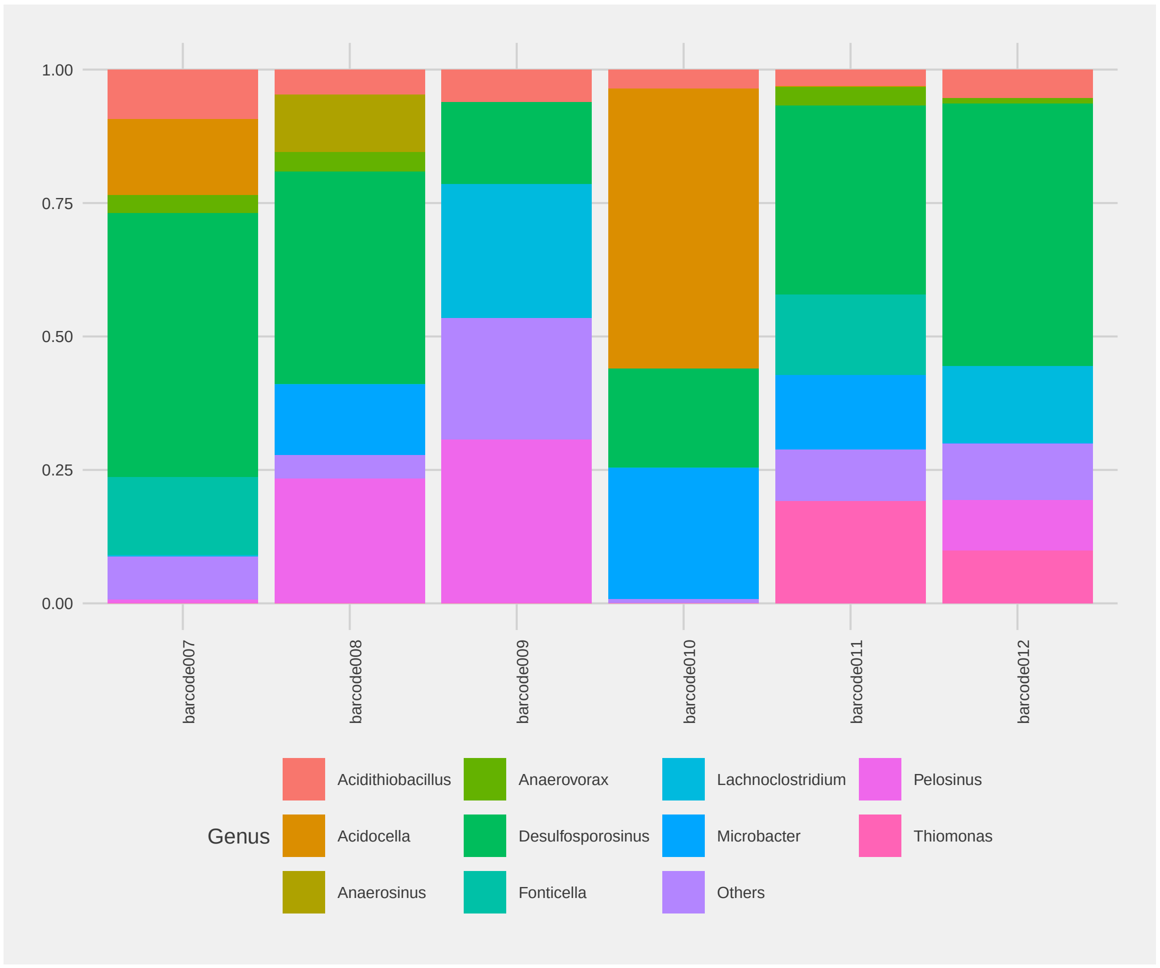
```
kable(table, digits=2) %>%
  kable_styling(latex_options = c("HOLD_position", "striped"),
              font_size = 10) %>%
  row_spec(0, background = "teal", color = "white")
```

Genus	barcode007	barcode008	barcode009	barcode010	barcode011	barcode012
Acidithiobacillus	0.09	0.05	0.06	0.04	0.03	0.05
Acidocella	0.14	0.00	0.00	0.53	0.00	0.00
Anaerosinus	0.00	0.11	0.00	0.00	0.00	0.00
Anaerovorax	0.03	0.04	0.00	0.00	0.03	0.01
Desulfosporosinus	0.49	0.40	0.15	0.19	0.35	0.49
Fonticella	0.15	0.00	0.00	0.00	0.15	0.00
Lachnoclostridium	0.00	0.00	0.25	0.00	0.00	0.14
Microbacter	0.00	0.13	0.00	0.25	0.14	0.00
Pelosinus	0.01	0.23	0.31	0.00	0.00	0.10
Thiomonas	0.00	0.00	0.00	0.00	0.19	0.10
Others	0.08	0.04	0.23	0.01	0.10	0.11

Composition plot

Transform data to long table format and create a barplot.

```
df_long <- table %>% pivot_longer(cols = starts_with("barcode"),
names_to = "Sample", values_to = "Abundance")
#Plot stacked barplot
ggplot(df_long, aes(x=Sample, y=Abundance, fill=Genus)) +
geom_bar(stat = "identity") +
   theme_fivethirtyeight(base_size=8) +
   theme(axis.text.x=element_text(angle=90))
```



Sample sizes

Prior diversity calculations, we check total count differences between samples.

```
totalcounts <- colSums(assays(tse)$counts)
totalcounts
```

```
barcode007 barcode008 barcode009 barcode010 barcode011 barcode012
    345406     212800     361049     367188     320530     273280
```

Alpha diversity

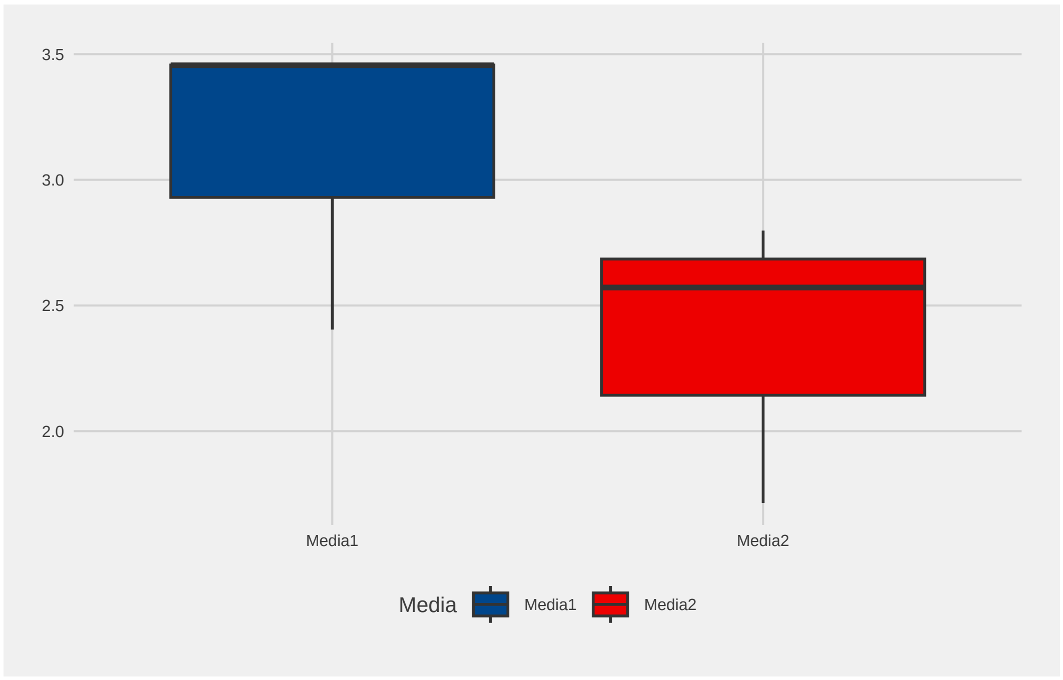
```
tse <- readRDS("set2/tse.rds")
tse <- addAlpha(tse, assay.type="counts", index="shannon_diversity", sample=212800)
shannon <- data.frame(Samples = colnames(tse),
                      Shannon_index = colData(tse)$shannon_diversity)
rownames(shannon) <- NULL
#colnames(shannon) <- c("Sample", "Shannon index")
#table
kable(shannon, digits=2, caption = "Shannon index") %>%
kable_styling(latex_options = c("HOLD_position", "striped"),
              font_size = 11) %>% row_spec(0, background = "teal",
                                             color = "ivory")
```

Table 2: Shannon index

Samples	Shannon_index
barcode007	3.46
barcode008	3.46
barcode009	2.40
barcode010	1.71
barcode011	2.80
barcode012	2.57

Shannon boxplot between two media groups

```
comparisons <- list(c("Media1", "Media2"))
media <- ggplot(colData(tse), aes(x=Media, y=shannon_diversity, fill=Media)) +
  geom_boxplot(ylim=c(0,4)) +
  theme_fivethirtyeight( base_size=8) + scale_fill_lancet()
media
```

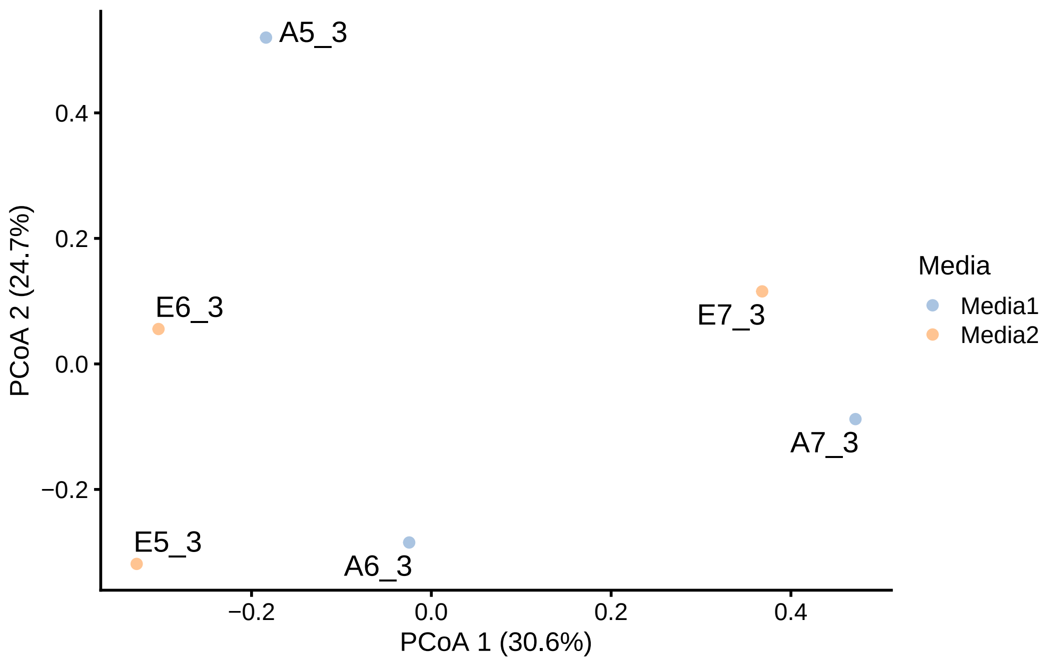


Beta diversity measured by Bray-Curtis dissimilarity

PCoA plot illustrating community composition differences using Bray-Curtis dissimilarity distances

```
tse <- runMDS(tse, FUN = getDissimilarity, method = "bray", assay.type = "counts",
name = "MDS_bray", sample=212800)

#Explained variance
e <- attr(reducedDim(tse, "MDS_bray"), "eig")
rel_eig <- e / sum(e[e > 0])
#Plot
plotReducedDim(tse, "MDS_bray", colour_by = "Media",
               text_by = "Name", text_size = 4) +
  labs(x = paste("PCoA 1 (", round(100 * rel_eig[[1]], 1),
                 "%", ")", sep = ""), y = paste("PCoA 2 (",
                 round(100 * rel_eig[[2]], 1),
                 "%", ")", sep = ""))
```



PCoA plot illustrates composition differences between samples using Bray-Curtis dissimilarity distance calculations.

Results

The read accuracy of Oxford Nanopore sequencers has improved significantly with the introduction of V14 chemistry and R10.4 flow cells. This is evident in read quality plots, which indicate a mean read accuracy of 99–99.5%. However, this does not change the fact that single-base accuracy cannot be predicted in the same way as with sequencers using SBS technology and signal capture at each flow. This inherent unpredictability makes it unsuitable to apply popular denoising algorithms in microbial population studies.

Here, we test a more traditional de novo OTU-picking strategy instead of mapping reads to a bacterial reference database. Our results suggest that, with the combination of long read lengths and improved accuracy, de novo clustering at 97% identity can accurately determine microbial populations. The main drawback appears to be an increase in low-abundance features, which are likely noise rather than biologically relevant features. This issue was also present in short-read sequencing data before the introduction of denoising algorithms. The advantage of de novo clustering over reference-based approaches is its ability to identify features that are underrepresented or absent in reference databases.

Although not shown here, using the Emu microbiome profiler for Nanopore reads resulted in roughly half as many detected features compared to OTU clustering, including those from closed-reference clustering against the SILVA database.

Shannon index values suggest that the samples contain enriched microbial communities, which aligns with expectations. There may be some differences in richness between culture media. While beta diversity analyses did not reveal significant similarities, the PCoA plot effectively captures Bray-Curtis distance differences between samples.

In conclusion, OTU clustering appears to be a viable approach for analyzing Nanopore sequencing data. Choice of analysis method rarely changes composition of most abundant features significantly. When measured in absolute numbers, alpha diversity or richness indices are most obviously affected. Perhaps also beta diversity indices in a lesser extent. However, it is extremely unlikely that analysis of nanopore sequences would result to incorrect conclusion, when studying two or more distinct microbial communities.