Supplementary_Methods–Infant Urobiome

Seth Reasoner Grace Morales Maria Hadjifrangiskou

2023-07-10

Contents

Introduction

This file includes documentation detailing the code used for bioinformatics and statistical analysis accompanying the manuscript **Survey of the Infant Male Urobiome and Genomic Analysis of Actinotignum spp.**. Catheterized urine samples were collected from 50 healthy male infants. Following DNA extraction, the V4 hypervariable 16S rRNA region was amplified using the 515F and 806R primers. Sequences can be accessed under BioProject PRJNA912725. Supplementary Table S7 includes all sample accession information. Two sequencing runs were completed–one on a Illumina MiSeq and one on a Illumina NovaSeq. These runs and the samples sequenced are detailed in the manuscript text and Supplementary Table S1 & S7. Initial bioinformatic processing of 16S rRNA reads will be presented twice, once for each run. As detailed in Supplementary Table S7, **each sequencing run included 8 extraction blanks, 4 PCR negative controls, and 8 serial dilutions of a mock microbial community**.

Processing of V4 16S rRNA reads using dada2

Processing of 16S rRNA reads from Illumina MiSeq run

```
##Load requisite libraries (libraries used in each code chunk are re-called)
library("tidyverse")
```

```
library("dada2")
library("here")
library("phyloseq")

##We would like to credit the following tutorial for guiding this code:
##"https://ycl6.github.io/16S-Demo/index.html"

##Create Folders
MiSeq_fastq = "/MiSeq_fastq/"         # raw fastq files
MiSeq_filtered = "/MiSeq_filtered"   # dada2 trimmed fastq files
MiSeq_output = "/MiSeq_output"       # output files
MiSeq_images = "/MiSeq_images"        # output images

if(!dir.exists(MiSeq_filtered)) dir.create(MiSeq_filtered)
if(!dir.exists(MiSeq_output)) dir.create(MiSeq_output)
if(!dir.exists(MiSeq_images)) dir.create(MiSeq_images)

##Create List of File Names
fns_MiSeq = sort(list.files(MiSeq_fastq, full.names = TRUE))
fnFwds_MiSeq = fns_MiSeq[grep("R1.fastq", fns_MiSeq)]
fnRevs_MiSeq = fns_MiSeq[grep("R2.fastq", fns_MiSeq)]

MiSeq_sample_names = gsub("_R1.fastq", "", basename(fnFwds_MiSeq))

##Visualize Read Quality
ii = 1:length(MiSeq_sample_names)
pdf(paste0(MiSeq_images, "/plotQualityProfile.pdf"), width = 8, height = 8, pointsize = 12)
for(i in ii) {
  message(paste0("[", i ,"/", length(MiSeq_sample_names), "] ", MiSeq_sample_names[i]))
  print(plotQualityProfile(fnFwds_MiSeq[i]) + ggtitle("Fwd"))
  print(plotQualityProfile(fnRevs_MiSeq[i]) + ggtitle("Rev"))
}
invisible(dev.off())

##Filter and Trim
filtFwds_MiSeq = file.path(MiSeq_filtered, basename(fnFwds_MiSeq))
filtRevs_MiSeq = file.path(MiSeq_filtered, basename(fnRevs_MiSeq))

out_MiSeq = filterAndTrim(fnFwds_MiSeq, filtFwds_MiSeq, fnRevs_MiSeq, filtRevs_MiSeq,
                    truncLen = c(140,140), minLen = 120, maxN = 0, truncQ = 2, maxEE =c(2,2),
                    rm.phix = TRUE, compress = TRUE, verbose = TRUE, multithread = TRUE)

out_MiSeq = as.data.frame(out_MiSeq)
rownames(out_MiSeq) = MiSeq_sample_names
head(out_MiSeq, 10)

##Learn the Error Rates
errFwd_MiSeq = learnErrors(filtFwds_MiSeq, multithread = TRUE)
errRev_MiSeq = learnErrors(filtRevs_MiSeq, multithread = TRUE)

pdf(paste0(MiSeq_images, "/plotErrors.pdf"), width = 10, height = 10, pointsize = 12)
plotErrors(errFwd_MiSeq, nominalQ = TRUE)
plotErrors(errRev_MiSeq, nominalQ = TRUE)
```

```
invisible(dev.off())

##Sample Inference
dadaFwds_MiSeq = dada(filtFwds_MiSeq, err = errFwd_MiSeq, pool = FALSE, multithread = TRUE)
dadaRevs_MiSeq = dada(filtRevs_MiSeq, err = errRev_MiSeq, pool = FALSE, multithread = TRUE)

##Merge Paired Reads
merged_MiSeq = mergePairs(dadaFwds_MiSeq, filtFwds_MiSeq, dadaRevs_MiSeq, filtRevs_MiSeq, verbose = TRU

##Construct the Sequence Table
seqtab_MiSeq <- makeSequenceTable(merged_MiSeq)

table(nchar(getSequences(seqtab_MiSeq)))

##Remove Chimeras
seqtab_nochim_MiSeq <- removeBimeraDenovo(seqtab_MiSeq, method="consensus", multithread=2, verbose=TRUE
##Percent of sequences that are non-chimeric
sum(seqtab_nochim_MiSeq)/sum(seqtab_MiSeq)

rownames(seqtab_nochim_MiSeq) <- MiSeq_sample_names

##Track Reads through the Pipeline
getN <- function(x) sum(getUniques(x))
track_MiSeq <- cbind(out_MiSeq, sapply(dadaFwds_MiSeq, getN), sapply(dadaRevs_MiSeq, getN), sapply(merg
colnames(track_MiSeq) <- c("input", "filtered", "denoisedF", "denoisedR", "merged","nonchim")
rownames(track_MiSeq) <- MiSeq_sample_names
head(track_MiSeq)

##Assign Taxonomy
seqs_MiSeq = getSequences(seqtab_nochim_MiSeq)
dbpath = "/16S_DB/"
ref_db = paste0(dbpath, "silva_nr99_v138.1_train_set.fa.gz")

taxonomy_tab_MiSeq <- assignTaxonomy(seqs_MiSeq, refFasta = ref_db)


##Create Phyloseq Object
MiSeq_phyloseq_unfiltered <- phyloseq(otu_table(seqtab_nochim_MiSeq, taxa_are_rows=FALSE),
               tax_table(taxonomy_tab_MiSeq))
```

Processing of 16S rRNA reads from Illumina NovaSeq run

```
##Load requisite libraries
library("tidyverse")
library("dada2")
library("here")
library("phyloseq")

##Create Folders
Nova_fastq = "/Nova_fastq/"         # raw fastq files
Nova_filtered = "/Nova_filtered"   # dada2 trimmed fastq files
Nova_output = "/Nova_output"       # output files
Nova_images = "/Nova_images"        # output images
```

```
if(!dir.exists(Nova_filtered)) dir.create(Nova_filtered)
if(!dir.exists(Nova_output)) dir.create(Nova_output)
if(!dir.exists(Nova_images)) dir.create(Nova_images)

##Create List of File Names
fns_Nova = sort(list.files(Nova_fastq, full.names = TRUE))
fnFwds_Nova = fns_Nova[grep("R1.fastq", fns_Nova)]
fnRevs_Nova = fns_Nova[grep("R2.fastq", fns_Nova)]

Nova_sample_names = gsub("_R1.fastq", "", basename(fnFwds_Nova))

##Visualize Read Quality
ii = 1:length(Nova_sample_names)
pdf(paste0(Nova_images, "/plotQualityProfile.pdf"), width = 8, height = 8, pointsize = 12)
for(i in ii) {
  message(paste0("[", i ,"/", length(Nova_sample_names), "] ", Nova_sample_names[i]))
  print(plotQualityProfile(fnFwds_Nova[i]) + ggtitle("Fwd"))
  print(plotQualityProfile(fnRevs_Nova[i]) + ggtitle("Rev"))
}
invisible(dev.off())

##Filter and Trim
filtFwds_Nova = file.path(Nova_filtered, basename(fnFwds_Nova))
filtRevs_Nova = file.path(Nova_filtered, basename(fnRevs_Nova))

out_Nova = filterAndTrim(fnFwds_Nova, filtFwds_Nova, fnRevs_Nova, filtRevs_Nova,
                    truncLen = c(140,140), minLen = 120, maxN = 0, truncQ = 2, maxEE =c(2,2),
                    rm.phix = TRUE, compress = TRUE, verbose = TRUE, multithread = TRUE)

out_Nova = as.data.frame(out_Nova)
rownames(out_Nova) = Nova_sample_names
head(out_Nova, 10)

##Learn the Error Rates
errFwd_Nova = learnErrors(filtFwds_Nova, multithread = TRUE)
errRev_Nova = learnErrors(filtRevs_Nova, multithread = TRUE)

pdf(paste0(Nova_images, "/plotErrors.pdf"), width = 10, height = 10, pointsize = 12)
plotErrors(errFwd_Nova, nominalQ = TRUE)
plotErrors(errRev_Nova, nominalQ = TRUE)
invisible(dev.off())

##Sample Inference
dadaFwds_Nova = dada(filtFwds_Nova, err = errFwd_Nova, pool = FALSE, multithread = TRUE)
dadaRevs_Nova = dada(filtRevs_Nova, err = errRev_Nova, pool = FALSE, multithread = TRUE)

##Merge Paired Reads
merged_Nova = mergePairs(dadaFwds_Nova, filtFwds_Nova, dadaRevs_Nova, filtRevs_Nova, verbose = TRUE)

##Construct the Sequence Table
seqtab_Nova <- makeSequenceTable(merged_Nova)

table(nchar(getSequences(seqtab_Nova)))
```

```
##Remove Chimeras
seqtab_nochim_Nova <- removeBimeraDenovo(seqtab_Nova, method="consensus", multithread=2, verbose=TRUE)
##Percent of sequences that are non-chimeric
sum(seqtab_nochim_Nova)/sum(seqtab_Nova)

rownames(seqtab_nochim_Nova) <- Nova_sample_names

##Track Reads through the Pipeline
getN <- function(x) sum(getUniques(x))
track_Nova <- cbind(out_Nova, sapply(dadaFwds_Nova, getN), sapply(dadaRevs_Nova, getN), sapply(merged_N
colnames(track_Nova) <- c("input", "filtered", "denoisedF", "denoisedR", "merged","nonchim")
rownames(track_Nova) <- Nova_sample_names
head(track_Nova)

##Assign Taxonomy
seqs_Nova = getSequences(seqtab_nochim_Nova)
dbpath = "/16S_DB/"
ref_db = paste0(dbpath, "silva_nr99_v138.1_train_set.fa.gz")

taxonomy_tab_Nova <- assignTaxonomy(seqs_Nova, refFasta = ref_db)

##Create Phyloseq Object
Nova_phyloseq_unfiltered <- phyloseq(otu_table(seqtab_nochim_Nova, taxa_are_rows=FALSE),
               tax_table(taxonomy_tab_Nova))
```

Using Decontam to remove potential contaminant sequences

We use the R package Decontam (citation: PMID 30558668) to identify sequences that are more prevalent in the negative controls (sampling controls, extraction blanks and PCR blanks) than the patient samples. First, using the known composition of the mock microbial community from each sequencing run, we benchmarked Decontam performance to retain true sequence variants. Next, we applied the same threshold to patient samples to remove contaminants.

Merge Phyloseq Objects from Separate Sequencing Runs

```
##Load requisite libraries
library("tidyverse")
library("phyloseq")

merged_output <- merge_phyloseq(MiSeq_phyloseq_unfiltered, Nova_phyloseq_unfiltered)

# add metadata
complete_metadata<- read.csv("https://raw.githubusercontent.com/reaset41/Infant-Urobiome/main/metadata_

sample_data(merged_output) <- complete_metadata

# this merged phyloseq object is available on the associated github
urobiome_phyloseq <- merged_output

# export ASV table prior to decontamination (Table S2)
full_tax <- tax_table(urobiome_phyloseq)
full_ASV <- otu_table(urobiome_phyloseq)
```

```
TableS2 <- cbind(full_tax,t(full_ASV))

write.csv(TableS2, "TableS2.csv", row.names=TRUE)
```

Applying Decontam to mock communities to identify optimal Decontam threshold

```
##Load requisite libraries
library("tidyverse")
library("decontam")
library("phyloseq")

##We would like to credit the following publication for guiding this code: Controlling for Contaminants
##https://lakarstens.github.io/ControllingContaminants16S/Analyses/ControllingContaminants16S_decontam.

##Using merged phyloseq object created above
urobiome_phyloseq

##Create subset with only mock communities
mock_phyloseq <-subset_samples(urobiome_phyloseq, SampleType=="Mock")
#mock community includes 8 bacteria
sorted <- head(sort(colSums(otu_table(mock_phyloseq)), decreasing = TRUE), 8)
mock_taxa <- cbind(as.data.frame(tax_table(mock_phyloseq)[names(sorted),]), Count = sorted)

##Create subset of data with only mock and negative controls
mock_blanks <- subset_samples(urobiome_phyloseq, !SampleType=="PATIENT")

##extract out original ASV table
original_ASVs <- as.matrix(as.data.frame(otu_table(mock_blanks)))

##identify the original proportion of contaminants
contaminants_original <- rowSums(original_ASVs[,!colnames(original_ASVs) %in% rownames(mock_taxa)])
##identify the original proportion of mock community ASVs
mock_original <- rowSums(original_ASVs[,colnames(original_ASVs) %in% rownames(mock_taxa)])

##Evaluate different thresholds of Decontam
contam_Mocks <- isNotContaminant(mock_blanks, neg = "is.neg", method = "prevalence", threshold = 0.4, n

##create a phyloseq object without contaminants
mock_blank_true <- prune_taxa(contam_Mocks$not.contaminant, mock_blanks)

##remove ASVs <1% prevalence in entire dataset
filter_level_mocks <- colSums(otu_table(mock_blank_true))>(0.01*sum(rowSums(otu_table(mock_blank_true))
mocks_final <-prune_taxa(filter_level_mocks, mock_blank_true)

##subset out the ASV table of recovered ASVs
recovered_ASVs <- as.matrix(as.data.frame(otu_table(mock_blank_true)))

##create a subset of removed ASVs
removed_ASVs <- as.matrix(as.data.frame(otu_table(mock_blanks)))
removed_ASVs <- removed_ASVs[,!colnames(removed_ASVs) %in% colnames(recovered_ASVs)]

# Summarize results
##proportion of contaminants removed (of all total contaminant ASVs)
```

```
contaminants_removed = (rowSums(removed_ASVs[,!colnames(removed_ASVs) %in% rownames(mock_taxa)])/ contar
##proportion of mock removed (of all total mock ASVs)
mock_ASVs_removed = removed_ASVs[,(colnames(removed_ASVs) %in% rownames(mock_taxa))]/ mock_original

##total amount of contaminants remaining in new phyloseq object
contaminants_remaining =  rowSums(recovered_ASVs[,!colnames(recovered_ASVs) %in% rownames(mock_taxa)])

# Calculate classifications

##Percent of mock community ASVs correctly classified as mock community ASVs
true_neg <- rowSums(recovered_ASVs[,colnames(recovered_ASVs) %in% rownames(mock_taxa)])
##Percent of mock community incorrectly classified as mock community ASVs
false_neg <- rowSums(recovered_ASVs[,!colnames(recovered_ASVs) %in% rownames(mock_taxa)])
##identify non-mock community ASVs correctly classified as not belonging to mock community
true_pos <- rowSums(removed_ASVs[,!colnames(removed_ASVs) %in% rownames(mock_taxa)])
# #identify mock community ASVs incorrectly classified as not belonging to mock community
false_pos <- rowSums(removed_ASVs[,colnames(removed_ASVs) %in% rownames(mock_taxa)])

sensitivity <- true_pos/(true_pos + false_neg)
specificty <- true_neg/(true_neg + false_pos)
accuracy <- (true_pos + true_neg) / (false_pos + true_pos + false_neg + true_neg)
prevalence <- (true_pos + false_neg) / (false_pos + true_pos + false_neg + true_neg)

##Following testing various thresholds, we settled on 0.4 as the ideal threshold for identifying contam
```

Applying Decontam to remove potential contaminants within patient urine samples

```
##Load requisite libraries
library("tidyverse")
library("decontam")
library("phyloseq")

##Using phyloseq object created above
urobiome_phyloseq

##create subset of data excluding mock community samples
Patient_Blanks <- subset_samples(urobiome_phyloseq, !SampleType =="Mock")

##apply Decontam
contam_Patient <- isNotContaminant(Patient_Blanks, neg = "is.neg", method ="prevalence",threshold = 0.4

##create physloseq object without contaminants
patient_true_blanks <- prune_taxa(contam_Patient$not.contaminant, Patient_Blanks)

##remove ASVs <1% prevalence in entire dataset
filter_level <- colSums(otu_table(patient_true_blanks))>(0.01*sum(rowSums(otu_table(patient_true_blanks)
patient_true <- prune_taxa(filter_level, patient_true_blanks)

##Subset to patient results only
patient_true <- subset_samples(patient_true, SampleType=="PATIENT")

##remove chloroplast, mitochondrial, and unspecified taxa
```

```
patient_true_final <-subset_taxa(patient_true, Kingdom == "Bacteria"& Family != "Mitochondria" & Genus

## this is Table S3
patient_tax <- tax_table(patient_true_final)
patient_ASV <- otu_table(patient_true_final)
TableS3 <- cbind(patient_tax,t(patient_ASV))

write.csv(TableS3, "TableS3.csv", row.names=TRUE)
```

Figures

Figure 2A

```
##Load requisite libraries
library(tidyverse)
library(phyloseq)
library(vegan)

# beta diversity prior to decontamination

patient_sampling <- subset_samples(urobiome_phyloseq, SampleType == c("PATIENT","SamplingControl"))
GroupsColors <- c("#0072b2","#D81B60")

dist <- phyloseq::distance(patient_sampling, method = "bray")
ord <- ordinate(patient_sampling, method="PCoA", distance = "bray")

Figure2A <- plot_ordination(patient_sampling, ord, color = "SampleType") +
  geom_point(size=3) +
  stat_ellipse(aes(group=SampleType))+
  scale_color_manual(values = GroupsColors)+
  theme_bw()+
  theme(panel.grid = element_blank(),
        panel.border = element_rect(color="black",fill=NA,size=1.5),
        axis.text=element_text(size=13),
        axis.title=element_text(size=18))
        panel.background = element_rect(fill = "transparent"),
        plot.background = element_rect(fill = "transparent", color = NA),
        legend.background = element_rect(fill = "transparent"))


samples <- data.frame(sample_data(patient_sampling))
adonis2(dist ~ Figure2A, data = samples)
```

Figure 2B

```
##Load requisite libraries
library("tidyverse")
library("microbiome")
library("phyloseq")

sample_list <- c("H1", "H2", "H3", "H4", "H5", "H6", "H7", "H8", "H9", "H10", "H11", "H12", "H13", "H14
```

```
Figure2B <- comp_barplot(patient_true_final, tax_level = "Genus", n_taxa = 6, bar_width = 0.7, sample_o
  theme(axis.text.x = element_text(size=12),
        axis.text.y = element_text(size=7),
        axis.title.x = element_text(size=15))
```

Figure 2C

```
##Load requisite libraries
library("tidyverse")
library("decontam")
library("phyloseq")
library("ggpubr")

alpha_stats <- estimate_richness(patient_true_final, measures = c("Chao1", "Shannon"))
GroupsColors <- c("#6db6ff", "#009292")
comparisons_delivery <- list( c("Vaginal", "Caesarean"))
comparisons_abx <- list( c("Yes", "No"))
symnum.args = list(cutpoints = c(0, 0.0001, 0.001, 0.01, 0.05, 1), symbols = c("****", "***", "**", "*"

alpha_stats <- estimate_richness(patient_true_final, measures = c("Chao1", "Shannon"))
GroupsColors <- c("#6db6ff", "#009292")
comparisons_delivery <- list( c("Vaginal", "Caesarean"))
comparisons_abx <- list( c("Yes", "No"))
symnum.args = list(cutpoints = c(0, 0.0001, 0.001, 0.01, 0.05, 1), symbols = c("****", "***", "**", "*"

Delivery_Stat <- plot_richness(patient_true_final, x="Delivery_Mode", measures=c("Chao1", "Shannon"), c
  geom_boxplot(alpha=0.6, size=1.5)+
  geom_point(size=4)+
  theme_bw()+
  theme(axis.text = element_text(size = 16))+
  scale_color_manual(values = GroupsColors)+
  theme(legend.position="none", axis.text.x=element_text(angle=0,hjust=0.5,vjust=1,size=8))+
  stat_compare_means(method = "wilcox.test", comparisons = comparisons_delivery, label = "p.signif", sy

Abx_Stat <- plot_richness(patient_true_final, x="PriorAbx", measures=c("Chao1", "Shannon"), color = "Pri
  geom_boxplot(alpha=0.6, size=1.5)+
  geom_point(size=4)+
  theme_bw()+
  theme(axis.text = element_text(size = 16))+
  scale_color_manual(values = GroupsColors)+
  theme(legend.position="none", axis.text.x=element_text(angle=0,hjust=0.5,vjust=1,size=9))+
  stat_compare_means(method = "wilcox.test", comparisons = comparisons_abx, label = "p.signif", symnum.a
```

Differential Abundance Testing

```
##Load requisite libraries
library("tidyverse")
library("phyloseq")
library("Maaslin2")

ASV_input_data <- read.csv("https://raw.githubusercontent.com/reaset41/Infant-Urobiome/main/asvs_maaslin
metadata_maaslin2_urobiome <-read.csv("https://raw.githubusercontent.com/reaset41/Infant-Urobiome/main/m
```

```
urobiome_maaslin2 <- Maaslin2(
  input_data = ASV_input_data,
  input_metadata = metadata_maaslin2_urobiome,
  min_prevalence = 0.1,
  min_abundance = 0.0,
  normalization = "TSS",
  transform= "NONE",
  output = "test_output_ASVs_LM",
  fixed_effects = c("DeliveryMode","PriorAbx","Age_days"),
  reference = c("DeliveryMode,Vaginal","PriorAbx,No"),
  analysis_method='LM',
  correction='BH',
  max_significance = 0.25,
  plot_heatmap = FALSE,
  plot_scatter = FALSE)
```

Figure 3

```
library(tidyverse)
library(reshape2)

concordance <- read.csv("https://raw.githubusercontent.com/reaset41/Infant-Urobiome/main/concordance_rev

melt_concord <- melt(concordance)

##0 = not detected
##1 = EQUC only
##2 = 16S rRNA only
##3 = Both methods
col1 <- c("0", "1", "2", "3")

Concord <- ggplot(melt_concord, aes(variable, Family, fill=factor(value))) +
  geom_tile(color = "white",linewidth = 0.25,linetype = 1)+
  coord_fixed()+
  scale_fill_manual(values = c("grey", "#800000", "#ADD8E6", "#000080"))+
  xlab("Patient Number")+
  theme(
    axis.text.x   = element_text(colour = "black", angle=45, size=5),
    axis.title.x = element_text(colour = "black", size = rel(0.5)),
    axis.text.y = element_text(colour = "black", size=5))
```

R session information

```
sessionInfo()
```

```
## R version 4.2.1 (2022-06-23)
## Platform: x86_64-apple-darwin17.0 (64-bit)
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib
```

```
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] digest_0.6.31   lifecycle_1.0.3 magrittr_2.0.3  evaluate_0.19
##  [5] rlang_1.0.6     stringi_1.7.8   cli_3.5.0       rstudioapi_0.14
##  [9] vctrs_0.5.1     rmarkdown_2.19  tools_4.2.1     stringr_1.5.0
## [13] glue_1.6.2      xfun_0.36       yaml_2.3.6      fastmap_1.1.0
## [17] compiler_4.2.1  htmltools_0.5.4 knitr_1.41
```