Analysis code for :

Evidence for the Existence of a Bacterial Etiology for Alzheimer Disease and Evidence for a Temporal-Spatial Development of a Pathogenic Microbiome in the Brain

Yves Moné, Joshua P. Earl, Jarosław E. Król, Azad Ahmed, Bhaswati Sen, W. Sue T. Griffin, Richard A. Jones, Sue Woodward, Garth D. Ehrlich, and Jeffrey R. Lapides

Yves Moné

08 April, 2022

The following code was used to perform a differential abundance analysis through a Bayesian approach using the Dirichlet Multinomial Model (DMM) described in Harrison *et al.* 2020 (doi:10.1111/1755-0998.13128).

Load packages

```
# Microbiome analysis
library("phyloseq")
library("decontam")
library("zCompositions")
library("CoDaSeq")

# Graphics
library("ggplot2")
library("ggrepel")
library("grid")
library("gridExtra")
library("ggpubr")
library("lemon")
```

```
# Set working directory
workDir = "brain_microbiome"
dir.create(file.path(getwd(), workDir))
setwd(file.path(getwd(), workDir))
```

```
# Create subdirectories for data and results
dir.create(file.path(getwd(), "data"), showWarnings = FALSE)
dir.create(file.path(getwd(), "results"), showWarnings = FALSE)
```

User-defined Functions

```
# Identify contaminants (Prevalence method)
## ps is a phyloseq object
isNotContam <- function(ps){
  sample_data(ps)$is.neg <- sample_data(ps)$sample_type == "NegCtrl"
  nc_05 <- isNotContaminant(ps,
    neg="is.neg",
    threshold = 0.5,
    normalize = TRUE, detailed = TRUE, method = "prevalence")
  return(nc_05)
}

# Create a composite feature of infrequent OTU across samples
# ps: phyloseq object
# prev: prevalence threshold (percentage)
# rab: relative abundance threshold (percentage)
# other: A logical.Default TRUE. The function returns the filtered
#        phyloseq object with a composite feature named "OTU_others".
#        If FALSE, then the function returns the filtered
#        phyloseq object and the list of OTU that have been removed.

filter.rare <- function(ps, prev, rab, other = TRUE){
  prev = 20
  rab = 0.005
  ps_prev<-filter_taxa(ps,
                       function(x) sum(x>0)<=((prev/100)*length(x)),
                       prune = FALSE)
  ps_rab <- transform_sample_counts(ps,
                                    function(x){round(x/sum(x),6)})
  ps_ab <- filter_taxa(ps_rab,
```

```
                        function(x) mean(x) <= (rab/100),
                        prune = FALSE)
  fTaxa <- ps_prev & ps_ab
  rmtaxa <- taxa_names(ps)[fTaxa]
  if (other == TRUE) {
    ps.new <- merge_taxa(ps, eqtaxa=rmtaxa, archetype=1)
    taxa_names(ps.new)[taxa_names(ps.new) == rmtaxa[1]] <- "OTU_others"
    return(list(ps.new, rmtaxa))
  } else {
    keepTaxa <- !fTaxa
    ps.new <- prune_taxa(keepTaxa, ps)
    return(list(ps.new, rmtaxa))
  }
}

# Process DMM results
## If 95% of the samples of the distribution of the difference
dmm.sig <- function(res_diff){
  # res_diff = differences in pi parameters
  sig <- vector()
  no_sig <- vector()
  for(i in 1:dim(res_diff)[2]){
    perc <- length(which(res_diff[,i] > 0 ))/ length(res_diff[,i])
    if(perc >= 0.95 | perc <= 0.05){
      sig <- c(sig, i)
    }else{no_sig <- c(no_sig, i)
  }
}
return(list(significant = sig,
  nosignificant = no_sig))
}

## Plot DMM results
plot.dmm <- function(comparison, otuFile, pi1.file, pi2.file,
  otu2tax, res_dmm_sig, out_tsv, OTU_all=FALSE){
```

```
# comparison = vector of conditions to compare. eg: c(AD, control)
# otuFile: OTU table (tab-separated text file)
# pi1.file: estimated pi parameters in AD group
# pi2.file: estimated pi parameters in control group
# otu2tax: species name assigned to each OTU
# res_dmm_sig: output list from dmm.sig function
# out_tsv: Table of bacteria that shift in relative abundance
# OTU_all: TRUE to display all the OTU names in the plot,
#          FALSE to display the name of differentially abundant OTU

# Input files
otu_df <- read.table(otuFile,sep="\t",row.names=1,h=T)
pi1_df <- read.table(pi1.file,sep="\t",h=T)
pi2_df <- read.table(pi2.file,sep="\t",h=T)
otu2sp_df <- read.table(otu2tax,sep="\t",h=T)

otu_table <- otu_df[, grep(pattern="OTU_", colnames(otu_df))]

# Differences in estimated pi parameters
diffs <- pi1_df - pi2_df

# Mean of differences in pi parameters
mean_diffs <- data.frame("mean"=apply(diffs, 2, FUN = mean))
mean_diffs$cilow <- apply(diffs, 2,
  FUN = quantile, probs = c(0.025))
mean_diffs$ciup <- apply(diffs, 2,
  FUN = quantile, probs = c(0.975))
mean_diffs$otu_id <- colnames(otu_table)
idx_sig <- colnames(otu_table[,(res_dmm_sig$significant),
                               drop=FALSE])
mean_diffs$dab <- mean_diffs$otu_id %in% idx_sig

# Prepare data to plot
plot_data <- mean_diffs[order(mean_diffs[,1]),]
plot_data$otu_idx <- c(1:dim(plot_data)[1])
otu_id_lvl <- plot_data$otu_id[order(plot_data$mean)]
```

```
plot_data$otu_id <- factor(plot_data$otu_id,
                           levels = otu_id_lvl)
plot_data$otu_sig <- plot_data$otu_id[ifelse(plot_data$dab==TRUE,
                                             plot_data$otu_id,NA),
                                      drop=FALSE]

# Table of differentailly abundant OTU
dab_tab <- merge(na.omit(plot_data[,c(1,2,7,8)]), otu2sp_df,
                 by.x="otu_sig", by.y="OTU_ID", all.x=TRUE)
names(dab_tab) <- c("OTU_id",
                    "mean.diff",
                    "Index",
                    "Phylum",
                    "Species")
dab_df <- data.frame(dab_tab[,c(3,1,4,5,1,2)],
                     "Species"=apply(dab_tab[5], 2,
                                     function(x)
                                     as.character(gsub("_",
                                                       " ",
                                                       x))),
                     row.names = NULL)
dab_df2 <- dab_df[order(dab_df$Index),]
write.table(dab_df2,
            paste0("results/",out_tsv),
            sep="\t",row.names=FALSE, quote=FALSE)

# Select OTU to be labeled on the plot
if(OTU_all == FALSE){
  otu_labs <- plot_data[na.omit(plot_data$otu_sig),]
}else{
  otu_labs <- plot_data
}
otu_labs <- merge(otu_labs, otu2sp_df,
                  by.x="otu_sig", by.y="OTU_ID", all.x=TRUE)
otu_labs$annot <- ifelse(is.na(otu_labs$species),
                         paste0(otu_labs$otu_sig),
```

```
                          paste0(otu_labs$otu_sig,"-",
                                 otu_labs$species))
otu_labs <- otu_labs[order(otu_labs$mean),]
otu_labs <- otu_labs[,c(2:dim(otu_labs)[2],1)]

# Select label position
mynudge.x <- ifelse(otu_labs[,1] < 0, 25, -0.002)
mynudge.y <- ifelse(otu_labs[,1] < 0, -0.009, 0.009)
myhjust <- ifelse(otu_labs[,1] < 0, 0, 1.8)

plot_list <- list()

# Mean_diff plot
p1 <- ggplot(plot_data) +
  geom_point(aes(x=otu_idx, y=mean,
                 color=relevel(factor(dab), "TRUE")),
             size = 1.5) +
  geom_errorbar(aes(x=otu_idx, y=mean,
                    color=relevel(factor(dab), "TRUE")),
                ymin = plot_data$cilow, ymax = plot_data$ciup) +
  geom_hline(yintercept = 0.00, color="grey28", linetype = 2) +
  geom_text_repel(data= otu_labs,
                  aes(x=otu_idx,y=mean, label=annot,
                      color=relevel(factor(dab), "TRUE")),
                  vjust=0.5, hjust=myhjust,
                  size=4,
                  fontface="bold",
                  colour = "black",
                  nudge_x = mynudge.x,
                  nudge_y = mynudge.y,
                  segment.color = "black",
                  segment.alpha = 0.5,
                  segment.size = 0.5
                  )+
  ylab("Effect size
       (difference in estimated relative abundance)") +
```

```
  ylim(c(min(plot_data$cilow)*1.1,
         max(plot_data$ciup)*1.5)) +
  coord_capped_cart(bottom ='both') +
  scale_x_continuous(
    breaks = c(seq(0, max(plot_data$otu_idx),by=20),
               max(plot_data$otu_idx)),
    labels = c(seq(0, max(plot_data$otu_idx),by=20),""),
    expand = expansion(mult=c(0.05,0.016)),
    name="Index of OTU (ordered by effect size)") +
  annotate("text",
           x=dim(plot_data)[1]-25,
           y=0.05,
           label= paste(comparison[1]),
           colour = "black", fontface = "bold", size = 5) +
  annotate("text",
           x=dim(plot_data)[1]-25,
           y=-0.025,
           label= paste(comparison[2]),
           colour = "black", fontface = "bold", size = 5) +
  theme_bw() +
  theme(legend.position="none",
        axis.line.x = element_line(colour="black", linetype = 1),
        axis.ticks.x = element_line(color="black"),
        axis.text.x = element_text(size=14, color="black"),
        axis.title.x = element_text(size=14, color="black"),
        axis.line.y = element_line(colour="black", linetype = 1),
        axis.title.y = element_text(size=14, color="black"),
        axis.text.y=element_text(size=14, color="black"),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        panel.border = element_blank(),
        plot.margin = unit(c(1,2,1,2), "cm")
        )
plot_list[[1]] <- p1
plot_list[[2]] <- plot_data
return(plot_list)
```

```
}
```

Load dataset

```
ps <- readRDS("data/brain_phyloseq.rds")
```

Detection of contaminant OTU and data filtering

Potential contaminant OTUs were detected based on their mean relative abundance and their prevalence in experimental samples vs. negative controls using the R package Decontam (Davis *et al.* 2018, doi:10.1186/s40168-018-0605-2). To qualify as contaminant, an OTU had to have a score ≥ 0.5 or a higher mean relative abundance in the negative controls than the biological samples.

```
# Mean relative abundance (RAb) in negative controls
ps_neg <- subset_samples(ps, sample_type == "NegCtrl")
pneg <- prune_taxa(taxa_sums(ps_neg) > 0, ps_neg)
pneg_rab<-transform_sample_counts(pneg,
                                  function(x){round(x / sum(x),6)})
neg_mean_rab <- apply(X = otu_table(pneg_rab),
                      MARGIN = ifelse(taxa_are_rows(pneg_rab),
                                      yes = 1, no = 2),
                      FUN = function(x){mean(x)})

# Mean RAb in experimental samples
ps_true <- subset_samples(ps, sample_type == "sample")
exp_rab<-transform_sample_counts(ps_true,
                                  function(x){round(x / sum(x),6)})
exp_mean_rab <- apply(X = otu_table(exp_rab),
                      MARGIN = ifelse(taxa_are_rows(exp_rab),
                                      yes = 1, no = 2),
                      FUN = function(x){mean(x)})

# Compare mean RAb between negative control and experimental samples
otu_exp_neg<-round(exp_mean_rab[which(names(exp_mean_rab) %in%
                                      names(neg_mean_rab))]*100, 4)
```

```
rab_df <- data.frame("OTU_ID" = names(neg_mean_rab),
                     "rab.neg" = round(neg_mean_rab*100,4),
                     "rab.sample" = otu_exp_neg)
rab_df$greater.in.ctrl <- ifelse(rab_df[,2] > rab_df[,3],
                                 TRUE, FALSE)

# decontam package - Prevalence-based method
## conf_lvl: confidence levels from suppl. table S11
tax_df <- as.data.frame(tax_table(ps))
otu2tax <- data.frame("OTU_ID"= rownames(tax_df),
                      "phylum"=tax_df$phylum,
                      "species"=tax_df$species)
conf_lvl <- read.table("data/brain_confidence.tsv", sep="\t", h=T)
otu_neg1 <- as.data.frame(otu_table(pneg))
otu_neg1$OTU_ID <- rownames(otu_neg1)
otu_neg <- merge(otu_neg1, conf_lvl[, c(1,7:8)], by="OTU_ID")
otu_tax_neg <- merge(otu_neg, otu2tax, by="OTU_ID")
otu_tax_neg <- otu_tax_neg[order(otu_tax_neg$OTU_ID),]

## Identification of contaminants
non_contam <- isNotContam(ps)
contam <- non_contam[which(!is.na(non_contam$p) &
                           non_contam$not.contaminant==FALSE),]
contam$OTU_ID <- rownames(contam)
contam_df <- merge(otu_tax_neg, contam, by="OTU_ID", all.x=TRUE)
write.table(contam_df,
            "results/OTU_decontamPrev.tsv",
            sep="\t", row.names=FALSE, quote=FALSE)

## Combine results from RelAb and decontam approaches
df1 <- merge(rab_df, contam, by="OTU_ID", all.x=TRUE)
df2 <- merge(otu_tax_neg[, c(1,6,7,9)], df1[,c(1:4,9,10)],
             by="OTU_ID", all.x=TRUE)

contam <- df2[order(df2$species),]
names(contam) <- c("OTU_ID",
```

```
                        "genus_conf",
                        "species_conf",
                        "species",
                        "RAb.NegCtrl",
                        "RAb.samples",
                        "RAb.contaminant",
                        "decontam.p_value",
                        "decontam.not.contaminant")
contam$contaminant <- ifelse(contam$RAb.contaminant==TRUE,
                                    TRUE,
                                    ifelse(!(is.na(contam[,8])),
                                           TRUE, FALSE))
write.table(contam,
            "results/OTU_ContamDetection.tsv",
            sep="\t",row.names=FALSE,quote=FALSE)
```

```
#
### Filter-out contaminant OTU
contam <- read.table("results/OTU_ContamDetection.tsv",
                     sep="\t",h=T)

rmOTU <- contam[which(contam$contaminant==TRUE),1]
ncOTU <- !(taxa_names(ps) %in% rmOTU)
ps_nc <- prune_taxa(ncOTU, ps)

# Remove negative control samples
ctrl <- c("UA_C1_2_lbc47", "UA_C2_2_lbc62",
          "NW_C1_lbc37", "NW_C2_lbc38",
          "UA_C1_1_lbc68", "UA_C2_1_lbc76")
ps_nc2 <- subset_samples(ps_nc, !(sample_names(ps_nc) %in% ctrl))

# Remove low yield samples (cut-off=100 reads)
ps100 <- prune_samples(sample_sums(ps_nc2)>=100, ps_nc2)

# Remove samples with 0 counts across all OTU
ps_s0 <- prune_samples(sample_sums(ps100) > 0, ps100)
```

```
# Remove OTU with 0 counts across all samples
ps_decontam <- prune_taxa(taxa_sums(ps_s0) > 0, ps_s0)

## Infrequent OTU are grouped into a composite feature "OTU_others"
### OTU_others: RAb cut-off = 0.005% and prevalence cut-off = 20%
ps_filtered <- filter.rare(ps_decontam, 20, 0.005, other = TRUE)
saveRDS(ps_filtered[[1]], "results/brain_phyloseq_filtered.rds")

other <- taxa_names(ps_decontam) %in% ps_filtered[[2]]
ps_other <- prune_taxa(other, ps_decontam)

write.table(tax_table(ps_other),
            "results/list_otu_other.tsv",
            sep="\t",row.names=FALSE,quote=FALSE)
#
```

Differential abundance testing

```
# Prepare data for DMM analysis
ps <- readRDS("results/brain_phyloseq_filtered.rds")
otu_df <- as.data.frame(t(as.matrix(otu_table(ps))))
meta_df <- data.frame(sample_data(ps))

dmm_df <- merge(meta_df[,c(5:7)], otu_df, by=0)
colnames(dmm_df)[1] <- "sample_ID"
dmm_df <- dmm_df[order(dmm_df$diagnosis, dmm_df$specimen_ID),]
write.table(dmm_df,
  "results/dmm_brain_table.tsv",
  sep="\t",row.names=FALSE,quote=FALSE)
```

Model specification

The Dirichlet Multinomial model was specified in the Stan probabilistic programming language.

```
// DM model
data {
  int<lower=1> notus; // total counts in each sample
  int<lower=1> nreps; // number of replicates
  int<lower=1> N; // number of sampling groups (AD vs. C)
  int<lower=1> M; // number of specimens (individual biopsies)
  int<lower=1> start[N]; // group sampling start indices
  int<lower=1> end[N]; // group sampling end indices
  int<lower=1> reptospe[nreps]; // index of replicate per specimen
  int datamatrix[nreps, notus];
}

parameters {
  real<lower=0> theta[N];
  real<lower=0> tau[M];
  simplex[notus] psi[M];
  simplex[notus] pi[N];
  simplex[notus] p[nreps];
}

model {
  for(m in 1:M){
    target += exponential_lpdf(tau[m] | 0.01);
    target += dirichlet_lpdf(psi[m] | rep_vector(0.0000001, notus));
  }
  for(i in 1:N){
    for(j in start[i]:end[i]){
      target += exponential_lpdf(theta[i] | 0.01);
      target += dirichlet_lpdf(pi[i] | tau[reptospe[j]]*psi[reptospe[j]]);
      target += dirichlet_lpdf(p[j] | theta[i]*pi[i]);
      target += multinomial_lpmf(datamatrix[j,] | p[j]);
    }
  }
}
```

Run the Dirchlet Multionmial model

The Dirichlet Multinomial model was run through Python using the Pystan package. DMM is computationally intensive and have been run on a high performance cluster.

```
import os, sys
import pickle
import time
import pystan
import pandas as pd
import numpy as np
#
n_cpu=os.cpu_count()
print("n_cpu:", n_cpu)
print("Python:", sys.version)
print("pystan:", pystan.__version__)
print("pandas:", pd.__version__)
print("numpy:", np.__version__)
#
method = 'HMC'
modelfile = "data/DMM_brain.stan"
count_table = "results/dmm_brain_table.tsv"
prefix = "ADvsC"
pathname = "results"
# Output directory
os.makedirs('{0}/DMM-{2}_output_{1}'.format(pathname,prefix,method),
            exist_ok=True)
# Load data
otu_table = pd.read_csv(count_table,sep='\t')
otu_dat = otu_table.sort_values(by=['diagnosis',
          'specimen_ID']).reset_index(drop=True)
#
# get group indices (Stan language uses 1-based indexing)
starts = [1,otu_dat['diagnosis'].value_counts()[0]+1]
ends = [otu_dat['diagnosis'].value_counts()[0],
        len(otu_dat['diagnosis'])]
groups = [starts,ends]
#
```

```
# get specimen indices (Stan language uses 1-based indexing)
codes, uniques = pd.factorize(otu_dat['specimen_ID'])
otu_dat['reptospe'] = codes
reptospe = list(codes +1)
#
# Data matrix
counts = otu_dat.filter(regex='OTU_.*',axis=1)
#
#
data = {
    'datamatrix': counts+1,
    'nreps': int(len(counts)),
    'notus': int(len(counts.columns)),
    'N': int(len(starts)),
    'M': int(len(uniques)),
    'start': groups[0],
    'end': groups[1],
    'reptospe': reptospe
    }
#
start_time = time.process_time()
#
sm = pystan.StanModel(file=modelfile, model_name='DMM')
fit = sm.sampling(data=data,
    chains=4,
    control = {'max_treedepth': 15},
    warmup = 1500,
    iter = 3500,
    algorithm = "NUTS",
    seed = 123,
    thin = 2,
    pars = ['pi','psi'],
    verbose = True,
    sample_file =
    '{0}/DMM-{2}_output_{1}/DMM_{2}_{1}_sample_file.csv'.format(
      pathname,prefix,method)
```

```
    )
# Save object fit
with open('{0}/DMM-{2}_output_{1}/DMM_fit_{2}_{1}.pkl'.format(
  pathname,prefix,method),'wb') as f:
    pickle.dump({'model' : sm, 'fit' : fit}, f, protocol=-1)
#
fit_df = fit.to_dataframe()
fit_df.to_csv('{0}/DMM-{2}_output_{1}/DMM_fit_{2}_{1}.tsv'.format(
  pathname,prefix,method), sep='\t', header=True, index=False)
#
# Extract pi parameters for each sampling group
est_pi = fit.extract(pars='pi')
mat_pi1 = np.asmatrix(est_pi['pi'][:,0,:])
mat_pi2 = np.asmatrix(est_pi['pi'][:,1,:])
pi1_df = pd.DataFrame(mat_pi1, )
pi1_df.to_csv('{0}/DMM-{2}_output_{1}/DMM_{2}_{1}_est_pi1.tsv'.format(
  pathname,prefix,method), sep='\t', header=True, index=False)
pi2_df = pd.DataFrame(mat_pi2)
pi2_df.to_csv('{0}/DMM-{2}_output_{1}/DMM_{2}_{1}_est_pi2.tsv'.format(
  pathname,prefix,method), sep='\t', header=True, index=False)
#
# Summary
with open('{0}/DMM-{2}_output_{1}/DMM_fit_{2}_{1}_summary.txt'.format(
  pathname,prefix,method),'w') as f:
    print(fit.stansummary(), file = f)
#
# Time elapsed
end_time = time.process_time()
time_elapsed = end_time - start_time
print("Time elapsed: ", time_elapsed, "seconds")
```

Processing DMM results

```
# Generate otu2tax table
ps <- readRDS("results/brain_phyloseq_filtered.rds")
tax_df <- as.data.frame(tax_table(ps))
otu2sp <- data.frame("OTU_ID"= rownames(tax_df),
                     "phylum"=tax_df$phylum,
                     "species"=tax_df$species)
write.table(otu2sp,
            "results/brain_otu2sp.tsv",
            sep="\t",row.names=FALSE,quote=FALSE)

comp <- c("AD", "C")
otu_file = "results/dmm_brain_table.tsv"
pi1 = "results/DMM_HMC_ADvsAMC_est_pi1.tsv"
pi2 = "results/DMM_HMC_ADvsAMC_est_pi2.tsv"
otu2tax = "results/brain_otu2sp.tsv"
out_tsv = "results/DMM_results.tsv"

# OTU differentially abundant
pi1_df <- read.table(pi1,sep="\t",h=T)
pi2_df <- read.table(pi2,sep="\t",h=T)
diffs <- pi1_df - pi2_df
res_dmm_sig <- dmm.sig(diffs)
length(res_dmm_sig$sig)
#
## Plot differences

plot_others <- plot.dmm(
  comp,
  otuFile = otu_file,
  pi1.file = pi1,
  pi2.file = pi2,
  otu2tax,
  res_dmm_sig,
  out_tsv,
  OTU_all = FALSE)
```

```
#
ggsave(plot = plot_others[[1]],
       "results/Differential_abundance_plot.tiff"),
       device="tiff", width = 38, height = 30,
       units = 'cm', dpi = 400, compression = "lzw")
```