City-scale distribution and dispersal routes of mycobiome in residences

Xinzhao Tong, Marcus H. Y. Leung, David Wilkins, and Patrick K. H. Lee*

School of Energy and Environment, City University of Hong Kong, Hong Kong

Running title: City-scale distribution and dispersal routes of mycobiomes in residences

Keywords: Mycobiome, indoor built environment, dispersal potentials, distance-decay, biogeography

Correspondence: *B5423-AC1, School of Energy and Environment, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong; E-mail: patrick.kh.lee@cityu.edu.hk; Tel: (852) 3442-4625; Fax: (852) 3442-0688.

This document contains original in-house codes and scripts used to generate the results described in the manuscript. The document is divided into the following sections:

1) In-house script for read quality-filtering
2) In-house scripts for OTU-clustering and quality control, including chimera and contaminants removal
3) In-house scripts for alpha-diversity analyses
4) In-house scripts for beta-diversity analysis
5) In-house script for taxonomic analysis
6) In-house script for indicator species analysis
7) In-house script for cross-domain analysis
8) In-house script for SourceTracking analysis
9) In-house script for distance-decay analysis

Please note that while in-house scripts required for results and figure generation are presented on this document, some minor tasks (such as table reformatting) were performed in Microsoft Excel as described below. For example, the determination of average alpha-diversity values for each sample following ten rounds of rarefaction was performed using Pivot Table function on Excel. Also note that the exact same scripts will not function across different computers. It is the responsibility of the readers to understand the scripts included here and modify accordingly.

Some of the R packages required for the following scripts are:

- devtools

- wilkoxmisc (install via github)
- ggplot2
- reshape2
- plyr
- pgirmess

They can be installed by the input in R (for example):
install.packages("pgirmess")

(1) In-house script for read quality-filtering

After downloading raw fastq files, paired-end reads were merged using the "-fastq_mergepairs" command of usearch (version 8.1.1861)

```
1.1 usearch -fastq_mergepairs sampleA_forward.fastq –fastqout sampleA_merged.fastq -relabel @
```

Merged raw reads were subjected to usearch QC, based on truncation length of 300-bp, and fastq maximum error rate of 0.5:

```
1.1 usearch -fastq_filter sampleA_merged.fastq -fastq_trunclen 300 -fastq_maxee 0.5 -fastaout sampleA_filtered.fasta
```

Filtered reads were subjected to dereplication:

```
1.3 usearch -derep_fulllength forward_filtered.fasta -fastaout dereplicated_sampleA.fasta -sizeout -minseqlength 300
```

(2) In-house scripts for OTU-clustering and quality control, including chimera and contaminants removal

OTU-clustering: following read quality-filtering and demutiplexing from usearch, usearch -cluster_otu was used to generate OTU fasta file. Using the OTU fasta file as an input, the following perl script was used to generate a fasta file with OTU named as numbers, and a fasta file with the renamed OTUs:

2.1) assign_OTU_numbers.pl

```
#!/usr/bin/perl

use 5.18.2;
use autodie;
$|++;
open IN, '<', 'OTUs.fasta';
open OUT, '>', 'OTUs_numbered.fasta';
open MAP, '>', 'OTU_to_reference_sequence.tidy.txt';
say MAP "OTU\tReferenceSequence\tSize";
my $OTU = 0;
while (<IN>) {
 chomp;
 print "\r$. lines processed" unless $. % 1000;
 if (/^>(?<read>.+);size=(?<size>\d+)$/) {
  $OTU++;
  say MAP "$OTU\t${read}\t${size}";
  say OUT ">OTU_$OTU";
 } else {
  say OUT;
 }
}
say "\r$. lines processed";
close IN;
close OUT;
close MAP;
```

This output file was used to perform taxonomic classification using QIIME's "assign_taxonomy.py" script. Following chimera detection using usearch -uchime_ref command, script below was used to generate txt file containing a list of chimeric OTUs:

2.2) make_chimeras_list.pl

```
#!/usr/bin/perl
```

```
use 5.18.2;
use autodie;
$|++;
open IN, '<', './chimeras.fasta';
open OUT, '>', 'chimeras.tidy.txt';
say OUT "OTU";
while (<IN>) {
 chomp;
 next unless /^>/;
 (my $OTU) = $_ =~ /^>(.+)/;
 say OUT $OTU;
}
close IN;
close OUT;
```

OTU table was prepared by compiling the following input files:

-OTU_to_reference_sequence.tidy.txt (output from 2.1)
-OTU_numbered_tax_assignments.txt (output of assign_taxonomy.py QIIME script)
-readmap.uc (output of usearch -usearch_global command)

And generates the following output files:

-OTU_table.tidy.txt (OTU table including OTUs that are chimeric and contaminant)
-singletons.txt (txt file containing a list of singleton OTUs)

2.3) prepare_OTU_table.pl

```
#!/usr/bin/perl

use 5.18.2;
use autodie;
$|++;
# Load OTU reference sequences
say "Loading OTU reference sequences";
open OTUREFMAP, '<', './OTU_to_reference_sequence.tidy.txt';
my %OTUofRefSeq;
while (<OTUREFMAP>) {
 next if $. == 1;
 chomp;
 print "\r$. lines processed" unless $. % 1000;
 (my $OTU, my $refSeq) = split(/\t/, $_);
 $OTUofRefSeq{$refSeq} = $OTU;
```

```
}
say "\r$. lines processed";
close OTUREFMAP;
# Load OTU taxonomies
say "Loading OTU taxonomies";
open TAX, '<', './OTUs_numbered_tax_assignments.txt';
my %taxonomy;
while (<TAX>) {
 chomp;
 print "\r$. lines processed" unless $. % 1000;
 my @line = split /\t/, $_;
 my $OTU = $line[0];
 my @taxonomy;
 if ($line[1] eq 'Unassigned') {
  @taxonomy = ('') x 7;
 } else {
  @taxonomy = split(/;\s/, $line[1]);
 }
 s/^.__// for @taxonomy;
 @{$taxonomy{$OTU}} = @taxonomy;
}
say "\r$. lines processed";
close TAX;
# Count reads for each OTU
say "Counting reads for each OTU";
open READMAP, '<', 'readmap.uc';
my %readCount;
my %OTUReadCount;
while (<READMAP>) {
 chomp;
 print "\r$. lines processed" unless $. % 1000;
 next if /^N/;
 my @hit = split(/\t/, $_);
 (my $read, my $OTU) = @hit[8,9];
 (my $sample) = $read =~ /^([^.]+)/;
 $readCount{$OTU}{$sample}++;
 $OTUReadCount{$OTU}++;
}
say "\r$. lines processed";
close READMAP;
# Produce list of singletons
say "Producing list of singletons";
```

```
my %singletons;
open SINGLETONS, '>', 'singletons.txt';
say SINGLETONS "OTU";
foreach my $OTU (keys %OTUReadCount) {
 if ($OTUReadCount{$OTU} == 1) {
  say SINGLETONS $OTU;
  $singletons{$OTU} = 1;
 }
}
close SINGLETONS;

# Produce OTU table
say "Producing OTU table";
open OTUTABLE, '>', 'OTU_table.tidy.txt';
say OTUTABLE
"OTU\tSample\tCount\tKingdom\tPhylum\tClass\tOrder\tFamily\tGenus\tSpecies";
foreach my $OTU (sort keys %readCount) {

 # Skip singletons
 next if exists $singletons{$OTU};

 foreach my $sample (sort keys %{$readCount{$OTU}}) {
  say OTUTABLE "$OTU\t$sample\t$readCount{$OTU}{$sample}\t", join("\t",
@{$taxonomy{$OTU}});
 }
}
close OTUTABLE;
```

Following creation of OTU table from 2.3), will need to identify contaminant OTUs from the table, and remove from the OTU table later. This is performed by detecting lineages that are present in negative controls in more than 3% of reads on average. The script below takes in OTU_table.tidy.txt file from 2.3), and generates two output files:

-contaminant_lineages.tidy.txt (a list of lineages deemed contaminants)
-contaminants.txt (a list of OTUs deemed contaminants)

2.4) classify_contaminants.R

```
# Libraries
library(wilkoxmisc)

# List of blank samples
BlankSamples <- c("name_of_negative_sample(s)")
```

```
# Read in OTU table
OTUTable <- read.tidy("OTU_table.tidy.txt")

# Collapse by lineage
OTUTable <- within(OTUTable, Lineage <- factor(paste(Kingdom, Phylum, Class, Order,
Family, Genus, Species)))
OTUsByLineage <- unique(OTUTable[, c("OTU", "Lineage")])
OTUTable <- ddply(OTUTable, .(Sample, Lineage), summarise, Count = sum(Count),
.progress = "time")

# Select blank samples
Blank <- subset(OTUTable, Sample %in% BlankSamples)

# Add relative abundances
Blank <- add.relative.abundance(Blank)

# Aggregate
Blank <- ddply(Blank, .(Lineage), summarise, RelativeAbundance =
sum(RelativeAbundance))

# Calculate value for cutoff
Cutoff <- sum(Blank$RelativeAbundance) * 0.03
# Trim contaminant list to lineages above cutoff
Blank <- Blank[which(Blank$RelativeAbundance > Cutoff), ]

# Write contaminant lineages to file
write.tidy(Blank, "contaminant_lineages.tidy.txt")

# Sort OTUs into Contaminant/Non-contaminant
Contaminants <- within(OTUsByLineage, Contaminant <- ifelse(Lineage %in%
Blank$Lineage, "Contaminant", "Non-contaminant"))
Contaminants$Lineage <- NULL

# Write to file
write.tidy(Contaminants, "contaminants.txt")
```

Having identified chimeric OTUs (chimeras.tidy.txt from 2.2) and contaminant OTUs (contaminants.txt from 2.4), these files will be used to identify OTUs to be removed from OTU_table.tidy.txt. The output file will be "OTU_table_clean.tidy.txt" containing high-qualilty, non-chimeric, and non-contaminating OTUs.

2.5) clean_OTU_table.R

```
# Libraries
library(wilkoxmisc)

# Read in OTU table
OTUTable <- read.tidy("OTU_table.tidy.txt")

# Add fate column
OTUTable$Fate <- rep(NA, nrow(OTUTable))

# BLANK SAMPLES
BlankSamples <- c("negative samples")

# Remove blank samples
OTUTable <- OTUTable[which( ! OTUTable$Sample %in% BlankSamples), ]

# CHIMERAS
## Read in list of chimeras
Chimeras <- read.tidy("chimeras.tidy.txt")
Chimeras <- as.character(Chimeras$OTU)

# Mark chimeric OTUs
OTUTable$Fate <- ifelse(
 OTUTable$OTU %in% Chimeras & is.na(OTUTable$Fate),
 'Chimera',
 OTUTable$Fate
)
# CONTAMINANTS
## Read in list of contaminants
Contaminants <- read.tidy("contaminants.txt")
Contaminants <- as.character(Contaminants[which(Contaminants$Contaminant ==
'Contaminant'), "OTU"])

# Mark contaminant OTUs
OTUTable$Fate <- ifelse(
 OTUTable$OTU %in% Contaminants & is.na(OTUTable$Fate),
 'Contaminant',
 OTUTable$Fate
)
## OUTPUT
# Summarise OTUs by fate and write to file
OTUFates <- unique(OTUTable[c("OTU", "Fate")])
```

```
write.tidy(OTUFates, "OTU_fates.tidy.txt")

# Remove failures from OTU table and write to file
OTUTable <- OTUTable[which(is.na(OTUTable$Fate)), ]
OTUTable$Fate <- NULL
write.tidy(OTUTable, "OTU_table_clean.tidy.txt")
```

Prepare clean OTU_table to format readable for biom_convert command in QIIME

2.6) cast_OTU_table.R

```
# Libraries
library(wilkoxmisc)
library(reshape2)

# Read in OTU table
OTUCounts <- read.tidy("OTU_table_clean.tidy.txt")

# Cast
OTUCounts <- dcast(OTUCounts, OTU ~ Sample, value.var = "Count", fill = 0)

# Write
write.tidy(OTUCounts, "OTU_table_clean.cast.txt")
```

The output "OTU_table_clean.cast.txt" can be used as input for biom_convert command in QIIME.

(3) In-house scripts for alpha-diversity analyses

Following read depth normalization (normalized to 1,058 reads per sample), multiple rarefaction, and alpha-diversity analyses using QIIME scripts "multiple_rarefaction.py" and "alpha_diversity.py," statistical significance of alpha diversity was determined using the following script:

```
library(ggplot2)
library(readr)
library(dplyr)
library(reshape2)

Alpha <- read.tidy("alpha_diversity_average_1058.txt")
Meta <- read.tidy("samples_fungi.txt")
Merge <- merge(Alpha, Meta, by = "Sample", all.X = FALSE)

#Perform statistical analyses
library(lme4)

#Statistics for air samples
Air <- Merge %>% filter(Type == "Air")
lmer.hyp <- lmer(observed_otus~Household + (1|Occupancy) +(1|Room) + (1|Window) +
(1|Temperature) + (1| Humidity), data=Air, REML=FALSE)
lmer.null <- lmer(observed_otus~1 + (1|Occupancy) +(1|Room) + (1|Window)
+(1|Temperature) + (1| Humidity) ,data=Air, REML=FALSE)
anova(lmer.hyp,lmer.null)
```

(4) In-house scripts for beta-diversity analysis

QIIME script “beta_diversity.py” generates two output matrix files, one for Bray-Curtis dissimilarity (abundance-weighted beta-diversity analysis) and one for Binary Jaccard distance (abundance-nonweighted beta-diversity analysis). For each matrix file, the following script is used to determine ANOSIM and statistical significance for predictive variables.

```
#Load libraries
library(devtools)
library(ggplot2)
library(vegan)

#Read samples in Bray-Curtis dissimilarity
BrayCurtis <- as.matrix(read.dist("bray_curtis_div/bray_curtis_OTU_table_even1058.txt"))
Meta <- read.tidy("samples_fungi.txt")
BrayCurtisMatrix <- as.matrix(BrayCurtis)

#Perform PERMANOVA
#First list samples from both beta diversity matrix file and metadata, to check if the two lists are identical
AllSamples <-data.frame(Sample = row.names(BrayCurtisMatrix))
AllSamples <- merge(AllSamples, Meta, by = "Sample", all.x = TRUE)

#Check that the two lists are identical (output is either TRUE or FALSE)
sum(row.names(BrayCurtisMatrix)==AllSamples$Sample) == length(AllSamples$Sample)

#Perform PERMANOVA based on sample group comparison, the example below is household factor
BrayCurtis <- as.dist(BrayCurtis)
Adonis <- adonis(BrayCurtis~Individual, data=AllSamples, permutations=999)
```

(5) In-house script for taxonomic analysis

R-script takes in clean OTU txt table, and creates txt file indicating top taxa of a particular taxonomic rank. This output can be used as input to construct visual plot in ggplot. Also requires Metadata txt file, containing sample information, created during sample collection.

5.1) make_tax_plot.R

```
library(reshape2)
library(ggplot2)

#Open taxonomy OTU table
OTU <- read.tidy("Unite_Findley_OTU_table_clean.tidy.txt")

#Open Metadata table
Meta <- read.tidy("samples_fungi.txt")

#Tabulate read counts by Genus
OTU <- ddply(OTU, .(Sample, Genus), summarise, Count = sum(Count))

#Convert count to relativeabundance and add column
OTU <- ddply(OTU, .(Sample), mutate, RelativeAbundance = (Count * 100) / sum(Count))

#collapse taxa table to only 15 top phyla, genus, family, etc (require reshape2).
OTUTable <- collapse.taxon.table(OTU, n = 15, Rank = "Genus")

#Merge relative abundance table and metatable together
OTUTable <- merge(OTUTable, Meta, by = "Sample", all.x = TRUE)

write.tidy(OTUTable, "Top15Genera.txt")

#To create different number of genera in file, or group top taxa at a different taxonomic level (e.g. phylum), change the n number and Rank accordingly:
# OTUTable <- collapse.taxon.table(OTU, n = 12, Rank = #"Phylum")
```

To plot top taxonomy, the following script is used:

5.2) make_taxonomy_heatmap.R

```
#load packages
library(readr)
library(dplyr)
library(reshape2)
library(ggplot2)
library(knitr)
library(viridis)
library(ggthemes)
library(OIdata)
library(RColorBrewer)
library(extrefont)

hm.palette <- colorRampPalette(rev(brewer.pal(11, 'Spectral')), space='Lab')

#Skin
OTUTable <- read_tsv("Top15Genus_Skin.txt")
OTUTable$Genus <- factor(OTUTable$Genus, levels =
c("Alternaria","Aspergillus","Candida","Cladosporium",    "Cryptococcus", "Debaryomyces",
"Emericella","Lentinula", "Malassezia", "Penicillium","Pichia","Rhodotorula",
"Sporobolomyces","Sterigmatomyces", "Minor/Unclassified"  ))
Plot <- ggplot(OTUTable, aes(x = Individual, y = Genus, fill = RelativeAbundance))
Plot <- Plot + geom_tile(color="white", size = 0.1)
Plot <- Plot + coord_equal()
Plot <- Plot + theme(plot.title=element_text(hjust=0))
Plot <- Plot + theme(axis.title =element_text(size=12, face = "bold", family="Arial"))
Plot <- Plot + theme(axis.text.y =element_text(size = 8, family="Arial"))
Plot <- Plot + theme(axis.text.x = element_text(angle = 90, hjust = 1, family="Arial"))
Plot <- Plot + scale_fill_gradientn(colours = hm.palette(100), limits = c(0, 85), name =
"Mean relative \nabundance (%)")
Plot <- Plot + xlab(paste0("Individual")) +ylab(paste0("Genus"))
Plot <- Plot + theme(axis.ticks = element_blank())
Plot <- Plot + ggtitle("Human Skin")
Plot <- Plot + theme(plot.title=element_text(hjust=0.5, face="bold", family="Arial"))
Plot <- Plot + theme(legend.position = "bottom")
Plot <- Plot + theme(legend.title=element_text(family="Arial"))
Plot <- Plot + theme(legend.text=element_text(family="Arial"))
Plot <- Plot + theme(legend.key.size=unit(0.7, "cm"))
Plot <- Plot + theme(legend.key.width=unit(1.5, "cm"))
Plot <- Plot + theme(panel.background = element_rect(colour = "grey", size = 0.8))
```

```
#Perform statistical test for Malassezia on household surfaces with different surface touch frequencies (high/low)
library(readr)
library(dplyr)
library(reshape2)

Genus <- read_tsv("Top12Genus.txt")
Table <- Genus %>%
        filter(Genus == "Malassezia") %>%
        filter(Type == "Surface")

wilcox.test(RelativeAbundance~SurfaceTouchFrequency,data=Table)
```

(6) In-house script for indicator species analysis

Only OTUs with a relative abundance greater than 1% in at least one sample were included in indicator species analysis. Indicator taxa were identified using the "indval" function in the R package "labdsv", with indicator values greater than 0.5 and significance determined based on a False Discovery Rate (FDR) corrected $p < 0.05$ after 1,000 permutations.

```
# load packages
library(labdsv)
library(wilkoxmisc)
library(readr)
library(dplyr)

# load OTU Table
OTUTable <- read.tidy("Unite_Findley_OTU_table_clean.tidy.txt")

# load sample data
Samples <- read.tidy("samples_fungi.txt")
OTUTable <- join(OTUTable, Samples, by = "Sample")

# select only OTUs with a relative abundance above 1% in at least one sample
OTUTable <- OTUTable %>%
        group_by(Sample) %>%
        mutate(Sum_Count = sum(Count)) %>%
        ungroup()

OTUTable <- OTUTable %>%
        group_by(OTU, Sample) %>%
        mutate(RelativeAbundance = (100*Count) / Sum_Count) %>%
        ungroup()

AbundantOTUs <- OTUTable %>%
        filter(RelativeAbundance >=1) %>%
        select(OTU) %>%
        unique()

OTUTable <- OTUTable %>%
         filter(OTU %in% AbundantOTUs$OTU)

# cast
OTUMatrix <- dcast(OTUTable, Sample + Type ~OTU, value.var="Count", fill = 0)
```

```
# Convert type to numeric (required by labdsv)
Cluster <- as.numeric(OTUMatrix$Type)
Types <- unique(data.frame(Class=Cluster, Type = OTUMatrix$Type))

#Clean up matrix
OTUMatrix$Type <- NULL
rownames(OTUMatrix) <- OTUMatrix$Sample
OTUMatrix$Sample <- NULL

#Calculate indicator values
Indicators <- indval(OTUMatrix, Cluster)

#Identify OTUs with a significant pval(< 0.05), FDR corrected
Indicators$pval <- p.adjust(Indicators$pval, method ="fdr")
IndicatorOTUs <- names(Indicators$pval[which(Indicator$pval<0.05)])

#Tidy indicator values
IndicatorValues <- Indicators$indval
IndicatorValues$OTU <- rownames(IndicatorValues)
#convert the cast table into tidy format
IndicatorValues <- melt(IndicatorValues,
        id.var = "OTU",
        variable.name = "Class",
        value.name = "IndicatorValue")
# replace the 1,2,3 with air, skin and surface
IndicatorValues <- join(IndicatorValues, Types, by = "Class")
IndicatorValues$Class <- NULL

# select only significant and strong indicators (indicator value > 0.5)
IndicatorValues <- IndicatorValues[which(IndicatorValues$OTU %in% IndicatorOTUs), ]
IndicatorValues <- IndicatorValues[which(IndicatorValues$IndicatorValue > 0.5),]

# Tidy table and write
IndicatorValues <- arrange(IndicatorValues, Type, desc(IndicatorValue))
write.tidy(IndicatorValues, "IndicatorValues_0.5.txt")

# add taxonomy
Tax <- OTUTable %>% select(OTU, Phylum, Class, Order, Family, Genus, Species) %>%
unique()
IndicatorValues <- join(IndicatorValues, Tax, "OTU")
write.tidy(IndicatorValues, "IndicatorValues_0.5_tax.txt")
```

(7) In-house script for cross-domain analysis

To perform cross-domain analysis, bacterial clean OTU table was normalized to a depth of 2, 268 reads per sample using the QIIME script "multiple_rarefaction.py". Bacterial alpha- and beta- diversity were generated using the QIIME scripts "alpha_diversity.py" and "beta_diversity.py", respectively.

7.1) cross-domain alpha-diversity analysis

```
library(readr)
library(stringr)
library(dplyr)
library(ggplot2)

# Load the average alpha diversity table of bacteria and fungi
Bacteria <- read_tsv("Bacteria_alpha_diversity_average_2268.txt")
Fungi <- read_tsv("Fungi_alpha_diversity_average_1058.txt")

# Merge the the alpha diversity tables of bacteria and fungi. The columns should be renamed before the merging. For example, rename fungal chao1 as chao1_f, and bacterial chao1 as chao1_b.
Merge <- merge(Fungi, Bacteria, by = "Sample", all.x = FALSE)

# Save the merged file as text file
write_tsv(Merge, "Bacteria_Fungi_alpha_diversity_average.txt")

# add metadata
Meta <- read_tsv("samples.txt")

# Merge the alpha diversity average table with metadata
Alpha <- read_tsv("Bacteria_Fungi_alpha_diversity_average.txt")
Merge <- merge (Alpha, Meta, by = "Sample", all.x = TRUE)
write_tsv(Merge, "Alpha_diversity_metadata_bacteria_fungi.txt")

#Spearman statistics
cor.test(Merge$chao1_f, Merge$chao1_b, method="spearman")
```

7.2) cross-domain beta-diversity analysis

```
library(readr)
library(stringr)
library(dplyr)
library(ggplot2)

# Load the bacteria and fungi beta diversity Bray-Curtis tables
Bacteria <- read_tsv("Beta_Comparison_Bray_Curtis_Bacteria.txt")
names(Bacteria)[3] <- c("Distance_b")
Bacteria <- Bacteria %>%
        select(Distance_b, Comparison)

Fungi <- read_tsv("Beta_Comparison_Bray_Curtis_Fungi.txt")
names(Fungi)[3] <- c("Distance_f")
Fungi <- Fungi %>%
        select(Distance_f, Comparison)

#Merge the tables into the "combined" text file
Merge <- merge(Fungi, Bacteria, by = "Comparison", all.x = FALSE)
write_tsv(Merge, "Bacteria_fungi_Beta_Comparison_Bray_Curtis.txt")

#Statistic
cor.test(Merge$Distance_f, Merge$Distance_b, method="spearman")

# Load the bacteria and fungi beta diversity Binary Jaccard tables
Bacteria <- read_tsv("Beta_Comparison_Binary_Jaccard_Bacteria.txt")
names(Bacteria)[3] <- c("Distance_b")
Bacteria <- Bacteria %>%
        select(Distance_b, Comparison)

Fungi <- read_tsv("Beta_Comparison_Binary_Jaccard_Fungi.txt")
names(Fungi)[3] <- c("Distance_f")
Fungi <- Fungi %>%
        select(Distance_f, Comparison)

#Merge the tables into the "combined" text file
Merge <- merge(Fungi, Bacteria, by = "Comparison", all.x = FALSE)
write_tsv(Merge, "Bacteria_fungi_Beta_Comparison_Binary_Jaccard.txt")

#Statistic
cor.test(Merge$Distance_f, Merge$Distance_b, method="spearman")
```

(8) In-house script for SourceTracker analysis

8.1 SourceTracker Accuracy model for indoor dispersal prediction

OTU table preparation and SourceTracker analysis are performed in QIIME

```
# Filter OTUs present in less than 10% of the samples from the OTU table, a total of 428 samples are included in the study
filter_otus_from_otus_table.py -i OTU_table_clean_json.biom -o filtered_otu_table_10%.biom -s 43
# Convert table from BIOM to tab-separated text format
biom convert –i filtered_otu_table_10%.biom –o filtered_otu_table.txt --to-tsv
```

map preparation

The mapping files are prepared for all pairwise sample groups based on different building and location factors, with each file containing three columns, #SampleID, SourceSink, and Env.

```
# Load packages
library(readr)
library(stringr)
library(dplyr)
library(wilkoxmisc)
library(reshape2)

# Load OTU Table
OTUTable <- read_tsv("samples.txt")
# Generate Air_Surface_map.txt
AirSurface <- OTUTable %>%
        filter(Type %in% c("Air", "Surface")) %>%
   filter(Count >= 1000) %>%
        mutate(SourceSink = ifelse(Type == "Air", "source", "sink")) %>%
        mutate(Env = paste(Household, Type)) %>%
        select(Sample, Env, SourceSink)

names(AirSurface)[1] <- c("#SampleID")
write.tidy(AirSurface, "Air_Surface_map.txt")

# Generate Air_Surface_High_Deposition_map.txt
AirSurfaceHighDeposition <- OTUTable %>%
        filter(Type %in% c("Air", "Surface")) %>%
        filter(Count >= 1000) %>%
        filter(SurfaceDeposition %in% c("High", NA)) %>%
        mutate(SourceSink = ifelse(Type == "Air", "source", "sink")) %>%
        mutate(Env = paste(Household, Type)) %>%
        select(Sample, Env, SourceSink)

names(AirSurfaceHighDeposition)[1] <- c("#SampleID")
```

```
write.tidy(AirSurfaceHighDeposition, "Air_Surface_High_Deposition_map.txt")

# Generate Air_Surface_Low_Deposition_map.txt
AirSurfaceLowDeposition <- OTUTable %>%
        filter(Type %in% c("Air", "Surface")) %>%
        filter(Count >= 1000) %>%
        filter(SurfaceDeposition %in% c("Low", NA)) %>%
        mutate(SourceSink = ifelse(Type == "Air", "source", "sink")) %>%
        mutate(Env = paste(Household, Type)) %>%
        select(Sample, Env, SourceSink)
        names(AirSurfaceLowDeposition)[1] <- c("#SampleID")
write.tidy(AirSurfaceLowDeposition, "Air_Surface_Low_Deposition_map.txt")
```

Take the Air_Surface_High_Deposition_map.txt map for example, the indoor air samples are used as sources, with household surfaces with high deposition potentials as sinks in the SourceSink column. Each row in the Env column contains a description of the corresponding sample type, in this case, air and surface.

```
# Run SourceTracker
R --slave --vanilla --args –i filtered_otu_table_10%.biom –m
Air_Skin_High_Deposition_map.txt –o sourcetracker_out_air_skin_high_deposition <
$SOURCETRACKER_PATH/sourcetracker_for_qiime.r
```

```
# The analysis of SourceTracker results is performed in R (version 3.3.0).
# Load sourcetracker result
load("results.RData")
# Tidy results
SourceTrackerResults <- as.data.frame(results$proportions)
SourceTrackerResults$Sink <- rownames(SourceTrackerResults)
SourceTrackerResults <- melt(SourceTrackerResults, id.var = "Sink", variable.name =
"Source", value.name = "Accuracy")

# Add type and household to sink samples
Samples <- read.tidy("samples.txt")[c("Sample", "Household", "Type")]
SourceTrackerResults <- merge(SourceTrackerResults, Samples, by.x = "Sink", by.y =
"Sample", all.x = TRUE)
SourceTrackerResults$Sink <- paste(SourceTrackerResults$Household,
SourceTrackerResults$Type)
SourceTrackerResults <- SourceTrackerResults[c("Sink", "Source", "Accuracy")]

# Take the average for each sink environment
SourceTrackerResults <- ddply(SourceTrackerResults, .(Sink, Source), summarise, Accuracy
```

```
= mean(Accuracy))

# Add "self" variable
SourceTrackerResults$Self <- factor(gsub("\\s\\S+$", "", SourceTrackerResults$Sink) ==
gsub("\\s\\S+$", "", SourceTrackerResults$Source))
SourceTrackerResults$Sink <- factor(SourceTrackerResults$Sink)

# For each sink, decide whether self is largest source
is.self.largest <- function(Sink) {
  # Skip if no self-self possible
if ( ! "TRUE" %in% Sink$Self) { return(FALSE)
}
Max <- max(Sink$Accuracy)
return(ifelse(Max == Sink[which(Sink$Self == "TRUE"), "Accuracy"], TRUE, FALSE)) }
SourceTrackerResults <- ddply(SourceTrackerResults, .(Sink), is.self.largest)

Accuracy <- sum(SourceTrackerResults[,2]) / nrow(SourceTrackerResults)*100
```

Calculate the accuracy for each route, and build a table with Subset (for example, overall, High Occupancy, Low Deposition), Pair (for example, Air-to-Skin, Surface-to-Air) and accuracy columns, and save as text file.

```
# draw SourceTracker accuracy heatmap
# Load packages
library(readr)
library(ggplot2)
library(knitr)
library(viridis)
library(ggthemes)
library(OIdata)

sourcetracker <- read_tsv("sourcetracker_accuracy.txt")
kable(head(sourcetracker))
Plot <- ggplot(sourcetracker, aes(y = Subset, x = Pair, fill = Accuracy))
Plot <- Plot + geom_tile(color="white", size = 0.1)
Plot <- Plot + coord_equal()
Plot <- Plot + scale_x_discrete(limits=c("Skin-to-Surface", "Surface-to-Skin", "Air-to-
Surface", "Surface-to-Air", "Skin-to-Air", "Air-to-Skin"))
Plot <- Plot + scale_y_discrete(limits=c("Total", "High Occupancy", "Low Occupancy", "High
Deposition", "Low Deposition","High Touch", "Low Touch"))
Plot <- Plot + theme_tufte(base_family = "Helvetica")
Plot <- Plot + theme(plot.title=element_text(hjust=0))
```

```
Plot <- Plot + theme(axis.ticks=element_blank())
Plot <- Plot + theme(axis.text=element_text(size=7.6))
Plot <- Plot + theme(axis.title = element_blank())
Plot <- Plot + scale_fill_viridis()
Plot <- Plot + theme(legend.title=element_text(size=10))
Plot <- Plot + theme(legend.text=element_text(size=8))
Plot <- Plot + theme(legend.key.size=unit(2.7, "cm"))
Plot <- Plot + theme(legend.key.width=unit(0.8, "cm"))
```

8.2 Contributor OTUs

In order to further study which OTUs can be transferred between air, skin and surfaces, and the extent to which the OTUs contribute to shaping the sink community, the relative contribution rates of the source OTUs to the correct sink communities were assessed for all households. Here, source OTUs are referred to as contributor OTUs if they contribute to the corresponding sink communities within the same household. The proportions of contributor OTUs were aggregated at the genus level.

Take skin-to-surface as example. All the skin samples as sources, and surface samples as sinks. Enter into the directory: sourcetracker_prediction/Skin_Surface/full_results. Calculate the contribution proportion of contributor OTUs to the surface community for each household (here we take household WU Kai Sha for example), and generate a new text file ("Skin_contribution_to_surface.txt") containing all the households.

```
# Load packages
library(readr)
library(stringr)
library(dplyr)
library(tidyr)

OTU <- read_tsv("sink_predictions_Wu_Kai_Sha_Surface_contributions.txt")
OTU <- OTU %>% gather(OTU, Percentage, 2:(ncol(OTU))) %>% filter(Percentage >0)
names(OTU)[1] <- c("Sample")
sample <- read_tsv("OTUTable_taxonomy_metadata.txt")
merge <- left_join(OTU, sample)
unique(merge$Household)
merge <- merge %>%
      filter(Household  == "Wu Kai Sha") %>%
      mutate(Percentage = Percentage * 100) %>%
      select(Sample, OTU, Percentage, Type, Household, Area, Phylum, Class, Order,
Family, Genus, Species)
write_tsv(merge, "Wu_Kai_Sha_surface_contribution.txt")
```

```
#For the surface community in each household, we can generate one text file containing the
contrition of skin community from all occupants. Combine the the all contribution text files
into one using the "rbind" function in R and rename it as
"Skin_contribution_to_surface.txt".
OTU <- read_tsv("Skin_contribution_to_surface.txt")

#Calculate the sum contribution of skin community from five body sites of each occupant
OTU <- OTU %>%
```

```
        select(Sample, Percentage, Household, Area, Genus) %>%
        group_by(Sample, Genus) %>%
        mutate(Percentage = sum(Percentage)) %>%
        ungroup() %>%
        unique()

Skin <- read_tsv("OTUTable_taxonomy_metadata.txt") %>%
           filter(Type == "Skin") %>%
           select(Sample, Household, Individual, Room) %>%
           unique()

# add household category by merging the variables “OTU” and “Skin”
OTU <- merge(Skin, OTU, by = "Sample", all.x = TRUE)
write_tsv(OTU, ("Skin_contribution_to_surface_average.txt

#Load packages
library(readr)
library(stringr)
library(dplyr)
library(tidyr)
library(extrafont)
library(ggplot2)
OTU <- read_tsv("Skin_contribution_to_surface_average.txt")
#Indicate the color names for each genus
cbPalette <- c("#a6cee3", "#1f78b4", "#b2df8a", "#33a02c","#fb9a99", "#e31a1c", "#fdbf6f",
"#ff7f00", "#cab2d6", "#6a3d9a", "#ffff99", "#b15928","#fbb4ae", "#b3cde3", "#ccebc5",
"#decbe4", "#fed9a6", "#ffffcc", "#e5d8bd","#fddaec", "#f2f2f2")
#Rearrange the order of the household code name according to the correction and false
prediction
OTU$Household <- factor(OTU$Household2, levels = c("FT", "FH", "HFC", "HHA", "KT",
"QB",  "TY", "TMB", "WTS", "WKS","ADMA", "MOS", "SW", "STW", "TK", "TH", "TMA"))
#Rearrange the order of the genera in the legend
OTU$Genus <- factor(OTU$Genus, levels = c("Alternaria", "Aspergillus", "Aureobasidium",
"Candida","Cladosporium","Cryptococcus", "Debaryomyces", "Emericella", "Fusarium",
"Lentinula","Malassezia" , "Nigrospora", "Paecilomyces",  "Penicillium",
"Pichia","Rhodotorula", "Sporobolomyces" ,  "Sterigmatomyces" ,
"unclassified_Capnodiales_genus" , "unclassified_Sclerotiniaceae_genus", "unidentified"))
#Make the plot
Plot <- ggplot(OTU, aes(x = Area, y = Percentage, fill = Genus))
Plot <- Plot + geom_bar(stat= "identity")
Plot <- Plot + theme_classic()
Plot <- Plot + scale_fill_manual(values = cbPalette)
```

```
Plot <- Plot + facet_wrap(~Household, scales = "free_x", nrow=1)
Plot <- Plot + theme(legend.position="none")
Plot <- Plot + theme(axis.text.x = element_text(size = 5, angle = 60, hjust = 1, family =
"Arial"))
Plot <- Plot + theme(axis.title = element_text(face = "bold", family = "Arial"))
Plot <- Plot + ylab(paste0("Percentage (%)")) + xlab(paste0("Residential surface (sink
community)"))
Plot <- Plot + scale_y_continuous(expand = c(0,0), breaks=c(seq(0,100,25)))
Plot <- Plot + theme(strip.text=element_text(hjust=0.5))
```

8.3 Tracking the source of the HK indoor air

To test the hypothesis that outdoor air was the dominant source for the indoor mycobiome, fungal community data from outdoor air samples from Beijing, China and California, USA were retrieved as surrogate sources as there is no similar dataset on the mycobiome of outdoor air in HK. The raw forward sequences from the indoor and outdoor air were processed using the UPARSE pipeline described above. The reads were filtered for quality, trimmed, and then clustered into OTUs using the QIIME scripts "pick_closed_reference_otus.py" and "pick_open_reference_otus.py" with default settings, respectively.

```
usearch –fastq_filter raw_sampleA_forward.fastq –fastq_trunclen 240 –fastq_maxee 0.5 –fastaout sampleA_filtered.fasta –relabel xx.1_

cat *_filtered.fasta > combined_air_filtered.fasta

pick_closed_reference_otus.py -i combined_air_filtered.fasta -r UNITE.fasta -t 97_otu_taxonomy.txt -o out_tax/

pick_open_reference_otus.py -i combined_air_filtered.fasta -r UNITE.fasta -t 97_otu_taxonomy.txt -o out_tax/
```

The ouput file is otu_table.biom, filter the OTU table using the QIIME script "filter_otus_from_otu_table.py", and the reads with less than 10% prevalence were excluded from the following study. The filtered OTU table was converted into text format using "biom convert" command.

```
filter_otus_from_otu_table –i otu_table.biom –o filtered_otu_table.biom –s 14

R --slave --vanilla --args -i filtered_otu_table_mc2_10%.txt -m map.txt -o sourcetracker_out < $SOURCETRACKER_PATH/sourcetracker_for_qiime.r
```

Plot the source tracking of indoor and outdoor air

```
#Load packages
library(plyr)
library(tidyr)
library(readr)
library(stringr)
library(dplyr)
library(wilkoxmisc)
library(reshape2)
library(ggplot2)

# Load SourceTracker data from file
```

```
 load("results.RData")
# Turn the matrix into data frame
 SourceTrackerResults <- results$proportions %>%
   as.data.frame()
# Add a new column called Sink containing all the sink samples
 SourceTrackerResults$Sink <- rownames(SourceTrackerResults)
# Rename the first and the second columns as Source and Proportion, the last column is Sink
 SourceTrackerResults <- SourceTrackerResults %>%
          gather(Source, Proportion, 1:(ncol(SourceTrackerResults) - 1))
write_tsv(SourceTrackerResults, "Source_predictions.txt")

SourceTrackerResults <- read_tsv("Source_predictions.txt")
names(SourceTrackerResults)[1] <- c("Sample")
Meta <- read_tsv("metadata.txt")
Merge <- merge(SourceTrackerResults, Meta, by = "Sample", all.x = FALSE)

#Calculate the average contribution of each outdoor air source on four air samples in one household
Merge <- Merge %>%
          select(Source, Proportion, Household) %>%
          group_by(Household, Source) %>%
          mutate(Proportion = mean(Proportion)) %>%
          ungroup() %>%
          unique()
write_tsv(Merge, "Source_prediction_average.txt")

Average <- read_tsv("Source_prediction_average.txt")
Meta <- read_tsv("metadata.txt") %>%
          select(Source, SourceType) %>%
          unique()
Merge <- merge(Average, Meta, by = "Source", all.x =TRUE)
Merge <- Merge %>%
          select(-Source) %>%
          group_by(Household, SourceType) %>%
          mutate(Proportion=sum(Proportion)) %>%
          unique()
write_tsv(Merge, "Source_Prediction_sum.txt")

Merge <- read_tsv("Source_Prediction_sum.txt")
cbPalette <- c("#291B4F","#CEEAE6","#FCD42B")
```

```
Merge$SourceType <- factor(Merge$SourceType, levels = c("Unknown Sources","USA Outdoor Air", "Beijing Outdoor Air"))
Plot <- ggplot(Merge[order(Merge$SourceType, decreasing= T),],aes(x = Household, y = Proportion * 100, fill = SourceType))
Plot <- Plot + geom_bar(aes(x=as.factor(Household)), stat="identity", position = "stack", width=0.5) + scale_y_continuous(breaks=c(0, 20, 40, 60, 80, 100))
Plot <- Plot + geom_bar(stat= "identity")
Plot <- Plot + coord_flip()
Plot <- Plot + theme_classic()
Plot <- Plot + scale_fill_manual(values = cbPalette)
Plot <- Plot + xlab(paste0("HK Household")) + ylab(paste0("Average proportion (%)"))
Plot <- Plot + theme(legend.position="bottom")
Plot <- Plot + theme(legend.title = element_blank())
Plot <- Plot + theme(axis.line=element_blank())

ggsave("sourcetracker_plot_open_reference.pdf", plot = Plot, height = 4.45, width = 7.98, units = "in")
```

(9) In-house script for Distance-decay analysis

Before performing the distance-decay analysis, bacterial and fungal community distances were generated using the Binary Jaccard distance metric respectively using QIIME script "beta_diversity.py". Mantel test in the P package "ade4" was used to determine the significance between the community dissimilarity and geographic distance.

Perform Mantel test

```
#Load packages
library(reshape2)
library(dplyr)
library(readr)
library(ade4)

#Load geographic distance and community distance text files
Household <- read_tsv("Household_distance_19.txt")
Fungi <- read_tsv("Beta_Comparison_Binary_Jaccard_Fungi_Air.txt")
Fungi$Comparison_Household <- paste0(Fungi$Household1, " ", "vs.", " ",
Fungi$Household2)
Merge <- merge(Fungi, Household, by = "Comparison_Household", all.x = TRUE)
Merge <- Merge %>%
        select(Comparison_Household, Comparison, Distance_f, Distance_Household)

#Convert the community distance and geographic distance into distance matrixes, which are
required for Mantel test
Fungi.dists <- dist(Merge$Distance_f)
Household.dists <- dist(Merge$Distance_Household)
as.matrix(Fungi.dists)[1:5, 1:5]
as.matrix(Household.dists)[1:5, 1:5]

#Perform Mantel test
mantel.rtest(Fungi.dists, Household.dists, nrepet = 9999)
```

Plot the distance-decay pattern for bacterial and fungal communities at three different spatial scales

```
#Load packages
library(reshape2)
library(ggplot2)
library(dplyr)
library(readr)
library(ade4)

#City-wide scale (19 households)
AllFungi <- read_tsv("Binary_Jaccard_Fungi_19_Households_Air.txt")
Plot1 <- ggplot(AllFungi, aes(x = Distance_Household, y = Distance_f))
Plot1 <- Plot1 + geom_point(size=1, color = "#b35806")
Plot1 <- Plot1 + scale_color_brewer(palette = "Set1")
Plot1 <- Plot1 + theme(axis.title.x = element_text(face="bold"))
Plot1 <- Plot1 + theme(axis.title.y = element_text(face="bold"))
Plot1 <- Plot1 + theme(axis.text.x = element_text(face="bold"))
Plot1 <- Plot1 + theme(axis.text.y = element_text(face="bold"))
Plot1 <- Plot1 + xlab(paste0("Geographic distances (km)")) + ylab(paste0("Fungal
community dissimilarity"))
Plot1 <- Plot1 + scale_y_continuous(breaks=seq(0, 1.0, 0.1))
Plot1 <- Plot1 + geom_smooth(method = "lm", se = FALSE, color = "#b35806")
Plot1 <- Plot1 + theme(panel.border = element_blank(),
panel.grid.major = element_blank())
Plot1 <- Plot1 + theme(axis.title= element_text(size=14))

 #City-wide scale (19 households)
AllBacteria <- read_tsv("Binary_Jaccard_Bacteria_19_Households_Air.txt")
AllBacteria <- read_tsv("Binary_Jaccard_Bacteria_19_Households_Air.txt")
Plot2 <- ggplot(AllBacteria, aes(x = Distance_Household, y = Distance_b))
Plot2 <- Plot2 + geom_point(size=1, colour = "#4a5b23")
Plot2 <- Plot2 + theme(axis.title.x = element_text(face="bold"))
Plot2 <- Plot2 + theme(axis.title.y = element_text(face="bold"))
Plot2 <- Plot2 + theme(axis.text.x = element_text(face="bold"))
Plot2 <- Plot2 + theme(axis.text.y = element_text(face="bold"))
Plot2 <- Plot2 + scale_color_brewer(palette = "Set1")
Plot2 <- Plot2 + xlab(paste0("Geographic distances (km)")) + ylab(paste0("Bacterial
community dissimilarity (Binary Jaccard)"))
Plot2 <- Plot2 + scale_y_continuous(breaks=seq(0, 1.0, 0.1))
Plot2 <- Plot2 + theme(panel.border = element_blank(),
panel.grid.major = element_blank())
Plot2 <- Plot2 + geom_smooth(method = "lm", se = FALSE, color = "#4a5b23")
```

```
Plot2 <- Plot2 + theme(axis.title= element_text(size=14))

multiplot(Plot2, Plot1, cols=2)

#North-South transect (7 households)
NSFungi <- read_tsv("Binary_Jaccard_Fungi_North_to_Sourth_Air.txt")
Plot5 <- ggplot(NSFungi, aes(x = Distance_Household, y = Distance_f))
Plot5 <- Plot5 + geom_point(size=0.5, color = "#b35806")
Plot5 <- Plot5 + scale_color_brewer(palette = "Set1")
Plot5 <- Plot5 + scale_y_continuous(breaks=seq(0, 1.0, 0.1))
Plot5 <- Plot5 + theme(axis.title.x = element_text(face="bold"))
Plot5 <- Plot5 + theme(axis.title.y = element_text(face="bold"))
Plot5 <- Plot5 + theme(axis.text.x = element_text(face="bold"))
Plot5 <- Plot5 + theme(axis.text.y = element_text(face="bold"))
Plot5 <- Plot5 + xlab(paste0("Geographic distance between households \n along the North-
South transect (km)")) + ylab(paste0("Fungal community dissimilarity"))
Plot5 <- Plot5 + theme(panel.border = element_blank(),
panel.grid.major = element_blank())
Plot5 <- Plot5 + geom_smooth(method = "lm", size = 0.7, se = FALSE, color = "#b35806")
Plot5 <- Plot5 + theme(axis.title= element_text(size=10))

#North-South transect (7 households)
NSBacteria <- read_tsv("Binary_Jaccard_Bacteria_North_to_Sourth_Air.txt")
Plot6 <- ggplot(NSBacteria, aes(x = Distance_Household, y = Distance_b))
Plot6 <- Plot6 + geom_point(size=0.5, colour = "#b35806")
Plot6 <- Plot6 + scale_y_continuous(breaks=seq(0, 1.0, 0.1))
Plot6 <- Plot6 + theme(axis.title.x = element_text(face="bold"))
Plot6 <- Plot6 + theme(axis.title.y = element_text(face="bold"))
Plot6 <- Plot6 + theme(axis.text.x = element_text(face="bold"))
Plot6 <- Plot6 + theme(axis.text.y = element_text(face="bold"))
Plot6 <- Plot6 + xlab(paste0("Geographic distance between households \n along the North-
South transect (km)")) + ylab(paste0("Bacterial community dissimilarity"))
Plot6 <- Plot6 + theme(panel.border = element_blank(),
panel.grid.major = element_blank())
Plot6 <- Plot6 + geom_smooth(method = "lm", size = 0.7, se = FALSE, color = "#b35806")
Plot6 <- Plot6 + theme(axis.title= element_text(size=10))

#West-East transect (8 households)
WEFungi <- read_tsv("Binary_Jaccard_Fungi_West_to_East_Air.txt")
Plot7 <- ggplot(WEFungi, aes(x = Distance_Household, y = Distance_f))
Plot7 <- Plot7 + geom_point(size=0.5, colour = "#4a5b23")
Plot7 <- Plot7 + scale_x_continuous(breaks=seq(0, 12, 3))
```

```
Plot7 <- Plot7 + scale_y_continuous(breaks=seq(0, 1.0, 0.1))
Plot7 <- Plot7 + theme(axis.title.x = element_text(face="bold"))
Plot7 <- Plot7 + theme(axis.title.y = element_text(face="bold"))
Plot7 <- Plot7 + theme(axis.text.x = element_text(face="bold"))
Plot7 <- Plot7 + theme(axis.text.y = element_text(face="bold"))
Plot7 <- Plot7 + xlab(paste0("Geographic distance between households \n along the East-
West transect (km)")) + ylab(paste0("Fungal community dissimilarity"))
Plot7 <- Plot7 + theme(panel.border = element_blank(),
panel.grid.major = element_blank())
Plot7 <- Plot7 + geom_smooth(method = "lm", size = 0.7, se = FALSE, color = "#4a5b23")
Plot7 <- Plot7 + theme(axis.title= element_text(size=10))

#West-East transect (8 households)
WEBacteria <- read_tsv("Binary_Bray_Curtis_Bacteria_West_to_East_Air.txt")
Plot8 <- ggplot(WEBacteria, aes(x = Distance_Household, y = Distance_b))
Plot8 <- Plot8 + geom_point(size=0.5, colour = "#4a5b23")
Plot8 <- Plot8 + scale_x_continuous(breaks=seq(0, 12, 3))
Plot8 <- Plot8 + scale_y_continuous(breaks=seq(0, 1.0, 0.1))
Plot8 <- Plot8 + theme(axis.title.x = element_text(face="bold"))
Plot8 <- Plot8 + theme(axis.title.y = element_text(face="bold"))
Plot8 <- Plot8 + theme(axis.text.x = element_text(face="bold"))
Plot8 <- Plot8 + theme(axis.text.y = element_text(face="bold"))
Plot8 <- Plot8 + xlab(paste0("Geographic distance between households \n along the East-
West transect (km)")) + ylab(paste0("Bacterial community dissimilarity"))
Plot8 <- Plot8 + theme(panel.border = element_blank(),
panel.grid.major = element_blank())
Plot8 <- Plot8 + geom_smooth(method = "lm",  size = 0.7, se = FALSE, color = "#4a5b23")
Plot8 <- Plot8 + theme(axis.title= element_text(size=10))

multiplot(Plot6, Plot5, Plot8, Plot7, cols=2)
```