**Optimizing methods to unveil the genetic potential and viable microbiome in the built environments (BEs)**

Cinta Gomez-Silvan (cgomezsilvan@lbl.gov)1,2

Marcus H. Y. Leung (mhyleung@cityu.edu.hk)3

Katherine A. Grue (katherine.grue@ucsf.edu) 1,4

Randeep Kaur (randeepkaur@lbl.gov)2

Xinzhao Tong (xinzhtong2-c@my.cityu.edu.hk)3

Patrick K. H. Lee (patrick.kh.lee@cityu.edu.hk)3, and

Gary L. Andersen (glandersen@lbl.gov)1,2

1Department of Environmental Science, Policy, and Management; University of California, Berkeley, CA, USA.

2Earth Sciences Division, Ecology Department, Lawrence Berkeley National Laboratory, Berkeley, CA, USA

3School of Energy and Environment, City University of Hong Kong, Hong Kong

4 Current affiliation: Department of Physical Therapy and Rehabilitation Science, University of California, San Francisco, CA, USA.

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 scripts for quality control and OTU-clustering
2. In-house script for taxonomic plot generation
3. In-house script for DeSeq2 plot generation
4. In-house script for dbRDA analysis
5. In-house script for RNA:DNA ratio analysis
6. In-house script for SPIEC-EASI network 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

1. **In-house script for OTU clustering**

OTU-clustering: following read merging, 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:

**1.1) assign\_OTU\_numbers.pl**

#!/usr/bin/perl

use Modern::Perl 2014;

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 UCHIME2, script below was used to generate txt file containing a list of chimeric OTUs:

**1.2) make\_chimeras\_list.pl**

#!/usr/bin/perl

use Modern::Perl 2014;

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

-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)

**1.3) prepare\_OTU\_table.pl**

#!/usr/bin/perl

use Modern::Perl 2014;

use autodie;

$|++;

# Load OTU reference sequences

say "Loading OTU reference seqences";

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, '>', [original OTU table]);

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 1.3), 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 5% of reads. The script below takes in OTU\_table.tidy.txt file from 1.3), and generates two output files:

-contaminant\_lineages.tidy.txt (a list of lineages deemed contaminants)

-contaminants.txt (a list of OTUs deemed contaminants)

**1.4) classify\_contaminants.R (perform for each negative sample to assess contaminating lineages)**

# Libraries

library(wilkoxmisc)

# List of blank samples

BlankSamples <- c("name\_of\_negative\_sample(s)")

# Read in OTU table

OTUTable <- read.tidy([original OTU table])

# 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) \* 0.05

# 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) and contaminant OTUs (contaminants.txt), 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.

**1.5) clean\_OTU\_table.R**

# Libraries

library(wilkoxmisc)

# Read in OTU table

OTUTable <- read.tidy([original OTU table])

# 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, [clean OTU table])

Prepare clean OTU\_table to format readable for biom\_convert command in QIIME:

**1.6) cast\_OTU\_table.R**

# Libraries

library(wilkoxmisc)

library(reshape2)

# Read in OTU table

OTUCounts <- read.tidy([clean OTU table])

# 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.

1. **In-house script for taxonomic plot generation**

**2.1 make\_taxo\_plot.R**

library(devtools)

library(wilkoxmisc)

library(reshape2)

library(ggplot2)

#Open taxonomy OTU table

OTU <- read.tidy([clean\_OTU\_table))

#Open Metadata table

Meta <- read.tidy([metadata file]")

#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 5 or 8 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, "Top15Genus\_wt\_control.txt")

#Plot (Genus)

OTUTable <- read.tidy("Top15Genus\_wt\_control\_average.txt")

OTUTable <- transform(OTUTable, Genus\_new = factor(Genus,levels=c("Sphingomonas", "Staphylococcus", "Micrococcus", "Porphyrobacter", "Mycobacterium", "Streptococcus", "Sphingopyxis", "Acinetobacter","Corynebacterium","Propionibacterium","Rothia","Haemophilus","Bradyrhizobium","Enhydrobacter","Minor/Unclassified")))

OTUTable$Genus <- factor(OTUTable$Genus,levels = c("Sphingomonas", "Staphylococcus", "Micrococcus", "Porphyrobacter", "Mycobacterium", "Streptococcus", "Sphingopyxis", "Acinetobacter","Corynebacterium","Propionibacterium","Rothia","Haemophilus","Bradyrhizobium","Enhydrobacter","Minor/Unclassified"))

OTUTable$Nucleic <- factor(OTUTable$Nucleic, levels =c("DNA","RNA"))

#Plot

Plot <- ggplot(OTUTable, aes(x = Nucleic, y = RelativeAbundance, fill = Genus))

Plot <- Plot + geom\_bar(data=OTUTable,aes(x=Nucleic,y=RelativeAbundance,fill=Genus),stat="identity",width=0.7)

Plot <- Plot + theme\_classic()

Plot <- Plot + scale\_fill\_manual(values = c(

# Brewer palette Set1, with eleventh value ("Other")

#  modified to a darker grey

"#E41A1C",

"#377EB8",

"#4DAF4A",

"#984EA3",

"#FF7F00",

"#FFFF33",

"#A65628",

"#54FF9F",

"#FF1493",

"#8B7500",

"#ffb90f",

"#caff70",

"#bf3eff",

"#528b8b",

"#cdc8b1",

"#458b00",

"#ff7f24",

"#cdc0b0",

"#cdcd00",

"#525252"

))

Plot <- Plot + theme(axis.title = element\_text(size = 12, face = "bold"))

Plot <- Plot + theme(legend.text = element\_text(size = 10))

Plot <- Plot + theme(legend.title=element\_blank())

Plot <- Plot + scale\_y\_continuous(expand=c(0,0))

Plot <- Plot + ylab(paste0("Relative Abundance (%)"))

Plot <- Plot + theme(axis.title.x = element\_blank())

Plot <- Plot + theme(legend.title=element\_blank())

ggsave("taxonomy\_plot.png",width=7,height=7,unit="in")

**(3) In-house script for DeSeq2 plot generation**

Following DeSeq2 analysis as described, OTUs that were not statistically significant following FDR correction were not included in input table.

**3.1 DeSeq\_plot.R**

library(devtools)

library(ggplot2)

library(wilkoxmisc)

#Open deseq table and merge with OTU taxonomic information

Table <- read.tidy([output file from deseq2 analysis, with non-sig otus removed])

Tax <- read.tidy([clean OTU table")

Tax$Count <- NULL

Tax$Sample <- NULL

Tax <- unique(Tax)

Merge <- merge(Table,Tax,by="OTU", all.X = TRUE)

write.tidy(Merge,[significant deseq table with taxonomy])

#Add extra column, negative deseq log values = “Neg”, positive values = “Pos”

#Draw deseq barplot

Table <- read.tidy(significant deseq table with taxonomy)

Table$Sign <- factor(Table$Sign, levels=c("Pos","Neg"))

Plot <- ggplot(Table, aes(x=reorder(Name,log2FoldChange), y=log2FoldChange, fill=Sign))

Plot <- Plot + geom\_bar(stat="identity")

Plot <- Plot + xlab(paste0("OTU")) + ylab(paste0("DeSeq2 Log-Fold Change"))

Plot <- Plot + theme\_classic()

Plot <- Plot + theme\_classic() + coord\_flip()

Plot <- Plot + theme(axis.text.y = element\_text(size=6))

Plot <- Plot + scale\_fill\_brewer(palette="Set1")

Plot <- Plot + theme(legend.position = "none")

ggsave("DeSeq2 figure")

1. **In-house script for dbRDA analysis**

**4.1 dbRDA.R**

#library(devtools)

library(wilkoxmisc)

library(ggplot2)

library(reshape2)

library(plyr)

library(vegan)

#Load UniFrac distance matrix

UniFrac <- read.dist([output distance matrix from beta\_diversity.py QIIME]")

#Load sample metadata

Samples <- read.tidy([meta file containing relative abundances of genera of interest in columns])

#Order and select sample metadata

Samples <- Samples[match(labels(UniFrac), Samples$Sample), ]

#Perform dbRDA

dbRDA <- capscale(UniFrac ~ Acinetobacter + Bradyrhizobium + Corynebacterium + Enhydrobacter + Haemophilus + Micrococcus + Mycobacterium + Porphyrobacter + Propionibacterium +  Rothia + Sphingomonas + Sphingopyxis + Staphylococcus + Streptococcus, Samples)

#Extract biplot

Biplot <- plot(dbRDA)

#Sample coordinates

SampleCoords <- as.data.frame(Biplot$sites)

SampleCoords$Sample <- factor(rownames(SampleCoords))

rownames(SampleCoords) <- NULL

SampleCoords <- merge(SampleCoords, Samples)

#Environmental coordinates

EnvCoords <- as.data.frame(Biplot$biplot) \* 1.5

EnvCoords$Variable <- rownames(EnvCoords)

rownames(EnvCoords) <- NULL

#Plot

Plot <- ggplot(SampleCoords, aes(x = CAP1, y = CAP2))

Plot <- Plot + geom\_point(aes(colour = Nucleic),size=4)

Plot <- Plot + theme\_classic()

Plot <- Plot + geom\_segment(data = EnvCoords, xend = 0, yend = 0,colour="grey")

Plot <- Plot + theme(axis.text = element\_text(size = 14, face="bold"))

Plot <- Plot + theme(axis.title = element\_text(size = 18, face = "bold"))

Plot <- Plot + theme(legend.text = element\_text(size = 14))

Plot <- Plot + theme(axis.line.x = element\_line(color="black", size=1))

Plot <- Plot + theme(axis.line.y = element\_line(color="black", size=1))

Plot <- Plot + theme(legend.title = element\_blank())

ggsave("dbRDA figure ",width=8,height=8,units="in")

1. **In-house script for RNA:DNA ratio analysis**

From read count data in clean OTU table, relative abundances of each OTU within DNA and RNA populations were calculated, and the ratio of relative abundance for the DNA and RNA populations for each OTU was added to a new data table. This ratio was plotted as Y-axis against the log DNA abundance of the OTU on the X-axis.

**5.1 plot\_ratio.R**

#Construct active:total ratio and total community abundance plot for each OTU and #include taxonomy data

Table <- read.tidy([abundance and ratio file for each OTU])

Tax <- read.tidy([OTU table with taxonomy data])

Tax$Sample <- NULL

Tax$Count <- NULL

Tax$Nucleic <- NULL

Tax <- unique(Tax)

Merge <- merge(Table,Tax,by="OTU", all.X = TRUE)

write.tidy(Merge,[ratio data with taxonomy])

#Plot ratio by DNA abundance

Merge <- read.tidy([ratio data with taxonomy])

Plot <- ggplot(Merge,aes(x=log\_ratio,y=DNA\_relative\_abund,colour=Class))

Plot <- Plot + geom\_point()

Plot <- Plot + scale\_colour\_manual(values = c(

# Brewer palette Set1, with eleventh value ("Other")

#  modified to a darker grey

"#E41A1C",

"#377EB8",

"#4DAF4A",

"#984EA3",

"#FF7F00",

"#FFFF33",

"#A65628",

"#54FF9F",

"#FF1493",

"#8B7500",

"#ffb90f",

"#caff70",

"#bf3eff",

"#528b8b",

"#cdc8b1",

"#458b00",

"#ff7f24",

"#cdc0b0",

"#cdcd00",

"#525252"

))

Plot <- Plot + xlab(paste0("log(Abundance in DNA Population)")) + ylab(paste0("RNA:DNA Ratio"))

Plot <- Plot + theme\_classic()

Plot <- Plot + geom\_hline(yintercept=1,linetype=2)

Plot <- Plot + theme(legend.title = element\_blank())

ggsave("DNA\_RNA\_ratio.png")

1. **In-house script for SPIEC-EASI network analysis**

**6.1 create\_network\_and\_structure\_properties\_plots.R**

# Libraries

library(readr)

library(SpiecEasi)

library(phyloseq)

library(tibble)

library(igraph)

library(dplyr)

library(stringr)

library(tidyr)

library(ggplot2)

library(Matrix)

library(orca)

# 1 Prepare phyloseq OTU table

OTUTableTotal <- [clean DNA-OTU table]%>%

read\_tsv %>%

select(OTU, Sample, Count) %>%

mutate(OTU = word(OTU, 1, sep = ";")) %>%

# 2 Select only most abundant OTUs

# Delete the marked lines below to include all OTUs

#group\_by(OTU) %>%                   # <- Delete

#mutate(TotalCount = sum(Count)) %>% # <- Delete

#ungroup %>%                         # <- Delete

#filter(TotalCount >= 150) %>%      # <- Delete

#select(-TotalCount) %>%             # <- Delete

spread(OTU, Count, fill = 0) %>%

as.data.frame %>%

remove\_rownames %>%

column\_to\_rownames("Sample") %>%

as.matrix %>%

otu\_table(taxa\_are\_rows = FALSE)

# 3 Prepare phyloseq taxa table

TaxaTableTotal <- [clean DNA-OTU table]%>%

read\_tsv %>%

select(OTU, Domain = Kingdom, Phylum, Class, Order, Family, Genus) %>%

mutate(OTU = word(OTU, 1, sep = ";")) %>%

unique %>%

as.data.frame %>%

remove\_rownames %>%

column\_to\_rownames("OTU") %>%

as.matrix %>%

tax\_table

# 4 Combine OTU and taxonomy into single phyloseq object

PhyseqTotal <- phyloseq(OTUTableTotal,TaxaTableTotal)

# 1 Prepare phyloseq OTU table

OTUTableActive <- [clean RNA OTU table] %>%

read\_tsv %>%

select(OTU, Sample, Count) %>%

mutate(OTU = word(OTU, 1, sep = ";")) %>%

# 2 Select only most abundant OTUs

# Delete the marked lines below to include all OTUs

#group\_by(OTU) %>%                   # <- Delete

#mutate(TotalCount = sum(Count)) %>% # <- Delete

#ungroup %>%                         # <- Delete

#filter(TotalCount >= 150) %>%      # <- Delete

#select(-TotalCount) %>%             # <- Delete

spread(OTU, Count, fill = 0) %>%

as.data.frame %>%

remove\_rownames %>%

column\_to\_rownames("Sample") %>%

as.matrix %>%

otu\_table(taxa\_are\_rows = FALSE)

# 3 Prepare phyloseq taxa table

TaxaTableActive <- [clean RNA OTU table] %>%

read\_tsv %>%

select(OTU, Domain = Kingdom, Phylum, Class, Order, Family, Genus) %>%

mutate(OTU = word(OTU, 1, sep = ";")) %>%

unique %>%

as.data.frame %>%

remove\_rownames %>%

column\_to\_rownames("OTU") %>%

as.matrix %>%

tax\_table

# 4 Combine OTU and taxonomy into single phyloseq object

PhyseqActive <- phyloseq(OTUTableActive,TaxaTableActive)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

#Determine degree statistics for each network

se.Active <- spiec.easi(PhyseqActive, method='mb', lambda.min.ratio=1e-2, nlambda=20,icov.select.params=list(rep.num=50, ncores=2))

ig.Active <- adj2igraph(se.Active$refit, vertex.attr=list(name=taxa\_names(PhyseqActive)))

dd.Active <- degree\_distribution(ig.Active, cumulative=FALSE)

sum(seq\_along(dd.Active)\*dd.Active)-1

se.Total <- spiec.easi(PhyseqTotal, method='mb', lambda.min.ratio=1e-2, nlambda=20,icov.select.params=list(rep.num=50, ncores=2))

ig.Total <- adj2igraph(se.Total$refit, vertex.attr=list(name=taxa\_names(PhyseqTotal)))

dd.Total <- degree\_distribution(ig.Total, cumulative=FALSE)

sum(seq\_along(dd.Total)\*dd.Total)-1

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## plot degree distributions

png("dd\_Active\_and\_total\_air.png",width=4,height=4,units="in",res=1200)

#par(mfrow=c(3,5))

par(mar=c(4,4,4,4))

plot(seq\_along(dd.Active)-1, dd.Active, type='b', xlim=c(0,6), ylim=c(0,0.8),

ylab="Frequency", xlab="Degree", col='red')

points(seq\_along(dd.Total)-1, dd.Total, type='b', col='blue',pch=8)

legend("topright", c("RNA Air", "DNA Air"), pch=c(1,8,0))

title(main="Degree Distribution")

dev.off()

#Network stability estimation by natural connectivity

#based on betweenness centrality

natcon <- function(ig) {

    N <- vcount(ig)

    adj <- get.adjacency(ig)

    evals <- eigen(adj)$value

    nc <- log(mean(exp(evals)))

    nc / (N - log(N))

}

nc.attack <- function(ig) {

    hubord <- order(rank(betweenness(ig)), decreasing=TRUE)

    sapply(1:round(vcount(ig)\*.95), function(i) {

        ind <- hubord[1:i]

        tmp <- delete\_vertices(ig, V(ig)$name[ind])

        natcon(tmp)

    })

}

#based on decreasing number of degrees

natcon <- function(ig) {

    N <- vcount(ig)

    adj <- get.adjacency(ig)

    evals <- eigen(adj)$value

    nc <- log(mean(exp(evals)))

    nc / (N - log(N))

}

nc.attack <- function(ig) {

    hubord <- order(rank(degree(ig)), decreasing=TRUE)

    sapply(1:round(vcount(ig)\*.95), function(i) {

        ind <- hubord[1:i]

        tmp <- delete\_vertices(ig, V(ig)$name[ind])

        natcon(tmp)

    })

}

nc.Active <- nc.attack(ig.Active)

nc.Total <- nc.attack(ig.Total)

#nc.ActiveSurface <- nc.attack(ig.ActiveSurface)

png("connectivity\_overall\_betweenness\_air\_only.png",width=4,height=4,units="in",res=1200)

plot(seq(0,0.8,len=length(nc.Active)),nc.Active, type='l', col='red', ylim=c(0,max(nc.Active)), xlab="Proportion of Removed Nodes", ylab="Natural Connectivity",font.lab=2)

points(seq(0,0.8,len=length(nc.Total)),nc.Total,type='l', col='blue')

#points(seq(0,0.8,len=length(nc.ActiveSurface)),nc.ActiveSurface,type='l', col='green')

title(main="Betweenness-Based")

dev.off()