Degree Distribution plots

Creating a vector with 5 colony names

```
nests=as.character(as.factor(c("v57", "v87", "v72", "v99", "v82")))
class(nests)

## [1] "character"

## [1] "character"
```

Getting out-degree and out-strength of each wasp in 5 colonies

```
library(stringi)

for (n in 1:length(nests)){
  nest_path=paste("D:/data/…/…/",nests[n],"/", sep="")
  setwd(nest_path)
  assign(paste(nests[n],"_vers",sep=""), read.csv(paste(nests[n],"_nodeVersat
ility.csv", sep=""), header=T))


  assign(paste(nests[n],"_1",sep=""),subset(eval(parse(text = paste(nests[n],
"_vers",sep=""))), Layer=="1-Multi")) # versatility in multilayer


  assign(paste(nests[n],"_agg",sep=""),subset(eval(parse(text =paste(nests[n]
,"_vers",sep=""))), Layer=="Aggr"))    #centrality in aggregate network


}
```

Multilayer network degree distribution (normalized for colony size)

```
x <- list(v99=v99_1$DegreeOut/nrow(v99_1), v87=v87_1$DegreeOut/nrow(v87_1),
          v57 = v57_1$DegreeOut/nrow(v57_1), v82=v82_1$DegreeOut/nrow(v82_1),
          v72=v72_1$DegreeOut/nrow(v72_1))

library(ggplot2);library(reshape2)
data<- melt(x)
data_count<-as.data.frame(aggregate(data$value, by = data[c('value','L1')], l
ength))

## Multilayer degree distribution
### High resolution image
safe_colorblind_palette <- c("#88CCEE", "#CC6677", "#DDCC77", "#117733", "#33
2288", "#AA4499",
                             "#44AA99", "#999933", "#882255", "#661100", "#66
```

```
99CC", "#888888")

#tiff("Multilayer_DD2.tiff", units="in", width=8, height=5, res=300)
par(xpd = F)
par(mar=c(4.15,4.15,4,8))
#par(xpd = F, mar = par()$mar + c(0,0,0,7))
plot(density(v99_1$DegreeOut/nrow(v99_1)), col = safe_colorblind_palette[1],
lwd = 3, lty =2,
     #main ="Degree distribution in multilayer network",
     main=NA,
     xlab="Out-degree versatility normalized by colony size",
     cex.lab = 1.45, cex.main =1.5)
lines(density(v87_1$DegreeOut/nrow(v87_1)), col = safe_colorblind_palette[2],
lwd = 3, lty = 1)
lines(density(v82_1$DegreeOut/nrow(v82_1)), col = safe_colorblind_palette[10]
, lwd = 3, lty = 4)
lines(density(v72_1$DegreeOut/nrow(v72_1)), col = safe_colorblind_palette[3],
lwd = 3,  lty = 5)
lines(density(v57_1$DegreeOut/nrow(v57_1)), col = safe_colorblind_palette[5],
lwd = 3,  lty = 6)


Colony <- c("v57", "v72", "v82", "v87","v99")  # one per row
colors <- c(safe_colorblind_palette[5],
            safe_colorblind_palette[3],
            safe_colorblind_palette[10],
            safe_colorblind_palette[2],
            safe_colorblind_palette[1])  # one color for each row; should be
same length as Vars
ltys <- c(6,5,4,1,2)  # one lty for each column
lwds<-rep(3,5)
nc <- length(ltys)
par(xpd = T)
legend(2.35, 2.35, Colony, col = colors, lty = ltys,lwd = lwds,
       #nrow = nc,
       cex = 1.25, bty = "n", title = "Colony")


text(2.55,3, "A", cex=2)

#dev.off()
```

Aggregate network degree distribution (normalized for colony size)

```
xA <- list(v99=v99_agg$DegreeOut/nrow(v99_1),v87=v87_agg$DegreeOut/nrow(v87_1
),
           v57 = v57_agg$DegreeOut/nrow(v57_1), v82=v82_agg$DegreeOut/nrow(v8
2_1),
           v72=v72_agg$DegreeOut/nrow(v72_1))
```

```r
library(ggplot2);library(reshape2)
dataA<- melt(xA)

### High resolution image
#tiff("Aggregate_DD2.tiff", units="in", width=8, height=5, res=300)
par(xpd = F)
par(mar=c(4.15,4.15,4,8))
#par(xpd = F, mar = par()$mar + c(0,0,0,7))
plot(density(v99_agg$DegreeOut/nrow(v99_agg)), col = safe_colorblind_palette[
1], lwd = 3, lty =2,
     #main ="Degree distribution in aggregate network",
     main = NA,
     xlab="Out-degree normalized by colony size",
     cex.lab = 1.45, cex.main =1.5, ylim = c(0, max(density(v57_agg$DegreeOut
/nrow(v57_agg))$y)+1))
lines(density(v87_agg$DegreeOut/nrow(v87_agg)), col = safe_colorblind_palette
[2], lwd = 3, lty = 1)
lines(density(v82_agg$DegreeOut/nrow(v82_agg)), col = safe_colorblind_palette
[10], lwd = 3, lty = 4)
lines(density(v72_agg$DegreeOut/nrow(v72_agg)), col = safe_colorblind_palette
[3], lwd = 3,  lty = 5)
lines(density(v57_agg$DegreeOut/nrow(v57_agg)), col = safe_colorblind_palette
[5], lwd = 3,  lty = 6)


Colony <- c("v57", "v72", "v82", "v87","v99")  # one per row
colors <- c(safe_colorblind_palette[5],
            safe_colorblind_palette[3],
            safe_colorblind_palette[10],
            safe_colorblind_palette[2],
            safe_colorblind_palette[1])  # one color for each row; should be
same length as Vars
ltys <- c(6,5,4,1,2)  # one lty for each column
lwds<-rep(3,5)
nc <- length(ltys)
par(xpd = T)
legend(1.04, 45, Colony, col = colors, lty = ltys,lwd = lwds,
       #nrow = nc,
       cex = 1.25, bty = "n", title = "Colony")


text(1.1,60, "B", cex=2)

#dev.off()
```

Randomization code for Degree in Multilayer Networks

R Markdown

Creating a vector with 5 colony names

```
nests=as.character(as.factor(c("v57", "v87", "v72", "v99", "v82")))
class(nests)

## [1] "character"

library(igraph)

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library(stringi)
```

#Randomization loop after sourcing different colony extended edgelists

```
for (n in 1:length(nests)){
  #n=1
  nest_path=paste("D:/data/",nests[n],"/", sep="")

  #recommended to set up wd this way for RMarkdown

#knitr::opts_chunk$set(echo = TRUE)
#knitr::opts_knit$set(root.dir = "")


#getwd()

  setwd(nest_path)
  ns_wasp_dataOriginal = read.csv(paste(nests[n],"_txtedges.csv", sep=""), he
ader=T)[,c(1:5)]

  library("dplyr")
ns_wasp_dataOrig<-as.data.frame(ns_wasp_dataOriginal %>% group_by(lyr1) %>% m
utate(NorWt = wt/max(wt)))

  ns_wasp_data <- ns_wasp_dataOrig[,c(1,2,3,4,6)]
  colnames(ns_wasp_data)[5]<-'wt'
```

```
  .libPaths( c( .libPaths(), "D:/Users/XXXXXX/Documents/R/win-library/3.6") )

  source("C:/Users/XXXXX/OneDrive/Documents/R/win-library/packages/muxViz-mas
ter//muxLib.R")

  ##Building the supra-adjacency matrix from extended edgelist

  sadjmat=BuildSupraAdjacencyMatrixFromExtendedEdgelist(
    mEdges=ns_wasp_data,
    Layers=length(unique(ns_wasp_data$lyr1)),
    Nodes=length(unique(ns_wasp_data$node1)), isDirected=T)

  ##calculating out-degree in multilayer

  multi_deg<-as.data.frame(GetMultiOutDegreeSum(sadjmat,
                                                Layers=length(unique(ns_was
p_data$lyr1)),
                                                Nodes=length(unique(ns_wasp
_data$node1)),DIRECTED))




  labels<-read.csv(paste(nests[n],"node_labels.csv",sep=""), header = T)

  ##Sort labels and degree in descending order

  library('dplyr')

  act_deg<-cbind(labels[,2],multi_deg)
  colnames(act_deg)<-c("ID","deg")
  ##assign a value to a customized name by using "assign"
  assign(paste(nests[n],"_PQobsDeg",sep=""),act_deg$deg[act_deg$ID=="PQ"])


 ##### RANDOMIZATION TEST  ######

  df=ns_wasp_data
  df$node1<-as.factor(as.character(df$node1))
  df$node2<-as.factor(as.character(df$node2))

  head(df)

  PQ_Rdegs<-list()


  for(i in 1:1000){ ## for each iteration
```

```
    rand_DF<-list() ## this gets empty because we are shuffling within each l
ayer and then making a combined extended edgelist for all 4 layers in a go

    for(j in 1: length(levels(as.factor(df$lyr1)))) { ##only shuffle within l
ayers
      df1<-subset(df, lyr1 == j) # subset for that layer
      #length(unique(levels(as.factor(df1[,3]))))
      #creating an igraph object
      el1=as.matrix(df1[,c(1,3,5)]) #igraph needs the edgelist to be in matri
x format
      g1=graph.edgelist(el1[,c(1,2)], directed=TRUE) #We first create a netwo
rk from the first two columns, which has the list of vertices
      E(g1)$weight=as.numeric(el1[,3])

      g2=g1
      V(g2)$name=sample(V(g1)$name) ## sampled/shuffled the vertices

      df_rand=get.data.frame(g2)

      nrow(df_rand)
      nrow(df1)
      df1_rand<-df_rand
      df1_rand$node1<-df_rand$from
      df1_rand$lyr1<-df1$lyr1 ## making it an extended edgelist by adding lay
ers too
      df1_rand$node2<-df_rand$to
      df1_rand$lyr2<-df1$lyr2
      df1_rand$wt<-df_rand$weight ##df1_rand is the shuffled vertices' extend
ed edgelist

      df2_rand=df1_rand[,c("node1","lyr1","node2","lyr2","weight")] ##pruning
      colnames(df2_rand)<-c("node1","lyr1","node2","lyr2","wt")
      tail(df2_rand)
      rand_DF<-c(rand_DF, list(df2_rand)) ##list of different shuffled within
layer extended edgelists

    }

    #convert shuffled list to shuffled dataframe

    shuff_DF<-do.call(rbind.data.frame, rand_DF) ## making it a single shuffl
ed dataframe for all 4 layers combined
    tail(shuff_DF)

    #subset(shuff_DF, shuff.nodes1==shuff.nodes2)
    nrow(shuff_DF)
    nrow(df)
    #View(as.data.frame(shuff_DF))
    ## create an igraph object to work on
```

```
    library(igraph)
    el=as.matrix(shuff_DF[,1:5]) #igraph needs the edgelist to be in matrix f
ormat
    g=graph.edgelist(el[,c(1,3)], directed=TRUE) #We first create a network f
rom the first two columns, which has the list of vertices
    E(g)$weight=as.numeric(el[,5]) #We then add the edge weights to this netw
ork by assigning an edge attribute called 'weight'.



    nrow(shuff_DF)
    ## Re-create supra-adjacency matrix for shuffled nodes with same node 2 a
nd weights (because still technically shuffled)
    class(shuff_DF$node2)
    shuff_DF[,1]<-as.numeric(as.character(shuff_DF[,1]))
    shuff_DF[,2]<-as.numeric(as.character(shuff_DF[,2]))
    shuff_DF[,3]<-as.numeric(as.character(shuff_DF[,3]))
    shuff_DF[,4]<-as.numeric(as.character(shuff_DF[,4]))
    shuff_DF[,5]<-as.numeric(as.character(shuff_DF[,5]))

    source("C:/Users/XXXXXX/OneDrive/Documents/R/win-library/packages/muxViz-
master//muxLib.R") #replace with your file path obviously...

    sadjmat1=BuildSupraAdjacencyMatrixFromExtendedEdgelist(
      mEdges=shuff_DF,
      Layers=length(unique(shuff_DF$lyr1)),
      Nodes=length(unique(shuff_DF$node1)), isDirected=T)

    ## calculate degree in multilayer network iteration

    multi_deg1<-as.data.frame(GetMultiOutDegreeSum(sadjmat1,
                                                   Layers=length(unique(shu
ff_DF$lyr1)),
                                                   Nodes=length(unique(shuf
f_DF$node1)), DIRECTED))

    labels<-read.csv(paste(nests[n],"node_labels.csv",sep=""), header = T)
    nrow(labels)
    ##Sort labels and degree in descending order

    library('dplyr')

    iter_deg<-cbind(labels[,2],multi_deg1) ## replacing numbers as nodes to n
ames as nodes
    ## assign multilayer versatility of shuffled dataframe to the ID
    colnames(iter_deg)<-c("ID","deg")


    ## degree of the PQ in randomized iteration
```

```
    PQ_Rdeg<-iter_deg$deg[iter_deg$ID=="PQ"]

    PQ_Rdegs<-c(PQ_Rdegs, list(PQ_Rdeg))

  }

  Degrees_df<-do.call(rbind.data.frame, PQ_Rdegs)



  assign(paste(nests[n],"_rand.deg",sep=""),as.vector(do.call(rbind.data.fram
e, PQ_Rdegs))[,1])
         #,inherits = TRUE)

  assign(paste(nests[n],"_DF",sep=""),as.data.frame(cbind(nest=rep(paste(nest
s[n]),1000),
                                                          assign(paste(nests[
n],"_randomRanks", sep=""),eval(parse(text=paste(nests[n],"_rand.deg", sep=""
)))))))
  hist(eval(parse(text = paste(nests[n],"_rand.deg",sep=""))), main = paste("
Nest ",nests[n],"(Out-degree)", sep=""), las=1,
       xlab="Out-degree") ## to evaluate the context of the text output of pa
ste, use eval(parse(text=...))
  assign(paste(nests[n],"_qts_deg", sep=""),
         quantile(eval(parse(text=paste(nests[n],"_rand.deg", sep=""))),probs
=c(.025,.975)))
  abline(v=eval(parse(text=paste(nests[n],"_PQobsDeg",sep=""))), col="red", l
wd=2)
  abline(v=eval(parse(text=paste(nests[n],"_qts_deg", sep=""))), col="blue",
lwd=2, lty=4)



  assign(paste(nests[n],"_lines", sep=""),
         as.data.frame(c(as.numeric(as.character(eval(parse(text=paste(nests[
n],"_qts_deg", sep="")))))),
                       as.numeric(as.character(eval(parse(text=paste(nests[
n],"_PQobsDeg",sep=""))))))))
  cat_lines=c("Quantile","Quantile",paste(nests[n],"_PQobs", sep=""))
  assign(paste(nests[n],"_lines", sep=""),
         as.data.frame(cbind(values=eval(parse(text=paste(nests[n],"_lines",
sep=""))))[,1],cat_lines)))

  library(ggplot2)
  #n=2
  p<-ggplot(eval(parse(text = paste(nests[n],"_DF",sep=""))),
            aes(x=eval(parse(text = paste(nest[n],"_rand.deg",sep="")))), fill
="orange",alpha=0.25)) +
    geom_density()+
```

```
    geom_vline(data=as.data.frame(eval(parse(text = paste(nests[n],"_lines",s
ep=""))))[1:2,]),
               aes(xintercept=as.numeric(as.character(eval(parse(text = paste
(nests[n],"_lines",sep=""))))[1:2,1])),
                   colour="blue",linetype="dashed",size=0.025))+
    geom_vline(data=as.data.frame(eval(parse(text = paste(nests[n],"_lines",
sep=""))))[3,]),
               aes(xintercept=as.numeric(as.character(eval(parse(text = paste
(nests[n],"_lines",sep=""))))[3,1])),colour="red",
               linetype="solid",size=0.025))

  #remove grey background
  p+theme_bw()+ #remove grey background
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blan
k())+#remove plot grids
    theme(text = element_text(size=20)) + labs(x = "Out-Degree in Multilayer"
, y="Frequency")



}
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:igraph':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Quantiles, Randomized and observed degrees of each of the 5 colonies

```
all_random_distributions <- list(v99=v99_rand.deg, v87=v87_rand.deg, v57 =v57
_rand.deg, v82=v82_rand.deg, v72=v72_rand.deg)

all_Quant_ablines<-list(v99=v99_lines,v87=v87_lines,v57=v57_lines,v82=v82_lin
es,v72=v72_lines)


library(ggplot2);library(reshape2)
data<- melt(all_random_distributions)
head(data)
```

```
data=data[c('L1','value')]
colnames(data)=c("nest","rand_dist")
Quant_ablines<-melt(all_Quant_ablines)

tail(Quant_ablines)

Quant_ablines<-Quant_ablines[c('L1','cat_lines','values')]
colnames(Quant_ablines)<-c("nest","cat_lines","line_value")
Quantiles_all=subset(Quant_ablines,cat_lines=="Quantile")
Quantiles_all$line_value<-as.numeric(as.character(Quantiles_all$line_value))
##change to numeric to avoid ggplot error of inputting factor instead of nume
ric (discrete in continuous error)
all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")
all_Observed$line_value<-as.numeric(as.character(all_Observed$line_value))##c
hange to numeric to avoid ggplot error of inputting factor instead of numeric
(discrete in continuous error)
```

ggplot histograms of all 5 colonies

```
library(ggplot2)
p<-ggplot(data, aes(x=rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=Quantiles_all, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=all_Observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-Degree in Multilayer",
y="Frequency")
```

Getting violin plots without having to run long randomizations each time by sourcing csv

```
#data=read.csv("NotStd_MultirandomStrength.csv")
#all_Observed=read.csv("NotStd_MultiObsPQStrength.csv")
#Quantiles_all=read.csv("NotStd_MultiQuantilesStrength.csv")

#all_Observed$line_value<-as.numeric(as.character(all_Observed$line_value))
```

Note:the draw_quantile argument in violin plots has discrepency so use stat_summary instead

Violin plots

```
#CBfriendly = c("#332288", "#009E73","#661100","#D55E00", "#CC79A7")

safe_colorblind_palette <- c("#88CCEE", "#CC6677", "#DDCC77", "#117733", "#33
```

```
2288", "#AA4499",
                                "#44AA99", "#999933", "#882255", "#661100", "#66
99CC", "#888888")

CBfriendly = c(safe_colorblind_palette[5], safe_colorblind_palette[3], safe_c
olorblind_palette[10],
               safe_colorblind_palette[2], safe_colorblind_palette[1])

q <- ggplot(data, aes(nest,rand_dist,group = factor(nest), fill=factor(nest))
)+
  geom_point(data=all_Observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  #scale_color_manual(values = c("#F8766D", "#A3A500", "#00BF7D","#00B0F6","#
E76BF3"),
  ##colorblindness friendly palette
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly) +
  scale_shape_identity()+ theme(legend.position="none")


r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,rand_dist,group = factor(nest), fill=factor(nest),colour=factor(ne
st))
  #, scale="width", width=1
  )+coord_flip()+
  ylab("Out-degree in Multilayer Network")+xlab("Colony ID")+
  geom_point(data=all_Observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")

r
```

Final violin plot for degree randomization comparison for all 5 colonies

```
################################
## EXPORT IMAGE AND CSVs
################################
setwd("D:/data/Degree")
### High resolution image
```

```
#tiff("MultiLines_outDegreeHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-degree in mutilayer
network", x="Colony ID")+
  #labs(caption = "A") +
  labs(tag = "A")+
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption A
```

```
#dev.off()

setwd("…/…/…/..")
#png("MultiLines_outDegree.png", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-degree in mutilayer
network", x="Colony ID")+
  #labs(caption = "A") +
  labs(tag ="A")
```

```
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption A

#dev.off()
```

Randomization code for Strength in Multilayer networks

R Markdown

Creating a vector with 5 colony names

```
nests=as.character(as.factor(c("v57", "v87", "v72", "v99", "v82")))
class(nests)

## [1] "character"

library(igraph)

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library(stringi)
```

Randomization loop after sourcing different colony extended edgelists

```
for (n in 1:length(nests)){
  #n=1
  nest_path=paste("D:/data/",nests[n],"/", sep="")

  setwd(nest_path)

  ns_wasp_dataOriginal <- read.csv(paste(nests[n],"_txtedges.csv", sep=""), h
eader=T)[,c(1:5)]

  library("dplyr")
ns_wasp_dataOrig<-as.data.frame(ns_wasp_dataOriginal %>% group_by(lyr1) %>% m
utate(NorWt = wt/max(wt)))

  ns_wasp_data <- ns_wasp_dataOrig[,c(1,2,3,4,6)]
  colnames(ns_wasp_data)[5]<-'wt'

  .libPaths( c( .libPaths(), "D:/Users/xxxxxx/Documents/R/win-library/3.6") )

  source("C:/Users/XXXXX/OneDrive/Documents/R/win-library/packages/muxViz-mas
ter//muxLib.R")

  ##Building the supra-adjacency matrix from extended edgelist
```

```
  sadjmat=BuildSupraAdjacencyMatrixFromExtendedEdgelist(
    mEdges=ns_wasp_data,
    Layers=length(unique(ns_wasp_data$lyr1)),
    Nodes=length(unique(ns_wasp_data$node1)), isDirected=T)

  ##calculating outstrength in multilayer

  multi_str<-as.data.frame(GetMultiOutStrengthSum(sadjmat,
                                                  Layers=length(unique(ns_was
p_data$lyr1)),
                                                  Nodes=length(unique(ns_wasp
_data$node1)),DIRECTED))




  labels<-read.csv(paste(nests[n],"node_labels.csv",sep=""), header = T)

  ##Sort labels and strength in descending order

  library('dplyr')

  act_str<-cbind(labels[,2],multi_str)
  colnames(act_str)<-c("ID","str")
  ##assign a value to a customized name by using "assign"
  assign(paste(nests[n],"_PQobsStr",sep=""),act_str$str[act_str$ID=="PQ"])
  #v87_PQobsStr

 ##### RANDOMIZATION TEST  ######

  df=ns_wasp_data
  df$node1<-as.factor(as.character(df$node1))
  df$node2<-as.factor(as.character(df$node2))

  head(df)

  PQ_Rstrs<-list()

  #i=2
  for(i in 1:1000){ ## for each iteration

    rand_DF<-list() ## this gets empty because we are shuffling within each l
ayer and then making a combined extended edgelist for all 4 layers in a go

    for(j in 1: length(levels(as.factor(df$lyr1)))) { ##only shuffle within l
ayers
      df1<-subset(df, lyr1 == j) # subset for that layer
```

```
      #length(unique(levels(as.factor(df1[,3]))))
      #creating an igraph object
      el1=as.matrix(df1[,c(1,3,5)]) #igraph needs the edgelist to be in matri
x format
      g1=graph.edgelist(el1[,c(1,2)], directed=TRUE) #We first create a netwo
rk from the first two columns, which has the list of vertices
      E(g1)$weight=as.numeric(el1[,3])

      g2=g1
      V(g2)$name=sample(V(g1)$name) ## sampled/shuffled the vertices

      df_rand=get.data.frame(g2)

      nrow(df_rand)
      nrow(df1)
      df1_rand<-df_rand
      df1_rand$node1<-df_rand$from
      df1_rand$lyr1<-df1$lyr1 ## making it an extended edgelist by adding lay
ers too
      df1_rand$node2<-df_rand$to
      df1_rand$lyr2<-df1$lyr2
      df1_rand$wt<-df_rand$weight ##df1_rand is the shuffled vertices' extend
ed edgelist

      df2_rand=df1_rand[,c("node1","lyr1","node2","lyr2","weight")] ##pruning
      colnames(df2_rand)<-c("node1","lyr1","node2","lyr2","wt")
      tail(df2_rand)
      rand_DF<-c(rand_DF, list(df2_rand)) ##list of different shuffled within
layer extended edgelists

    }

    #convert shuffled list to shuffled dataframe

    shuff_DF<-do.call(rbind.data.frame, rand_DF) ## making it a single shuffl
ed dataframe for all 4 layers combined
    tail(shuff_DF)

    #subset(shuff_DF, shuff.nodes1==shuff.nodes2)
    nrow(shuff_DF)
    nrow(df)
    #View(as.data.frame(shuff_DF))
    ## create an igraph object to work on
    library(igraph)
    el=as.matrix(shuff_DF[,1:5]) #igraph needs the edgelist to be in matrix f
ormat
    g=graph.edgelist(el[,c(1,3)], directed=TRUE) #We first create a network f
rom the first two columns, which has the list of vertices
    E(g)$weight=as.numeric(el[,5]) #We then add the edge weights to this netw
```

```
ork by assigning an edge attribute called 'weight'.



    nrow(shuff_DF)
    ## Re-create supra-adjacency matrix for shuffled nodes with same node 2 a
nd weights (because still technically shuffled)
    class(shuff_DF$node2)
    shuff_DF[,1]<-as.numeric(as.character(shuff_DF[,1]))
    shuff_DF[,2]<-as.numeric(as.character(shuff_DF[,2]))
    shuff_DF[,3]<-as.numeric(as.character(shuff_DF[,3]))
    shuff_DF[,4]<-as.numeric(as.character(shuff_DF[,4]))
    shuff_DF[,5]<-as.numeric(as.character(shuff_DF[,5]))

    source("C:/Users/XXXXXX/OneDrive/Documents/R/win-library/packages/muxViz-
master//muxLib.R") #replace with your file path obviously...

    sadjmat1=BuildSupraAdjacencyMatrixFromExtendedEdgelist(
      mEdges=shuff_DF,
      Layers=length(unique(shuff_DF$lyr1)),
      Nodes=length(unique(shuff_DF$node1)), isDirected=T)

    ## calculate strength in multilayer network iteration

    multi_str1<-as.data.frame(GetMultiOutStrengthSum(sadjmat1,
                                                     Layers=length(unique(shu
ff_DF$lyr1)),
                                                     Nodes=length(unique(shuf
f_DF$node1)), DIRECTED))

    labels<-read.csv(paste(nests[n],"node_labels.csv",sep=""), header = T)
    nrow(labels)
    ##Sort labels and strength in descending order

    library('dplyr')

    iter_str<-cbind(labels[,2],multi_str1) ## replacing numbers as nodes to n
ames as nodes
    ## assign multilayer versatility of shuffled dataframe to the ID
    colnames(iter_str)<-c("ID","str")



    ## strength of the PQ in randomized iteration
    PQ_Rstr<-iter_str$str[iter_str$ID=="PQ"]

    PQ_Rstrs<-c(PQ_Rstrs, list(PQ_Rstr))

  }
```

```
  Strengths_df<-do.call(rbind.data.frame, PQ_Rstrs)



  assign(paste(nests[n],"_rand.str",sep=""),as.vector(do.call(rbind.data.fram
e, PQ_Rstrs))[,1])
         #,inherits = TRUE)

  assign(paste(nests[n],"_DF",sep=""),as.data.frame(cbind(nest=rep(paste(nest
s[n]),1000),
                                                          assign(paste(nests[
n],"_randomRanks", sep=""),eval(parse(text=paste(nests[n],"_rand.str", sep=""
)))))))
  hist(eval(parse(text = paste(nests[n],"_rand.str",sep=""))), main = paste("
Nest ",nests[n],"(Out-strength)", sep=""), las=1,
       xlab="Out-strength") ## to evaluate the context of the text output of
paste, use eval(parse(text=...))
  assign(paste(nests[n],"_qts_str", sep=""),
         quantile(eval(parse(text=paste(nests[n],"_rand.str", sep=""))),probs
=c(.025,.975)))
  abline(v=eval(parse(text=paste(nests[n],"_PQobsStr",sep=""))), col="red", l
wd=2)
  abline(v=eval(parse(text=paste(nests[n],"_qts_str", sep=""))), col="blue",
lwd=2, lty=4)



  assign(paste(nests[n],"_lines", sep=""),
         as.data.frame(c(as.numeric(as.character(eval(parse(text=paste(nests[
n],"_qts_str", sep="")))),
                         as.numeric(as.character(eval(parse(text=paste(nests[
n],"_PQobsStr",sep=""))))))))
  cat_lines=c("Quantile","Quantile",paste(nests[n],"_PQobs", sep=""))
  assign(paste(nests[n],"_lines", sep=""),
         as.data.frame(cbind(values=eval(parse(text=paste(nests[n],"_lines",
sep=""))))[,1],cat_lines)))

  library(ggplot2)
  #n=2
  p<-ggplot(eval(parse(text = paste(nests[n],"_DF",sep=""))),
            aes(x=eval(parse(text = paste(nest[n],"_rand.str",sep=""))), fill
="orange",alpha=0.25)) +
    geom_density()+
    geom_vline(data=as.data.frame(eval(parse(text = paste(nests[n],"_lines",s
ep=""))))[1:2,]),
               aes(xintercept=as.numeric(as.character(eval(parse(text = paste
(nests[n],"_lines",sep=""))))[1:2,1])),
                   colour="blue",linetype="dashed",size=0.025))+
```

```
    geom_vline(data=as.data.frame(eval(parse(text = paste(nests[n],"_lines",
sep=""))))[3,]),
               aes(xintercept=as.numeric(as.character(eval(parse(text = paste
(nests[n],"_lines",sep=""))))[3,1])),colour="red",
               linetype="solid",size=0.025))

  #remove grey background
  p+theme_bw()+ #remove grey background
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blan
k())+#remove plot grids
    theme(text = element_text(size=20)) + labs(x = "Out-Degree in Multilayer"
, y="Frequency")



}
```

Quantiles, Randomized and observed strengths of each of the 5 colonies

```
all_random_distributions <- list(v99=v99_rand.str, v87=v87_rand.str, v57 =v57
_rand.str, v82=v82_rand.str, v72=v72_rand.str)

all_Quant_ablines<-list(v99=v99_lines,v87=v87_lines,v57=v57_lines,v82=v82_lin
es,v72=v72_lines)



library(ggplot2);library(reshape2)
data<- melt(all_random_distributions)
head(data)

data=data[c('L1','value')]
colnames(data)=c("nest","rand_dist")
Quant_ablines<-melt(all_Quant_ablines)

tail(Quant_ablines)

Quant_ablines<-Quant_ablines[c('L1','cat_lines','values')]
colnames(Quant_ablines)<-c("nest","cat_lines","line_value")
Quantiles_all=subset(Quant_ablines,cat_lines=="Quantile")
Quantiles_all$line_value<-as.numeric(as.character(Quantiles_all$line_value))
##change to numeric to avoid ggplot error of inputting factor instead of nume
ric (discrete in continuous error)
all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")
all_Observed$line_value<-as.numeric(as.character(all_Observed$line_value))##c
hange to numeric to avoid ggplot error of inputting factor instead of numeric
(discrete in continuous error)
```

ggplot histograms of all 5 colonies

```
library(ggplot2)
p<-ggplot(data, aes(x=rand_dist, fill=nest,alpha=0.1)) +
```

```
  geom_density()+
  geom_vline(data=Quantiles_all, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=all_Observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-Degree in Multilayer",
y="Frequency")
```

Getting violin plots without having to run long randomizations each time by sourcing csv

```
#data=read.csv("NotStd_MultirandomStrength.csv")
#all_Observed=read.csv("NotStd_MultiObsPQStrength.csv")
#Quantiles_all=read.csv("NotStd_MultiQuantilesStrength.csv")

#all_Observed$line_value<-as.numeric(as.character(all_Observed$line_value))
```

Note:the draw_quantile argument in violin plots has discrepency so use stat_summary instead

Violin plots

```
#CBfriendly = c(#"#E69F00"
#  "#332288",
#  "#009E73","#661100",#"#0072B2",
#  "#D55E00", "#CC79A7")

safe_colorblind_palette <- c("#88CCEE", "#CC6677", "#DDCC77", "#117733", "#33
2288", "#AA4499",
                             "#44AA99", "#999933", "#882255", "#661100", "#66
99CC", "#888888")

CBfriendly = c(safe_colorblind_palette[5], safe_colorblind_palette[3], safe_c
olorblind_palette[10],
               safe_colorblind_palette[2], safe_colorblind_palette[1])

q <- ggplot(data, aes(nest,rand_dist,group = factor(nest), fill=
                        #c("#E69F00", "#56B4E9", "#009E73","#F0E442","#0072B2
")
                        factor(nest)
                      ))+
  geom_point(data=all_Observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=
```

```
                    #c("#E69F00", "#56B4E9", "#009E73","#F0E442","#0072B2")
                    factor(nest)
                    ),colour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  #scale_color_manual(values = c("#F8766D", "#A3A500", #"#00BF7D","#00B0F6","
#E76BF3"),
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly) +
  scale_shape_identity()+ theme(legend.position="none")


r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,rand_dist,group = factor(nest), fill=factor(nest),colour=factor(ne
st))
  #, scale="width", width=1
  )+coord_flip()+
  ylab("Out-strength in Multilayer Network")+xlab("Colony ID")+
  geom_point(data=all_Observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")

r
```

Final violin plot for strength randomization comparison for all 5 colonies

```
################################
## EXPORT IMAGE AND CSVs
################################
### High resolution image
tiff("MultiLines_outStrengthHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-strength in mutilaye
r network", x="Colony ID")+
  labs(tag = "A") +
```

```
    theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption A


dev.off()

## png
##   2

### High resolution image
png("MultiLines_outStrength.png", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-strength in mutilaye
r network", x="Colony ID")+
  labs(tag = "A") +
    theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption A


dev.off()
```

Randomization code for Degree in Aggregate network

Creating a vector with 5 colony names

```
nests=as.character(as.factor(c("v57", "v87", "v72", "v99", "v82")))
class(nests)

## [1] "character"

library(igraph)

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library(stringi)
```

#Randomization loop after sourcing different colony extended edgelists

```
for (n in 1:length(nests)){
  nest_path=paste("D:/data/",nests[n],"/", sep="")
  setwd(nest_path)
  ns_wasp_dataOriginal = read.csv(paste(nests[n],"_txtedges.csv", sep=""), he
ader=T)[,c(1:5)]

  library("dplyr")
ns_wasp_dataOrig<-as.data.frame(ns_wasp_dataOriginal %>% group_by(lyr1) %>% m
utate(NorWt = wt/max(wt)))

  ns_wasp_data <- ns_wasp_dataOrig[,c(1,2,3,4,6)]
  colnames(ns_wasp_data)[5]<-'wt'

  .libPaths( c( .libPaths(), "D:/Users/XXXXXXX/Documents/R/win-library/3.6")
)

  source("C:/Users/XXXXX/OneDrive/Documents/R/win-library/packages/muxViz-mas
ter//muxLib.R")

  ##Building the supra-adjacency matrix from extended edgelist

  sadjmat=BuildSupraAdjacencyMatrixFromExtendedEdgelist(
    mEdges=ns_wasp_data,
    Layers=length(unique(ns_wasp_data$lyr1)),
    Nodes=length(unique(ns_wasp_data$node1)), isDirected=T)
```

```
  ##calculating out-degree in aggregate

  ##Aggregate degree
NodesTensor <- SupraAdjacencyToNodesTensor(binarizeMatrix(sadjmat),Layers=len
gth(unique(ns_wasp_data$lyr1)),
                                           Nodes=length(unique(ns_wasp_data$n
ode1)))
AggrMatrix <- GetAggregateMatrix(NodesTensor, Layers=length(unique(ns_wasp_da
ta$lyr1)),
                                 Nodes=length(unique(ns_wasp_data$node1)))

agg_degree<-colSums(as.matrix(AggrMatrix) != 0) ## all elements that are non-
zero, as.matrix replaces '.'  with '0'

 labels<-read.csv(paste(nests[n],"node_labels.csv",sep=""), header = T)

obs_agg_df=cbind(labels,agg_degree)
obs_agg_deg=obs_agg_df[,3]
#as.matrix(AggrMatrix)[,10]



##Sort labels and degrees in descending order

library('dplyr')

act_deg<-as.data.frame(cbind(as.character(labels[,2]),obs_agg_deg))
colnames(act_deg)<-c("ID","Agg_deg")


  ##assign a value to a customized name by using "assign"
  assign(paste(nests[n],"_PQobsDeg",sep=""),act_deg$Agg_deg[act_deg$ID=="PQ"]
)




 ##### RANDOMIZATION TEST  ######

  df=ns_wasp_data
  df$node1<-as.factor(as.character(df$node1))
  df$node2<-as.factor(as.character(df$node2))

  head(df)

  PQ_Rdegs<-list()
```

```r
  for(i in 1:1000){ ## for each iteration

    rand_DF<-list() ## this gets empty because we are shuffling within each l
ayer and then making a combined extended edgelist for all 4 layers in a go

    for(j in 1: length(levels(as.factor(df$lyr1)))) { ##only shuffle within l
ayers
      df1<-subset(df, lyr1 == j) # subset for that layer
      #length(unique(levels(as.factor(df1[,3]))))
      #creating an igraph object
      el1=as.matrix(df1[,c(1,3,5)]) #igraph needs the edgelist to be in matri
x format
      g1=graph.edgelist(el1[,c(1,2)], directed=TRUE) #We first create a netwo
rk from the first two columns, which has the list of vertices
      E(g1)$weight=as.numeric(el1[,3])

      g2=g1
      V(g2)$name=sample(V(g1)$name) ## sampled/shuffled the vertices

      df_rand=get.data.frame(g2)

      nrow(df_rand)
      nrow(df1)
      df1_rand<-df_rand
      df1_rand$node1<-df_rand$from
      df1_rand$lyr1<-df1$lyr1 ## making it an extended edgelist by adding lay
ers too
      df1_rand$node2<-df_rand$to
      df1_rand$lyr2<-df1$lyr2
      df1_rand$wt<-df_rand$weight ##df1_rand is the shuffled vertices' extend
ed edgelist

      df2_rand=df1_rand[,c("node1","lyr1","node2","lyr2","weight")] ##pruning
      colnames(df2_rand)<-c("node1","lyr1","node2","lyr2","wt")
      tail(df2_rand)
      rand_DF<-c(rand_DF, list(df2_rand)) ##list of different shuffled within
layer extended edgelists

    }

    #convert shuffled list to shuffled dataframe

    shuff_DF<-do.call(rbind.data.frame, rand_DF) ## making it a single shuffl
ed dataframe for all 4 layers combined
    tail(shuff_DF)

    #subset(shuff_DF, shuff.nodes1==shuff.nodes2)
    nrow(shuff_DF)
```

```
    nrow(df)
    #View(as.data.frame(shuff_DF))
    ## create an igraph object to work on
    library(igraph)
    el=as.matrix(shuff_DF[,1:5]) #igraph needs the edgelist to be in matrix f
ormat
    g=graph.edgelist(el[,c(1,3)], directed=TRUE) #We first create a network f
rom the first two columns, which has the list of vertices
    E(g)$weight=as.numeric(el[,5]) #We then add the edge weights to this netw
ork by assigning an edge attribute called 'weight'.



    nrow(shuff_DF)
    ## Re-create supra-adjacency matrix for shuffled nodes with same node 2 a
nd weights (because still technically shuffled)
    class(shuff_DF$node2)
    shuff_DF[,1]<-as.numeric(as.character(shuff_DF[,1]))
    shuff_DF[,2]<-as.numeric(as.character(shuff_DF[,2]))
    shuff_DF[,3]<-as.numeric(as.character(shuff_DF[,3]))
    shuff_DF[,4]<-as.numeric(as.character(shuff_DF[,4]))
    shuff_DF[,5]<-as.numeric(as.character(shuff_DF[,5]))

    source("C:/Users/XXXXXX/OneDrive/Documents/R/win-library/packages/muxViz-
master//muxLib.R") #replace with your file path obviously...



    sadjmat1=BuildSupraAdjacencyMatrixFromExtendedEdgelist(
    mEdges=shuff_DF,
    Layers=length(unique(shuff_DF$lyr1)),
    Nodes=length(unique(shuff_DF$node1)), isDirected=T)

    ## calculate degree in aggregate network iteration

     ## calculate degree in aggregate network iteration

  ##Aggregate degree
  NodesTensor1 <- SupraAdjacencyToNodesTensor(binarizeMatrix(sadjmat1),Layers
=length(unique(ns_wasp_data$lyr1)),
                                              Nodes=length(unique(ns_wasp_dat
a$node1)))
  AggrMatrix1 <- GetAggregateMatrix(NodesTensor1, Layers=length(unique(ns_was
p_data$lyr1)),
                                    Nodes=length(unique(ns_wasp_data$node1)))

  agg_degree1<-colSums(as.matrix(AggrMatrix1) != 0) ## all elements that are
non-zero, as.matrix replaces '.'  with '0'
  rand_agg_df1=cbind(labels,agg_degree1)
  rand_agg_deg1=rand_agg_df1[,3]
```

```
  #as.data.frame(table(shuff_DF$node1)) ## occurence of each node



 labels<-read.csv(paste(nests[n],"node_labels.csv",sep=""), header = T)


    ##Sort labels and degree in descending order

    library('dplyr')

    iter_deg<-as.data.frame(cbind(as.character(labels[,2]),rand_agg_deg1)) ##
replacing numbers as nodes to names as nodes
    ## assign aggregate out-degree of shuffled dataframe to the ID
    colnames(iter_deg)<-c("ID","rand_agg_deg")


    ## degree of the PQ in randomized iteration
    PQ_Rdeg<-as.numeric(as.character(iter_deg$rand_agg_deg[iter_deg$ID=="PQ"]
))

    PQ_Rdegs<-c(PQ_Rdegs, list(PQ_Rdeg))

  }

  Degrees_df<-do.call(rbind.data.frame, PQ_Rdegs)



  assign(paste(nests[n],"_rand.deg",sep=""),as.vector(do.call(rbind.data.fram
e, PQ_Rdegs))[,1])
         #,inherits = TRUE)

  assign(paste(nests[n],"_DF",sep=""),as.data.frame(cbind(nest=rep(paste(nest
s[n]),1000),
                                                          assign(paste(nests[
n],"_randomRanks", sep=""),eval(parse(text=paste(nests[n],"_rand.deg", sep=""
)))))))


  hist(eval(parse(text = paste(nests[n],"_rand.deg",sep=""))), main = paste("
Nest ",nests[n],"(Out-degree)", sep=""), las=1,
       xlab="Out-degree") ## to evaluate the context of the text output of pa
ste, use eval(parse(text=...))
  assign(paste(nests[n],"_qts_deg", sep=""),
         quantile(eval(parse(text=paste(nests[n],"_rand.deg", sep=""))),probs
=c(.025,.975)))
```

```
  abline(v=eval(parse(text=paste(nests[n],"_PQobsDeg",sep=""))), col="red", l
wd=2)
  abline(v=eval(parse(text=paste(nests[n],"_qts_deg", sep=""))), col="blue",
lwd=2, lty=4)



  assign(paste(nests[n],"_lines", sep=""),
         as.data.frame(c(as.numeric(as.character(eval(parse(text=paste(nests[
n],"_qts_deg", sep="")))),
                         as.numeric(as.character(eval(parse(text=paste(nests[
n],"_PQobsDeg",sep="")))))))))
  cat_lines=c("Quantile","Quantile",paste(nests[n],"_PQobs", sep=""))
  assign(paste(nests[n],"_lines", sep=""),
         as.data.frame(cbind(values=eval(parse(text=paste(nests[n],"_lines",
sep="")))[,1],cat_lines)))

  library(ggplot2)
  #n=2
  p<-ggplot(eval(parse(text = paste(nests[n],"_DF",sep=""))),
            aes(x=eval(parse(text = paste(nest[n],"_rand.deg",sep="")))), fill
="orange",alpha=0.25)) +
    geom_density()+
    geom_vline(data=as.data.frame(eval(parse(text = paste(nests[n],"_lines",s
ep="")))[1:2,]),
               aes(xintercept=as.numeric(as.character(eval(parse(text = paste
(nests[n],"_lines",sep="")))[1:2,1])),
                   colour="blue",linetype="dashed",size=0.025))+
    geom_vline(data=as.data.frame(eval(parse(text = paste(nests[n],"_lines",
sep="")))[3,]),
               aes(xintercept=as.numeric(as.character(eval(parse(text = paste
(nests[n],"_lines",sep="")))[3,1])),colour="red",
               linetype="solid",size=0.025))

  #remove grey background
  p+theme_bw()+ #remove grey background
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blan
k())+#remove plot grids
    theme(text = element_text(size=20)) + labs(x = "Out-Degree in Aggregate n
etwork", y="Frequency")



}
```

Quantiles, Randomized and observed degrees of each of the 5 colonies

```
all_random_distributions <- list(v99=v99_rand.deg, v87=v87_rand.deg, v57 =v57
_rand.deg, v82=v82_rand.deg, v72=v72_rand.deg)
```

```
all_Quant_ablines<-list(v99=v99_lines,v87=v87_lines,v57=v57_lines,v82=v82_lin
es,v72=v72_lines)


library(ggplot2);library(reshape2)
data<- melt(all_random_distributions)
head(data)

data=data[c('L1','value')]
colnames(data)=c("nest","rand_dist")
Quant_ablines<-melt(all_Quant_ablines)

## Using values, cat_lines as id variables
## Using values, cat_lines as id variables
## Using values, cat_lines as id variables
## Using values, cat_lines as id variables
## Using values, cat_lines as id variables

tail(Quant_ablines)

Quant_ablines<-Quant_ablines[c('L1','cat_lines','values')]
colnames(Quant_ablines)<-c("nest","cat_lines","line_value")
Quantiles_all=subset(Quant_ablines,cat_lines=="Quantile")
Quantiles_all$line_value<-as.numeric(as.character(Quantiles_all$line_value))
##change to numeric to avoid ggplot error of inputting factor instead of nume
ric (discrete in continuous error)
all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")
all_Observed$line_value<-as.numeric(as.character(all_Observed$line_value))##c
hange to numeric to avoid ggplot error of inputting factor instead of numeric
(discrete in continuous error)
```

ggplot histograms of all 5 colonies

```
library(ggplot2)
p<-ggplot(data, aes(x=rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=Quantiles_all, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=all_Observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-Degree in Multilayer",
y="Frequency")
```

Getting violin plots without having to run long randomizations each time by sourcing csv

```
#knitr::opts_knit$set(root.dir = 'D:/data/')
#setwd("D:/data/")

#data=read.csv("NotStd_MultirandomStrength.csv")
#all_Observed=read.csv("NotStd_MultiObsPQStrength.csv")
#Quantiles_all=read.csv("NotStd_MultiQuantilesStrength.csv")

#all_Observed$line_value<-as.numeric(as.character(all_Observed$line_value))
```

Violin plots

```
#CBfriendly = c("#332288", "#009E73","#661100","#D55E00", "#CC79A7")

safe_colorblind_palette <- c("#88CCEE", "#CC6677", "#DDCC77", "#117733", "#33
2288", "#AA4499",
                             "#44AA99", "#999933", "#882255", "#661100", "#66
99CC", "#888888")

CBfriendly = c(safe_colorblind_palette[5], safe_colorblind_palette[3], safe_c
olorblind_palette[10],
               safe_colorblind_palette[2], safe_colorblind_palette[1])


q <- ggplot(data, aes(nest,rand_dist,group = factor(nest), fill=factor(nest))
)+
  geom_point(data=all_Observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly) +
  scale_shape_identity()+ theme(legend.position="none")


r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,rand_dist,group = factor(nest), fill=factor(nest),colour=factor(ne
st))
  #, scale="width", width=1
  )+coord_flip()+
  ylab("Out-degree in Multilayer Network")+xlab("Colony ID")+
  geom_point(data=all_Observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")
```

```
r
```

Final violin plot for degree randomization comparison for all 5 colonies

```
################################
## EXPORT IMAGE AND CSVs
################################

### High resolution image
tiff("Agglines2_outDegHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-degree in Aggregate
network", x="Colony ID")+
  labs(tag = 'B') +
  theme(plot.caption = element_text(size = 22)) +
  theme(plot.margin = margin(t = 1, r = 1, b = 1, l = 1)) +
  theme(plot.tag.position = "topright")


dev.off()

### High resolution image
png("Agglines2_outDeg.png", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-degree in Aggregate
network", x="Colony ID")+
  labs(tag = 'B)') +
  theme(plot.caption = element_text(size = 22)) +
  theme(plot.margin = margin(t = 1, r = 1, b = 1, l = 1)) +
  theme(plot.tag.position = "topright")


dev.off()
```

Randomization code for strength in aggregate network

Creating a vector with 5 colony names

```
nests=as.character(as.factor(c("v57", "v87", "v72", "v99", "v82")))
class(nests)

## [1] "character"

library(igraph)

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library(stringi)
```

#Randomization loop after sourcing different colony extended edgelists

```
for (n in 1:length(nests)){
  nest_path=paste("D:/data/",nests[n],"/", sep="")
  setwd(nest_path) #replace with your file path obviously...
  ns_wasp_dataOriginal = read.csv(paste(nests[n],"_txtedges.csv", sep=""), he
ader=T)[,c(1:5)]

  library("dplyr")
ns_wasp_dataOrig<-as.data.frame(ns_wasp_dataOriginal %>% group_by(lyr1) %>% m
utate(NorWt = wt/max(wt)))

  ns_wasp_data <- ns_wasp_dataOrig[,c(1,2,3,4,6)]
  colnames(ns_wasp_data)[5]<-'wt'

  labels<-read.csv(paste(nests[n],"node_labels.csv",sep=""), header = T)

  ##For AGGREGATE STRENGTH:
  df=ns_wasp_data
  df$node1<-as.character(df$node1)
  df$node2<-as.character(df$node2)

  el=as.matrix(df[,c(1,3,5)]) #igraph needs the edgelist to be in matrix form
at
  g=graph.edgelist(el[,c(1,2)], directed=TRUE) #We first create a network fro
m the first two columns, which has the list of vertices
  E(g)$weight=as.numeric(el[,3])
```

```
  all_strengths=as.data.frame(cbind(V(g)$name,as.numeric(as.character(strengt
h(g, mode="out")))))
  colnames(all_strengths)<-c("nodeID","strength")
  id_strengths=merge(labels, all_strengths, by="nodeID")
  PQ_obsStr=id_strengths[id_strengths$nodeLabel=="PQ",]$strength


  ##assign a value to a customized name by using "assign"
  assign(paste(nests[n],"_PQobsStr",sep=""),as.numeric(as.character(id_streng
ths[id_strengths$nodeLabel=="PQ",]$strength)))
  #v87_PQobsStr

  ##### RANDOMIZATION TEST  ######

  df=ns_wasp_data
  df$node1<-as.factor(as.character(df$node1))
  df$node2<-as.factor(as.character(df$node2))

  head(df)

  PQ_Rstrs<-list()

  #i=2
  for(i in 1:1000){ ## for each iteration

    rand_DF<-list() ## this gets empty before we shuffle
    df1=ns_wasp_data
    df1$node1<-as.factor(as.character(df1$node1))
    df1$node2<-as.factor(as.character(df1$node2))
    ##No need to shuffle only within layers because aggregate doesn't differe
ntiate interactions arising from diff layers


    el1=as.matrix(df1[,c(1,3,5)]) #igraph needs the edgelist to be in matrix
format
      g1=graph.edgelist(el1[,c(1,2)], directed=TRUE) #We first create a netwo
rk from the first two columns, which has the list of vertices
      E(g1)$weight=as.numeric(el1[,3])

      g2=g1
      V(g2)$name=sample(V(g1)$name) ## sampled/shuffled the vertices

      rand_degs<-as.data.frame(cbind(V(g2)$name,as.numeric(as.character(stren
gth(g2, mode="out")))))
      colnames(rand_degs)<-c("nodeID","strength")
      rand_id_str<-merge(labels, rand_degs, by="nodeID")
      rand_id_str$nest<-rep(nests[n],nrow(rand_id_str))
      colnames(rand_id_str)=c("nodeID","nodeLabel","out_strength","nest")
```

```
    PQ_Rstr<-as.numeric(as.character(rand_id_str$out_strength[rand_id_str$nod
eLabel=="PQ"]))

    PQ_Rstrs<-c(PQ_Rstrs, list(PQ_Rstr))

  }

  Strengths_df<-do.call(rbind.data.frame, PQ_Rstrs)



  assign(paste(nests[n],"_rand.str",sep=""),as.vector(do.call(rbind.data.fram
e, PQ_Rstrs))[,1])
  #,inherits = TRUE)

  assign(paste(nests[n],"_DF",sep=""),as.data.frame(cbind(nest=rep(paste(nest
s[n]),1000),
                                                          assign(paste(nests[
n],"_randomRanks", sep=""),eval(parse(text=paste(nests[n],"_rand.str", sep=""
))))))))
  hist(eval(parse(text = paste(nests[n],"_rand.str",sep=""))), main = paste("
Nest ",nests[n],"(Out-strength)", sep=""), las=1,
       xlab="Out-strength") ## to evaluate the context of the text output of
paste, use eval(parse(text=...))
  assign(paste(nests[n],"_qts_str", sep=""),
         quantile(eval(parse(text=paste(nests[n],"_rand.str", sep=""))),probs
=c(.025,.975)))
  abline(v=eval(parse(text=paste(nests[n],"_PQobsStr",sep=""))), col="red", l
wd=2)
  abline(v=eval(parse(text=paste(nests[n],"_qts_str", sep=""))), col="blue",
lwd=2, lty=4)



  assign(paste(nests[n],"_lines", sep=""),
         as.data.frame(c(as.numeric(as.character(eval(parse(text=paste(nests[
n],"_qts_str", sep="")))))),
                       as.numeric(as.character(eval(parse(text=paste(nests[
n],"_PQobsStr",sep=""))))))))
  cat_lines=c("Quantile","Quantile",paste(nests[n],"_PQobs", sep=""))
  assign(paste(nests[n],"_lines", sep=""),
         as.data.frame(cbind(values=eval(parse(text=paste(nests[n],"_lines",
sep=""))))[,1],cat_lines)))

  library(ggplot2)
  #n=2
  p<-ggplot(eval(parse(text = paste(nests[n],"_DF",sep=""))),
```

```
            aes(x=eval(parse(text = paste(nest[n],"_rand.str",sep=""))), fill
="orange",alpha=0.25)) +
    geom_density()+
    geom_vline(data=as.data.frame(eval(parse(text = paste(nests[n],"_lines",s
ep=""))))[1:2,]),
               aes(xintercept=as.numeric(as.character(eval(parse(text = paste
(nests[n],"_lines",sep=""))))[1:2,1])),
                   colour="blue",linetype="dashed",size=0.025))+
    geom_vline(data=as.data.frame(eval(parse(text = paste(nests[n],"_lines",
sep=""))))[3,]),
               aes(xintercept=as.numeric(as.character(eval(parse(text = paste
(nests[n],"_lines",sep=""))))[3,1])),colour="red",
                   linetype="solid",size=0.025))

  #remove grey background
  p+theme_bw()+ #remove grey background
    theme(panel.grid.major = element_blank(), panel.grid.minor = element_blan
k())+#remove plot grids
    theme(text = element_text(size=20)) + labs(x = "Out-Degree in Multilayer"
, y="Frequency")



}
```

To save randomization CSVs

```
#setwd("D:/data/Strength")

#data=read.csv("22Apr21_Std_AggrandomStrength.csv")
#all_Observed=read.csv("22Apr21_Std_AggObsPQStrength.csv")
#Quantiles_all=read.csv("22Apr21_Std_AggQuantilesStrength.csv")
```

Quantiles, Randomized and observed degrees of each of the 5 colonies

```
all_random_distributions <- list(v99=v99_rand.str, v87=v87_rand.str, v57 =v57
_rand.str, v82=v82_rand.str, v72=v72_rand.str)

all_Quant_ablines<-list(v99=v99_lines,v87=v87_lines,v57=v57_lines,v82=v82_lin
es,v72=v72_lines)


library(ggplot2);library(reshape2)
data<- melt(all_random_distributions)
head(data)
```

```
data=data[c('L1','value')]
colnames(data)=c("nest","rand_dist")
Quant_ablines<-melt(all_Quant_ablines)

tail(Quant_ablines)

Quant_ablines<-Quant_ablines[c('L1','cat_lines','values')]
colnames(Quant_ablines)<-c("nest","cat_lines","line_value")
Quantiles_all=subset(Quant_ablines,cat_lines=="Quantile")
Quantiles_all$line_value<-as.numeric(as.character(Quantiles_all$line_value))
##change to numeric to avoid ggplot error of inputting factor instead of nume
ric (discrete in continuous error)
all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")
all_Observed$line_value<-as.numeric(as.character(all_Observed$line_value))##c
hange to numeric to avoid ggplot error of inputting factor instead of numeric
(discrete in continuous error)
```

ggplot histograms of all 5 colonies

```
library(ggplot2)
p<-ggplot(data, aes(x=rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=Quantiles_all, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=all_Observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-strength in Aggregate N
etwork", y="Frequency")
```

Getting violin plots without having to run long randomizations each time by sourcing csv

```
#data=read.csv("NotStd_MultirandomStrength.csv")
#all_Observed=read.csv("NotStd_MultiObsPQStrength.csv")
#Quantiles_all=read.csv("NotStd_MultiQuantilesStrength.csv")

#all_Observed$line_value<-as.numeric(as.character(all_Observed$line_value))
```

Note:the draw_quantile argument in violin plots has discrepency so use stat_summary instead

Violin plots

```
#CBfriendly = c("#332288", "#009E73","#661100","#D55E00", "#CC79A7")

safe_colorblind_palette <- c("#88CCEE", "#CC6677", "#DDCC77", "#117733", "#33
2288", "#AA4499",
```

```
                              "#44AA99", "#999933", "#882255", "#661100", "#66
99CC", "#888888")

CBfriendly = c(safe_colorblind_palette[5], safe_colorblind_palette[3], safe_c
olorblind_palette[10],
               safe_colorblind_palette[2], safe_colorblind_palette[1])

q <- ggplot(data, aes(nest,rand_dist,group = factor(nest), fill=factor(nest))
)+
  geom_point(data=all_Observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly) +
  scale_shape_identity()+ theme(legend.position="none")


r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,rand_dist,group = factor(nest), fill=factor(nest),colour=factor(ne
st))
  #, scale="width", width=1
  )+coord_flip()+
  ylab("Out-degree in Multilayer Network")+xlab("Colony ID")+
  geom_point(data=all_Observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")

r
```

Final violin plot for degree randomization comparison for all 5 colonies

```
################################
## EXPORT IMAGE AND CSVs
################################
### High resolution image
tiff("Agglines_outStrengthHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
```

```
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-strength in Aggregat
e network", x="Colony ID")+
  labs(tag = "B") +
    theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption B


dev.off()

### High resolution image
png("Agglines_outStrength.png", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-strength in Aggregat
e network", x="Colony ID")+
  labs(tag = "B") +
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption B


dev.off()
```

Randomization code for Degree in single layer networks

Creating a vector with 5 colony names

```
nests=as.character(as.factor(c("v57", "v87", "v72", "v99", "v82")))
class(nests)

## [1] "character"

library(igraph)

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library(stringi)
```

#Randomization loop after sourcing different colony extended edgelists

```
for (n in 1:length(nests)){
  nest_path=paste("D:/data/",nests[n],"/", sep="")
  setwd(nest_path)
  #ns_wasp_data = read.csv(paste(nests[n],"_txtedges.csv", sep=""), header=T)
[,c(1:5)]

  ns_wasp_dataOriginal = read.csv(paste(nests[n],"_txtedges.csv", sep=""), he
ader=T)[,c(1:5)]

  library("dplyr")
ns_wasp_dataOrig<-as.data.frame(ns_wasp_dataOriginal %>% group_by(lyr1) %>% m
utate(NorWt = wt/max(wt)))

  ns_wasp_data <- ns_wasp_dataOrig[,c(1,2,3,4,6)]

  labels<-read.csv(paste(nests[n],"node_labels.csv",sep=""), header = T)

  ##For monolayer degree:
  library(igraph)
df=ns_wasp_data
df$node1<-as.factor(as.character(df$node1))
df$node2<-as.factor(as.character(df$node2))

head(df)
PQranks<-list()
```

```
PQ_Rdegs<-list()
rand_allLayers<-list()


  ## this gets empty because we are shuffling within each layer and then maki
ng a combined extended edgelist for all 4 layers in a go

  for(i in 1:1000){ ## for each iteration
    Random_perdeg<-list()

  orig_allLayers<-list() ##empty before running a cycle through all layers

  for(j in 1: length(levels(as.factor(df$lyr1)))) { ##only shuffle within lay
ers
    #j=4
    df1<-subset(df, lyr1 == j) # subset for that layer
    #length(unique(levels(as.factor(df1[,3]))))
    #creating an igraph object
    el1=as.matrix(df1[,c(1,3,5)]) #igraph needs the edgelist to be in matrix
format
    g1=graph.edgelist(el1[,c(1,2)], directed=TRUE) #We first create a network
from the first two columns, which has the list of vertices
    E(g1)$weight=as.numeric(el1[,3])

    #vertex_attr_names(g1)
    #g2=graph.empty(n=length(V(g1)$name), directed=TRUE)
    g2=g1
    V(g2)$name=sample(V(g1)$name) ## sampled/shuffled the vertices
    #V(g2)$name<-V(g1)$name
    #class(V(g1))
    #get.data.frame(g1)
    df_rand=get.data.frame(g2)
    #el_g1$weight=E(g1)$weight


    ## DEGREE OF ORIGINAL NETWORK

    orig_degs<-as.data.frame(cbind(V(g1)$name,as.numeric(as.character(degree(
g1,mode="out")))))
    colnames(orig_degs)<-c("nodeID","degree")
    org_lyr_deg<-merge(labels, orig_degs, by="nodeID")
    org_lyr_deg$layerNo<-rep(j,nrow(org_lyr_deg))
    org_lyr_deg$nest<-rep("v87",nrow(org_lyr_deg))
    colnames(org_lyr_deg)=c("nodeID","nodeLabel","out_degree","LayerNo","nest
")

    ## DEGREE OF RANDOMIZED NETWORK

    rand_degs<-as.data.frame(cbind(V(g2)$name,as.numeric(as.character(degree(
```

```
g2, mode="out")))))
    colnames(rand_degs)<-c("nodeID","degree")
    rand_lyr_deg<-merge(labels, rand_degs, by="nodeID")
    rand_lyr_deg$layerNo<-rep(j,nrow(rand_lyr_deg))
    rand_lyr_deg$nest<-rep("v87",nrow(rand_lyr_deg))
    colnames(rand_lyr_deg)=c("nodeID","nodeLabel","out_degree","LayerNo","nes
t")

    PQ_origlyr_deg<-org_lyr_deg[org_lyr_deg$nodeLabel=="PQ",]
    PQ_randlyr_deg<-rand_lyr_deg[rand_lyr_deg$nodeLabel=="PQ",]
    Random_perdeg<-c(Random_perdeg,list(PQ_randlyr_deg))
    orig_allLayers<-c(orig_allLayers, list(PQ_origlyr_deg))


  }


  #convert shuffled list to shuffled dataframe
  rand_allLayers<-c(rand_allLayers, list(do.call(rbind.data.frame, Random_per
deg)))
  nrow(rand_allLayers)

  }
#                   Single layer assignments to each nest




#allRand_df<-do.call(rbind.data.frame, rand_allLayers)

##assign a value to a customized name by using "assign"
  assign(paste(nests[n],"_allRand_df",sep=""),do.call(rbind.data.frame,rand_a
llLayers))

assign(paste(nests[n],"_Rand_lyr1",sep=""),subset(eval(parse(text=paste(nests
[n],"_allRand_df",sep=""))), LayerNo=="1"))
assign(paste(nests[n],"_Rand_lyr2",sep=""),subset(eval(parse(text=paste(nests
[n],"_allRand_df",sep=""))), LayerNo=="2"))
assign(paste(nests[n],"_Rand_lyr3",sep=""),subset(eval(parse(text=paste(nests
[n],"_allRand_df",sep=""))), LayerNo=="3"))
assign(paste(nests[n],"_Rand_lyr4",sep=""),subset(eval(parse(text=paste(nests
[n],"_allRand_df",sep=""))), LayerNo=="4"))


assign(paste(nests[n],"_allOrig_df",sep=""),do.call(rbind.data.frame,orig_all
Layers))

assign(paste(nests[n],"_Orig_lyr1",sep=""),subset(eval(parse(text=paste(nests
```

```
[n],"_allOrig_df",sep=""))), LayerNo=="1"))
assign(paste(nests[n],"_Orig_lyr2",sep=""),subset(eval(parse(text=paste(nests
[n],"_allOrig_df",sep=""))), LayerNo=="2"))
assign(paste(nests[n],"_Orig_lyr3",sep=""),subset(eval(parse(text=paste(nests
[n],"_allOrig_df",sep=""))), LayerNo=="3"))
assign(paste(nests[n],"_Orig_lyr4",sep=""),subset(eval(parse(text=paste(nests
[n],"_allOrig_df",sep=""))), LayerNo=="4"))

##Quantiles

assign(paste(nests[n],"_qts_deg1",sep=""),quantile((as.numeric(as.character(e
val(parse(text= paste(nests[n],"_Rand_lyr1",sep="")))[,3]))),probs = c(0.025,
0.975)))


assign(paste(nests[n],"_qts_deg2",sep=""),quantile((as.numeric(as.character(e
val(parse(text= paste(nests[n],"_Rand_lyr2",sep="")))[,3]))),probs = c(0.025,
0.975)))


assign(paste(nests[n],"_qts_deg3",sep=""),quantile((as.numeric(as.character(e
val(parse(text= paste(nests[n],"_Rand_lyr3",sep="")))[,3]))),probs = c(0.025,
0.975)))


assign(paste(nests[n],"_qts_deg4",sep=""),quantile((as.numeric(as.character(e
val(parse(text= paste(nests[n],"_Rand_lyr4",sep="")))[,3]))),probs = c(0.025,
0.975)))

}

##
```

Histogram of spatial network

```
#all_random_distributions <- list(v99=v99_rand.dist, v87=v87_rand.dist, v57 =
v57_rand.dist, v82=v82_rand.dist, v72=v72_rand.dist)



SP_random_distributions <- list(v99=as.numeric(as.character(v99_Rand_lyr1$out
_degree)),
                                v87=as.numeric(as.character(v87_Rand_lyr1$out
_degree)),
                                v57 =as.numeric(as.character(v57_Rand_lyr1$ou
t_degree)),
                                v82=as.numeric(as.character(v82_Rand_lyr1$out
_degree)),
```

```
                                  v72=as.numeric(as.character(v72_Rand_lyr1$out
_degree)))

SP_Quant_ablines<-list(v99=v99_qts_deg1,v87=v87_qts_deg1,v57=v57_qts_deg1,
                       v82=v82_qts_deg1,v72=v72_qts_deg1)

library(ggplot2);library(reshape2)
SP_observed<-melt(list(v99=as.numeric(as.character(v99_Orig_lyr1$out_degree))
, v87=as.numeric(as.character(v87_Orig_lyr1$out_degree)),
                  v57=as.numeric(as.character(v57_Orig_lyr1$out_degree)), v82
=as.numeric(as.character(v82_Orig_lyr1$out_degree)),
                  v72=as.numeric(as.character(v72_Orig_lyr1$out_degree))))


data_SP<- melt(SP_random_distributions)
head(data_SP)

data_SP=data_SP[c('L1','value')]
colnames(data_SP)=c("nest","SP_rand_dist")

SP_ablines<-melt(SP_Quant_ablines)
#tail(Quant_ablines)
SP_ablines$cat_lines=rep("Quantile", nrow(SP_ablines))
SP_observed$cat_lines=rep("Obs",nrow(SP_observed))
#SPQuant_ablines<-as.data.frame(rbind(SP_ablines,SP_observed))

SP_ablines<-SP_ablines[c('L1','cat_lines','value')]
SP_observed<-SP_observed[c('L1','cat_lines','value')]

colnames(SP_ablines)<-c("nest","cat_lines","line_value")
colnames(SP_observed)<-c("nest","cat_lines","line_value")
SP_observed$line_value<-as.numeric(as.character(SP_observed$line_value))
SP_ablines$line_value<-as.numeric(as.character(SP_ablines$line_value))

#Quantiles_all=subset(SPQuant_ablines,cat_lines=="Quantile")
#all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")

p<-ggplot(data_SP, aes(x=SP_rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=SP_ablines, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=SP_observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
```

```
  theme(text = element_text(size=20)) + labs(x = "Out-Degree in spatial netwo
rk")
```

Violin plots - Spatial

```
#library(ggplot2)
safe_colorblind_palette <- c("#88CCEE", "#CC6677", "#DDCC77", "#117733", "#33
2288", "#AA4499",
                             "#44AA99", "#999933", "#882255", "#661100", "#66
99CC", "#888888")

CBfriendly = c(safe_colorblind_palette[5], safe_colorblind_palette[3], safe_c
olorblind_palette[10],
               safe_colorblind_palette[2], safe_colorblind_palette[1])


q <- ggplot(data_SP, aes(nest,SP_rand_dist,fill=factor(nest)))+
  geom_point(data=SP_observed,shape=23,size=2,alpha = 1,aes(factor(nest),line
_value,colour=factor(nest)),colour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly, labels = c("v57", "v72", "v82", "v8
7","v99")) +
  scale_fill_manual(values = CBfriendly)+
  scale_shape_identity()+ theme(legend.position="none")
r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,SP_rand_dist,group = factor(nest), fill=factor(nest),
      colour=factor(nest))
  #, scale="width", width=1
)+coord_flip()+
  ylab("Out-degree in Spatial overlap Network")+xlab("Colony ID")+
  geom_point(data=SP_observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")
r
```

final violin plot for Spatial layer for all five colonies

```
################################
## EXPORT IMAGE AND CSVs
################################
### High resolution image
tiff("SPlines_outDegHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code
```

```
#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-degree in Spatial ov
erlap network", x="Colony ID")+
  labs(tag = "C") +
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption A

dev.off()
```

To save randomization CSVs

Histogram of Aggression network

```
DB_random_distributions <- list(v99=as.numeric(as.character(v99_Rand_lyr2$out
_degree)),
                                v87=as.numeric(as.character(v87_Rand_lyr2$out
_degree)),
                                v57 =as.numeric(as.character(v57_Rand_lyr2$ou
t_degree)),
                                v82=as.numeric(as.character(v82_Rand_lyr2$out
_degree)),
                                v72=as.numeric(as.character(v72_Rand_lyr2$out
_degree)))

DB_Quant_ablines<-list(v99=v99_qts_deg2,v87=v87_qts_deg2,v57=v57_qts_deg2,
                       v82=v82_qts_deg2,v72=v72_qts_deg2)

DB_observed<-melt(list(v99=as.numeric(as.character(v99_Orig_lyr2$out_degree))
, v87=as.numeric(as.character(v87_Orig_lyr2$out_degree)),
                       v57=as.numeric(as.character(v57_Orig_lyr2$out_degree))
, v82=as.numeric(as.character(v82_Orig_lyr2$out_degree)),
                       v72=as.numeric(as.character(v72_Orig_lyr2$out_degree))
))


library(ggplot2);library(reshape2)
data_DB<- melt(DB_random_distributions)
tail(data_DB,20)

nrow(data_DB)

data_DB=data_DB[c('L1','value')]
colnames(data_DB)=c("nest","DB_rand_dist")
```

```
DB_ablines<-melt(DB_Quant_ablines)
Lowers=DB_ablines[c(1,3,5,7,9),]
Uppers=DB_ablines[c(2,4,6,8,10),]

#tail(Quant_ablines)
DB_ablines$cat_lines=rep("Quantile", nrow(DB_ablines))
DB_observed$cat_lines=rep("Obs",nrow(DB_observed))
#DBQuant_ablines<-as.data.frame(rbind(DB_ablines,DB_observed))

DB_ablines<-DB_ablines[c('L1','cat_lines','value')]
DB_observed<-DB_observed[c('L1','cat_lines','value')]

colnames(DB_ablines)<-c("nest","cat_lines","line_value")
colnames(DB_observed)<-c("nest","cat_lines","line_value")
DB_observed$line_value<-as.numeric(as.character(DB_observed$line_value))
DB_ablines$line_value<-as.numeric(as.character(DB_ablines$line_value))

#Quantiles_all=subset(DBQuant_ablines,cat_lines=="Quantile")
#all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")

p<-ggplot(data_DB, aes(x=DB_rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=DB_ablines, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=DB_observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-Degree in Aggregate net
work")
```

Violin plot - Aggression network

```
q <- ggplot(data_DB, aes(nest,DB_rand_dist,group = factor(nest), fill=factor(
nest)))+
  geom_point(data=DB_observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly)+
  scale_shape_identity()+ theme(legend.position="none")
```

```
#library(devtools)
#source_gist("https://gist.github.com/4578531")
r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
                   aes(nest,DB_rand_dist,group = factor(nest), fill=factor(ne
st),
                       colour=factor(nest))
                   #, scale="width", width=1
                   )+coord_flip()+
  ylab("Out-degree in Aggression Network")+xlab("Colony ID")+
  geom_point(data=DB_observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")
r
```

Final violin plot for aggression randomization networks for all 5 colonies

```
################################
## EXPORT IMAGE AND CSVs
################################
### High resolution image
tiff("DBlines_outDegHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-degree in Aggression
network", x="Colony ID")+
  labs(tag = "D") +
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption D

dev.off()
```

ggplots of Trophallaxis network

```
SC_random_distributions <- list(v99=as.numeric(as.character(v99_Rand_lyr3$out
_degree)),
                                v87=as.numeric(as.character(v87_Rand_lyr3$out
```

```
_degree)),
                                  v57 =as.numeric(as.character(v57_Rand_lyr3$ou
t_degree)),
                                  v82=as.numeric(as.character(v82_Rand_lyr3$out
_degree)),
                                  v72=as.numeric(as.character(v72_Rand_lyr3$out
_degree)))

SC_Quant_ablines<-list(v99=v99_qts_deg3,v87=v87_qts_deg3,v57=v57_qts_deg3,
                       v82=v82_qts_deg3,v72=v72_qts_deg3)

library(ggplot2);library(reshape2)

SC_observed<-melt(list(v99=as.numeric(as.character(v99_Orig_lyr3$out_degree))
, v87=as.numeric(as.character(v87_Orig_lyr3$out_degree)),
                       v57=as.numeric(as.character(v57_Orig_lyr3$out_degree))
, v82=as.numeric(as.character(v82_Orig_lyr3$out_degree)),
                       v72=as.numeric(as.character(v72_Orig_lyr3$out_degree))
))


data_SC<- melt(SC_random_distributions)
tail(data_SC,20)

nrow(data_SC)

data_SC=data_SC[c('L1','value')]
colnames(data_SC)=c("nest","SC_rand_dist")

SC_ablines<-melt(SC_Quant_ablines)
Lowers=SC_ablines[c(1,3,5,7,9),]
Uppers=SC_ablines[c(2,4,6,8,10),]

#tail(Quant_ablines)
SC_ablines$cat_lines=rep("Quantile", nrow(SC_ablines))
SC_observed$cat_lines=rep("Obs",nrow(SC_observed))
#SCQuant_ablines<-as.data.frame(rbind(SC_ablines,SC_observed))

SC_ablines<-SC_ablines[c('L1','cat_lines','value')]
SC_observed<-SC_observed[c('L1','cat_lines','value')]

colnames(SC_ablines)<-c("nest","cat_lines","line_value")
colnames(SC_observed)<-c("nest","cat_lines","line_value")
SC_observed$line_value<-as.numeric(as.character(SC_observed$line_value))
SC_ablines$line_value<-as.numeric(as.character(SC_ablines$line_value))

#Quantiles_all=subset(SCQuant_ablines,cat_lines=="Quantile")
#all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")
```

```
p<-ggplot(data_SC, aes(x=SC_rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=SC_ablines, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=SC_observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-Degree in Trophallaxis
network")
```

```
q <- ggplot(data_SC, aes(nest,SC_rand_dist,group = factor(nest), fill=factor(
nest)))+
  geom_point(data=SC_observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly)+
  scale_shape_identity()+ theme(legend.position="none")


#library(devtools)
#source_gist("https://gist.github.com/4578531")
r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,SC_rand_dist,group = factor(nest), fill=factor(nest),
      colour=factor(nest))
  #, scale="width", width=1
)+coord_flip()+
  ylab("Out-degree in Aggression Network")+xlab("Colony ID")+
  geom_point(data=SC_observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")
```

final violin plot of Trophallaxis network - degree

```
### High resolution image
tiff("SClines_outDegHD.tiff", units="in", width=8, height=5, res=300)
```

```
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-degree in Trophallax
is network", x="Colony ID")+
  labs(tag = "E") +
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption E

dev.off()
```

without re-running long randomizations over and over

```
#data_SC<-read.csv("NotStd_SCrandomDegree.csv")
#SC_observed<-read.csv("NotStd_SCObsDegree.csv")
#SC_ablines<-read.csv("NotStd_SCQuantilesDegrees.csv")
```

Exporting trophallaxis CSVs

```
#write.csv(data_SC, "NotStd_SCrandomDegree.csv")
#write.csv(SC_observed,"NotStd_SCObsDegree.csv")
#write.csv(SC_ablines,"NotStd_SCQuantilesDegrees.csv")
```

ggplots of solid food exchange network

```
ST_random_distributions <- list(v99=as.numeric(as.character(v99_Rand_lyr4$out
_degree)),
                                v87=as.numeric(as.character(v87_Rand_lyr4$out
_degree)),
                                v57 =as.numeric(as.character(v57_Rand_lyr4$ou
t_degree)),
                                v82=as.numeric(as.character(v82_Rand_lyr4$out
_degree)),
                                v72=as.numeric(as.character(v72_Rand_lyr4$out
_degree)))

ST_Quant_ablines<-list(v99=v99_qts_deg4,v87=v87_qts_deg4,v57=v57_qts_deg4,
                       v82=v82_qts_deg4,v72=v72_qts_deg4)

library(ggplot2);library(reshape2)

ST_observed<-melt(list(v99=as.numeric(as.character(v99_Orig_lyr4$out_degree))
, v87=as.numeric(as.character(v87_Orig_lyr4$out_degree)),
                       v57=as.numeric(as.character(v57_Orig_lyr4$out_degree))
, v82=as.numeric(as.character(v82_Orig_lyr4$out_degree)),
```

```
                        v72=as.numeric(as.character(v72_Orig_lyr4$out_degree))
))


data_ST<- melt(ST_random_distributions)
tail(data_ST,20)

nrow(data_ST)

## [1] 5000

data_ST=data_ST[c('L1','value')]
colnames(data_ST)=c("nest","ST_rand_dist")

ST_ablines<-melt(ST_Quant_ablines)
Lowers=ST_ablines[c(1,3,5,7,9),]
Uppers=ST_ablines[c(2,4,6,8,10),]

#tail(Quant_ablines)
ST_ablines$cat_lines=rep("Quantile", nrow(ST_ablines))
ST_observed$cat_lines=rep("Obs",nrow(ST_observed))
#STQuant_ablines<-as.data.frame(rbind(ST_ablines,ST_observed))

ST_ablines<-ST_ablines[c('L1','cat_lines','value')]
ST_observed<-ST_observed[c('L1','cat_lines','value')]

colnames(ST_ablines)<-c("nest","cat_lines","line_value")
colnames(ST_observed)<-c("nest","cat_lines","line_value")
ST_observed$line_value<-as.numeric(as.character(ST_observed$line_value))
ST_ablines$line_value<-as.numeric(as.character(ST_ablines$line_value))

#Quantiles_all=subset(STQuant_ablines,cat_lines=="Quantile")
#all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")

p<-ggplot(data_ST, aes(x=ST_rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=ST_ablines, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=ST_observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-Degree in solid food ex
change network")
```

```
##Violin plots

library(ggplot2)



q <- ggplot(data_ST, aes(nest,ST_rand_dist,group = factor(nest), fill=factor(
nest)))+
  geom_point(data=ST_observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly)+
  scale_shape_identity()+ theme(legend.position="none")

r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,ST_rand_dist,group = factor(nest), fill=factor(nest),
      colour=factor(nest))
  #, scale="width", width=1
)+coord_flip()+
  ylab("Out-degree in solid food exchange network")+xlab("Colony ID")+
  geom_point(data=ST_observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")
r
```

final violin plot of solid food exchange network - degree

```
setwd("D:/data/")

### High resolution image
tiff("STlines_outDegHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #STale_shape_diSTrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
```

```
  theme(text = element_text(size=20)) + labs(y = "PQ out-degree in solid food
exchange network", x="Colony ID")+
  labs(tag = "F") +
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption F



dev.off()
```

Randomization code for strength in single layer networks

Creating a vector with 5 colony names

```
nests=as.character(as.factor(c("v57", "v87", "v72", "v99", "v82")))
class(nests)

## [1] "character"

library(igraph)

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library(stringi)
```

#Randomization loop after sourcing different colony extended edgelists

```
for (n in 1:length(nests)){
  nest_path=paste("D:/data/",nests[n],"/", sep="")
  setwd(nest_path)

  ns_wasp_dataOriginal = read.csv(paste(nests[n],"_txtedges.csv", sep=""), he
ader=T)[,c(1:5)]

  library("dplyr")
ns_wasp_dataOrig<-as.data.frame(ns_wasp_dataOriginal %>% group_by(lyr1) %>% m
utate(NorWt = wt/max(wt)))

  ns_wasp_data <- ns_wasp_dataOrig[,c(1,2,3,4,6)]

  labels<-read.csv(paste(nests[n],"node_labels.csv",sep=""), header = T)

  ##For monolayer strength:
  library(igraph)
df=ns_wasp_data
df$node1<-as.factor(as.character(df$node1))
df$node2<-as.factor(as.character(df$node2))

head(df)
PQranks<-list()
PQ_Rstrs<-list()
rand_allLayers<-list()
```

```r
  ## this gets empty because we are shuffling within each layer and then maki
ng a combined extended edgelist for all 4 layers in a go

  for(i in 1:1000){ ## for each iteration
    Random_perstr<-list()

  orig_allLayers<-list() ##empty before running a cycle through all layers

  for(j in 1: length(levels(as.factor(df$lyr1)))) { ##only shuffle within lay
ers
    #j=4
    df1<-subset(df, lyr1 == j) # subset for that layer
    #length(unique(levels(as.factor(df1[,3]))))
    #creating an igraph object
    el1=as.matrix(df1[,c(1,3,5)]) #igraph needs the edgelist to be in matrix
format
    g1=graph.edgelist(el1[,c(1,2)], directed=TRUE) #We first create a network
from the first two columns, which has the list of vertices
    E(g1)$weight=as.numeric(el1[,3])

    #vertex_attr_names(g1)
    #g2=graph.empty(n=length(V(g1)$name), directed=TRUE)
    g2=g1
    V(g2)$name=sample(V(g1)$name) ## sampled/shuffled the vertices
    #V(g2)$name<-V(g1)$name
    #class(V(g1))
    #get.data.frame(g1)
    df_rand=get.data.frame(g2)
    #el_g1$weight=E(g1)$weight



    ## strength OF ORIGINAL NETWORK

    orig_strs<-as.data.frame(cbind(V(g1)$name,as.numeric(as.character(strengt
h(g1,mode="out")))))
    colnames(orig_strs)<-c("nodeID","strength")
    org_lyr_str<-merge(labels, orig_strs, by="nodeID")
    org_lyr_str$layerNo<-rep(j,nrow(org_lyr_str))
    org_lyr_str$nest<-rep("v87",nrow(org_lyr_str))
    colnames(org_lyr_str)=c("nodeID","nodeLabel","out_strength","LayerNo","ne
st")

    ## strength OF RANDOMIZED NETWORK

    rand_strs<-as.data.frame(cbind(V(g2)$name,as.numeric(as.character(strengt
h(g2, mode="out")))))
    colnames(rand_strs)<-c("nodeID","strength")
```

```
    rand_lyr_str<-merge(labels, rand_strs, by="nodeID")
    rand_lyr_str$layerNo<-rep(j,nrow(rand_lyr_str))
    rand_lyr_str$nest<-rep("v87",nrow(rand_lyr_str))
    colnames(rand_lyr_str)=c("nodeID","nodeLabel","out_strength","LayerNo","n
est")

    PQ_origlyr_str<-org_lyr_str[org_lyr_str$nodeLabel=="PQ",]
    PQ_randlyr_str<-rand_lyr_str[rand_lyr_str$nodeLabel=="PQ",]
    Random_perstr<-c(Random_perstr,list(PQ_randlyr_str))
    orig_allLayers<-c(orig_allLayers, list(PQ_origlyr_str))


  }


  #convert shuffled list to shuffled dataframe
  rand_allLayers<-c(rand_allLayers, list(do.call(rbind.data.frame, Random_per
str)))
  nrow(rand_allLayers)

  }
#                   Single layer assignments to each nest




#allRand_df<-do.call(rbind.data.frame, rand_allLayers)

##assign a value to a customized name by using "assign"
  assign(paste(nests[n],"_allRand_df",sep=""),do.call(rbind.data.frame,rand_a
llLayers))

assign(paste(nests[n],"_Rand_lyr1",sep=""),subset(eval(parse(text=paste(nests
[n],"_allRand_df",sep=""))), LayerNo=="1"))
assign(paste(nests[n],"_Rand_lyr2",sep=""),subset(eval(parse(text=paste(nests
[n],"_allRand_df",sep=""))), LayerNo=="2"))
assign(paste(nests[n],"_Rand_lyr3",sep=""),subset(eval(parse(text=paste(nests
[n],"_allRand_df",sep=""))), LayerNo=="3"))
assign(paste(nests[n],"_Rand_lyr4",sep=""),subset(eval(parse(text=paste(nests
[n],"_allRand_df",sep=""))), LayerNo=="4"))


assign(paste(nests[n],"_allOrig_df",sep=""),do.call(rbind.data.frame,orig_all
Layers))

assign(paste(nests[n],"_Orig_lyr1",sep=""),subset(eval(parse(text=paste(nests
[n],"_allOrig_df",sep=""))), LayerNo=="1"))
assign(paste(nests[n],"_Orig_lyr2",sep=""),subset(eval(parse(text=paste(nests
```

```
[n],"_allOrig_df",sep=""))), LayerNo=="2"))
assign(paste(nests[n],"_Orig_lyr3",sep=""),subset(eval(parse(text=paste(nests
[n],"_allOrig_df",sep=""))), LayerNo=="3"))
assign(paste(nests[n],"_Orig_lyr4",sep=""),subset(eval(parse(text=paste(nests
[n],"_allOrig_df",sep=""))), LayerNo=="4"))

##Quantiles

assign(paste(nests[n],"_qts_str1",sep=""),quantile((as.numeric(as.character(e
val(parse(text= paste(nests[n],"_Rand_lyr1",sep="")))[,3]))),probs = c(0.025,
0.975)))


assign(paste(nests[n],"_qts_str2",sep=""),quantile((as.numeric(as.character(e
val(parse(text= paste(nests[n],"_Rand_lyr2",sep="")))[,3]))),probs = c(0.025,
0.975)))


assign(paste(nests[n],"_qts_str3",sep=""),quantile((as.numeric(as.character(e
val(parse(text= paste(nests[n],"_Rand_lyr3",sep="")))[,3]))),probs = c(0.025,
0.975)))


assign(paste(nests[n],"_qts_str4",sep=""),quantile((as.numeric(as.character(e
val(parse(text= paste(nests[n],"_Rand_lyr4",sep="")))[,3]))),probs = c(0.025,
0.975)))

}
```

Histogram of spatial network

```
#all_random_distributions <- list(v99=v99_rand.dist, v87=v87_rand.dist, v57 =
v57_rand.dist, v82=v82_rand.dist, v72=v72_rand.dist)

SP_random_distributions <- list(v99=as.numeric(as.character(v99_Rand_lyr1$out
_strength)),
                                v87=as.numeric(as.character(v87_Rand_lyr1$out
_strength)),
                                v57 =as.numeric(as.character(v57_Rand_lyr1$ou
t_strength)),
                                v82=as.numeric(as.character(v82_Rand_lyr1$out
_strength)),
                                v72=as.numeric(as.character(v72_Rand_lyr1$out
_strength)))

SP_Quant_ablines<-list(v99=v99_qts_str1,v87=v87_qts_str1,v57=v57_qts_str1,
                       v82=v82_qts_str1,v72=v72_qts_str1)

library(ggplot2);library(reshape2)
```

```
SP_observed<-melt(list(v99=as.numeric(as.character(v99_Orig_lyr1$out_strength
)), v87=as.numeric(as.character(v87_Orig_lyr1$out_strength)),
                  v57=as.numeric(as.character(v57_Orig_lyr1$out_strength)), v
82=as.numeric(as.character(v82_Orig_lyr1$out_strength)),
                  v72=as.numeric(as.character(v72_Orig_lyr1$out_strength))))


data_SP<- melt(SP_random_distributions)
head(data_SP)

data_SP=data_SP[c('L1','value')]
colnames(data_SP)=c("nest","SP_rand_dist")

SP_ablines<-melt(SP_Quant_ablines)
#tail(Quant_ablines)
SP_ablines$cat_lines=rep("Quantile", nrow(SP_ablines))
SP_observed$cat_lines=rep("Obs",nrow(SP_observed))
#SPQuant_ablines<-as.data.frame(rbind(SP_ablines,SP_observed))

SP_ablines<-SP_ablines[c('L1','cat_lines','value')]
SP_observed<-SP_observed[c('L1','cat_lines','value')]

colnames(SP_ablines)<-c("nest","cat_lines","line_value")
colnames(SP_observed)<-c("nest","cat_lines","line_value")
SP_observed$line_value<-as.numeric(as.character(SP_observed$line_value))
SP_ablines$line_value<-as.numeric(as.character(SP_ablines$line_value))

#Quantiles_all=subset(SPQuant_ablines,cat_lines=="Quantile")
#all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")

p<-ggplot(data_SP, aes(x=SP_rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=SP_ablines, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=SP_observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-strength in spatial net
work")
```

Violin plots - Spatial

```
#library(ggplot2)
```

```
safe_colorblind_palette <- c("#88CCEE", "#CC6677", "#DDCC77", "#117733", "#33
2288", "#AA4499",
                             "#44AA99", "#999933", "#882255", "#661100", "#66
99CC", "#888888")

CBfriendly = c(safe_colorblind_palette[5], safe_colorblind_palette[3], safe_c
olorblind_palette[10],
               safe_colorblind_palette[2], safe_colorblind_palette[1])


q <- ggplot(data_SP, aes(nest,SP_rand_dist,fill=factor(nest)))+
  geom_point(data=SP_observed,shape=23,size=2,alpha = 1,aes(factor(nest),line
_value,colour=factor(nest)),colour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly, labels = c("v57", "v72", "v82", "v8
7","v99")) +
  scale_fill_manual(values = CBfriendly)+
  scale_shape_identity()+ theme(legend.position="none")
r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,SP_rand_dist,group = factor(nest), fill=factor(nest),
      colour=factor(nest))
  #, scale="width", width=1
)+coord_flip()+
  ylab("Out-strength in Spatial overlap Network")+xlab("Colony ID")+
  geom_point(data=SP_observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")
r
```

final violin plot for Spatial layer for all five colonies

```
################################
## EXPORT IMAGE AND CSVs
################################
### High resolution image
tiff("SPlines_outstrHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
```

```
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-strength in Spatial
overlap network", x="Colony ID")+
  labs(tag = "C") +
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption A


dev.off()
```

Histogram of Aggression network

```
DB_random_distributions <- list(v99=as.numeric(as.character(v99_Rand_lyr2$out
_strength)),
                                v87=as.numeric(as.character(v87_Rand_lyr2$out
_strength)),
                                v57 =as.numeric(as.character(v57_Rand_lyr2$ou
t_strength)),
                                v82=as.numeric(as.character(v82_Rand_lyr2$out
_strength)),
                                v72=as.numeric(as.character(v72_Rand_lyr2$out
_strength)))

DB_Quant_ablines<-list(v99=v99_qts_str2,v87=v87_qts_str2,v57=v57_qts_str2,
                       v82=v82_qts_str2,v72=v72_qts_str2)

DB_observed<-melt(list(v99=as.numeric(as.character(v99_Orig_lyr2$out_strength
)), v87=as.numeric(as.character(v87_Orig_lyr2$out_strength)),
                       v57=as.numeric(as.character(v57_Orig_lyr2$out_strength
)), v82=as.numeric(as.character(v82_Orig_lyr2$out_strength)),
                       v72=as.numeric(as.character(v72_Orig_lyr2$out_strength
))))


library(ggplot2);library(reshape2)
data_DB<- melt(DB_random_distributions)
tail(data_DB,20)

## [1] 5000

data_DB=data_DB[c('L1','value')]
colnames(data_DB)=c("nest","DB_rand_dist")

DB_ablines<-melt(DB_Quant_ablines)
Lowers=DB_ablines[c(1,3,5,7,9),]
Uppers=DB_ablines[c(2,4,6,8,10),]

#tail(Quant_ablines)
```

```
DB_ablines$cat_lines=rep("Quantile", nrow(DB_ablines))
DB_observed$cat_lines=rep("Obs",nrow(DB_observed))
#DBQuant_ablines<-as.data.frame(rbind(DB_ablines,DB_observed))

DB_ablines<-DB_ablines[c('L1','cat_lines','value')]
DB_observed<-DB_observed[c('L1','cat_lines','value')]

colnames(DB_ablines)<-c("nest","cat_lines","line_value")
colnames(DB_observed)<-c("nest","cat_lines","line_value")
DB_observed$line_value<-as.numeric(as.character(DB_observed$line_value))
DB_ablines$line_value<-as.numeric(as.character(DB_ablines$line_value))

#Quantiles_all=subset(DBQuant_ablines,cat_lines=="Quantile")
#all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")

p<-ggplot(data_DB, aes(x=DB_rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=DB_ablines, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=DB_observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-strength in Aggregate n
etwork")
```

Getting violin plots without having to run long randomizations each time by sourcing aggression csv

```
#data_DB<-read.csv("NotStd_DBrandomstrength.csv")
#DB_observed<-read.csv("NotStd_DBObsstrength.csv")
#DB_ablines<-read.csv("NotStd_DBQuantilesstrengths.csv")
```

Violin plot - Aggression network

```
q <- ggplot(data_DB, aes(nest,DB_rand_dist,group = factor(nest), fill=factor(
nest)))+
  geom_point(data=DB_observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly)+
  scale_shape_identity()+ theme(legend.position="none")
```

```
r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
                    aes(nest,DB_rand_dist,group = factor(nest), fill=factor(ne
st),
                        colour=factor(nest))
                    #, scale="width", width=1
                    )+coord_flip()+
  ylab("Out-strength in Aggression Network")+xlab("Colony ID")+
  geom_point(data=DB_observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")

r
```

Final violin plot for aggression randomization networks for all 5 colonies

```
################################
## EXPORT IMAGE AND CSVs
################################

### High resolution image
tiff("DBlines_outstrHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-strength in Aggressi
on network", x="Colony ID")+
  labs(tag = "D") +
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption D

dev.off()
```

ggplots of Trophallaxis network

```
SC_random_distributions <- list(v99=as.numeric(as.character(v99_Rand_lyr3$out
_strength)),
                                v87=as.numeric(as.character(v87_Rand_lyr3$out
_strength)),
                                v57 =as.numeric(as.character(v57_Rand_lyr3$ou
t_strength)),
                                v82=as.numeric(as.character(v82_Rand_lyr3$out
_strength)),
                                v72=as.numeric(as.character(v72_Rand_lyr3$out
_strength)))

SC_Quant_ablines<-list(v99=v99_qts_str3,v87=v87_qts_str3,v57=v57_qts_str3,
                       v82=v82_qts_str3,v72=v72_qts_str3)

library(ggplot2);library(reshape2)

SC_observed<-melt(list(v99=as.numeric(as.character(v99_Orig_lyr3$out_strength
)), v87=as.numeric(as.character(v87_Orig_lyr3$out_strength)),
                       v57=as.numeric(as.character(v57_Orig_lyr3$out_strength
)), v82=as.numeric(as.character(v82_Orig_lyr3$out_strength)),
                       v72=as.numeric(as.character(v72_Orig_lyr3$out_strength
))))


data_SC<- melt(SC_random_distributions)


data_SC=data_SC[c('L1','value')]
colnames(data_SC)=c("nest","SC_rand_dist")

SC_ablines<-melt(SC_Quant_ablines)
Lowers=SC_ablines[c(1,3,5,7,9),]
Uppers=SC_ablines[c(2,4,6,8,10),]

#tail(Quant_ablines)
SC_ablines$cat_lines=rep("Quantile", nrow(SC_ablines))
SC_observed$cat_lines=rep("Obs",nrow(SC_observed))
#SCQuant_ablines<-as.data.frame(rbind(SC_ablines,SC_observed))

SC_ablines<-SC_ablines[c('L1','cat_lines','value')]
SC_observed<-SC_observed[c('L1','cat_lines','value')]

colnames(SC_ablines)<-c("nest","cat_lines","line_value")
colnames(SC_observed)<-c("nest","cat_lines","line_value")
SC_observed$line_value<-as.numeric(as.character(SC_observed$line_value))
SC_ablines$line_value<-as.numeric(as.character(SC_ablines$line_value))

#Quantiles_all=subset(SCQuant_ablines,cat_lines=="Quantile")
#all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")
```

```
p<-ggplot(data_SC, aes(x=SC_rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=SC_ablines, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=SC_observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-strength in Trophallaxi
s network")
```

```
q <- ggplot(data_SC, aes(nest,SC_rand_dist,group = factor(nest), fill=factor(
nest)))+
  geom_point(data=SC_observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly)+
  scale_shape_identity()+ theme(legend.position="none")

r<-q + geom_violin(
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,SC_rand_dist,group = factor(nest), fill=factor(nest),
      colour=factor(nest))
  #, scale="width", width=1
)+coord_flip()+
  ylab("Out-strength in Aggression Network")+xlab("Colony ID")+
  geom_point(data=SC_observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")
```

final violin plot of Trophallaxis network - strength

```
### High resolution image
tiff("SClines_outstrHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code
```

```
#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #scale_shape_discrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-strength in Trophall
axis network", x="Colony ID")+
  labs(tag = "E") +
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption E

dev.off()
```

ggplots of solid food exchange network

```
ST_random_distributions <- list(v99=as.numeric(as.character(v99_Rand_lyr4$out
_strength)),
                                v87=as.numeric(as.character(v87_Rand_lyr4$out
_strength)),
                                v57 =as.numeric(as.character(v57_Rand_lyr4$ou
t_strength)),
                                v82=as.numeric(as.character(v82_Rand_lyr4$out
_strength)),
                                v72=as.numeric(as.character(v72_Rand_lyr4$out
_strength)))

ST_Quant_ablines<-list(v99=v99_qts_str4,v87=v87_qts_str4,v57=v57_qts_str4,
                       v82=v82_qts_str4,v72=v72_qts_str4)

library(ggplot2);library(reshape2)

ST_observed<-melt(list(v99=as.numeric(as.character(v99_Orig_lyr4$out_strength
)), v87=as.numeric(as.character(v87_Orig_lyr4$out_strength)),
                       v57=as.numeric(as.character(v57_Orig_lyr4$out_strength
)), v82=as.numeric(as.character(v82_Orig_lyr4$out_strength)),
                       v72=as.numeric(as.character(v72_Orig_lyr4$out_strength
))))


data_ST<- melt(ST_random_distributions)


data_ST=data_ST[c('L1','value')]
colnames(data_ST)=c("nest","ST_rand_dist")

ST_ablines<-melt(ST_Quant_ablines)
Lowers=ST_ablines[c(1,3,5,7,9),]
Uppers=ST_ablines[c(2,4,6,8,10),]
```

```r
#tail(Quant_ablines)
ST_ablines$cat_lines=rep("Quantile", nrow(ST_ablines))
ST_observed$cat_lines=rep("Obs",nrow(ST_observed))
#STQuant_ablines<-as.data.frame(rbind(ST_ablines,ST_observed))

ST_ablines<-ST_ablines[c('L1','cat_lines','value')]
ST_observed<-ST_observed[c('L1','cat_lines','value')]

colnames(ST_ablines)<-c("nest","cat_lines","line_value")
colnames(ST_observed)<-c("nest","cat_lines","line_value")
ST_observed$line_value<-as.numeric(as.character(ST_observed$line_value))
ST_ablines$line_value<-as.numeric(as.character(ST_ablines$line_value))

#Quantiles_all=subset(STQuant_ablines,cat_lines=="Quantile")
#all_Observed<-subset(Quant_ablines,cat_lines!="Quantile")

p<-ggplot(data_ST, aes(x=ST_rand_dist, fill=nest,alpha=0.1)) +
  geom_density()+
  geom_vline(data=ST_ablines, aes(xintercept=line_value,colour=nest),
             linetype="dashed",size=1)+
  geom_vline(data=ST_observed, aes(xintercept=line_value,colour=nest),
             linetype="solid",size=1.5)

#remove grey background
p+theme_bw()+ #remove grey background
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(x = "Out-strength in solid food
exchange network")
```

```r
##Violin plots

library(ggplot2)

q <- ggplot(data_ST, aes(nest,ST_rand_dist,group = factor(nest), fill=factor(
nest)))+
  geom_point(data=ST_observed,shape=23,size=2,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  theme(legend.position = "bottom")+
  scale_color_manual(values = CBfriendly,
                     labels = c("v57", "v72", "v82", "v87","v99")) +
  scale_fill_manual(values = CBfriendly)+
  scale_shape_identity()+ theme(legend.position="none")

r<-q + geom_violin(
```

```
  #draw_quantiles = c(0.025, 0.975),
  size=0.2, alpha=0.75,
  aes(nest,ST_rand_dist,group = factor(nest), fill=factor(nest),
      colour=factor(nest))
  #, scale="width", width=1
)+coord_flip()+
  ylab("Out-strength in solid food exchange network")+xlab("Colony ID")+
  geom_point(data=ST_observed,shape=23,size=3,alpha = 1,
             aes(factor(nest),line_value,group = nest, colour=factor(nest)),c
olour="black", stroke=1.5)+
  stat_summary(fun= function(x) quantile(x,0.975), geom="point", size=5.5, co
lour="black", shape="|")+
  stat_summary(fun= function(x) quantile(x,0.025), geom="point", size=5.5, co
lour="black", shape="|")
```

final violin plot of solid food exchange network - strength

```
### High resolution image
tiff("STlines_outstrHD.tiff", units="in", width=8, height=5, res=300)
# insert ggplot code

#remove grey background
r+theme_bw()+theme(legend.position="none")+#remove grey background
  #scale_fill_discrete(name="Colony ID")+
  #STale_shape_diSTrete(name = "Observed PQ rank")
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(
))+#remove plot grids
  theme(text = element_text(size=20)) + labs(y = "PQ out-strength in solid fo
od exchange network", x="Colony ID")+
  labs(tag = "F") +
  theme(plot.tag.position = "topright")+
  theme(plot.caption = element_text(colour = "black", hjust = 0, angle = 0, s
ize=22)) ##Adding caption F

dev.off()
```