Supplemental online material

Contents:

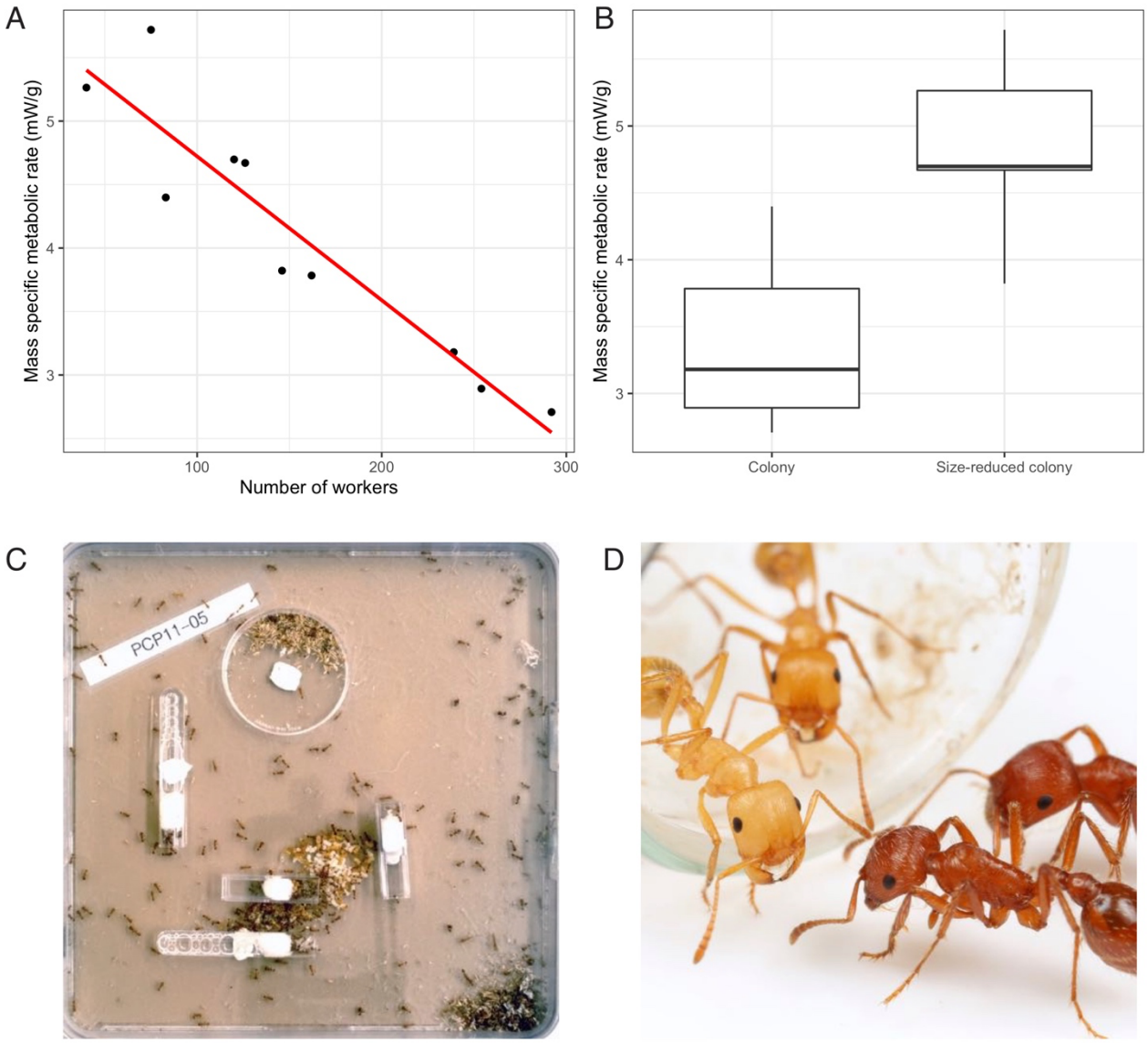


Figure S1. California seed-harvester ants, *Pogonomyrmex californicus*, are a model system to investigate how behavior and metabolism scale with colony size. (A) As previously reported [30], these colonies exhibit a metabolic allometry in which mass-specific metabolic rates decrease with increasing colony size. (B) Following an experimental manipulation that reduced the size of colonies by 50%, the mass-specific metabolic rates of the remaining individuals increased, indicating support for colony size as a driver for the metabolic allometry. These graphs (A-B) summarize the effects of colony size on the mass-specific metabolic rate of colonies including only the data for the colonies whose social networks were analyzed in this study. (C) This image is a frame of the video recording of one of the focal colonies while it is inside its artificial nest enclosure and also simultaneously inside a respirometry chamber. Also visible in the enclosure are water tubes, food within a small petri dish, a central pile with the queens, larvae, and pupae, and a refuse pile in the corner. (D) This photograph shows four *P. californicus* sisters interacting with each other by direct antennal contact.

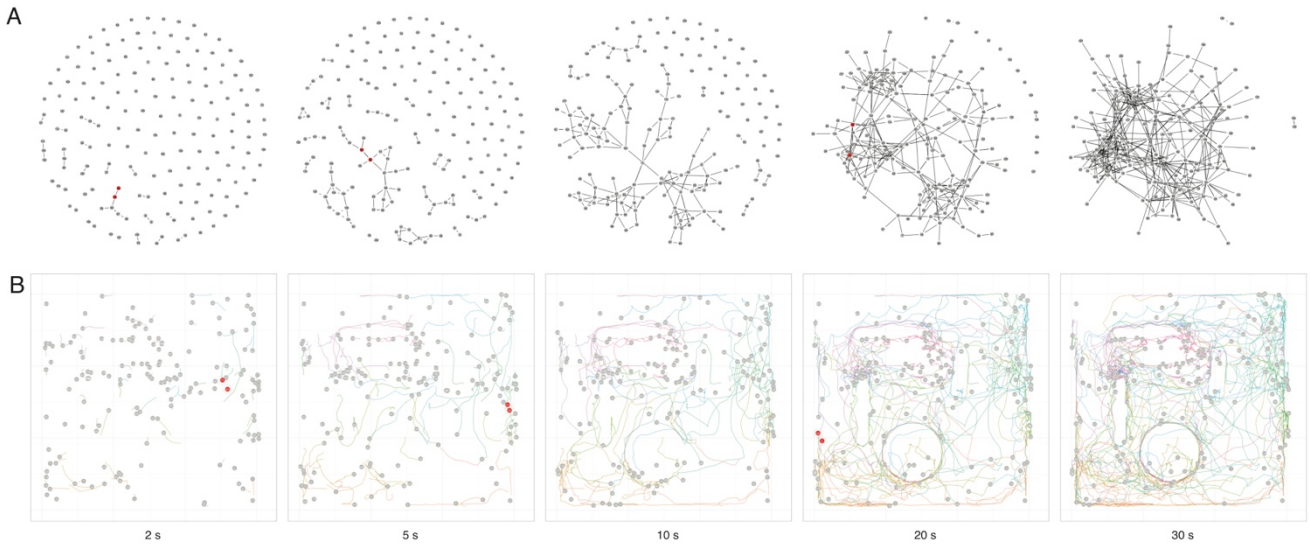


Figure S2. Antennation-based social network construction and spatial trajectories for a colony of *P. californicus* with 239 workers. (A) The social network is shown as it developed over time, with nodes representing individual ants and edges based on interactions manually identified by reviewing video of the colony for direct antennal contacts between workers. (B) These panels illustrate the trajectories of individual ants within this colony simultaneous with the corresponding social network graphs above. Individuals interacting at the specific time of each of these snapshots are highlighted in red.

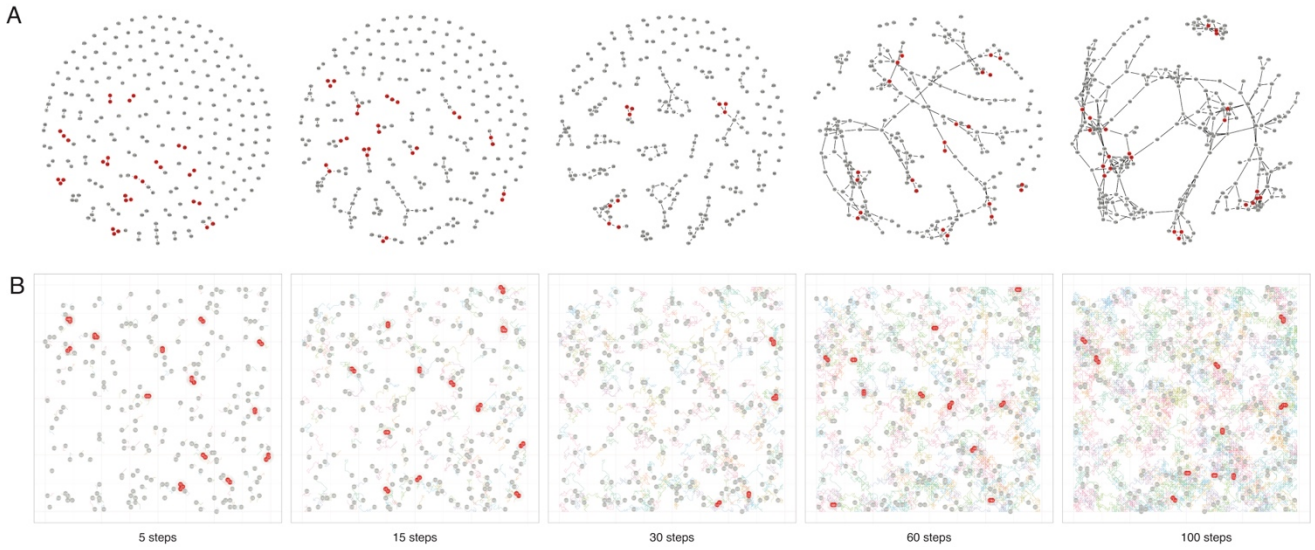


Figure S3. Social network construction and spatial trajectories for a simulated colony of 239 randomly moving virtual individuals. (A) The social network is shown as it developed over 100 timesteps of the model, with nodes representing individuals and edges based on proximity. (B) These panels illustrate the trajectories of individuals within this group's simulation, simultaneous with the corresponding social network graphs above. Individuals interacting at the specific time of each of these snapshots are highlighted in red.

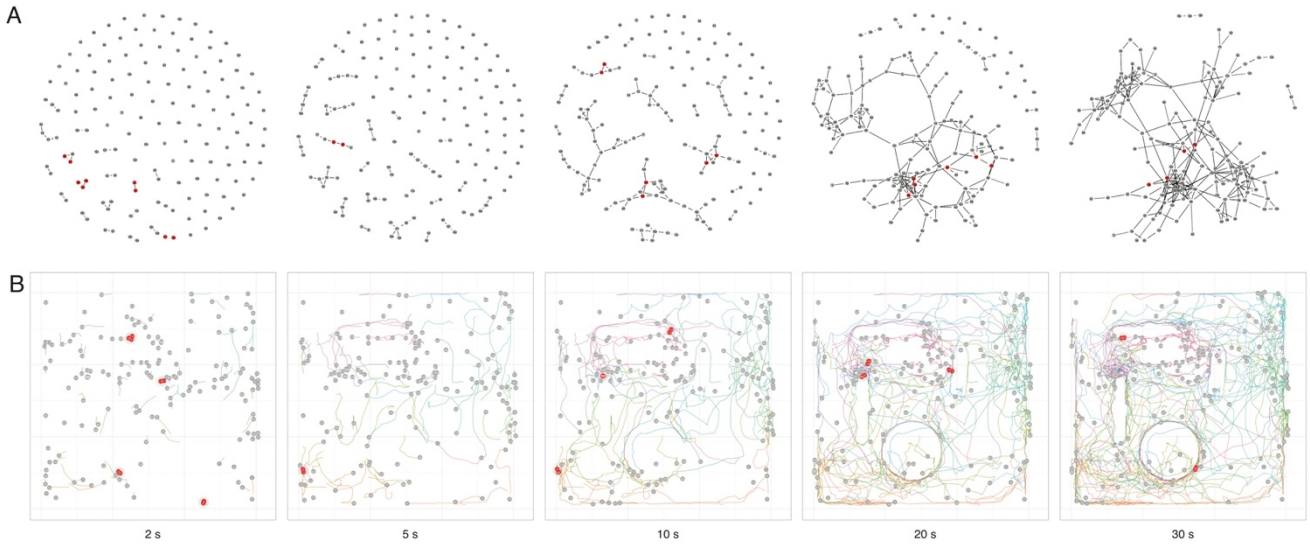


Figure S4. Proximity-based social network construction and spatial trajectories for a colony of *P. californicus* with 239 workers. (A) The social network is shown as it developed over time, with nodes representing individual ants and edges based on interactions inferred based on the proximity between individuals as calculated from their tracked positions. (B) These panels illustrate the spatial trajectories of individual ants (the same as shown in Figure 2B) within this colony simultaneous with the corresponding social network graphs above. Individuals inferred to be interacting at the specific time of each of these snapshots are highlighted in red.

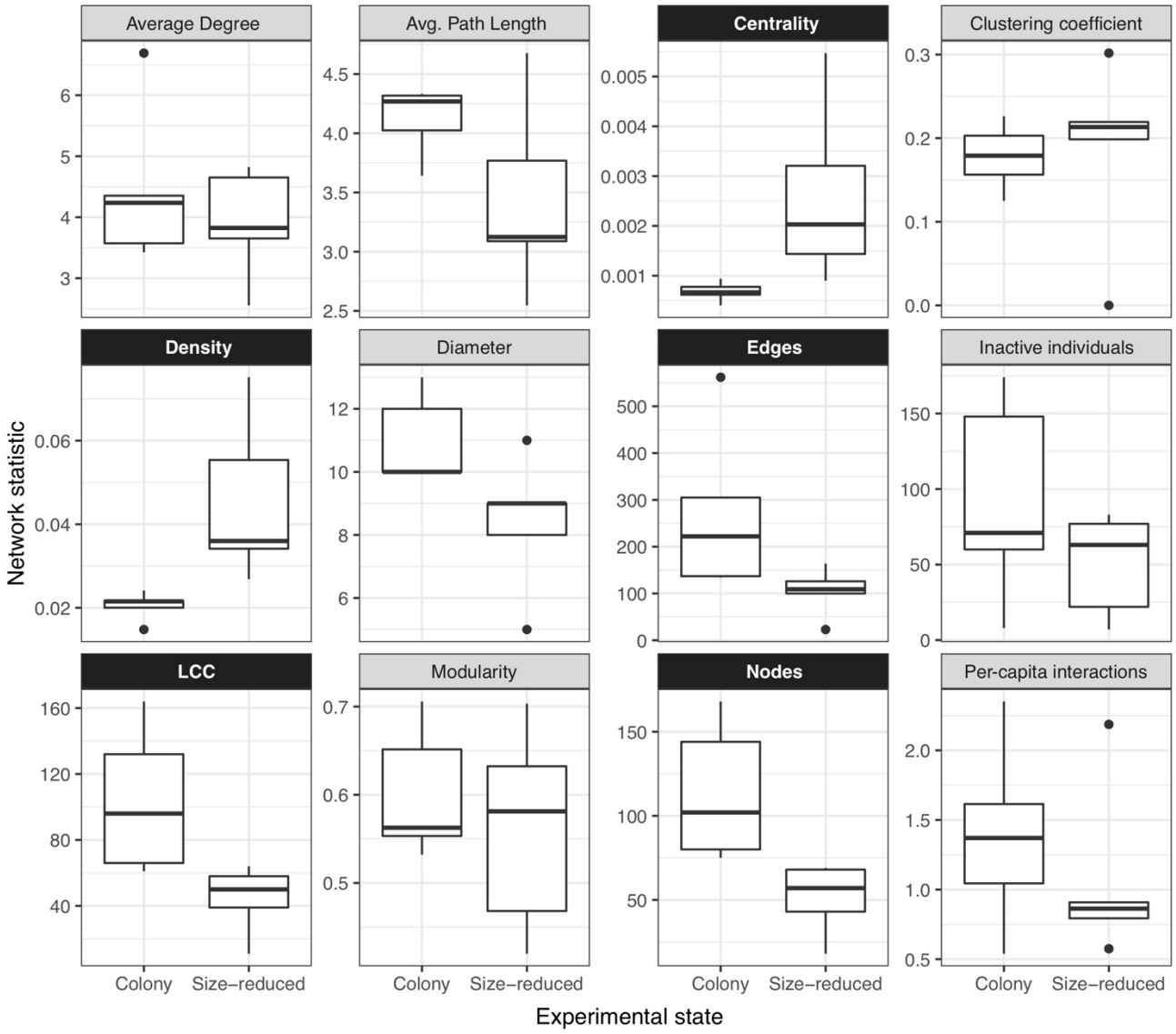


Figure S5. Effects of the experimental colony size reduction on network properties. Five of the properties were significantly affected by the manipulation, as indicated by solid black facet labels, including the number of nodes, the number of edges, closeness centrality, graph density, and the largest connected component.

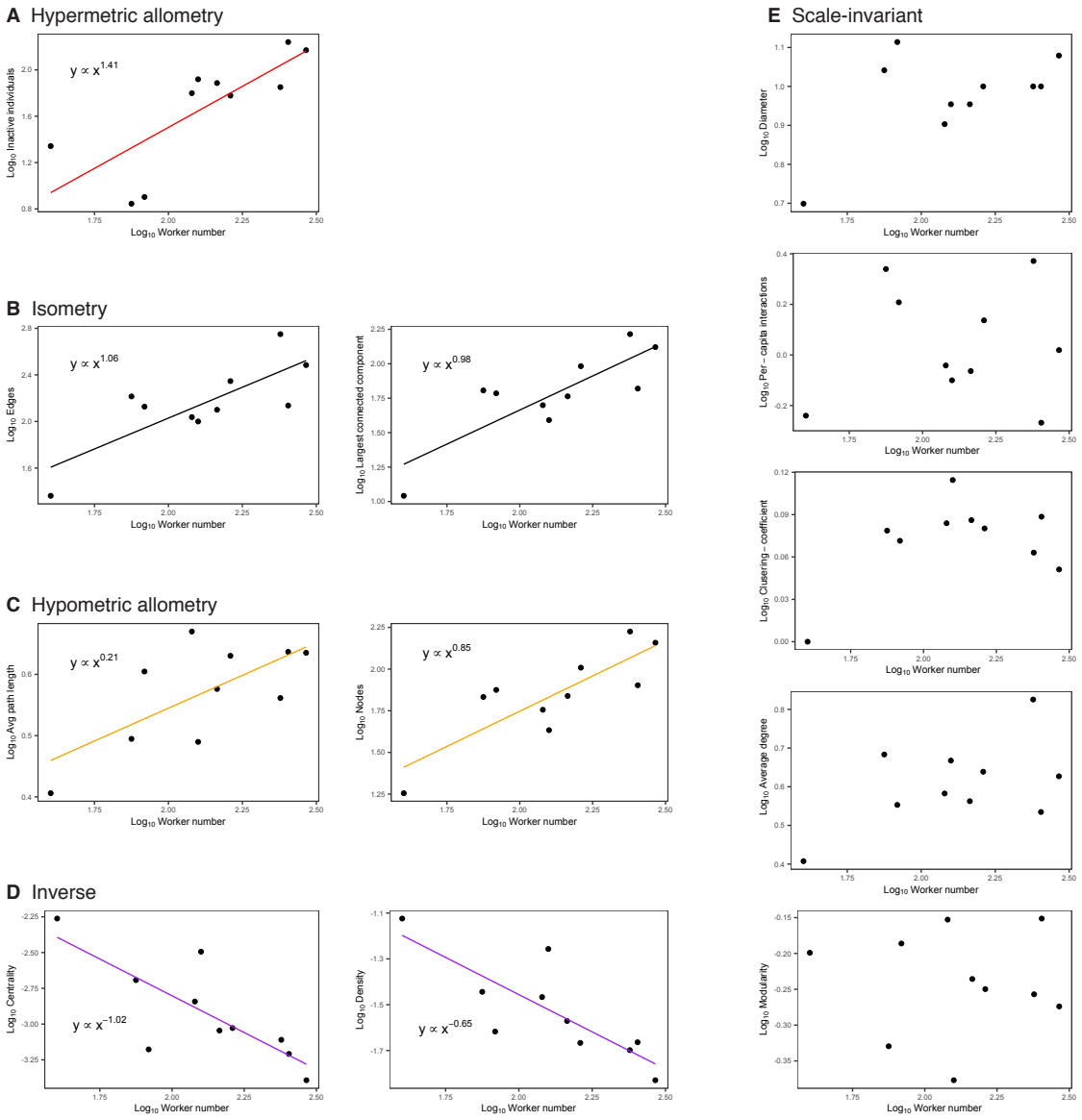
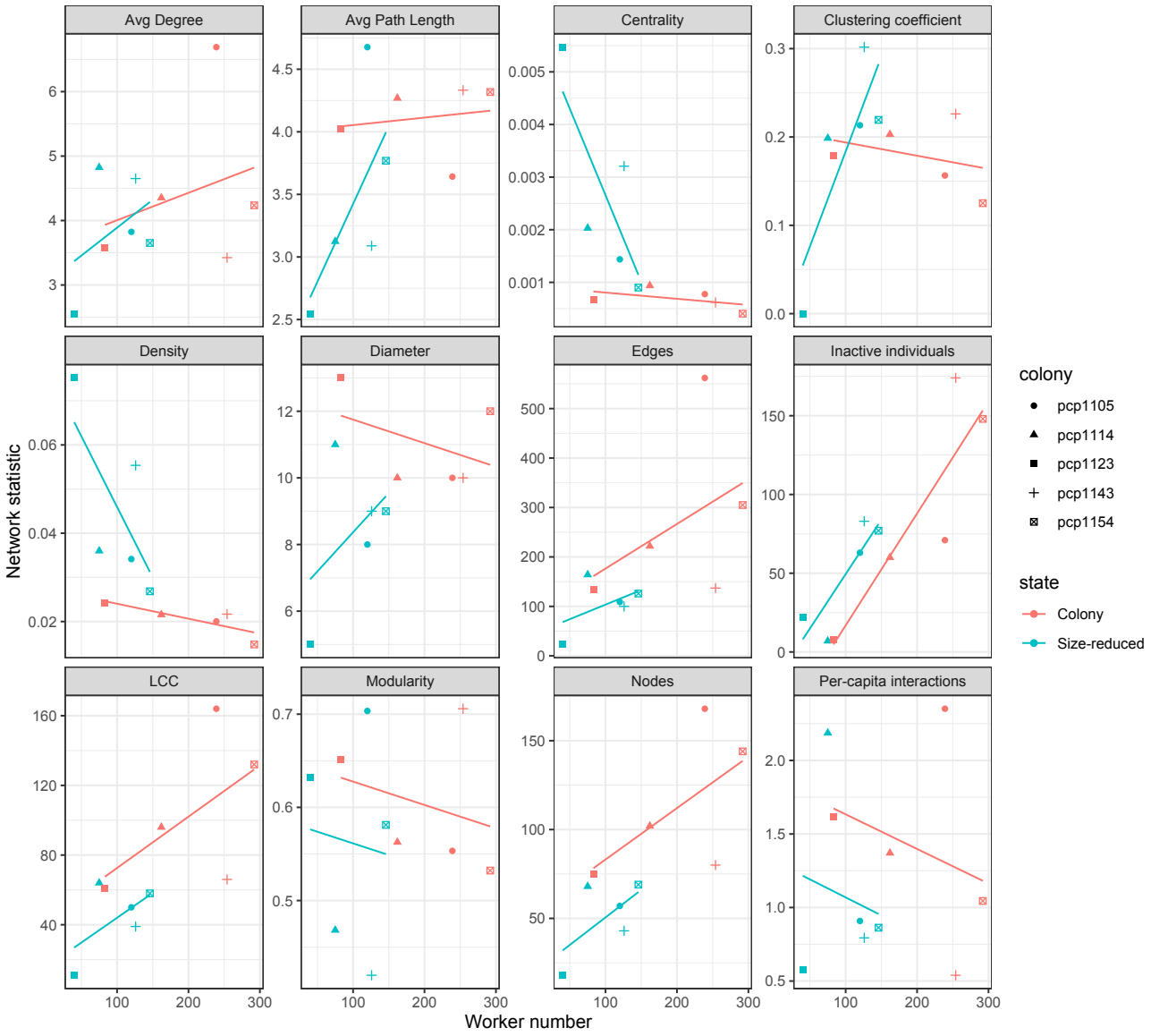


Figure S6. Scaling relationships in *P. californicus* interaction networks. (A) The number of inactive individuals (i.e. individuals not connected to the network) increases hypermetrically with colony size. (B) The number of edges and the size of the largest connected component scale proportional to colony size. (C) The average path length and the number of nodes in the network increase hypometrically with colony size. (D) Centrality and density both have an inverse relationship with colony size. (E) Many aspects of the colony network structure including diameter, per-capita interaction rate, clustering coefficient, average node degree, and modularity did not show any consistent changes with colony size.



106

107 **Figure S7.** Scaling of network attributes identified by originating colony (point shapes) for each of
108 the two experimental states (distinguished by color). Regression lines are plotted separately for the
109 five whole colonies (red lines) and the same five source colonies, but reduced to half their size (green
110 lines).

111

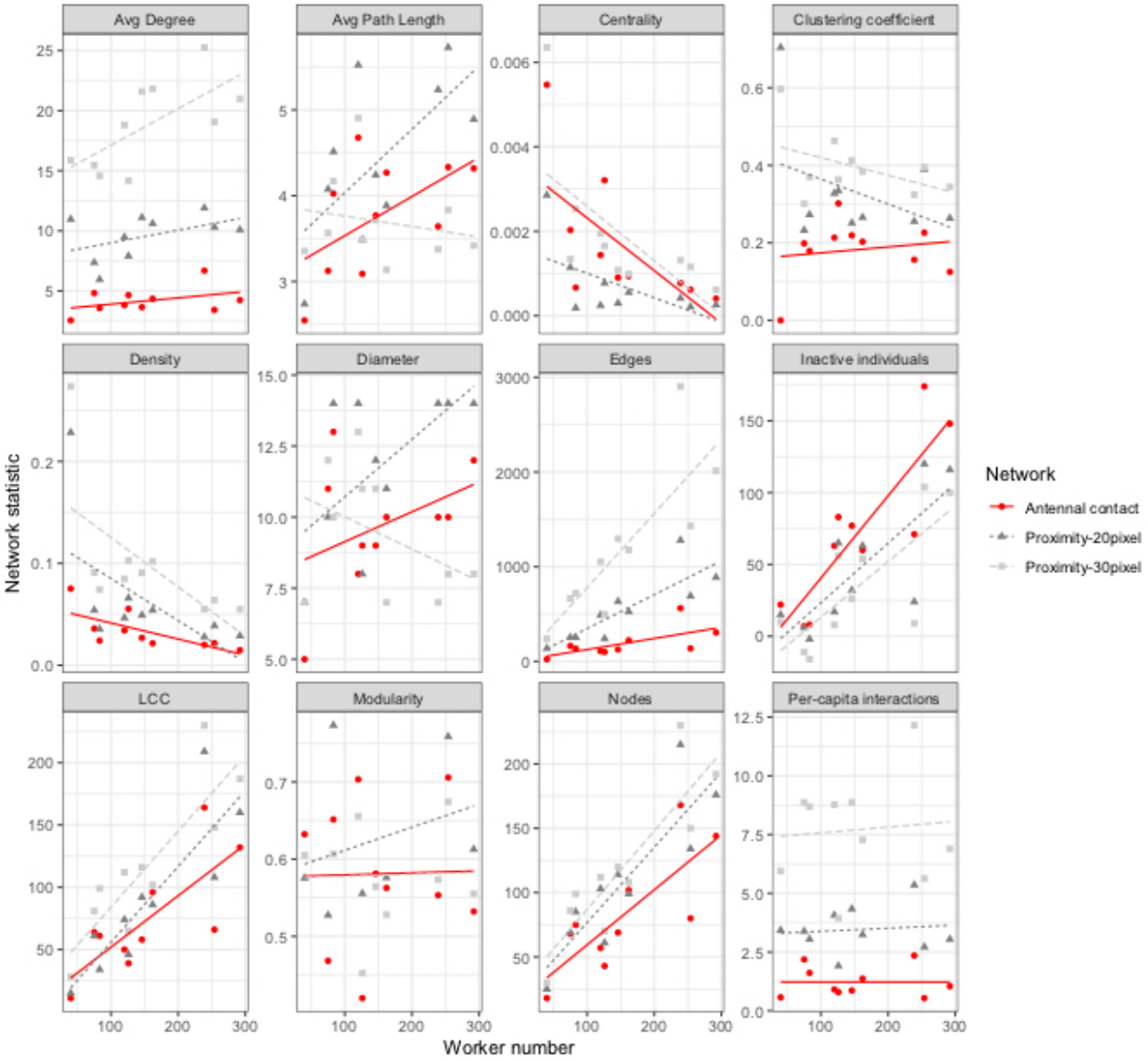


Figure S8. Comparing the scaling of social insect networks constructed using different methods for identifying interactions. We constructed networks based on observing direct antennal contacts ("antennation"; red points and lines) and by indirectly inferring contacts based on spatial proximity ("proximity"; grey points and dotted lines). Results for inferring networks using two different proximity thresholds are included, a 30-pixel distance between ants which was selected to match observed distances between ants making antennal contact, and a 20-pixel threshold selected to require a closer proximity. To evaluate trends in the data, all regression lines are plotted regardless of statistical significance, but complete summary statistics are provided in Table S3. The antennal contact network metrics in this figure are the same data plotted (also in red) in Figure 3 in the main manuscript text where they are compared with the results of the random interaction simulation. The proximity networks are sensitive to the threshold parameter and over-estimated the number of interactions, but the scaling patterns were not significantly different between the 20-pixel proximity networks and those based on observing direct antennal contacts.

Table S1. Colony demographic characteristics and composition. Note that data are from the previous study with these colonies, Waters et al. 2017.

Colony	State	Metabolic rate (mW)	Mass (g)	Queens	Larvae	Pupae	Workers
pcp1105	Whole colony	3.075	0.967	3	25	24	239
pcp1114	Whole colony	3.254	0.860	3	38	44	162
pcp1123	Whole colony	1.403	0.319	1	2	0	83
pcp1143	Whole colony	3.121	1.079	3	49	0	254
pcp1154	Whole colony	3.279	1.211	3	29	0	292
pcp1105	Size-reduced	2.349	0.500	3	12	12	120
pcp1114	Size-reduced	2.402	0.420	3	16	22	75
pcp1123	Size-reduced	0.858	0.163	1	2	0	40
pcp1143	Size-reduced	2.578	0.552	3	24	0	126
pcp1154	Size-reduced	2.400	0.628	3	15	0	146

Table S2. Network structure metrics.

Colony	State	Vertices	Edges	Diameter	Avg. Path Length	Modularity
pcp1105	Whole colony	168	562	10	3.642	0.553
pcp1114	Whole colony	102	222	10	4.268	0.562
pcp1123	Whole colony	75	134	13	4.024	0.651
pcp1143	Whole colony	80	137	10	4.332	0.705
pcp1154	Whole colony	144	305	12	4.317	0.532
pcp1105	Size-reduced	57	109	8	4.676	0.703
pcp1114	Size-reduced	68	164	11	3.123	0.468
pcp1123	Size-reduced	18	23	5	2.546	0.632
pcp1143	Size-reduced	43	100	9	3.088	0.419
pcp1154	Size-reduced	69	126	9	3.768	0.581

Colony	State	Density	Transitivity	Mean degree	Mean centrality	Inactive
pcp1105	Whole colony	0.020	0.156	6.690	0.0008	71
pcp1114	Whole colony	0.022	0.203	4.353	0.0009	60
pcp1123	Whole colony	0.024	0.179	3.573	0.0007	8
pcp1143	Whole colony	0.022	0.226	3.425	0.0006	174
pcp1154	Whole colony	0.015	0.125	4.236	0.0004	148
pcp1105	Size-reduced	0.033	0.211	3.793	0.0014	63
pcp1114	Size-reduced	0.036	0.199	4.824	0.0020	7
pcp1123	Size-reduced	0.075	0.000	2.556	0.0055	22
pcp1143	Size-reduced	0.055	0.302	4.651	0.0032	83
pcp1154	Size-reduced	0.027	0.219	3.652	0.0009	77

Table S3. Scaling coefficients. This table summarizes the scaling of network characteristics for the ten networks as determined by either watching for antennal contact or inferring interaction based on proximity. The exponents and their 95% confidence intervals reported below are the power-law scaling exponents for models using the colony size (number of workers) as the independent variable. Network characteristics indicated with superscript=1 indicates an aspect of the antennal contact networks that scaled with an exponent significantly different than 0, ones labeled with an asterisk (*) indicate significantly different scaling exponents for the antennal contact networks compared to the proximity networks.

	Antennal contact networks		Proximity networks (20 pixel)	
	Exponent	**95% C.I.**	**Exponent**	**95% C.I.**
Avg. degree	0.21 +/- 0.12 p=0.12, R^2=0.27	(-0.07, 0.50)	0.14 +/- 0.12 p=0.27, R^2=0.15	(-0.13, 0.40)
Avg. path length[1]	0.21 +/- 0.08 p=0.03, R^2=0.48	(0.03, 0.40)	0.27 +/- 0.09 p=0.02, R^2=0.53	(0.06, 0.47)
Centrality	-1.02 +/- 0.28 p=0.007, R^2=0.62	(-1.69, -0.37)	-0.96 +/-0.37 p=0.03, R^2=0.46	(-1.81, -0.10)
Clustering[1]	0.21 +/- 0.08 p= 0.03, R^2=0.48	(0.03, 0.40)	-0.27 +/- 0.16 p=0.14, R^2=0.25	(-0.64, 0.10)
Density[1]	-0.65 +/- 0.17 p=0.004, R^2=0.66	(-1.03, -0.27)	-0.76 +/-0.21 p=0.006, R^2=0.63	(-1.25, -0.29)
Diameter	0.23 +/- 0.13 p=0.1, R^2=0.30	(-0.05, 0.53)	0.28 +/- 0.11 p=0.03, R^2=0.47	(0.04, 0.53)
Edges[1]	1.05 +/- 0.29 p=0.007, R^2=0.62	(0.38, 1.73)	1.03 +/- 0.16 p=0.0002, R^2=0.84	(0.66, 1.40)
Inactive individuals[1]	1.42 +/- 0.39 p=0.008, R^2=0.61	(0.49, 2.33)	1.16 +/- 0.42 p=0.03, R^2=0.52	(0.17, 2.14)
LCC[1]	0.99 +/- 0.24 p=0.003, R^2=0.68	(0.43, 1.54)	1.14 +/- 0.17 p=0.0001, R^2=0.85	(0.75, 1.53)
Modularity	-0.004 +/- 0.1 p=0.97, R^2=0.0002	(-0.23, 0.22)	0.06 +/- 0.09 p=0.52, R^2=0.09	(-0.17, 0.30)
Nodes[1]	0.84 +/- 0.20 p=0.003, R^2=0.67	(0.38, 1.31)	0.89 +/- 0.14 p=0.0003, R^2=0.83	(0.56, 1.22)
Per-capita interactions	0.06 +/- 0.29 p=0.84, R^2= 0.006	(-0.62, 0.73)	0.03 +/- 0.16 p=0.85, R^2=0.004	(-0.34, 0.40)

```
#    AntSim-01-function.R
#    Spatially explicit random interaction simulation paper
#    JM Toth & JS Waters
#    June 2021

#    The below code is the actual algorithm to run the simulation
#    See manuscript for more information

antsim <- function(size,arena,time,trackingout,graphout){
  unlink(trackingout)
  unlink(graphout)
  tfile <- file(trackingout,"w")
  dfile <- file(graphout,"w")

  #Initialize ants
  X = sample(1:arena,size,replace=T)
  Y = sample(1:arena,size,replace=T)

  lastint = rep(0,size) #last interaction to prevent duplication

  minBound = 0;
  maxBound = arena;

  interaction = ""
  coords = ""
  for(t in 1:timesteps){
    print(t)

    #Calculate Motion
    for(i in 1:size){
      generating = TRUE
      while(generating){
        invalidX = TRUE
        while(invalidX){
          newX = X[i] + sample(-1:1,1,replace=T)
          if(newX >= minBound && newX < maxBound){
            invalidX = FALSE;
          }
        }
        invalidY = TRUE
        while(invalidY){
          newY = Y[i] + sample(-1:1,1,replace=T)
          if(newY >= minBound && newY < maxBound){
            invalidY = FALSE;
          }
        }
        #Ensure no collisions
        generating = FALSE
        for(j in 1:size){
          if(X[j]==newX && Y[j]==newY && i!=j){
```

```
197               generating = TRUE;
198             }
199           }
200           X[i] = newX
201           Y[i] = newY
202         }
203       }
204       #Calculate Interactions
205       for(i in 1:size){
206         for(j in 1:size){
207           if( abs(X[i]-X[j])<=1 && abs(Y[i]-Y[j])<=1 && lastint[i]!=j && i!=j){
208             #Interaction
209             lastint[i] = j
210             interaction = paste(interaction,i,",",j,",",t,"\n",sep="")
211           }
212         }
213       }
214       #Print Coordinates
215       for(i in 1:size){
216         coords = paste(coords,i,",",X[i],",",Y[i],",",t,"\n",sep="")
217       }
218
219       if(t%%diskcache == 0){
220         cat(interaction,file=dfile,append=T)
221         cat(coords,file=tfile,append=T)
222         interaction = ""
223         coords = ""
224       }
225     }
226
227     close(tfile)
228     close(dfile)
229   }
230
```

```
#   AntSim-02-simulation-loop.R
#   Spatially explicit random interaction simulation paper
#   JM Toth & JS Waters
#   June 2021

#   The below code is the loop to run the antsim() simulation
#   based on the antsim() function defined in the AntSim-01-function.R file
#   See manuscript for more information

#   set working directory
setwd("~/Desktop")

#   1. Define the range of group sizes that should be simulated
#   2. Specify how large the (squre) virtual arena should be
#   3. Indicate how long the simulation should run
#   4. Optionally, edit the file names of the saved output files

for (i in 0:510){
 size <- 5+i
 arena <- 100
 timesteps <- 100
 trackingout <- paste("coordinates_", size, ".txt", sep="")
 graphout <- paste("interactions_", size, ".txt", sep="")

 # This input decides how much the program will store before writing to disk.
 # Tune this to whatever makes the program run fast.
 diskcache = 10

 #  This section stores metadata
 metadata <- paste("Colony Size: ",size,"\n",
       "Arena Size: ",arena,"\n",
       "Time Steps: ",timesteps,"\n"
       ,sep="")
 metafile <- file("metadata.txt")
 writeLines(metadata,metafile)
 close(metafile)

 antsim(size, arena, time, trackingout, graphout)
      }
```

```
#   AntSim-03a-analysis_network-graphs.R
#   Spatially explicit random interaction simulation paper
#   JM Toth & JS Waters
#   June 2021

#   This script plots the ant colony networks and a selection of the simulated
 networks
#   See manuscript for more information

#   LOAD NETWORK PACKAGES
library(igraph)
library(ggplot2)
library(ggraph)
library(tidygraph)
library(patchwork)

#   SET WORKING DIRECTORY
setwd("~/Desktop/pogo-data/")

# set files to be the contents of the working directory
files <- dir()
file.number <- length(files)

#Turn each file in the directory into an igraph directly
for(i in 1:length(files)){
  nam <- files[i]
  assign(files[i], graph.data.frame(read.table(nam, sep="\t", header=F)))
}

# newfile <- ls(pattern="interactions*")
newfile <- ls(pattern="pcp*")


pcal <- c("g01", "g02", "g03", "g04", "g05", "g06", "g07", "g08", "g09", "g10")

for(i in 1:length(newfile)){
 g <- get(newfile[i])
 gg <- as_tbl_graph(g)
 assign(pcal[i], ggraph(gg, layout=layout_with_fr(g)) + geom_edge_link() +
 geom_node_point(shape=21, fill="white") +
 theme_graph(plot_margin=margin(0,0,0,0)) + labs(subtitle=newfile[i]))
}

(g05 + g07 + g03 + g09 + g01 + plot_layout(ncol=5)) / (g06 + g08 + g04 + g10 +
 g02 + plot_layout(ncol=5))

g <- get(newfile[i])
gg <- as_tbl_graph(g)
```

```
assign(pcal[i], ggraph(gg, layout=layout_with_fr(g)) + geom_edge_link() +
 geom_node_point(shape=21, fill="white") +
 theme_graph(plot_margin=margin(0,0,0,0)) + labs(subtitle=newfile[i]))


m1 <- graph.data.frame(read.csv("interactions_24.txt"))
m1t <- as_tbl_graph(m1)
m1g <- ggraph(m1t, layout=layout_with_fr(m1t)) + geom_edge_link() +
 geom_node_point(shape=23, fill="white") +
 theme_graph(plot_margin=margin(0,0,0,0)) + labs(subtitle="simulation n=24")

m2 <- graph.data.frame(read.csv("interactions_75.txt"))
m2t <- as_tbl_graph(m2)
m2g <- ggraph(m2t, layout=layout_with_fr(m2t)) + geom_edge_link() +
 geom_node_point(shape=23, fill="white") +
 theme_graph(plot_margin=margin(0,0,0,0)) + labs(subtitle="simulation n=75")

m3 <- graph.data.frame(read.csv("interactions_150.txt"))
m3t <- as_tbl_graph(m3)
m3g <- ggraph(m3t, layout=layout_with_fr(m3t)) + geom_edge_link() +
 geom_node_point(shape=23, fill="white") +
 theme_graph(plot_margin=margin(0,0,0,0)) + labs(subtitle="simulation n=150")

m4 <- graph.data.frame(read.csv("interactions_225.txt"))
m4t <- as_tbl_graph(m4)
m4g <- ggraph(m4t, layout=layout_with_fr(m4t)) + geom_edge_link() +
 geom_node_point(shape=23, fill="white") +
 theme_graph(plot_margin=margin(0,0,0,0)) + labs(subtitle="simulation n=225")

m5 <- graph.data.frame(read.csv("interactions_300.txt"))
m5t <- as_tbl_graph(m5)
m5g <- ggraph(m5t, layout=layout_with_fr(m5t)) + geom_edge_link() +
 geom_node_point(shape=23, fill="white") +
 theme_graph(plot_margin=margin(0,0,0,0)) + labs(subtitle="simulation n=300")

(g05 + g07 + g03 + g09 + g01 + plot_layout(ncol=5)) / (g06 + g08 + g04 + g10 +
 g02 + plot_layout(ncol=5)) / (m1g + m2g + m3g + m4g + m5g + plot_layout(ncol=5))
```

```
#   AntSim-03b-analysis-pcal-networks.R
#   Spatially explicit random interaction simulation paper
#   JM Toth & JS Waters
#   June 2021

#   The below code calculates network statistics based
#   on the observed Pogonomyrmex californicus colonies

#   LOAD NETWORK PACKAGES
library(igraph)
library(ggplot2)
library(ggraph)
library(tidygraph)
library(patchwork)
library(tidyverse)
library(brainGraph)

#   SET WORKING DIRECTORY
setwd("~/Desktop/pogo-data/Pcal-networks")

# set files to be the contents of the working directory
files <- dir()
file.number <- length(files)

#Turn each file in the directory into an igraph directly
for(i in 1:length(files)){
  nam <- files[i]
  assign(files[i], graph.data.frame(read.table(nam, sep="\t", header=F)))
}

newfile <- ls(pattern="pcp*")

#   Creating empty vectors with same length as number of networks
Network.nodes <- rep(NA, length(newfile))
Network.edges <- rep(NA, length(newfile))
Diameters <- rep(NA, length(newfile))
Diameters.2 <- rep(NA, length(newfile))
Diameters.3 <- rep(NA, length(newfile))
Diameters.4 <- rep(NA, length(newfile))
Avg.degree <- rep(NA, length(newfile))
Network.density <- rep(NA, length(newfile))
Network.centrality <- rep(NA, length(newfile))
Clustering.coefficient <- rep(NA, length(newfile))
Average.path.length.un <- rep(NA, length(newfile))
Average.path.length.dir <- rep(NA, length(newfile))
StDev.degree <- rep(NA, length(newfile))
Variance.degree <- rep(NA, length(newfile))
Variance.degree <- rep(NA, length(newfile))
Efficiency <- rep(NA, length(newfile))
LCC <- rep(NA, length(newfile))
```

```
Modularity <- rep(NA, length(newfile))

#   define this function
mean.centrality <- function(x){mean(closeness(x, mode=c("all")))}

#   Calculate the network metrics for each network
for(i in 1:length(newfile)){
  Network.nodes[i] <- vcount(get(newfile[i]))
  Network.edges[i] <- ecount(get(newfile[i]))
  Diameters[i] <- diameter(get(newfile[i]))
  Diameters.2[i] <- diameter(get(newfile[i]), unconnected=F)
  Diameters.3[i] <- diameter(get(newfile[i]), unconnected=F, directed=F)
  Diameters.4[i] <- diameter(get(newfile[i]), directed=F)
  Avg.degree[i] <- mean(degree(get(newfile[i])))
  Network.density[i] <- graph.density(get(newfile[i]))
  Network.centrality[i] <- mean.centrality(get(newfile[i]))
  Clustering.coefficient[i] <- transitivity(get(newfile[i]))
  Average.path.length.un[i]<- average.path.length(get(newfile[i]), directed=F)
  Average.path.length.dir[i]<- average.path.length(get(newfile[i]), directed=T)
  StDev.degree[i]<- sd(degree(get(newfile[i])))
  Variance.degree[i]<- sd(degree(get(newfile[i])))^2
  Efficiency[i] <- efficiency(get(newfile[i]), type="global")
  LCC[i] <- max(clusters(get(newfile[i]), mode="weak")$csize)
  Modularity[i] <- modularity(cluster_leading_eigen(get(newfile[i])))
}

#   Bring the vectors together as a data frame
net.df <- data.frame(Nodes=Network.nodes, Edges=Network.edges,
 Diameter=Diameters, Avg.Degree=Avg.degree, Density=Network.density,
 Centrality=Network.centrality, Diameter.2=Diameters.2, Diameter.3=Diameters.3,
 Diameter.4=Diameters.4, Clustering.coefficient=Clustering.coefficient,
 Avg.Path.Length.un=Average.path.length.un,
 Avg.Path.Length.dir=Average.path.length.dir, Efficiency=Efficiency, LCC=LCC,
 Modularity=Modularity)

#   Add additional information, including colony sizes and IDs
net.df$id <- newfile
net.df$worker.number <- c(239, 120, 162, 75, 83, 40, 254, 126, 292, 146)
net.df$metrate <- c(3.075, 2.349, 3.254, 2.402, 1.403, 0.858, 3.121, 2.578,
 3.279, 2.400)
net.df$mass <- c(.967, .5, .860, .420, .319, .163, 1.079, .552, 1.211, .628)
net.df$state <- rep(c("Colony", "Size-reduced"), 5)
net.df$metrate.per.g <- net.df$metrate/net.df$mass
net.df$connected.fraction <- net.df$Nodes/net.df$worker.number
net.df$per.capita.interactions <- net.df$Edges/net.df$worker.number
net.df$inactives <- net.df$worker.number - net.df$Nodes
net.df$inactive.fr <- net.df$inactives/net.df$worker.number


# Metabolic rate figure
```

```
metrate.plot.1 <- ggplot(net.df, aes(x=worker.number, y=metrate.per.g)) +
 geom_point() + stat_smooth(method="lm", se=F, col="red") + theme_bw() +
 ylab("Mass specific metabolic rate (mW/g)") + xlab("Number of workers")
metrate.plot.2 <- ggplot(net.df, aes(x=state, y=metrate.per.g)) + geom_boxplot()
 + ylab("Mass specific metabolic rate (mW/g)") + xlab("") + theme_bw()
metrate.plot.1 + metrate.plot.2


#   Network scaling figures
pcal.scaling <- pivot_longer(net.df, names_to="metric", values_to="value", -
 c("Diameter.2", "Diameter.3", "Diameter.4", "Avg.Path.Length.dir", "id",
 "worker.number", "metrate", "mass", "metrate.per.g", "state", "Efficiency",
 "connected.fraction"))
ggplot(pcal.scaling, aes(x=worker.number, y=value)) + facet_wrap(~metric,
 scales="free_y", ncol=5) + geom_point() + theme_bw()

ggplot(pcal.scaling, aes(x=worker.number, y=value)) + facet_wrap(~metric,
 scales="free_y", ncol=4) + geom_point(aes(shape=state)) + theme_bw() +
 stat_smooth(method="lm", se=F, lty=2, size=0.5, color="grey") + guides(shape=F)
 + xlab("Worker number") + ylab("Network statistic")

ggplot(pcal.scaling, aes(x=state, y=value)) + facet_wrap(~metric,
 scales="free_y", ncol=4) + geom_boxplot() + theme_bw() +
 stat_smooth(method="lm", se=F, lty=2, size=0.5, color="grey") + guides(shape=F)
 + ylab("Network statistic")


#   T-tests
wilcox.test(Centrality ~ state, data=net.df, paired=F)
wilcox.test(Density ~ state, data=net.df, paired=F)
wilcox.test(Diameter ~ state, data=net.df, paired=F)
wilcox.test(Edges ~ state, data=net.df, paired=F)
wilcox.test(LCC ~ state, data=net.df, paired=F)
wilcox.test(Nodes ~ state, data=net.df, paired=F)

#   Scaling analysies

#   HYPERMETRY
summary(lm(log10(inactives) ~ log10(worker.number), data=net.df))
# p=0.007, R2=0.61, F1,8=12.61, exponent 1.41 +/- 0.39 SE
s1 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(inactives))) +
 geom_point() + theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Inactive~individuals)) + stat_smooth(method="lm", se=F,
 col="red", size=0.5) + annotate("text", x=1.75, y=2,
 label=expression(y%prop%x^1.41))  + theme(panel.grid.major = element_blank(),
 panel.grid.minor = element_blank(), axis.title=element_text(size=8),
 axis.text=element_text(size=6))

#   ISOMETRY
summary(lm(log10(LCC) ~ log10(worker.number), data=net.df))
```

```
# p=0.003, R2=0.67, F1,8=16.75, exponent 0.98 +/- 0.24 SE
s2 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(LCC))) + geom_point() +
 theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Largest~connected~component)) +
 stat_smooth(method="lm", se=F, col="black", size=0.5) + annotate("text", x=1.75,
 y=2, label=expression(y%prop%x^0.98))  + theme(panel.grid.major =
 element_blank(), panel.grid.minor = element_blank(),
 axis.title=element_text(size=8), axis.text=element_text(size=6))

summary(lm(log10(Edges) ~ log10(worker.number), data=net.df))
# p=0.006, R2=0.62, F1,8=13.1, exponent 1.06 +/- 0.29 SE
s3 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(Edges))) + geom_point()
 + theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Edges)) + stat_smooth(method="lm", se=F, col="black",
 size=0.5) + annotate("text", x=1.75, y=2.5, label=expression(y%prop%x^1.06))  +
 theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
 axis.title=element_text(size=8), axis.text=element_text(size=6))

# ALLOMETRY: HYPOMETRIC
summary(lm(log10(Nodes) ~ log10(worker.number), data=net.df))
# p=0.003, R2=0.69, F1,8=17.52, exponent 0.85 +/- 0.20 SE
s4 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(Nodes))) + geom_point()
 + theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Nodes)) + stat_smooth(method="lm", se=F, col="orange",
 size=0.5) + annotate("text", x=1.75, y=2.1, label=expression(y%prop%x^0.85))  +
 theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
 axis.title=element_text(size=8), axis.text=element_text(size=6))

summary(lm(log10(Avg.Path.Length.un) ~ log10(worker.number), data=net.df))
# p=0.03, R2=0.48, F1,8=7.25, exponent 0.21 +/- 0.08 SE
s5 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(Avg.Path.Length.un))) +
 geom_point() + theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Avg~path~length)) + stat_smooth(method="lm", se=F,
 col="orange", size=0.5) + annotate("text", x=1.75, y=.62,
 label=expression(y%prop%x^0.21))  + theme(panel.grid.major = element_blank(),
 panel.grid.minor = element_blank(), axis.title=element_text(size=8),
 axis.text=element_text(size=6))

# INVERSE
summary(lm(log10(Density) ~ log10(worker.number), data=net.df))
# p=0.004, R2=0.65, F1,8=15.5, exponent -0.65 +/- 0.17 SE
s6 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(Density))) +
 geom_point() + theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Density)) + stat_smooth(method="lm", se=F,
 col="purple", size=0.5) + annotate("text", x=1.75, y=-1.6,
 label=expression(y%prop%x^-0.65))  + theme(panel.grid.major = element_blank(),
 panel.grid.minor = element_blank(), axis.title=element_text(size=8),
 axis.text=element_text(size=6))

summary(lm(log10(Centrality) ~ log10(worker.number), data=net.df))
```

```
# p=0.007, R2=0.62, F1,8=12.95, exponent -1.02 +/- 0.28 SE
s7 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(Centrality))) +
 geom_point() + theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Centrality)) + stat_smooth(method="lm", se=F,
 col="purple", size=0.5) + annotate("text", x=1.75, y=-3,
 label=expression(y%prop%x^-1.02))  + theme(panel.grid.major = element_blank(),
 panel.grid.minor = element_blank(), axis.title=element_text(size=8),
 axis.text=element_text(size=6))

# SCALE INVARIANT:
summary(lm(log10(per.capita.interactions) ~ log10(worker.number), data=net.df))
s8 <- ggplot(net.df, aes(x=log10(worker.number),
 y=log10(per.capita.interactions))) + geom_point() + theme_bw() +
 xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~per.capita.interactions)) + theme(panel.grid.major =
 element_blank(), panel.grid.minor = element_blank(),
 axis.title=element_text(size=8), axis.text=element_text(size=6))

summary(lm(log10(Modularity) ~ log10(worker.number), data=net.df))
s9 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(Modularity))) +
 geom_point() + theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Modularity)) + theme(panel.grid.major =
 element_blank(), panel.grid.minor = element_blank(),
 axis.title=element_text(size=8), axis.text=element_text(size=6))

summary(lm(log10(Diameter) ~ log10(worker.number), data=net.df))
s10 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(Diameter))) +
 geom_point() + theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Diameter))   + theme(panel.grid.major =
 element_blank(), panel.grid.minor = element_blank(),
 axis.title=element_text(size=8), axis.text=element_text(size=6))

summary(lm(log10(Clustering.coefficient+1) ~ log10(worker.number), data=net.df))
s11 <- ggplot(net.df, aes(x=log10(worker.number),
 y=log10(Clustering.coefficient+1))) + geom_point() + theme_bw() +
 xlab(expression(Log[10]*~Worker~number)) + ylab(expression(Log[10]*~Clusering-
 coefficient))  + theme(panel.grid.major = element_blank(), panel.grid.minor =
 element_blank(), axis.title=element_text(size=8),
 axis.text=element_text(size=6))

summary(lm(log10(Avg.Degree) ~ log10(worker.number), data=net.df))
s12 <- ggplot(net.df, aes(x=log10(worker.number), y=log10(Avg.Degree))) +
 geom_point() + theme_bw() + xlab(expression(Log[10]*~Worker~number)) +
 ylab(expression(Log[10]*~Average~degree))  + theme(panel.grid.major =
 element_blank(), panel.grid.minor = element_blank(),
 axis.title=element_text(size=8), axis.text=element_text(size=6))

#   Composite figure
```

```
(s1 + plot_spacer() + plot_spacer()) / (s2 + s3 + plot_spacer()) / (s4 + s5 +
 plot_spacer()) / (s6 + s7 + plot_spacer()) / (s8 + s9 + s10) / (s11 + s12 +
 plot_spacer())

#   Summary statistic (Mean +/- SEM)
std <- function(x) {
 print(mean(x))
 print(sd(x)/sqrt(length(x)))
      }
```

```
#   AntSim-03c-analysis-simulated-networks.R
#   Spatially explicit random interaction simulation paper
#   JM Toth & JS Waters
#   June 2021

#   The below code calculates network statistics based
#   on the random simulation results

#   LOAD NETWORK PACKAGES
library(igraph)
library(ggplot2)
library(ggraph)
library(tidygraph)
library(patchwork)
library(tidyverse)
library(brainGraph)

#   SET WORKING DIRECTORY
setwd("~/Desktop/simulation-data/interactions")

# set files to be the contents of the working directory
files <- dir()
file.number <- length(files)

#Turn each file in the directory into an igraph directly
for(i in 1:length(files)){
  nam <- files[i]
  assign(files[i], graph.data.frame(read.csv(nam, sep=",", header=F)))
}

newfile <- ls(pattern="interactions*")

#   Creating empty vectors with same length as number of networks
Network.nodes <- rep(NA, length(newfile)) #create an empty vector with same
 length as number of networks
Network.edges <- rep(NA, length(newfile)) #create an empty vector with same
 length as number of networks
Diameters <- rep(NA, length(newfile))
Diameters.2 <- rep(NA, length(newfile))
Diameters.3 <- rep(NA, length(newfile))
Diameters.4 <- rep(NA, length(newfile))
Avg.degree <- rep(NA, length(newfile))
Network.density <- rep(NA, length(newfile))
Network.centrality <- rep(NA, length(newfile))
Clustering.coefficient <- rep(NA, length(newfile))
Average.path.length.un <- rep(NA, length(newfile))
Average.path.length.dir <- rep(NA, length(newfile))
StDev.degree <- rep(NA, length(newfile))
Variance.degree <- rep(NA, length(newfile))
Variance.degree <- rep(NA, length(newfile))
```

```
Efficiency <- rep(NA, length(newfile))
LCC <- rep(NA, length(newfile))
Modularity <- rep(NA, length(newfile))

#   define this function
mean.centrality <- function(x){mean(closeness(x, mode=c("all")))}

#   Needed to make this function because some of the simulated
#   networks were returning errors in the modularity function
james.modularity <- function(x){
var1 <- try(modularity(cluster_leading_eigen(x)), silent=T)
print(ifelse(is.numeric(var1), var1, NA))
}

#   Calculate the network metrics for each network
for(i in 1:length(newfile)){
  Network.nodes[i] <- vcount(get(newfile[i]))
  Network.edges[i] <- ecount(get(newfile[i]))
  Diameters[i] <- diameter(get(newfile[i]))
  Diameters.2[i] <- diameter(get(newfile[i]), unconnected=F)
  Diameters.3[i] <- diameter(get(newfile[i]), unconnected=F, directed=F)
  Diameters.4[i] <- diameter(get(newfile[i]), directed=F)
  Avg.degree[i] <- mean(degree(get(newfile[i])))
  Network.density[i] <- graph.density(get(newfile[i]))
  Network.centrality[i] <- mean.centrality(get(newfile[i]))
  Clustering.coefficient[i] <- transitivity(get(newfile[i]))
  Average.path.length.un[i]<- average.path.length(get(newfile[i]), directed=F)
  Average.path.length.dir[i]<- average.path.length(get(newfile[i]), directed=T)
  StDev.degree[i]<- sd(degree(get(newfile[i])))
  Variance.degree[i]<- sd(degree(get(newfile[i])))^2
  Efficiency[i] <- efficiency(get(newfile[i]), type="global")
  LCC[i] <- max(clusters(get(newfile[i]), mode="weak")$csize)
  Modularity[i] <- james.modularity(get(newfile[i]))
}

#   Bring the vectors together as a data frame
sim.df <- data.frame(Nodes=Network.nodes, Edges=Network.edges,
 Diameter=Diameters, Avg.Degree=Avg.degree, Density=Network.density,
 Centrality=Network.centrality, Diameter.2=Diameters.2, Diameter.3=Diameters.3,
 Diameter.4=Diameters.4, Clustering.coefficient=Clustering.coefficient,
 Avg.Path.Length.un=Average.path.length.un,
 Avg.Path.Length.dir=Average.path.length.dir, Efficiency=Efficiency, LCC=LCC,
 Modularity=Modularity)

#   Add additional information, including group sizes and IDs
sim.df$id <- newfile
sim.df$number <- as.numeric(gsub(".*?([0-9]+).*", "\\1", sim.df$id))
sim.df$connected.fraction <- sim.df$LCC/sim.df$number
sim.df$per.capita.interactions <- sim.df$Edges/sim.df$number
sim.df$inactives <- sim.df$number - sim.df$Nodes
```

```r
#   Network scaling figure
sim.scaling <- pivot_longer(sim.df, names_to="metric", values_to="value", -
 c("Diameter.2", "Diameter.3", "Diameter.4", "Avg.Path.Length.dir", "id",
 "number", "Efficiency", "connected.fraction"))

ggplot(sim.scaling, aes(x=number, y=value)) + facet_wrap(~metric,
 scales="free_y", ncol=5) + geom_point() + theme_bw()
ggplot(sim.scaling, aes(x=number, y=value)) + facet_wrap(~metric,
 scales="free_y", ncol=4) + geom_point(alpha=0.5, size=.5) + theme_bw() +
 guides(shape=F) + xlab("Simulated colony size") + ylab("Network statistic")

#   Merging simulation data with colony data
#   to create the overaly in Figure 7
sim.scaling.2 <- sim.scaling[,c(7, 9, 10)]
sim.scaling.2$data <- "simulation"

#   Note that this requires running code in the 03b.R file
pcal.scaling.2 <- pcal.scaling[,c(7, 13, 14)]
names(pcal.scaling.2)[1] <- "number"
pcal.scaling.2$data <- "colony"

#   Creating the merged data frame with matching columns
mega <- rbind(sim.scaling.2, pcal.scaling.2)

#   Plot for Figure 7
ggplot(mega, aes(x=number, y=value, col=data)) + facet_wrap(~metric,
 scales="free_y", ncol=4) + geom_point(size=.5) + theme_bw() + guides(shape=F) +
 xlab("Group size") + ylab("Network statistic") + stat_smooth(data=subset(mega,
 data=="colony"), method = "loess", formula = y ~ x, size = 1, se=T, span=1) +
 scale_color_manual("Data source: ", values=c("red", "black"))+
 theme(legend.position="bottom")

#   Modified Fig 7 plot, minimum size 25
#   to avoid centrality/density sensitivity at small group sizes
ggplot(mega[mega$number > 25,], aes(x=number, y=value, col=data)) +
facet_wrap(~metric, scales="free_y", ncol=4) + geom_point(size=.5) + theme_bw() +
guides(shape=F) + xlab("Group size") + ylab("Network statistic") +
stat_smooth(data=subset(mega, data=="colony"), method = "loess", formula = y ~ x,
size = 1, se=T, span=1) + scale_color_manual("Data source: ", values=c("red",
"black"))+ theme(legend.position="bottom") + scale_x_continuous(breaks=c(0,100,
200, 300), limits=c(0,350))
```

```
#   AntSim-04-network-and-spatial-animation.R
#   Spatially explicit random interaction simulation paper
#   JM Toth & JS Waters
#   June 2021

#   The below code generates frames for an animation
#   showing the development of a social network at the
#   same time as the spatial trajectories of the individuals
#   This is the code for one of the simulated data sets,
#   but it's the same for real colony data too.

#   LOAD NETWORK PACKAGES
library(ggplot2)
library(igraph)
library(patchwork)
library(ggraph)
library(tidygraph)

setwd("~/Desktop/simulation-data")

#   Import interaction data from the simulation
net <- read.table("interactions/interactions_239.txt", sep=",")
names(net) <- c("ant1", "ant2", "t")

#   Import spatial data from the simulation
map <- read.table("coordinates/coordinates_239.txt", sep=",")
head(map)
names(map) <- c("ant", "x", "y", "t")
map$interaction <- 0

 #  Label the coordinates when two ants interact
 for(i in 1:length(net$t)){
 time <- net$t[i]
 ant1 <- net$ant1[i]
 ant2 <- net$ant2[i]
 map$interaction[map$t == time & map$ant == ant1] <- 1
 map$interaction[map$t == time & map$ant == ant2] <- 1
 }

#   Make graph
g <- graph.data.frame(net, directed=F)
E(g)$time <- net[,3]

gg_color_hue <- function(n) {
  hues = seq(15, 375, length=n+1)
  hcl(h=hues, l=65, c=100)[1:n]
}

#   Make tidy graph and set initial layout
g1 <- as_tbl_graph(g)
```

```
old.fr.coordinates <- layout_randomly(g1)
# ggraph(g1, layout=layout_with_fr(g1)) + geom_edge_link() + geom_node_point() +
 theme_graph()

###
###
###

# Directory to save animation frames to
setwd("~/Desktop/Animations/vis04-n239")

png("SimulationModeln234-%03d.png", width=1600, height=900)

for(ti in seq(1,max(map$t),1)){

#   Making the left panel (network)
    E(g)$edge.weight <- ifelse(E(g)$time < ti+1,1,0)
    E(g)$status <- ifelse(E(g)$time < ti+1,"a","b")
    E(g)$color <- ifelse(E(g)$time < ti+1,"gray",rgb(0,0,0,0))
    V(g)$color <- ifelse(graph.strength(g)==0,rgb(0,0,0,0),"grey")
 interacting.now <- as.character(map$ant[map$t == ti & map$interaction == 1])
 V(g)$color[V(g)$name %in% interacting.now] <- gg_color_hue(1)

 g1 <- as_tbl_graph(g)
 coordinates.fr <- layout_with_fr(g1, coords = old.fr.coordinates, start.temp=.5,
 weights=E(g)$edge.weight)

p.1 <- ggraph(g1, layout=coordinates.fr) + geom_edge_link(aes(width=status,
 start_cap=label_rect(node1.name), end_cap=label_rect(node2.name))) +
 theme_graph() + scale_edge_width_manual(values=c(.5,0), breaks=c("a",
 "b"),drop=F)  + geom_node_point(aes(color=color), size=5) +
 geom_node_text(aes(label=name), size=3) + scale_color_manual(values=c("grey",
 "red"), breaks=c("grey", "#F8766D")) + guides(edge_width=F, color=F) +
 ggtitle("Social network") + theme(plot.title = element_text(size = 10, face =
 "bold", family="sans"))

old.fr.coordinates <- coordinates.fr

#   Making the right panel (spatial trajectories)
 map.t <- map[map$t == ti,]
 df.1 <- map[map$t < ti+1,]
 df.2 <- map[map$t == ti,]

p.2 <- ggplot(df.1, aes(x=x, y=y, group=ant, color=as.factor(ant))) +
 geom_path(size=0.4, alpha=.5) + theme_bw() + scale_color_discrete(guide=F) +
 labs(x=NULL, y=NULL) + scale_x_continuous(limits=c(0,100))+
 scale_y_continuous(limits=c(0,100)) + annotate("point", x=df.2$x[df.2$ant %in%
 interacting.now], y=df.2$y[df.2$ant %in% interacting.now], size=15, color="light
 grey", alpha=0.2) + annotate("point", x=df.2$x, y=df.2$y, size=6, color="grey")
 + annotate("point", x=df.2$x[df.2$ant %in% interacting.now], y=df.2$y[df.2$ant
```

```
 %in% interacting.now], size=6, color="red") + annotate("text", x=df.2$x,
 y=df.2$y, label=df.2$ant, size=3, color="white") +
 theme(axis.line=element_blank(),
 axis.text.x=element_blank(),axis.text.y=element_blank(),
 axis.ticks=element_blank(),axis.title.x=element_blank(),axis.title.y=element_bla
 nk()) + ggtitle("Spatial trajectories") + theme(plot.title = element_text(size =
 10, face = "bold", family="sans"))

print((p.1 + p.2) + plot_annotation(title="Spatially explicit random interaction
 social network simulation", subtitle="Colony size: N=239 virtual ants", caption=
 paste("Simulation frame: ", ti, "/100", sep="")))
}
dev.off()

# NOTES
# The code above only saves a sequence of frames.
# Beware: For large networks each frame can take 1-5 minutes to render and save.
#
#To put the frames together and save as a movie,
# we used the "Open Image Sequence..." option in an old version of Quicktime
 Player,
# but you could also use ImageJ to do the same thing.
```

```
#   AntSim-05-trajectory-based-proximity-networks.R
#   Extracting social network based on proximity
#   June 2021

library(ggplot2)
library(igraph)

#   Set working directory (to find data files)
setwd("~/Desktop/source-trajectories")

#   Set files to be the contents of the directory
files <- dir()
file.number <- length(files)

for(i in 1:length(files)){
 setwd("~/Desktop/source-trajectories")
 nam <- files[i]
 assign(files[i], read.csv(nam, header=T))
 map <- get(files[i])
label <- nam

interaction_distance <- 30
diskcache = 10

names(map) <- c("ant", "frame", "x", "y")

# This file had some extra data, so limiting it for consistency
map <- map[map$frame<=90,]

#   Plot of all trajectories
setwd("~/Desktop/figures")
pdf(file=paste(label, "-traj", ".pdf", sep=""), width=8, height=8)
p <- ggplot(map, aes(x=x, y=y, col=ant, group=ant)) + geom_path() +
 guides(col="none") + labs(caption=label)
print(p)
dev.off()

setwd("~/Desktop/results-proximity-networks")

#Summary data about the recording
ant_list = unique(map$ant)
num_ants = max(ant_list) #This may not actually be the exact number of ants as
 some are not counted on occasion
num_frames <- max(map$frame)

#Variables for calculating interactions
lastint = rep(0,num_ants) #last interaction to prevent duplication
distance_reset =
 kronecker(matrix(1,num_ants,num_ants),TRUE)#rep(TRUE,[num_ants,num_ants]) #Must
 exit and re-enter interaction distance
```

```
934
935 #Setting up write file
936 graphout <- paste("interactions_", "trajectories_", label, ".txt", sep="")
937 dfile <- file(graphout,"w")
938 interaction <- ""
939
940 #Run analysis
941 for(t in 1:num_frames){
942   #Select relevant time point
943   print(t)
944   mapt <- map[map$frame==t,]
945
946   #Process each pair of ants
947   for(ant1 in ant_list){
948
949     if( !(ant1 %in% mapt$ant)){
950       print(paste("ant number ",ant1," not tracked on timestamp ",t,sep=""))
951       next
952     }
953
954     for(ant2 in ant_list){
955
956       if(! (ant2 %in% mapt$ant)){
957         next
958       }
959
960       if(ant1 != ant2){
961
962         #Finding indices of X and Y is necessary because some ants are skipped
963         ant1x <- mapt$x[which(mapt$ant==ant1)]
964         ant2x <- mapt$x[which(mapt$ant==ant2)]
965         ant1y <- mapt$y[which(mapt$ant==ant1)]
966         ant2y <- mapt$y[which(mapt$ant==ant2)]
967         #Need to check distance, last interaction, and whether the ants have
968 moved apart
969         distance <- sqrt(  (ant1x - ant2x)**2 + (ant1y - ant2y)**2  )
970         if(  (distance < interaction_distance) && (lastint[ant1] != ant2) &&
971 (distance_reset[ant1,ant2]>0)  ){
972           #Interaction
973           lastint[ant1] <- ant2
974           interaction = paste(interaction,ant1,",",ant2,",",t,"\n",sep="")
975           distance_reset[ant1,ant2] <- 0
976         }
977         else if(distance >= interaction_distance){
978           distance_reset[ant1,ant2] <- 1
979         }
980         #Uncomment this to turn off the requirement for moving apart before
981 interacting
982         #distance_reset[ant1,ant2] = TRUE
983
```

```
      }
    }
  }

  #Write data
  if(t%%diskcache == 0){
    cat(interaction,file=dfile,append=T)
    interaction <- ""
  }
  #ggplot(mapt, aes(x=x, y=y, col=ant)) + geom_point() + guides(col="none")
}
cat(interaction,file=dfile,append=T)
close(dfile)

}
```