Code

1. Code for generating targets sets in the network

2. Code for running diffusion simulations

3. Code for plotting simulation results

4. Code for similarity indices and logistic regressions

#In addition, four data files are supplied:

#1. ‘Edgelists.csv’ – this contains edgelists for the household visit and co-residence networks. It can #be #used to generate key player sets or analyse the network structures

#2. ‘network data.csv’ – this is the raw survey data we used to produce the network, it also contains #the demographic and other data used to generate conventional sets.

#3. ‘theset.csv’ – This is a matrix of the combined network which is used to run the simulations

#4. ‘ResultsTable.csv’ – This is a file containing the outputs of all the simulations, combined and #cleaned up. It can be used for plotting and for the cost-benefit analyses

library(tidyverse)

library(ggplot2)

library(igraph)

library(network)

library(keyplayer)

library(aod)

library(sets)

library(lme4)

#########1. Code for generating targets sets in the network############

networkdata <- read\_csv("networkdata.csv")

MasterEdgeList <- read\_csv("Intermediate files/MasterEdgeList.csv")

#Convert the edgelist into graph and network objects

graphs = list()

networks=list()

for (i in c(1:2)){

a <- MasterEdgeList[!is.na(MasterEdgeList[i+2]),]

edgevalues <- a[i+2]

a <- as.matrix(a[1:2])

graphs[[i]] <- graph\_from\_edgelist(a, directed=TRUE)

edge\_attr(graphs[[i]], "value") <- edgevalues

graphs[[i]] <- set\_vertex\_attr(graph = graphs[[i]], "age", value = networkdata$prelim.age)

graphs[[i]] <- set\_vertex\_attr(graph = graphs[[i]], "wealth", value = networkdata$hh.wealth1)

graphs[[i]] <- set\_vertex\_attr(graph = graphs[[i]], "gender", value = networkdata$prelim.gender)

graphs[[i]] <- set\_vertex\_attr(graph = graphs[[i]], "positions",value = networkdata$prelim.positions)

graphs[[i]] <- set\_vertex\_attr(graph = graphs[[i]], "jobs", value = networkdata$prelim.jobs)

graphs[[i]] <- set\_vertex\_attr(graph = graphs[[i]], "hh", value = networkdata$prelim.household)

networks[[i]] <- network(a, directed=TRUE, matrix.type="edgelist")

network::set.edge.attribute(networks[[i]], "value", edgevalues)

}

#Calculate some simple Descriptive Statistics

network.size(networks[[1]])

lgc <- component.largest(networks[[1]], result="graph")

gd <- geodist(lgc)

max(gd$gdist)

gtrans(networks[[1]], mode="graph", use.adjacency=FALSE)

components(networks[[1]], connected="weak")

network.density(networks[[1]])

#generate sets of key players

hhvisit1matrix <- as\_adjacency\_matrix(graphs[[2]])

#size = 2

kpindeg2 <- kpset(hhvisit1matrix, size=2, type="degree", method="max", cmode="indegree")

kpbet2 <- kpset(hhvisit1matrix, size=2, type="betweenness")

kpclose2 <- kpset(hhvisit1matrix, size=2, type="closeness", large=FALSE)

kpeigen2 <- kpset(hhvisit1matrix, size=2, type="evcent", method="max")

#size = 10

kpindeg10 <- kpset(hhvisit1matrix, size=10, type="degree", method="max", cmode="indegree")

kpbet10 <- kpset(hhvisit1matrix, size=10, type="betweenness")

kpclose10 <- kpset(hhvisit1matrix, size=10, type="closeness", large=FALSE)

kpeigen10 <- kpset(hhvisit1matrix, size=10, type="evcent", method="max")

#size = 20

kpindeg20 <- kpset(hhvisit1matrix, size=20, type="degree", method="max", cmode="indegree")

kpbet20 <- kpset(hhvisit1matrix, size=20, type="betweenness")

kpclose20 <- kpset(hhvisit1matrix, size=20, type="closeness", large=FALSE)

kpeigen20 <- kpset(hhvisit1matrix, size=20, type="evcent", method="max")

#size = 30

kpindeg30 <- kpset(hhvisit1matrix, size=30, type="degree", method="max", cmode="indegree")

kpbet30 <- kpset(hhvisit1matrix, size=30, type="betweenness")

kpclose30 <- kpset(hhvisit1matrix, size=30, type="closeness", large=FALSE)

kpeigen30 <- kpset(hhvisit1matrix, size=30, type="evcent", method="max")

#Add this last set as attribute in the graph

kpset <- data.frame(kpindeg30 $keyplayers, kpbet30 $keyplayers, kpclose30 $keyplayers, kpeigen30 $keyplayers)

colnames(kpset) <- c("indegree", "betweenness", "closeness", "eigenvector")

kpsetattribute <- data.frame(Indeg=integer(length=365), Betweenness=integer(length=365), Closeness=integer(length=365), Eigenvector=integer(length=365))

for(i in 1:4){

for(j in 1:10){

kpsetattribute[kpset [j,i],i] <- 1

}

}

graphs[[2]] <- set\_vertex\_attr(graph=graphs[[2]], "indegree", value= kpsetattribute $Indegree)

graphs[[2]] <- set\_vertex\_attr(graph=graphs[[2]], "betweennness", value= kpsetattribute $Betweenness)

graphs[[2]] <- set\_vertex\_attr(graph=graphs[[2]], "closeness", value= kpsetattribute $Closeness)

graphs[[2]] <- set\_vertex\_attr(graph=graphs[[2]], "eigenvector", value= kpsetattribute $Eigenvector)

##Repeat with half-sets for clustering

#size = 1

kpindeg1 <- kpset(hhvisit1matrix, size=1, type="degree", method="max", cmode="indegree")

kpbet1 <- kpset(hhvisit1matrix, size=1, type="betweenness")

kpclose1 <- kpset(hhvisit1matrix, size=1, type="closeness", large=FALSE)

kpeigen1 <- kpset(hhvisit1matrix, size=1, type="evcent", method="max")

#size = 5

kpindeg5 <- kpset(hhvisit1matrix, size=5, type="degree", method="max", cmode="indegree")

kpbet5 <- kpset(hhvisit1matrix, size=5, type="betweenness")

kpclose5 <- kpset(hhvisit1matrix, size=5, type="closeness", large=FALSE)

kpeigen5 <- kpset(hhvisit1matrix, size=5, type="evcent", method="max")

#size = 10 already exists above

#size = 15

kpindeg15 <- kpset(hhvisit1matrix, size=15, type="degree", method="max", cmode="indegree")

kpbet15 <- kpset(hhvisit1matrix, size=15, type="betweenness")

kpclose15 <- kpset(hhvisit1matrix, size=15, type="closeness", large=FALSE)

kpeigen15 <- kpset(hhvisit1matrix, size=15, type="evcent", method="max")

##cluster

#Code for n=2 given for illustration

combisets2 <- rbind(kpindeg1$keyplayers, kpindeg1$keyplayers, kpindeg1$keyplayers, kpindeg1$keyplayers)

combisets2matrix <- matrix(0, nrow=4, ncol=2)

combisets2matrix[,1] <- combisets2

for(i in 1:4){

node <- combisets2matrix[i,1]

alter = vector(mode="numeric", length=11)

alter[1] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit1name))

alter[2] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit2name))

alter[3] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit3name))

alter[4] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit4name))

alter[5] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit5name))

alter[6] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit6name))

alter[7] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visitor1name))

alter[8] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visitor2name))

alter[9] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visitor3name))

alter[10] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visitor4name))

alter[11] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visitor5name))

alter\_rmna <- alter[!is.na(alter)]

combisets2matrix[i,2] <- alter\_rmna[1]

}

#combisets10, combisets20, etc…

#####Characterstic based targeting

#Lowest attitudes to reporting

lowatt2 <- subset(networkdata, base.attitudes<11, select=c(prelim.name, base.attitudes))

lowatt2 <- lowatt2[2:3,1]

lowatt10 <- subset(networkdata, base.attitudes<12, select=c(prelim.name))

lowatt20 <- subset(networkdata, base.attitudes<13, select=c(prelim.name))

a <-subset(networkdata,BaseAttitudes==13, select=prelim.name)

lowatt20 <- rbind(lowatt20, a[1,])

lowatt30 <- subset(networkdata, base.attitudes<14, select=c(prelim.name, base.attitudes))

lowatt30 <- lowatt30[order(lowatt30$base.attitudes),]

lowatt30 <- lowatt30[1:30,1]

#highest attitudes to reporting

highatt2 <- subset(networkdata, base.attitudes>18, select=c(prelim.name, base.attitudes))

highatt2 <- highatt2[order(-highatt2$base.attitudes),]

highatt2 <- highatt2[1:2,1]

highatt10 <- subset(networkdata, base.attitudes>18, select=c(prelim.name))

highatt20 <- subset(networkdata, base.attitudes>17, select=c(prelim.name))

highatt30 <- subset(networkdata, base.attitudes>16, select=c(prelim.name, base.attitudes))

highatt30 <- highatt30[order(-highatt30$base.attitudes),]

highatt30 <- highatt30[1:30,1]

#Heads of wealthy households

wealthset <- subset(networkdata, prelim.gender==1, select=c(prelim.name, prelim.age, hh.wealth1, prelim.household)) %>%

subset(prelim.age>20) %>%

subset(prelim.age<55)

wealthset <- wealthset[order(-wealthset$hh.wealth1, wealthset$prelim.household, -wealthset$prelim.age),]

wealthset <- wealthset[!duplicated(wealthset$prelim.household),]

wealthset2 <- wealthset$prelim.name[1:2]

wealthset10 <- wealthset$prelim.name[1:10]

wealthset20 <- wealthset$prelim.name[1:20]

wealthset30 <- wealthset$prelim.name[1:30]

#leadership positions

leaderset <- subset(networkdata, prelim.positions==1, select=c(prelim.name, prelim.positions2))

a <- subset(networkdata, prelim.jobs==1, select=c(prelim.name, prelim.jobs2))

#I manually inspect the jobs for any positions of leadership

a <- a[c(7,28,57,77,93),]

leaderset <- merge(leaderset, a, by="prelim.name", all=TRUE)

#2: village chief and sub-chief

leaderset2 <- leaderset[c(2,364),1]

#10: village chief, sub-chiefs, committee members

leaderset10 <- leaderset[c(2,6,8,9,12,14,15,17,19,20),1]

leaderset20 <- leaderset[,1]

#gatekeeper (chief and his contacts)

gateset2 <- c(364,147) #the chief and his one visit nominee

gateset10 <- c(364, 147, 288, 230, 227, 226, 51, 270, 167, 161) #as above, plus drinking, meeting, labour, news & money nominees

gateset20 <- c(364, 147, 288, 230, 227, 226, 51, 270, 167, 161,

248, 363, 362, 361, 86, 30, 31, 32, 163, 309) #as above, plus respect & visitor nominees, siblings, uncaun

gateset30 <- c(364, 147, 288, 230, 227, 226, 51, 270, 167, 161,

248, 363, 362, 361, 86, 30, 31, 32, 163, 309,

50, 226, 136, 282, 40, 148, 282, 219, 53, 183) #as above, plus final nominations & 2nd-degree visit connections

#event targeting. Random sample skewed to women head of households near in the old village

#i.e. 3:1 and 2:1 likelihoods respectively

eventhhset <- c(1:156)

eventhhset2 <- sample(eventhhset, size=2, prob=rep(c(2,1), c(79, 156-79)), replace=FALSE)

eventhhset10 <- sample(eventhhset, size=10, prob=rep(c(2,1), c(79, 156-79)), replace=FALSE)

eventhhset20 <- sample(eventhhset, size=20, prob=rep(c(2,1), c(79, 156-79)), replace=FALSE)

eventhhset30 <- sample(eventhhset, size=30, prob=rep(c(2,1), c(79, 156-79)), replace=FALSE)

eventset <- select(networkdata, prelim.name, prelim.age, prelim.gender,prelim.household)

eventset <- eventset[order(eventset$prelim.gender),]

eventset2 <- eventset[eventset$prelim.household %in% eventhhset2,] %>%

group\_by(prelim.household) %>%

slice(1)

eventset2 <- eventset2$prelim.name

eventset10 <- eventset[eventset$prelim.household %in% eventhhset10,] %>%

group\_by(prelim.household) %>%

slice(1)

eventset10 <- eventset10$prelim.name

eventset20 <- eventset[eventset$prelim.household %in% eventhhset20,] %>%

group\_by(prelim.household) %>%

slice(1)

eventset20 <- eventset20$prelim.name

eventset30 <- eventset[eventset$prelim.household %in% eventhhset30,] %>%

group\_by(prelim.household) %>%

slice(1)

eventset30 <- eventset30$prelim.name

#targeting conservationists, manually select

conservationset2 <- c(274, 117) #members of forest community

conservationset10 <- c(306, 117, 162, 274, 339, 364, 100, 149, 210, 148) #including some random ibis rice farmers

conservationset20 <- c(306, 117, 162, 274, 339, 364, 100, 149, 210, 148,

209, 165, 193, 58, 229, 360, 117, 257, 118, 323) #including all ibis rice farmers

#####Compile the target sets, except for the clustered sets, into lists

#sets of two targets

targetsets2 = list()

targetsets2[[1]] <- kpindeg2$keyplayers

targetsets2[[2]] <- kpbet2$keyplayers

targetsets2[[3]] <- kpclose2$keyplayers

targetsets2[[4]] <- kpeigen2$keyplayers

targetsets2[[5]] <- lowatt2$prelim.name

targetsets2[[6]] <- highatt2$prelim.name

targetsets2[[7]] <- wealthset2

targetsets2[[8]] <- leaderset2

targetsets2[[9]] <- gateset2

targetsets2[[10]] <- eventset2

targetsets2[[11]] <- conservationset2

targetsets2[[12]] <- combisets2matrix[1,]

targetsets2[[13]] <- combisets2matrix[2,]

targetsets2[[14]] <- combisets2matrix[3,]

targetsets2[[15]] <- combisets2matrix[4,]

#sets of ten targets

targetsets10 = list()

targetsets10[[1]] <- kpindeg10$keyplayers

targetsets10[[2]] <- kpbet10$keyplayers

targetsets10[[3]] <- kpclose10$keyplayers

targetsets10[[4]] <- kpeigen10$keyplayers

targetsets10[[5]] <- lowatt10$prelim.name

targetsets10[[6]] <- highatt10$prelim.name

targetsets10[[7]] <- wealthset10

targetsets10[[8]] <- leaderset10

targetsets10[[9]] <- gateset10

targetsets10[[10]] <- eventset10

targetsets10[[11]] <- conservationset10

targetsets10[[12]] <- combisets10matrix[1,]

targetsets10[[13]] <- combisets10matrix[2,]

targetsets10[[14]] <- combisets10matrix[3,]

targetsets10[[15]] <- combisets10matrix[4,]

#sets of twenty targets

targetsets20 = list()

targetsets20[[1]] <- kpindeg20$keyplayers

targetsets20[[2]] <- kpbet20$keyplayers

targetsets20[[3]] <- kpclose20$keyplayers

targetsets20[[4]] <- kpeigen20$keyplayers

targetsets20[[5]] <- lowatt20$prelim.name

targetsets20[[6]] <- highatt20$prelim.name

targetsets20[[7]] <- wealthset20

targetsets20[[8]] <- leaderset20

targetsets20[[9]] <- gateset20

targetsets20[[10]] <- eventset20

targetsets20[[11]] <- conservationset20

targetsets20[[12]] <- combisets20matrix[1,]

targetsets20[[13]] <- combisets20matrix[2,]

targetsets20[[14]] <- combisets20matrix[3,]

targetsets20[[15]] <- combisets20matrix[4,]

#sets of thirty targets

targetsets30 = list()

targetsets30[[1]] <- kpindeg30$keyplayers

targetsets30[[2]] <- kpbet30$keyplayers

targetsets30[[3]] <- kpclose30$keyplayers

targetsets30[[4]] <- kpeigen30$keyplayers

targetsets30[[5]] <- lowatt30$prelim.name

targetsets30[[6]] <- highatt30$prelim.name

targetsets30[[7]] <- wealthset30

targetsets30[[8]] <- 0

targetsets30[[9]] <- gateset30

targetsets30[[10]] <- eventset30

targetsets30[[11]] <- 0

targetsets30[[12]] <- combisets30matrix[1,]

targetsets30 [[13]] <- combisets30matrix[2,]

targetsets30 [[14]] <- combisets30matrix[3,]

targetsets30 [[15]] <- combisets30matrix[4,]

###########Final List of lists

AllTargetSets <- list(targetsets2, targetsets10, targetsets20, targetsets30)

#######Generate random and random cluster sets

Randomset2 = list()

Randomset10 = list()

Randomset20 = list()

Randomset30 = list()

samplenumber <- sample(c(1:365), size = 30, replace=FALSE)

for(i in 1:30){

j <- samplenumber[i]

Randomset2[[i]] <- sample(c(1:365), size = 2, replace=FALSE)

Randomset10[[i]] <- sample(c(1:365), size = 10, replace=FALSE)

Randomset20[[i]] <- sample(c(1:365), size = 20, replace=FALSE)

Randomset30[[i]] <- sample(c(1:365), size = 30, replace=FALSE)

}

AllRandomSets = list()

AllRandomSets[[1]] <- Randomset2

AllRandomSets[[2]] <- Randomset10

AllRandomSets[[3]] <- Randomset20

AllRandomSets[[4]] <- Randomset30

##Cluster Sets

Clusterset2 = list()

Clusterset10 = list()

Clusterset20 = list()

Clusterset30 = list()

clustersets2 <- matrix(0, nrow=2, ncol=30)

clustersets10 <- matrix(0, nrow=10, ncol=30)

clustersets20 <- matrix(0, nrow=20, ncol=30)

clustersets30 <- matrix(0, nrow=30, ncol=30)

for(i in 1:30){

clustersets2[1,i] <- sample(c(1:365), size = 1, replace=TRUE)

clustersets10[1:5,i] <- sample(c(1:365), size = 5, replace=TRUE)

clustersets20[1:10,i] <- sample(c(1:365), size = 10, replace=TRUE)

clustersets30[1:15,i] <- sample(c(1:365), size = 15, replace=TRUE)

}

for(i in 1:30){

for(j in 1:15){

node <- clustersets30[j,i]

alter = vector(mode="numeric", length=6)

alter[1] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit1name))

alter[2] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit2name))

alter[3] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit3name))

alter[4] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit4name))

alter[5] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit5name))

alter[6] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit6name))

alter\_rmna <- alter[!is.na(alter)]

clustersets30[j+15,i] <- alter\_rmna[1]

}

for(j in 1:10){

node <- clustersets20[j,i]

alter = vector(mode="numeric", length=6)

alter[1] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit1name))

alter[2] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit2name))

alter[3] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit3name))

alter[4] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit4name))

alter[5] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit5name))

alter[6] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit6name))

alter\_rmna <- alter[!is.na(alter)]

clustersets20[j+10,i] <- alter\_rmna[1]

}

for(j in 1:5){

node <- clustersets10[j,i]

alter = vector(mode="numeric", length=6)

alter[1] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit1name))

alter[2] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit2name))

alter[3] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit3name))

alter[4] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit4name))

alter[5] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit5name))

alter[6] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit6name))

alter\_rmna <- alter[!is.na(alter)]

clustersets10[j+5,i] <- alter\_rmna[1]

}

node <- clustersets2[1,i]

alter = vector(mode="numeric", length=6)

alter[1] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit1name))

alter[2] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit2name))

alter[3] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit3name))

alter[4] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit4name))

alter[5] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit5name))

alter[6] <- as.numeric(subset(networkdata, prelim.name==node, prelim.visit6name))

alter\_rmna <- alter[!is.na(alter)]

clustersets2[2,i] <- alter\_rmna[1]

Clusterset2[[i]] <- clustersets2[,i]

Clusterset10[[i]] <- clustersets10[,i]

Clusterset20[[i]] <- clustersets20[,i]

Clusterset30[[i]] <- clustersets30[,i]

}

AllClusterSets = list()

AllClusterSets[[1]] <- Clusterset2

AllClusterSets[[2]] <- Clusterset10

AllClusterSets[[3]] <- Clusterset20

AllClusterSets[[4]] <- Clusterset30

####################2. Code for running diffusion simulations############################

#We now have three Lists of different target sets: 1) AllTargetSets, 2)AllRandomSets, and 3) #AllClusterSets. These will be passed through the simulation code separately. Make sure to ‘point’ to #these objects at the start, and save the outputs in different ‘newtable’ files at the end.

#############################################

#############Simulation model##################

#############################################

# Code adapted from Dobson et al. 2019, Phil. Trans.

# (a) threshvals - threshold number of knowledge-holding individuals to whom

# the recipient of information must be directly connected

# before 'listening' occurs

# (b) abprobvals - likelihood of 'listening' (and subsequently potentially

# passing on information)

library(network)

library(igraph)

theset<-as.data.frame(read.csv("theset.csv"))

nodes<-dim(theset)[1] #40

names(theset)[c(1,(nodes+2))]<-c("Node","Degree") #

for (i in 1:nodes){ #

names(theset)[i+1]<-paste("N",i,"\_link",sep="")} #

for (j in 1:nodes){ #

theset$Degree[j]<-sum(theset[j,c(2:(nodes+1))])} #

ttr<-20

threshvals<-c(1, 3)

abprobvals<-c(0.2, 0.8)

efforts <- c(2, 10, 20, 30)

strategies <- c("KP In-degree", "KP Betweenness", "KP Closeness", "KP Eigenvector",

"Low attitudes", "High attitudes", "Wealth", "Leaders", "Gatekeeper",

"Event", "Conservation","In-degree Cluster", "Betweenness Cluster", "Closeness Cluster",

"Eigenvector Cluster") #Use this for AllTargets

#strategies <- c(1:30) #Use this for the random sets

noreps<-20 # How many repetitions

dtable<-rep(0,(noreps+6))

rm(whoknows, iseed, alldegrees, allinfo, thelist, foundout, nodeorder)

for(eff in 1:length(AllTargetSets)){ #replace this with the target set list

for(strat in 1:length(AllTargetSets [[1]])){ #ditto

targetset <- AllTargetSets [[eff]][strat] #ditto

seeded <- length(targetset)

for (th in 1:length(threshvals)){

for (ab in 1:length(abprobvals)){

thresh<-threshvals[th]

abprob<-abprobvals[ab]

effo <- efforts[eff]

strategy <- strategies[strat]

seeds<-0

alldegrees<-matrix(rep(0,nodes\*noreps),nrow=nodes)

allinfo<-matrix(rep(0,(ttr+1)\*noreps),nrow=noreps) # +1 because time starts at 0

#rr<-1#

for (rr in 1:noreps){

whoknows<-as.data.frame(matrix(rep(0,ttr\*nodes),nrow=nodes))

nn<-rep(0,ttr)

for (i in 1:ttr){

nn[i]<-paste("t=",i,sep="")}

names(whoknows)<-nn

iseed<-rep(0,nodes) #modified by me to manually select seeds

toseed <- targetset[[1]]

iseed[toseed]<-1

seeds<-sum(iseed)

thelist=0

for (j in 1:ttr){

foundout<-rep(0,nodes)

nodeorder<-sample(1:nodes,nodes,replace=F)

for (i in nodeorder){

if(j==1){thelist<-iseed}else{thelist<-whoknows[,j-1]}

if(thelist[i]!=1){

#palswhoknow<-which((thelist==1) & as.double(theset[i,3:(nodes+2)])==1) #OLD

palswhoknow<-which((thelist==1) & as.double(theset[i,2:(nodes+1)])>0) #NEW #this is where it searches the network for pals who know

#numpals<-length(palswhoknow) #OLD

if(length(palswhoknow)!=0){

numpals<-sum(theset[i,2:(nodes+1)][palswhoknow]) #NEW

}else{

numpals<-0}

if (numpals>=thresh){

threshpals<-1+(numpals-thresh)

absorbprob=abprob\*threshpals

if (absorbprob>1){

absorbprob<-1

}

foundout[i]<-sample(c(0,1),1,prob=c((1-absorbprob),absorbprob),replace=F)

}else{foundout[i]<-0

}

}else{foundout[i]<-1

}

}

whoknows[,j]<-foundout

}

for (i in 1:nodes){

if (whoknows[i,1]<iseed[i]){print("Someone's forgotten something")}

}

for (j in 2:ttr){

for (i in 1:nodes){

if (whoknows[i,j]<whoknows[i,j-1]){print("Someone's forgotten something")}

}}

allinfo[rr,]<-colSums(cbind(iseed,whoknows))

#whoknows

thecount<-((th-1)\*length(abprobvals)\*noreps)+

((ab-1)\*noreps)+rr

print(paste("Rep number ",thecount," of ",length(threshvals)\*length(abprobvals)\*noreps,

" of strategy ",strat," of ",length(AllTargetSets[[eff]])," effort level ",eff," of 4 ", sep=""))

} #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ End of rr loop

integ<-rowSums(allinfo)

if(th==1 & ab==1 & eff==1 & strat==1){

dtable[1]<-effo->degreetable[1]

dtable[2]<-strategy->degreetable[2]

dtable[3]<-thresh->degreetable[3]

dtable[4]<-abprob->degreetable[4]

dtable[5:(noreps+4)]<-integ

}else{

dtable<-rbind(dtable,c(effo,strategy,thresh,abprob,integ,rep(0,2)))

}

} #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ End of ab loop

} #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ End of th loop

} #~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ End of eff loop

} # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ End of strat loop

for(jj in 1:dim(dtable)[1]){

dtable[jj,(noreps+5)]<-mean(as.numeric(dtable[jj,5:(noreps+4)]))

dtable[jj,(noreps+6)]<-sd(as.numeric(dtable[jj,5:(noreps+4)]))

}

repstring<-paste("AUC\_Rep",c(1:noreps),sep="\_")

colnames(dtable)<-c("Effort", "Strategy","Threshold","Listening\_prob",repstring,"Mean\_AUC","SD\_AUC")

write.csv(dtable,"dtable50.csv")

maxAUC <- 7300 #This is the size of the network \* number of time periods

dtable<-read.csv("dtable50.csv")[,-1] #modify the filename depending on which table

orig.cols<-dim(dtable)[2]

dtable<-cbind(dtable,rep(0,dim(dtable)[1]))

dtable<-cbind(dtable,rep(0,dim(dtable)[1]))

#Adjust dtable to remove the number of individuals to whom the info was seeded -

# otherwise this gives you potential max AUC greater than 2000 (i.e. >100%)

seeded <- as.numeric(as.character(dtable[,1]))

for(i in 1:length(dtable[,1])){

for (j in 5:(noreps+4)){

dtable[i,j]<-dtable[i,j]-seeded[i]

}

}

for (i in 1:dim(dtable)[1]){

dtable[i,(orig.cols+1)]<-mean(as.double(dtable[i,c(5:(noreps+4))]))

dtable[i,(orig.cols+2)]<-sd(as.double(dtable[i,c(5:(noreps+4))]))

}

bootsize<-1000 # for use with bootstrap

# get the zlims

zmin<-min(dtable[,c(5:(noreps+4))])

zmax<-max(dtable[,c(5:(noreps+4))])

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

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

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

newtable <- dtable #newtable to write results into

newtableorigcol <- dim(newtable)[2]

for(k in 1:length(strategies)){

strategy <- strategies[k]

# Now plot the AUC views, one for each th value

for (th in 1:2){

indat<-as.data.frame(dtable[which(dtable[,2]==strategy & dtable[,3]==threshvals[th]),])

names(indat)[1:4]<-c("ef","str","th","ab")

auc<-indat[,(noreps+7)]

aucSD<-rep(0,length(auc))

for (gg in 1:length(auc)){

aucSD[gg]<-sd(indat[gg,c(5:(noreps+4))])

}

# Need to transform the dataset to fit with the persp function requirements

efvals<-sort(unique(indat$ef))

abvals<-sort(unique(indat$ab))

AUC<-matrix(rep(0,length(efvals)\*length(abvals)),nrow=length(abvals))

for (i in 1:length(abvals)){

for (j in 1:length(efvals)){

AUC[i,j]<-auc[which(indat$ab==abvals[i] & indat$ef==efvals[j])]

}}

# Now generate data for errors

# bootstrapped 95%

AUC95upper<-matrix(rep(0,length(efvals)\*length(abvals)),nrow=length(abvals))

AUC95lower<-matrix(rep(0,length(efvals)\*length(abvals)),nrow=length(abvals))

for (i in 1:length(abvals)){

for(j in 1:length(efvals)){

dats<-indat[which(indat$ab==abvals[i] & indat$ef==efvals[j]),c(5:(noreps+4))]

booted<-sample(dats,bootsize,replace=T)

AUC95lower[i,j]<-as.double(sort(booted)[round(bootsize\*0.025)])

AUC95upper[i,j]<-as.double(sort(booted)[round(bootsize\*0.975)])

}

}

AUClowP<-(AUC95lower/maxAUC)\*100

AUCuppP<-(AUC95upper/maxAUC)\*100

AUC<-(AUC/maxAUC)\*100 # <--- change the max possible AUC

#Write the bootstrapped values back into newtable

for (i in 1:length(abvals)){

for(j in 1:length(efvals)){

newtable[which(newtable[,2]==strategy & newtable[,3]==threshvals[th]

& newtable[,1]==efvals[j] & newtable[,4]==abvals[i]),

newtableorigcol+1] <- AUC[i,j]

newtable[which(newtable[,2]==strategy & newtable[,3]==threshvals[th]

& newtable[,1]==efvals[j] & newtable[,4]==abvals[i]),

newtableorigcol+2] <- AUClowP[i,j]

newtable[which(newtable[,2]==strategy & newtable[,3]==threshvals[th]

& newtable[,1]==efvals[j] & newtable[,4]==abvals[i]),

newtableorigcol+3] <- AUCuppP[i,j]

}

}

} #<--------- end of th loop

} #<----------- end of K loop

names(newtable)[29:31] <- c("AUCResult", "AUCLow", "AUCUpp")

write.csv(newtable, "newtable.csv") ###Add a number after the filename for each set of targets

########################3. Code for plotting simulation results###############

################################################################################

################################################################################

################################################################################

################################################################################

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

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

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

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ PLOTTING ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

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

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

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

################################################################################

################################################################################

################################################################################

################################################################################

##############Presentation Tables#########

#Combine all the strategy tables

newtable1 <- read.csv("newtable-1.csv") #This is generated from AllTargetSets

for(i in 1:length(newtable1$Strategy)){ #assign a type

newtable1$type[i] <- ifelse(grepl("KP", newtable1$Strategy[i]),"Network", "Conventional")

newtable1$type[i] <- ifelse(grepl("Cluster", newtable1$Strategy[i]),"Network", "Conventional")

}

newtable2 <- read.csv("newtable-2.csv") #This is from the 30 random sets

orig.cols<-dim(newtable2)[2] #Add Random labels

newtable2$Strategy2 <- "Random"

newtable22$Strategy3 <- paste(newtable2$Strategy2, newtable2$Strategy)

newtable2$Strategy <- newtable2$Strategy3

newtable2 <- newtable2[,1:orig.cols]

newtable2$type <- "other"

newtable3 <- read.csv("newtable-3.csv") #This is from the 30 Random clusters

orig.cols<-dim(newtable3)[2] #Add …

newtable3$Strategy2 <- "Cluster"

newtable3$Strategy3 <- paste(newtable3$Strategy2, newtable3$Strategy)

newtable3$Strategy <- newtable3$Strategy3

newtable3 <- newtable3[,1:orig.cols]

newtable3$type <- "other"

ResultsTable <- rbind(newtable1, newtable2, newtable3)

ResultsTable <- ResultsTable[,c(2:5,30:33)]

write.csv(ResultsTable, "Outputs/ResultsTable.csv")

##Aggregate results for the null simulations

RandomTable <- matrix(rep(0,16\*8), nrow=16)

RandomTable <- as.data.frame(RandomTable)

names(RandomTable) <- c("Rank",names(ResultsTable)[c(1,3,4)],"Strategy", "Result", "Max", "Min")

RandomTable$Effort <- rep(c(2,10,20,30),4)

RandomTable$Threshold <- c(rep(1, 8), rep(3, 8))

RandomTable$Listening\_prob <- c(rep(0.2, 4), rep(0.8, 4),rep(0.2, 4), rep(0.8, 4))

ClusterTable <- RandomTable

for(lp in 1:2){

for(th in 1:2){

for(eff in 1:4){

listp <- abvals[lp]

thresh <- threshvals[th]

effort <- efvals[eff]

indat <-as.matrix(newtable3[which(newtable3$Effort==effort & newtable3$Threshold==thresh & newtable3$Listening\_prob==listp),c(6:25)])

mean <- mean(indat)

booted<-sample(indat,bootsize,replace=T)

AUC95lower<-as.double(sort(booted)[round(bootsize\*0.025)])

AUC95upper<-as.double(sort(booted)[round(bootsize\*0.975)])

ClusterTable$Min[which(ClusterTable$Effort==effort & ClusterTable$Threshold==thresh & ClusterTable$Listening\_prob==listp)]<-(AUC95lower/maxAUC)\*100

ClusterTable$Max[which(ClusterTable$Effort==effort & ClusterTable$Threshold==thresh & ClusterTable$Listening\_prob==listp)]<-(AUC95upper/maxAUC)\*100

ClusterTable$Result[which(ClusterTable$Effort==effort & ClusterTable$Threshold==thresh & ClusterTable$Listening\_prob==listp)]<-(mean/maxAUC)\*100 # <--- change the max possible AUC

indat <-as.matrix(newtable5[which(newtable5$Effort==effort & newtable5$Threshold==thresh & newtable5$Listening\_prob==listp),c(6:25)])

mean <- mean(indat)

booted<-sample(indat,bootsize,replace=T)

AUC95lower<-as.double(sort(booted)[round(bootsize\*0.025)])

AUC95upper<-as.double(sort(booted)[round(bootsize\*0.975)])

RandomTable$Min[which(RandomTable$Effort==effort & RandomTable$Threshold==thresh & RandomTable$Listening\_prob==listp)]<-(AUC95lower/maxAUC)\*100

RandomTable$Max[which(RandomTable$Effort==effort & RandomTable$Threshold==thresh & RandomTable$Listening\_prob==listp)]<-(AUC95upper/maxAUC)\*100

RandomTable$Result[which(RandomTable$Effort==effort & RandomTable$Threshold==thresh & RandomTable$Listening\_prob==listp)]<-(mean/maxAUC)\*100 # <--- change the max possible AUC

}

}

}

RandomTable$Strategy <- "Random"

ClusterTable$Strategy <- "Cluster"

########################################################################################## Plotting ##############################################

##################################################################################

library(tidyverse)

library(ggplot2)

# Load data and shorten some variable names

# NB Probably need to set your working directory first

d <- read\_csv("Outputs/ResultsTable.csv") %>%

select(-X1) %>%

rename(mn = AUCResult, lo = AUCLow, hi = AUCUpp) %>%

subset(mn>0)

#find the medians of the null results

medrandoms <- as.data.frame(matrix(rep(0,length(d)\*16), nrow=16))

names(medrandoms) <- names(d)

medclusters <- medrandoms

a <- 0

for(lp in c(0.2, 0.8)){

for(th in c(1,3)){

for(eff in c(2,10,20,30)){

a <- a+1

randoms <- d%>%

filter(Listening\_prob==lp & Threshold == th & Effort == eff & str\_detect(Strategy, "^Random"))

median <- median(randoms$mn)

medrandoms[a,] <- randoms[which.min(abs(randoms$mn- median)),]

clusters <- d%>%

filter(Listening\_prob==lp & Threshold == th & Effort == eff & str\_detect(Strategy, "^Cluster"))

median <- median(clusters$mn)

medclusters[a,] <- clusters[which.min(abs(clusters$mn- median)),]

}

}

}

medrandoms$Strategy <- "Random"

medclusters$Strategy <- "Cluster"

#Get the RandomTable & ClusterTable from the other script

medclusters <- merge(medclusters, ClusterTable, all=T)

medclusters$lo <- medclusters$Min

medclusters$hi <- medclusters$Max

medclusters <- medclusters[,1:14]

#add the median back in

d <- d %>%

filter(!str\_detect(Strategy, "^Cluster") & !str\_detect(Strategy, "^Random"))

d <- rbind(d, medrandoms,medclusters)

#add Max and Min from RandomTable

#if calculating costs, do so for Random table

NullTable <- RandomTable

NullTable$totalcost <- 0

NullTable[which(NullTable$Effort==2),]$totalcost <- 52 #These costs are manually determined

NullTable[which(NullTable$Effort==10),]$totalcost <- 146

NullTable[which(NullTable$Effort==20),]$totalcost <- 348

NullTable[which(NullTable$Effort==30),]$totalcost <- 502

NullTable$randomcostbenefit <- NullTable$Result/NullTable$totalcost

NullTable$maxrandomcost <- NullTable$Max/NullTable$totalcost

NullTable$minrandomcost <- NullTable$Min/NullTable$totalcost

dd <- merge(d, NullTable, by=c("Effort","Threshold","Listening\_prob"), all=T)

names(dd)[4] <- "Strategy"

#

plt1 <- dd %>%

# Filter out the subset of results to be plotted

# NB Just using Random 1 and Cluster 1 as examples rather than bothering to

# find the median here

filter(Listening\_prob==0.2 & !str\_detect(Strategy, "Random")) %>%

mutate(

# Order strategies to be Network then Conventional then Cluster, then other, and within

# each group to be ordered by the median AUC

Strategy = fct\_reorder(Strategy, ifelse(type=="Network", 1000 \* mn, ifelse(type=="Conventional", 100\*mn,

ifelse(type=="Cluster", 1\*mn, ifelse(type=="other", -100 \* mn, mn)))), .desc = T),

# Set up duplicate of Strategy variable - used later to add background lines

# in panels

Strat = Strategy,

# Give threshold variable more descriptive names

Threshold = ifelse(Threshold==3, "Complex (Threshold = 3)", "Simple (Threshold = 1)")

)

ggplot(plt1, aes(x = Effort, y = mn, ymin = lo, ymax = hi)) +

geom\_ribbon(aes(ymin=Min, ymax=Max, fill=Threshold, alpha=0.5)) +

# Add background lines for non-focal strategies

#geom\_line(data = transform(plt1, Strategy = NULL), alpha = 0.5, size = 0.1,

# aes(group = interaction(Strat, Threshold), colour = Threshold)) +

# Add foreground lines, points and error bars for focal strategy

geom\_line(aes(group = interaction(Strategy, Threshold), colour = Threshold)) +

geom\_point(aes(colour = Threshold)) +

geom\_linerange(aes(colour = Threshold)) +

# Use custom colours for plotting symbols

scale\_color\_manual(values = c("Simple (Threshold = 1)" = rgb(0.7,0.2,0.3,1),

"Complex (Threshold = 3)" = rgb(0.2,0.4,0.5,1))) +

scale\_fill\_manual(values = c("Simple (Threshold = 1)" = rgb(0.7,0.2,0.3,1),

"Complex (Threshold = 3)" = rgb(0.2,0.4,0.5,1)), guide=FALSE) +

guides(alpha=FALSE) +

facet\_wrap( ~ Strategy, ncol = 4) +

# Constrain y-axis since measured as a percent

ylim(c(0,100)) +

# Add labels

ylab("AUC") +

labs(color = "Type of behaviour", shape = "Type of approach",

title = "Listening probability = 0.2", alpha=NULL, fill=NULL) +

# Tweak appearance

theme\_bw() +

theme(

plot.title = element\_text(face = "bold", hjust = 0.5),

legend.position = "right",

panel.grid = element\_blank(),

strip.background = element\_blank(),

strip.text = element\_text(face = "bold", size = 10)

)

ggsave("strategy comparison w Nulls (lp=0.2).png", height = 225, width = 200,

units = "mm", scale = 1.2)

##repeat for higher listening prob

plt2 <- dd %>%

# Filter out the subset of results to be plotted

# NB Just using Random 1 and Cluster 1 as examples rather than bothering to

# find the median here

filter(Listening\_prob==0.8 & !str\_detect(Strategy, "Random")) %>%

mutate(

# Order strategies to be Network then Conventional then Cluster, then other, and within

# each group to be ordered by the median AUC

Strategy = fct\_reorder(Strategy, ifelse(type=="Network", 1000 \* mn, ifelse(type=="Conventional", 100\*mn,

ifelse(type=="Cluster", 1\*mn, ifelse(type=="other", -100 \* mn, mn)))), .desc = T),

# Set up duplicate of Strategy variable - used later to add background lines

# in panels

Strat = Strategy,

# Give threshold variable more descriptive names

Threshold = ifelse(Threshold==3, "Complex (Threshold = 3)", "Simple (Threshold = 1)")

)

ggplot(plt2, aes(x = Effort, y = mn, ymin = lo, ymax = hi)) +

geom\_ribbon(aes(ymin=Min, ymax=Max, fill=Threshold, alpha=0.5)) +

# Add background lines for non-focal strategies

#geom\_line(data = transform(plt1, Strategy = NULL), alpha = 0.5, size = 0.1,

# aes(group = interaction(Strat, Threshold), colour = Threshold)) +

# Add foreground lines, points and error bars for focal strategy

geom\_line(aes(group = interaction(Strategy, Threshold), colour = Threshold)) +

geom\_point(aes(colour = Threshold)) +

geom\_linerange(aes(colour = Threshold)) +

# Use custom colours for plotting symbols

scale\_color\_manual(values = c("Simple (Threshold = 1)" = rgb(0.7,0.2,0.3,1),

"Complex (Threshold = 3)" = rgb(0.2,0.4,0.5,1))) +

scale\_fill\_manual(values = c("Simple (Threshold = 1)" = rgb(0.7,0.2,0.3,1),

"Complex (Threshold = 3)" = rgb(0.2,0.4,0.5,1)), guide=FALSE) +

guides(alpha=FALSE) +

facet\_wrap( ~ Strategy, ncol = 4) +

# Constrain y-axis since measured as a percent

ylim(c(0,100)) +

# Add labels

ylab("AUC") +

labs(color = "Type of behaviour", shape = "Type of approach",

title = "Listening probability = 0.8", alpha=NULL, fill=NULL) +

# Tweak appearance

theme\_bw() +

theme(

plot.title = element\_text(face = "bold", hjust = 0.5),

legend.position ="right",

panel.grid = element\_blank(),

strip.background = element\_blank(),

strip.text = element\_text(face = "bold", size = 10)

)

ggsave("strategy comparison w Nulls (lp=0.8).png", height = 225, width = 200,

units = "mm", scale = 1.2)

###################################################

###########

plt5 <- dd %>%

# Filter out the subset of results to be plotted

filter(Listening\_prob==0.2 & Threshold == 3 & Effort==20) %>%

mutate(

# Order strategies to be Network then Conventional then Cluster, then other, and within

# each group to be ordered by the median AUC

Strategy = fct\_reorder(Strategy, ifelse(type=="Network", 1000 \* mn, ifelse(type=="Conventional", 100\*mn,

ifelse(type=="Cluster", 1\*mn, ifelse(type=="other", -100 \* mn, mn)))), .desc = T),

# Set up duplicate of Strategy variable - used later to add background lines

# in panels

Strat = Strategy,

# Give threshold variable more descriptive names

Threshold = ifelse(Threshold==3, "Complex (Threshold = 3)", "Simple (Threshold = 1)")

)

#Create the random null

plt5ran <- plt5 %>%

filter(Strategy=="Random")

plt5ran$lo <- plt5ran$Min

plt5ran$hi <- plt5ran$Max

#reinsert to plt5

plt5 <- filter(plt5, Strategy!="Random")

plt5 <- plt5 %>%

arrange(-mn)

plt5 <- rbind(plt5ran,plt5) %>%

mutate(Strategy = factor(Strategy, unique(Strategy)))

ggplot(plt5, aes(x=mn, y=Strategy, xmin=lo, xmax=hi))+

geom\_pointrange(position=position\_dodge(0.2)) +

geom\_vline(xintercept=plt5$mn[which(plt5$Strategy=="Random" & plt5$Effort==20)], linetype=2, colour="blue") +

xlab("AUC") +

theme\_bw() +

theme(

plot.title = element\_text(face = "bold", hjust = 0.5),

legend.position = c(0.85,0.6),

strip.background = element\_blank(),

strip.text = element\_text(face = "bold", size = 10)

)

ggsave("strategy AUC pointrange complex 20 (lp=0.2).png", height = 150, width = 150,

units = "mm", scale = 1.2)

###Interpolated plots

interpolateplot <- dd %>%

filter(Threshold==3 & Listening\_prob==0.2) %>%

select(Effort, Strategy, mn) %>%

pivot\_wider(names\_from=Strategy, values\_from=mn)

TargetAUC10 <- as.data.frame(names(interpolateplot)[-1])

for(i in 1:length(TargetAUC10[,1])){

TargetAUC10$Effort[i] <- approx(unlist(interpolateplot[1:4,i+1]),unlist(interpolateplot[1:4,1]), xout=10)$y

}

names(TargetAUC10) <- c("Strategy", "Effort")

TargetAUC10$Effort <- unlist(TargetAUC10$Effort)

TargetAUC10 <- arrange(TargetAUC10, Effort) %>%

mutate(Strategy = factor(Strategy, unique(Strategy)))

ggplot(TargetAUC10, aes(x=Strategy, y=Effort))+

geom\_point(stat="Identity") +

theme(axis.text.x=element\_text(angle=90))

TargetAUC15 <- as.data.frame(names(interpolateplot)[-1])

for(i in 1:length(TargetAUC15[,1])){

TargetAUC15$Effort[i] <- approx(unlist(interpolateplot[1:4,i+1]),unlist(interpolateplot[1:4,1]), xout=15)$y

}

names(TargetAUC15) <- c("Strategy", "Effort")

TargetAUC15$Effort <- unlist(TargetAUC15$Effort)

TargetAUC15 <- arrange(TargetAUC15, Effort) %>%

mutate(Strategy = factor(Strategy, unique(Strategy)))

ggplot(TargetAUC15, aes(x=Strategy, y=Effort))+

geom\_point(stat="Identity") +

theme(axis.text.x=element\_text(angle=90))

TargetAUC20 <- as.data.frame(names(interpolateplot)[-1])

for(i in 1:length(TargetAUC20[,1])){

TargetAUC20$Effort[i] <- approx(unlist(interpolateplot[1:4,i+1]),unlist(interpolateplot[1:4,1]), xout=20)$y

}

names(TargetAUC20) <- c("Strategy", "Effort")

TargetAUC20$Effort <- unlist(TargetAUC20$Effort)

TargetAUC20 <- arrange(TargetAUC20, Effort) %>%

mutate(Strategy = factor(Strategy, unique(Strategy)))

ggplot(TargetAUC20, aes(x=Strategy, y=Effort))+

geom\_point(stat="Identity") +

theme(axis.text.x=element\_text(angle=90))

#convert into costs

#intervention costs depends on effort

TargetAUC10$Cost <- 0

TargetAUC10$Cost[which(TargetAUC10$Effort<5)] <- 32

TargetAUC10$Cost[which(TargetAUC10$Effort>5 & TargetAUC10$Effort<15)] <- 56

TargetAUC10$Cost[which(TargetAUC10$Effort>15 & TargetAUC10$Effort <25)] <- 148

TargetAUC10$Cost[which(TargetAUC10$Effort>25)] <- 222

for(i in 1:length(TargetAUC10$Cost)){

TargetAUC10$Cost[i] <- TargetAUC10$Cost[i]+TargetAUC10$Effort[i]\*10

}

#add the data costs

for(i in c(1:3,5:8,13)){ #<- these numbers are manual

TargetAUC10$Cost[i] <- TargetAUC10$Cost[i]+5160

}

for(i in c(9,11,14)){

TargetAUC10$Cost[i] <- TargetAUC10$Cost[i]+2200

}

TargetAUC10 <- arrange(TargetAUC10, Cost) %>%

mutate(Strategy = factor(Strategy, unique(Strategy)))

TargetAUC15$Cost <- 0

TargetAUC15$Cost[which(TargetAUC15$Effort<15)] <- 56

TargetAUC15$Cost[which(TargetAUC15$Effort>15 & TargetAUC15$Effort <25)] <- 148

TargetAUC15$Cost[which(TargetAUC15$Effort>25)] <- 222

for(i in 1:length(TargetAUC15$Cost)){

TargetAUC15$Cost[i] <- TargetAUC15$Cost[i]+TargetAUC15$Effort[i]\*10

}

#add the data costs

for(i in c(1:3,5,6,8)){ #<- these numbers are manual

TargetAUC15$Cost[i] <- TargetAUC15$Cost[i]+5160

}

for(i in c(7)){

TargetAUC15$Cost[i] <- TargetAUC15$Cost[i]+2200

}

TargetAUC15 <- arrange(TargetAUC15, Cost) %>%

mutate(Strategy = factor(Strategy, unique(Strategy)))

ggplot(TargetAUC15, aes(x=Strategy, y=Cost))+

geom\_point(stat="Identity") +

theme(axis.text.x=element\_text(angle=90)) +

labs(title="Cost to achieve AUC = 10%",y="Cost (USD)")

ggsave("Cost to achieve AUC 10.png", height=150, width=200, units="mm", scale=1.2)

TargetAUC20$Cost <- 0

TargetAUC20$Cost[which(TargetAUC20$Effort<15)] <- 56

TargetAUC20$Cost[which(TargetAUC20$Effort>15 & TargetAUC20$Effort <25)] <- 148

TargetAUC20$Cost[which(TargetAUC20$Effort>25)] <- 222

for(i in 1:length(TargetAUC20$Cost)){

TargetAUC20$Cost[i] <- TargetAUC20$Cost[i]+TargetAUC20$Effort[i]\*10

}

#add the data costs

for(i in c(1:4)){ #<- these numbers are manual

TargetAUC20$Cost[i] <- TargetAUC20$Cost[i]+5160

}

#for(i in c(6)){

# TargetAUC20$Cost[i] <- TargetAUC20$Cost[i]+2200

#}

TargetAUC20 <- arrange(TargetAUC20, Cost) %>%

mutate(Strategy = factor(Strategy, unique(Strategy)))

ggplot(TargetAUC20, aes(x=Strategy, y=Cost))+

geom\_point(stat="Identity") +

theme(axis.text.x=element\_text(angle=90)) +

labs(title="Cost to achieve AUC = 10%",y="Cost (USD)")

ggsave("Cost to achieve AUC 10.png", height=150, width=200, units="mm", scale=1.2)

###########4. Code for similarity indices and logistic regressions##################

#effort level 10

SimilarityMatrix10 <- matrix(0, nrow=12, ncol=12)

Sets10 <- list()

for(i in 1:12){

Sets10[[i]] <- as.set(targetsets10[[i]])

}

for(i in 1:12){

for(j in 1:12){

SimilarityMatrix10[i,j] <- set\_similarity(Sets10[[i]],Sets10[[j]], method="Jaccard")

}

}

rownames(SimilarityMatrix10) <- c("KP In-degree", "KP Betweenness", "KP Closeness", "KP Eigenvector",

"Low attitudes", "High attitudes", "Wealth", "Leaders", "Gatekeeper",

"Event", "Conservation", "General")

colnames(SimilarityMatrix10) <- c("KP In-degree", "KP Betweenness", "KP Closeness", "KP Eigenvector",

"Low attitudes", "High attitudes", "Wealth", "Leaders", "Gatekeeper",

"Event", "Conservation", "General")

write.csv(SimilarityMatrix10, "Similaritymatrix10.csv")

# effort level 20

SimilarityMatrix20 <- matrix(0, nrow=12, ncol=12)

Sets20 <- list()

for(i in 1:12){

Sets20[[i]] <- as.set(targetsets20[[i]])

}

for(i in 1:12){

for(j in 1:12){

SimilarityMatrix20[i,j] <- set\_similarity(Sets20[[i]],Sets20[[j]], method="Jaccard")

}

}

rownames(SimilarityMatrix20) <- c("KP In-degree", "KP Betweenness", "KP Closeness", "KP Eigenvector",

"Low attitudes", "High attitudes", "Wealth", "Leaders", "Gatekeeper",

"Event", "Conservation", "General")

colnames(SimilarityMatrix20) <- c("KP In-degree", "KP Betweenness", "KP Closeness", "KP Eigenvector",

"Low attitudes", "High attitudes", "Wealth", "Leaders", "Gatekeeper",

"Event", "Conservation", "General")

write.csv(SimilarityMatrix20, "Similaritymatrix20.csv")

#effort level 30

SimilarityMatrix30 <- matrix(0, nrow=12, ncol=12)

Sets30 <- list()

for(i in 1:12){

Sets30[[i]] <- as.set(targetsets30[[i]])

}

for(i in 1:12){

for(j in 1:12){

SimilarityMatrix30[i,j] <- set\_similarity(Sets30[[i]],Sets30[[j]], method="Jaccard")

}

}

rownames(SimilarityMatrix30) <- c("KP In-degree", "KP Betweenness", "KP Closeness", "KP Eigenvector",

"Low attitudes", "High attitudes", "Wealth", "Leaders", "Gatekeeper",

"Event", "Conservation", "General")

colnames(SimilarityMatrix30) <- c("KP In-degree", "KP Betweenness", "KP Closeness", "KP Eigenvector",

"Low attitudes", "High attitudes", "Wealth", "Leaders", "Gatekeeper",

"Event", "Conservation", "General")

write.csv(SimilarityMatrix30, "Similaritymatrix30.csv")

##Heatmap

matrix10 <- read.csv("Similaritymatrix10.csv")

rownames(matrix10) <- matrix10[,1]

matrix10 <- matrix10[-12,-c(1,13)]

matrix10 <- as.matrix(matrix10)

heatmap(matrix10, Rowv=NA, Colv=NA)

matrix20 <- read.csv("Similaritymatrix20.csv")

rownames(matrix20) <- matrix20[,1]

matrix20 <- matrix20[-12,-c(1,13)]

matrix20 <- as.matrix(matrix20)

heatmap(matrix20, Rowv=NA, Colv=NA)

matrix30 <- read.csv("Similaritymatrix30.csv")

rownames(matrix30) <- matrix30[,1]

matrix30 <- matrix30[-c(8,11,12),-c(1,9,12,13)]

matrix30 <- as.matrix(matrix30)

jpeg(filename="simHeatmaps10.jpg", width=220, height=250, units="mm", res=72)

par(oma=c(4,1,1,4), mar=c(5,2,1,2))

heatmap(matrix10, Rowv=NA, Colv=NA, verbose=T)

dev.off()

jpeg(filename="simHeatmaps20.jpg", width=220, height=250, units="mm", res=72)

par(oma=c(4,1,1,4), mar=c(5,2,1,2))

heatmap(matrix20, Rowv=NA, Colv=NA, verbose=T)

dev.off()

jpeg(filename="simHeatmaps30.jpg", width=220, height=250, units="mm", res=72)

par(oma=c(4,1,1,4), mar=c(5,2,1,2))

heatmap(matrix30, Rowv=NA, Colv=NA, verbose=T)

dev.off()

############

#Logistic regressions

attributes <- data.frame(Age = rep(0, 365), Gender = rep(0, 365), Wealth = rep(0, 365),

Positions = rep(0, 365), Indeg = rep(0, 365), Between = rep(0, 365),

Close = rep(0, 365), Eigen = rep(0, 365))

vertex\_attr(graphs[[2]])

attributes$Age <- (vertex\_attr(graphs[[2]], "age")/sd(vertex\_attr(graphs[[2]], "age")))

attributes$Gender <- vertex\_attr(graphs[[2]], "gender")

attributes$Wealth <- vertex\_attr(graphs[[2]], "wealth")

attributes$Positions <- vertex\_attr(graphs[[2]], "positions")

attributes$Indeg <- vertex\_attr(graphs[[2]], "indegree")

attributes$Between <- vertex\_attr(graphs[[2]], "betweennness")

attributes$Close <- vertex\_attr(graphs[[2]], "closeness")

attributes$Eigen <- vertex\_attr(graphs[[2]], "eigenvector")

variables <- c("Intercept","Age", "Gender", "Wealth", "Positions")

#In-degree

indeglogit <- glm(Indeg ~ Age + Gender + Wealth + Positions, data=attributes, family="binomial")

summary(indeglogit)

indegdata <- as.data.frame(indeglogit$coefficients)

intercept <- indegdata[1,]

indegdata$variables <- variables

indegdata <- cbind(indegdata, confint(indeglogit), rep("In-degree", 5))

names(indegdata) <- c("coefficients","variables","min","max","measure")

indegdata <- indegdata[2:4,]

indegdata$coefficients <- indegdata$coefficients+intercept

indegdata$min <- indegdata$min+intercept

indegdata$max <- indegdata$max+intercept

#Betweenness

betweenlogit <- glm(Between ~ Age + Gender + Wealth + Positions, data=attributes, family="binomial")

summary(betweenlogit)

betweendata <- as.data.frame(betweenlogit$coefficients)

intercept <- betweendata[1,]

betweendata$variables <- variables

betweendata <- cbind(betweendata, confint(betweenlogit), rep("Betweenness", 5))

names(betweendata) <- c("coefficients","variables","min","max","measure")

betweendata <- betweendata[2:4,]

betweendata$coefficients <- betweendata$coefficients+intercept

betweendata$min <- betweendata$min+intercept

betweendata$max <- betweendata$max+intercept

#Closeness

closelogit <- glm(Close ~ Age + Gender + Wealth + Positions, data=attributes, family="binomial")

summary(closelogit)

closedata <- as.data.frame(closelogit$coefficients)

intercept <- closedata[1,]

closedata$variables <- variables

closedata <- cbind(closedata, confint(closelogit), rep("Closeness", 5))

names(closedata) <- c("coefficients","variables","min","max","measure")

closedata <- closedata[2:5,]

closedata$coefficients <- closedata$coefficients+intercept

closedata$min <- closedata$min+intercept

closedata$max <- closedata$max+intercept

#Eigenvector

eigenlogit <- glm(Eigen ~ Age + Gender + Wealth + Positions, data=attributes, family="binomial")

summary(eigenlogit)

eigendata <- as.data.frame(eigenlogit$coefficients)

intercept <- eigendata[1,]

eigendata$variables <- variables

eigendata <- cbind(eigendata, confint(eigenlogit), rep("Eigenvector", 5))

names(eigendata) <- c("coefficients","variables","min","max","measure")

eigendata <- eigendata[2:5,]

eigendata$coefficients <- eigendata$coefficients+intercept

eigendata$min <- eigendata$min+intercept

eigendata$max <- eigendata$max+intercept

##Plotting

logitdata <- rbind(indegdata, betweendata, closedata, eigendata)

ggplot(logitdata, aes(x=coefficients, y=variables))+

geom\_dotplot(binaxis='y', stackdir='center',dotsize=0.4) +

geom\_errorbarh(aes(xmin=min, xmax=max), height=0) +

facet\_wrap(~measure) +

geom\_vline(data=filter(logitdata, measure=="In-degree"), aes(xintercept=indeglogit$coefficients[1]), linetype="dashed") +

geom\_vline(data=filter(logitdata, measure=="Betweenness"), aes(xintercept=betweenlogit$coefficients[1]), linetype="dashed") +

geom\_vline(data=filter(logitdata, measure=="Closeness"), aes(xintercept=closelogit$coefficients[1]), linetype="dashed") +

geom\_vline(data=filter(logitdata, measure=="Eigenvector"), aes(xintercept=eigenlogit$coefficients[1]), linetype="dashed")