### Table 1: Modelling approaches used in the analyses.

<table>
<thead>
<tr>
<th>Serial interval derived from observed data on onset dates from hospital patients during outbreaks (N=149)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analysis 1</strong>: (main analysis in results) Include only outbreaks with complete data on patient position at time of onset. (N=65)</td>
</tr>
<tr>
<td><strong>Sensitivity analyses</strong></td>
</tr>
<tr>
<td>Analysis 2: Include outbreaks with missing values on patient position at time of onset. Patients allocated to distinct bays (N=85)</td>
</tr>
<tr>
<td>Analysis 3: Include outbreaks with missing values on patient position at time of onset, but exclude observations with missing values on patient position, i.e. patients deleted from dataset (N=85)</td>
</tr>
<tr>
<td>Analysis 4: Remove all information on patient position and randomly allocate bays</td>
</tr>
<tr>
<td>Using gamma distributions of serial intervals from Heijne et al (From healthy population)</td>
</tr>
</tbody>
</table>

Figure 1_supplement. Comparison of results from modelling approaches, main analysis and sensitivity analyses.
R. Code.

This first part of the program creates functions that will be called in the main part of the program later on.

```r
# create the distance matrix function which codes 1 for patients sharing a bay and 0 for those in different bays
makeDistancematrix <- function(pid, bay) {
  n <- length(pid)
  Dist <- matrix(data = 0, nrow = n, ncol = n)
  for (i in 1:n) {
    for (j in 1:n) {
      if (i != j & bay[i] == bay[j]) Dist[i, j] <- 1
    }
  }
  return(Dist)
}

# Create the simulated outbreak function this simulates one outbreak
sim1outbreak <- function(pkl, onset.times.percase) {
  n <- dim(pkl)[1]
  number.of.possible.index.cases <- sum(onset.times.percase == min(onset.times.percase))
  indexcase <- 1 + as.integer((number.of.possible.index.cases) * runif(1))
  whoinfectswhom <- rep(NA, n)
  for (i in 1:n) {
    probnsource = max(1 - sum(pkl[i, ]), 0)
    source <- which(rmultinom(1, 1, c(pkl[i, ], probnsource)) == 1)
    if (source == n + 1) source <- 0
    whoinfectswhom[i] <- source
    if (i == indexcase) whoinfectswhom[i] <- 0 # i.e. infected outside
  }
  return(whoinfectswhom)
}

# Create the function to randomise the bays in which patients are located
permutebays <- function(bays) {
  n <- length(bays)
  return(bays[order(runif(n))])
}

# Create the function to calculate the infection trees
calcOutbreakDist <- function(whoinfectswhom, Dist) {
  obDist = 0
  for (i in 1:length(whoinfectswhom)) {
    if (whoinfectswhom[i] != 0) {
      obDist = obDist + Dist[i, whoinfectswhom[i]]
    }
  }
  return(obDist)
}
```

# This section is the main program, it begins by reading in the data “data.csv” which is the comma delimited data file containing the data.
```r
dataset <- read.table("data.csv", header = T, sep = ",")
```
# To ensure that the dates in the CSV file are interpreted as dates by R

dataset$dateonset <- as.Date(as.character(dataset$dateonset), "%d/%m/%Y")

S <- 1000  # Note this sets the number of simulated distance matrices to create

numobks <- length(unique(dataset$obnumb))  # To calculate cumulative distance over all simulated outbreak trees

reallcumulativeDist <- rep(0, numobks)  # To calculate cumulative distance over all simulated outbreak trees

permutedDistances <- matrix(data = rep(0, S * numobks), nrow = numobks, ncol = S)

for (obnumb in 1: numobks){
  outbreak.id <- obnumb
  onset.times.percase <- dataset$dateonset[which(dataset$obnumb == outbreak.id)]
  unique.onset.times.percase <- unique(onset.times.percase)
  unique.onset.times.percase <- 1 + unique.onset.times.percase - min(unique.onset.times.percase)
  N <- length(onset.times.percase)
  maxtimediff <- max(onset.times.percase) - min(onset.times.percase)
  w <- rep(NA, maxtimediff + 1)  # Probability that the serial is i - 1 days
  w[1] <- 0.6377
  w[2] <- 0.1678
  w[3] <- 0.1074
  w[4] <- 0.0403
  w[5] <- 0.0000
  w[6] <- 0.0134
  w[7] <- 0.0000
  w[8] <- 0.0201
  w[9] <- 0.0000
  w[10] <- 0.0067
  w[11] <- 0.0000
  w[12] <- 0.0000
  w[13] <- 0.0000
  w[14] <- 0.0000
  w[15] <- 0.0000
  w[16] <- 0.0000
  w[17] <- 0.0000
  w[18] <- 0.0067
  w[19] <- 0.0000
  w[20] <- 0.0000
  w[21] <- 0.0000
  w[22] <- 0.0000
  w[23] <- 0.0000
  w[24] <- 0.0000
  w[25] <- 0.0000
  w[26] <- 0.0000
  w[27] <- 0.0000
  w[28] <- 0.0000
  w[29] <- 0.0000
  w[30] <- 0.0000
  wij <- matrix(rep(0, N^2), nrow = N)

  number.of.possible.index.cases <- sum(onset.times.percase == min(onset.times.percase))
  for (i in 1:N)
    j <- 1
    onissettime <- onset.times.percase[j]
    while (onissettime <= onset.times.percase[i] & & j <= N)
      timediff <- onset.times.percase[i] - onset.times.percase[j]
      if (i != j) {
        if (i > number.of.possible.index.cases) {
          ...
wij[i,j] <- w[time+1]
} else {
    wij[i,j] <- w[time+1]*(number.of.possible.index.cases-1)/(number.of.possible.index.cases)
}
}
j <- j+1
onsettime <- onset.times.percase[j]
}

# denoms is the sum of likelihoods that person i was infected by person i-1,i-2,....,1
denoms <- c(NULL,rep(0,N-1))
for(i in 1:N){
    denomfori <- 0
    for(j in 1:N) {
        denomfori <- denomfori + wij[i,j]
    }
    denoms[i] <- denomfori
}
pkl <- matrix(rep(0,N^2),nrow=N)  # relative likelihood that k was infected by l
for(l in 1:N){
    # print(l)
    for(k in 1:N){
        numerator <- wij[k,l]
        denom <- denoms[k]
        if(denom>0){
            pkl[k,l] <- numerator/denom
        } else {
            pkl[k,l] <- 0
        }  
    }  
}
for(i in 1:number.of.possible.index.cases) pkl[i,] <- pkl[i,]*(number.of.possible.index.cases-1)/number.of.possible.index.cases

pid <- dataset$caseid[which(dataset$obnumb==outbreak.id)]  # This limits the variable pid to one outbreak
bay <- dataset$sbay[which(dataset$obnumb==outbreak.id)]  # This limits the variable bay to one outbreak

Dist <- makeDistancematrix(pid, bay)
for (i in 1:S) {
    oneoutbreaksim <- sim1outbreak(pkl, onset.times.percase)  # a possible reconstruction of outbreak who infects whom.
    realDist <- calcOutbreakDist(oneoutbreaksim, Dist)
    realcumulativeDist[obnumb] <- realcumulativeDist[obnumb] + realDist
    permD <- makeDistancematrix(pid, permutebays(bay))  # this is for the permuted bays
    DistwithpermutedDistancematrix <- calcOutbreakDist(oneoutbreaksim, permD)
    permutedDistances[obnumb,i] <- permutedDistances[obnumb,i] + DistwithpermutedDistancematrix
}  # end for i in 1:S
realcumulativeDist[obnumb]<-realcumulativeDist[obnumb]/S
permutedDistancesAllOutbreaks<-colSums(permutedDistances)
test.statistic<-sum(realcumulativeDist)  # for all outbreaks
pvalue.2sided<-2*min(sum(permutedDistancesAllOutbreaks<=test.statistic)/length(permutedDistancesAllOutbreaks),
sum(permutedDistancesAllOutbreaks>=test.statistic)/length(permutedDistancesAllOutbreaks))
}

# Note this section can be used instead of the observed serial interval distribution if using the gamma
distribution estimate of serial intervals

w[1]<-pgamma(0.5,shape=shape,scale=scale) # probability of a serial interval recorded as 0 days (i.e.
# 0 to .5 days)
for(i in 1:maxtimediff) w[i+1]<-pgamma(i+0.5,shape=shape,scale=scale)-sum(w[1:i])

# Note this section can be used instead of the observed serial interval distribution if using the gamma
distribution estimate of serial intervals
Sample data.

<table>
<thead>
<tr>
<th>obnumb</th>
<th>dateonset</th>
<th>Bed number</th>
<th>Bed number</th>
<th>bay</th>
<th>caseid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22/12/2007</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>25/12/2007</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>25/12/2007</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>25/12/2007</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>26/12/2007</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>26/12/2007</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>27/12/2007</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>28/12/2007</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>