Risk_model_Code_and_description

Courtney Schreiner

2024-10-08

Step by step for creating risk model code

For this document I hope to go through my code for the risk model and double check that each piece does what it is supposed to do. The main document has functions first and then parameter declaration and then running the simulations. For this document I will go in order of the pieces needed, so I will start withe the community level model, then to adjacency matrix, transition, and then lastly all the code need to solve the system of equations

```
library(deSolve)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
##
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
##
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
##
## The following object is masked from 'package:tidyr':
```

```
##
##     crossing
##
## The following object is masked from 'package:tibble':
##
##     as_data_frame
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

```
library(ggnetwork)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(intergraph)
library(ggthemes)
library(paletteer)
library(pals)
library(ggrepel)
library(ggpubr)
```

Community model parameters

```
community_pop <- 100000 #pop size of the larger community
init_conds <- c(S = community_pop-1, I = 1, R = 0)
print(init_conds)
```

```
##     S     I     R
## 99999     1     0
```

```
parms <- data.frame(bet =0.0000035 , gam = 1/21)
print(parms)
```

```
##       bet        gam
## 1 3.5e-06 0.04761905
```

```
times <- seq(from = 1, to = 144, by = 1)
print(times)
```

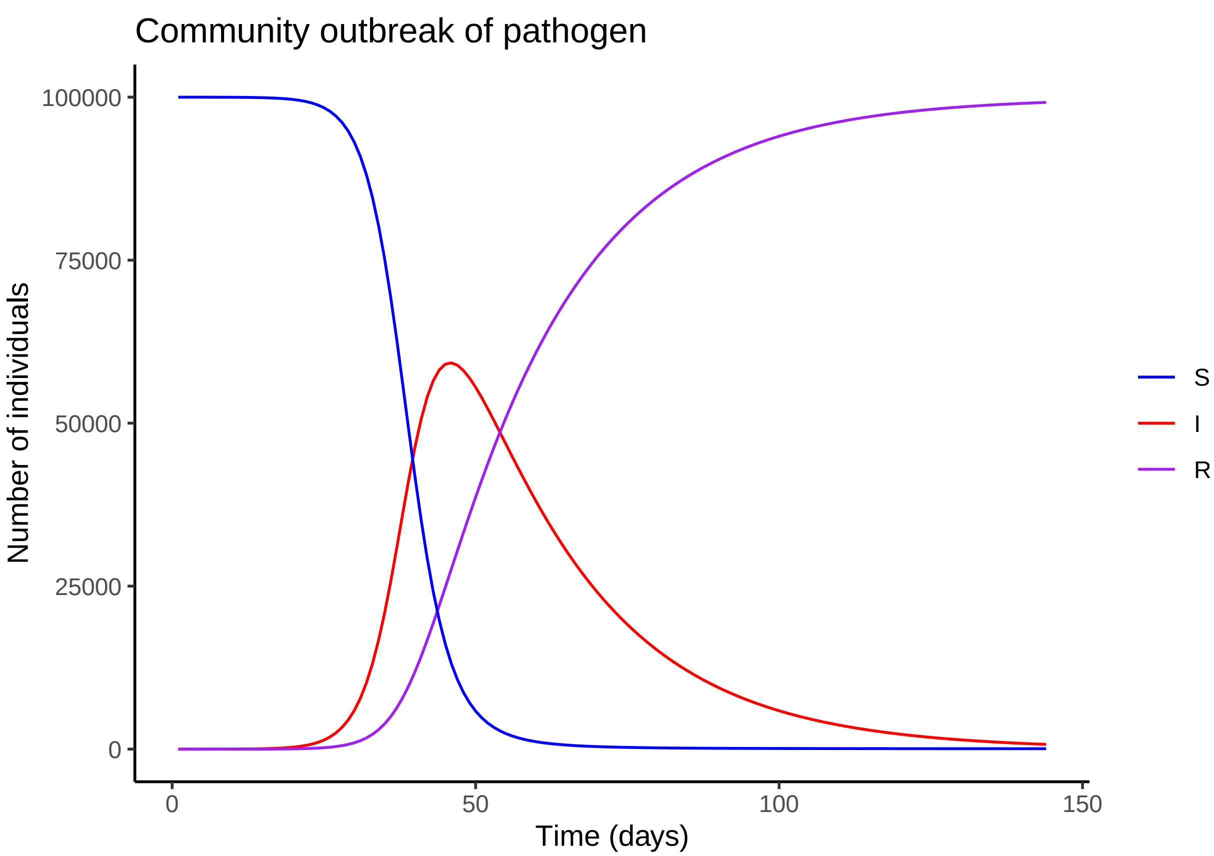
```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
```

```
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
```

Community modelfunction/equations

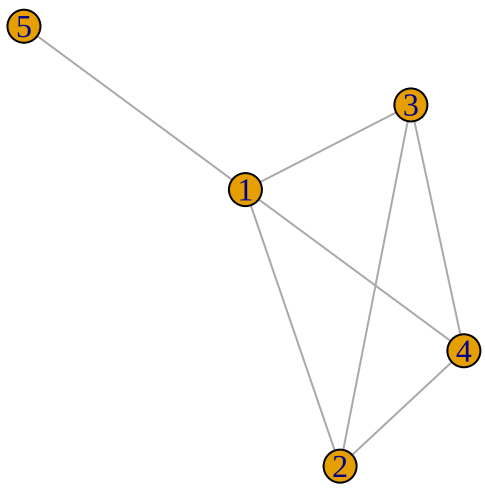
```
SIR_community_model <- function(t, x, parms) {
  S <- x[1]
  I <- x[2]
  R <- x[3]

  with(parms, {
    dS <- -bet*S*I
    dI <- bet*S*I - gam*I
    dR <- gam*I

    dt <- c(dS,dI,dR)
    return(list(dt))
  })}
```

Community model output



Make building

```r
church_adjacency_matrix <- matrix(c(c(0,1,1,1,1),
                                    c(1,0,1,1,0),
                                    c(1,1,0,1,0),
                                    c(1,1,1,0,0),
                                    c(1,0,0,0,0)),nrow=5, ncol = 5)
church_graph<- graph_from_adjacency_matrix(church_adjacency_matrix, mode = "undirected")
#take a look
plot(church_graph)
```




```r
graph_to_use <- church_graph
adjacency_matrix_to_use <- church_adjacency_matrix
N_rooms <- ncol(adjacency_matrix_to_use) #number of rooms
Church_C<- c(200, #column 1 - main area / Hallway
       100, # column 2 - Hallway
       100, # column 3 - Hallway
       400, #column 4 Main room
       200
)

length(Church_C)
```

```
## [1] 5
```

Now that we have building defined by the adjacency matrix, we need to specify the conditions in the building

```r
Max_Building_Capacity <- sum(Church_C)
Prop_full <- 0.8 # how full do we want our building capacity to be
Adj_Max_Building_Capacity <- Prop_full*Max_Building_Capacity #Adjusted capacity - number of individuals
delt <- Adj_Max_Building_Capacity/community_pop # proportionality constant ( what proportion of individ
```

Next, we need to put people in our building. These will be our initial conditions for the building level model.

```r
day <- 31 #What day to extract results from the community level model
N_rooms <- length(Church_C)
Bld_setup_func <- function(Community_output,day, delt,N_rooms){
  Building_ICs <- Community_output[day,] #Retrieves the number of S, I, and R individuals in the commun
```

```r
  #S_c + I_c + R_c = N_c | sum of the number of susceptible, infectious and recovered in the community
  # we want the proportion of S, I, and R in the community to match the proportion of S, I, and R in th
  # (S_c + I_c + R_c)*delt = building_pop
  # we set building_pop based on how full we want the building to be: this is our adjusted_building cap
  #i.e. Prop_Full*Building_Max_Capacity= Adjusted_building_capacity.
  # then building_pop = Adjusted_building_capacity so that we can get our desired amount of individuals
  # and then we solve for delt above: delt = adjusted_building_pop/community_pop

  Sb <- Community_output$S[day]*delt # number of susceptible in building
  Ib <- Community_output$I[day]*delt # number of Infectious in building
  Rb <- Community_output$R[day]*delt # number of Recovered in building

  #Sb, Ib, Rb are the number of Susceptible, Infected and Recovered individuals that will be in the bui

  #initial conditions for each room. Randomly distribute individuals throughout rooms

  # first assign a random number between 0 and 1 for each room,
  #broken up by S, I, and R
  S_x <- c(runif(N_rooms, min = 0, max = 1))
  I_x <- c(runif(N_rooms, min = 0, max = 1))
  R_x <- c(runif(N_rooms, min = 0, max = 1))

  #normalize and then assign the correct number of S, I, and R based on Sb, Ib, and Rb
  S_x <- ((S_x/sum(S_x))*Sb)
  I_x <- ((I_x/sum(I_x))*Ib)
  R_x <- ((R_x/sum(R_x))*Rb)


  S_prop <- ((S_x/sum(S_x))*Sb)/(Sb+Ib+Rb)
  I_prop <- ((I_x/sum(I_x))*Ib)/(Sb+Ib+Rb)
  R_prop <- ((R_x/sum(R_x))*Rb)/(Sb+Ib+Rb)

  #this model is of the proportion of S, I and R in each room so now we need to switch from numbers to 
  # but we need to keep the total number of individuals in each room since that is in our model
  Init_conds_nums <- data.frame(S_num = S_x, I_num = I_x, R_num = R_x)
  Init_conds_nums <- Init_conds_nums %>% mutate(N_x = S_num + I_num + R_num)
  Init_conds_props <- Init_conds_nums %>%
    mutate(S_prop = S_num/N_x, I_prop = I_num/N_x, R_prop = R_num/N_x)%>%
    select(S_prop,I_prop,R_prop,N_x)

  # we start with no particles in the building
  P_x <- c(rep(0,N_rooms))

  Init_conds_props <- Init_conds_props %>% mutate(S_prop = S_prop,I_prop =I_prop,R_prop= R_prop, P_x =P
  #Init_conds <-c(S=S_x/sum(Sb+Ib+Rb), I = I_x/sum(Sb+Ib+Rb), R = R_x/sum(Sb+Ib+Rb), P = P_x)

  return(data.frame(S=S_prop,I=I_prop,R=R_prop, P=P_x, N_x = S_x+I_x+R_x))
}

Church_setup <- Bld_setup_func(Community_output = Community_output,day = day,delt = delt,N_rooms =N_roo
```

Checking initial conditions Now lets check that our building setup function did what I wanted it to. Each room's proportion should sum to 1. And if we convert back to numbers using N__x the numbers should equal

```
#Church_setup %>% mutate(total_prop = S_prop+I_prop+R_prop)
```

props sum to 1 so that is good

```
#prop of susceptible in building
sum(Church_setup$S_prop*Church_setup$N_x)/sum(Church_setup$N_x)
```

```
## [1] 0
```

```
#prop of susceptible in community
Community_output$S[day]/(Community_output$S[day]+Community_output$I[day]+Community_output$R[day])
```

```
## [1] 0.909126
```

```
#prop of infectious in building
sum(Church_setup$I_prop*Church_setup$N_x)/sum(Church_setup$N_x)
```

```
## [1] 0
```

```
#prop of infectious in community
Community_output$I[day]/(Community_output$S[day]+Community_output$I[day]+Community_output$R[day])
```

```
## [1] 0.07791328
```

```
#prop of recovered in building
sum(Church_setup$R_prop*Church_setup$N_x)/sum(Church_setup$N_x)
```

```
## [1] 0
```

```
#prop of susceptible in community
Community_output$R[day]/(Community_output$S[day]+Community_output$I[day]+Community_output$R[day])
```

```
## [1] 0.01296077
```

Okay, great all of that checks out

Now people and particles need to move (Transition matrices)

```
Create_T_Matrix <-function(adjacency_matrix_to_use){
  #set.seed(123145) # <- easier for debugging
  T_mov <- data.frame(matrix(runif(N_rooms^2), nrow = N_rooms)) #populates a square matrix/dataframe wi
  diag(T_mov) <- 0 #set diagonal to 0
  T_mov <- adjacency_matrix_to_use*T_mov #restrict the movement according to our network/adjacency matr
  T_mov_norm <- t(apply(T_mov, 1, function(x) x / sum(x))) # normalize so that there aren't more people
  T_mov <-T_mov_norm
  return(T_mov)
}

Church_T_mov <- Create_T_Matrix(adjacency_matrix_to_use = adjacency_matrix_to_use)
sum(Church_T_mov[1,])
```

```
## [1] 1
```

```
sum(Church_T_mov[2,])
```

```
## [1] 1
```

```
sum(Church_T_mov[3,])
```

```
## [1] 1
```

```
sum(Church_T_mov[4,])
```

```
## [1] 1
```

```
sum(Church_T_mov[5,])
```

```
## [1] 1
```

```
# We can use the same function for the particle matrix
Church_theta_mov <- Create_T_Matrix(adjacency_matrix_to_use = adjacency_matrix_to_use)
```

Now set parameters for the model

```
parms <-data.frame(s=100,a=5, d=3,lam = 1)
# s = shedding, a = absorption, d = decay, lam = scalar for room capacities

Maxtime <- 24*3
times <- seq(from = 0, to = Maxtime, by = 0.2)
m <-5 #number of equations per room
```

For the model we are going to have to sum over some of the vectors/matrices so we will put those steps in functions. This is needed for the movement of individuals and particles. Particles and people will have to have a different function because there re capacities on rooms for people but not particles.

```
flux_in_people <- function(N_rooms, Transition_matrix,State,Room_pops,Carrying_capacity,t){
  # Transition_matrix <- matrix(c(0,0.5,1,0.7,0,0,0.3,0.5,0), nrow = 3,ncol = 3)
  # State <- Church_setup$S_prop
  # Room_pops <- Church_setup$N_x
  # Carrying_capacity <- Church_C
  all_room_change <- c(seq(N_rooms))
  for(x in 1:N_rooms){ #flow in to room x from other rooms (j)
    flux_in_temp <- 0
    for(j in 1:N_rooms){
      flux_in_temp <- flux_in_temp + State[j]*Transition_matrix[j,x]*(1-(Room_pops[x]/Carrying_capacity
    }
    all_room_change[x] <- flux_in_temp
  }

  test <- c(M = as.vector(all_room_change))
  return(test)
```

```
}

flux_out_people <- function(N_rooms, Transition_matrix,State,Room_pops,Carrying_capacity){
  all_room_change <- c(seq(N_rooms))
  for(x in 1:N_rooms){
    flux_out_temp <- 0
    for(j in 1:N_rooms){
      flux_out_temp <- flux_out_temp + State[x]*Transition_matrix[x,j]*(1-(Room_pops[j]/Carrying_capaci
    }
    all_room_change[x] <- flux_out_temp
  }
  test <- c(M = as.vector(all_room_change))
  return(test)
}

flux_in_particles <- function(N_rooms, Transition_matrix,State){
  all_room_change <- c(seq(N_rooms))
  for(x in 1:N_rooms){
    flux_in_temp <- 0
    for(i in 1:N_rooms){
      flux_in_temp <- flux_in_temp + State[i]*Transition_matrix[i,x]
    }
    all_room_change[x] <- flux_in_temp
  }

  test <- c(M = as.vector(all_room_change))
  return(test)
}

flux_out_particles <- function(N_rooms, Transition_matrix,State){
  all_room_change <- c(seq(N_rooms))
  for(x in 1:N_rooms){
    flux_out_temp <- 0
    for(i in 1:N_rooms){
      flux_out_temp <- flux_out_temp + State[x]*Transition_matrix[x,i]
    }
    all_room_change[x] <- flux_out_temp
  }
  test <- c(M = as.vector(all_room_change))
  return(test)
}
```

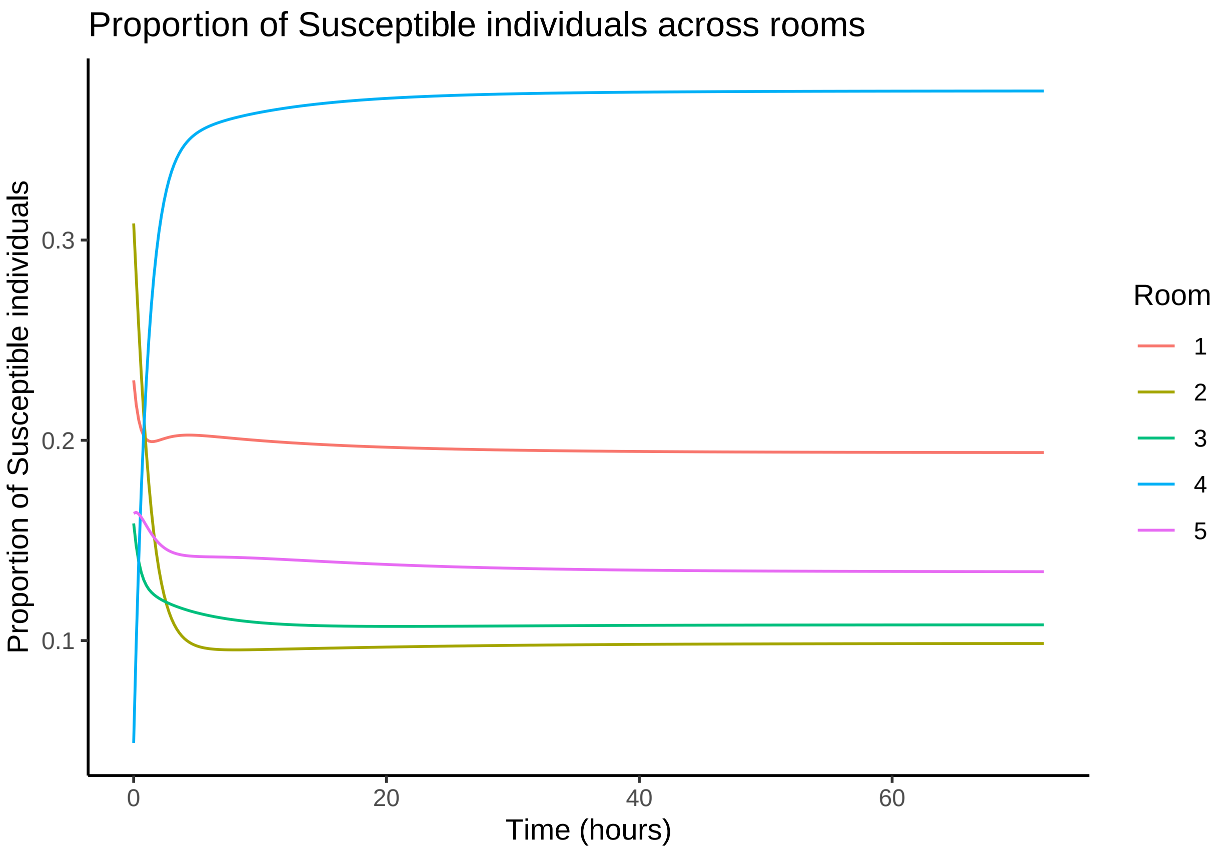
Function for the whole model

```
Particle_model <- function(t, x, parms,T_mov, theta_mov, adjacency_matrix_to_use,C_x){
  ncompartment <- 5
  n_rooms <- length(x)/ncompartment
  S <- as.matrix(x[1:n_rooms])
  I <- as.matrix(x[(n_rooms+1):(2*n_rooms)])
  R <- as.matrix(x[(2*n_rooms+1):(3*n_rooms)])
  P <- as.matrix(x[(3*n_rooms+1):(4*n_rooms)])
  N_x <- as.matrix(x[(4*n_rooms+1):(5*n_rooms)])
```

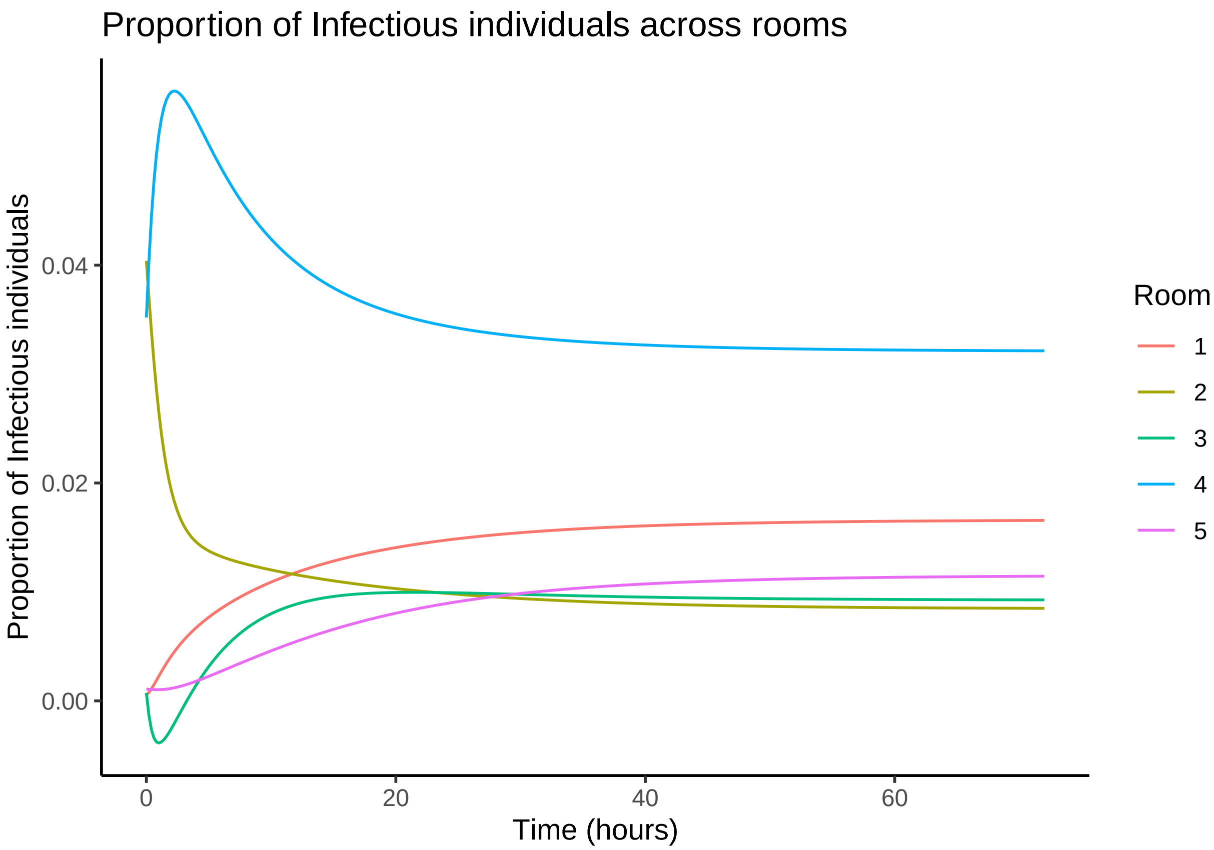
```r
  with(parms,{


    dS <- as.matrix((flux_in_people(N_rooms, Transition_matrix =T_mov, State=S,Room_pops = N_x,Carrying_

    dI <- as.matrix((flux_in_people(N_rooms, Transition_matrix =T_mov, State=I,Room_pops = N_x,Carrying_

    dR <- as.matrix((flux_in_people(N_rooms, Transition_matrix =T_mov, State=R,Room_pops = N_x,Carrying_
    #last step will be the particle EQ

    dP <- s*as.matrix(I)*as.matrix(N_x) - as.matrix(a*P/(lam*C_x*N_x))-d*as.matrix(P) + as.matrix(as.ma

    dN_x <- as.matrix((flux_in_people(N_rooms, Transition_matrix =T_mov, State=N_x,Room_pops = N_x,Carr

    dt <- c(dS,dI,dR,dP,dN_x)
    return(list(dt))})

}
```

```r
Church_Init_conds <-c(S=Church_setup$S, I = Church_setup$I, R = Church_setup$R, P = Church_setup$P, N_x

Church_output <- data.frame(lsoda(y = Church_Init_conds, func = Particle_model,times = times,
                                  parms = parms,
                                  adjacency_matrix_to_use=adjacency_matrix_to_use,
                                  theta_mov =Church_theta_mov,
                                  T_mov = Church_T_mov,
                                  C_x=Church_C))
```

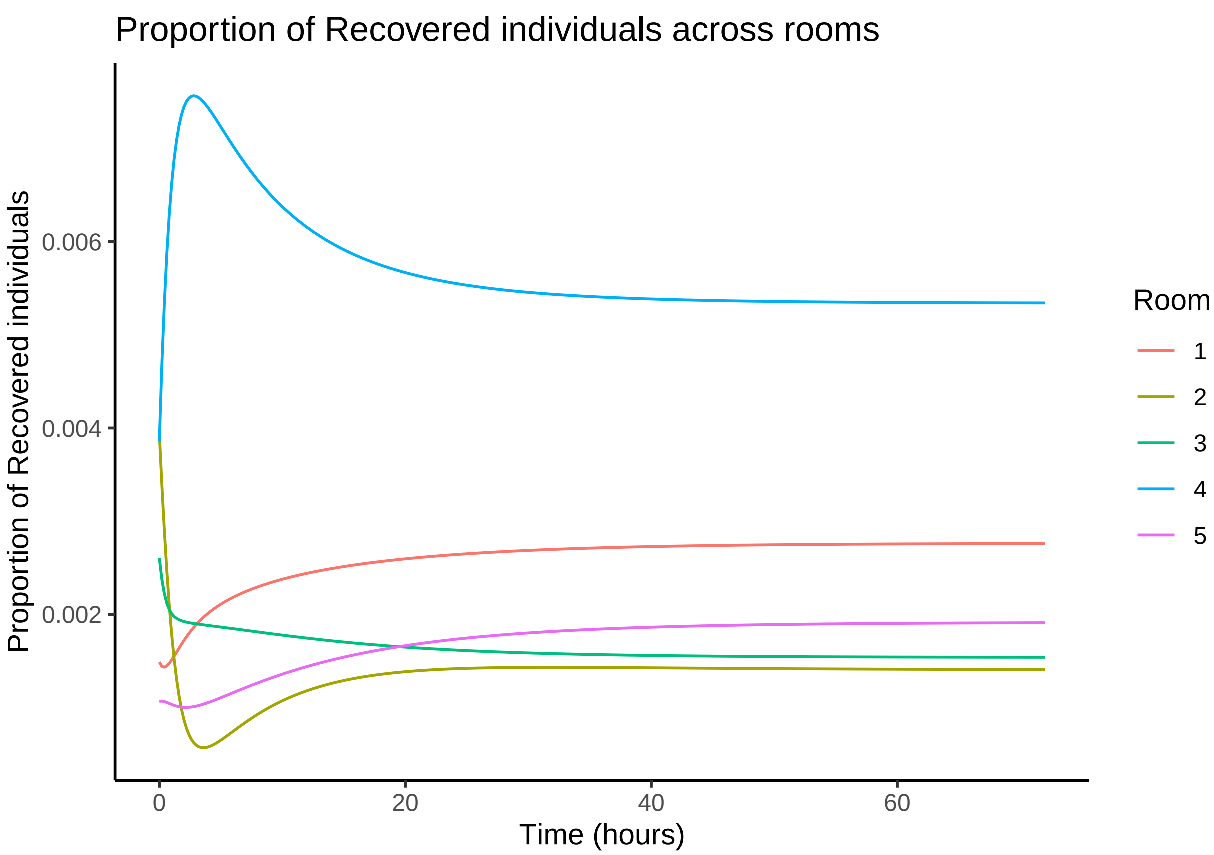
```r
Church_data_clean <- Church_output%>% pivot_longer(cols = !time,
                                                   names_to = c("State", "Room"),
                                                   names_pattern = "([A-Za-z]+)(\\d+)",
                                                   values_to = "Number")

# Church_data_ratios <- Church_data_clean %>% pivot_wider(names_from = c(State),values_from = c(Number)
#   mutate(N_x = S+I+R) %>% group_by(time,Room) %>%
#   mutate(K_x = ((parms$s)*I)/((parms$a)*N_x), prop_to_K = P/K_x)
# Church_P_x_K_x_plot<-ggplot(Church_data_ratios,aes(x =time, y =prop_to_K,group= Room,color=Room))+geom
# Church_P_x_K_x_plot
# Church_P_x_plot <- ggplot(Church_data_ratios, aes(x=time, y = P, color = Room))+geom_line()+labs(titl
# Church_P_x_plot

Church_data_clean %>% filter(State == "S") %>%
  group_by(State,Room)%>%
  ggplot( aes(x=time, y = Number, group =Room, color = Room))+
  geom_line()+theme_classic()+
  labs(x = "Time (hours)", y= "Proportion of Susceptible individuals")+ggtitle("Proportion of Susceptib
```



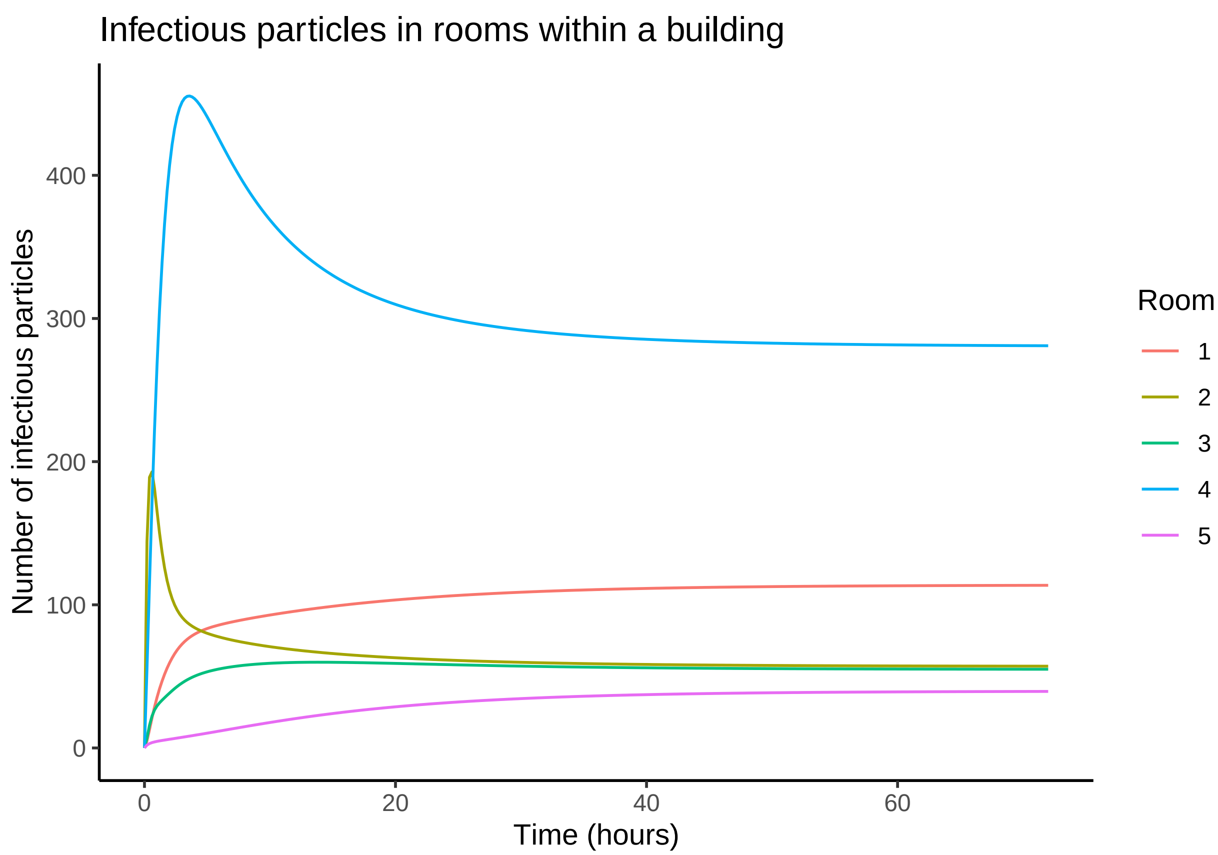

```
Church_data_clean %>% filter(State == "I") %>%
  group_by(State,Room)%>%
  ggplot( aes(x=time, y = Number, group =Room, color = Room))+
  geom_line()+theme_classic()+
  labs(x = "Time (hours)", y= "Proportion of Infectious individuals")+ggtitle("Proportion of Infectious
```



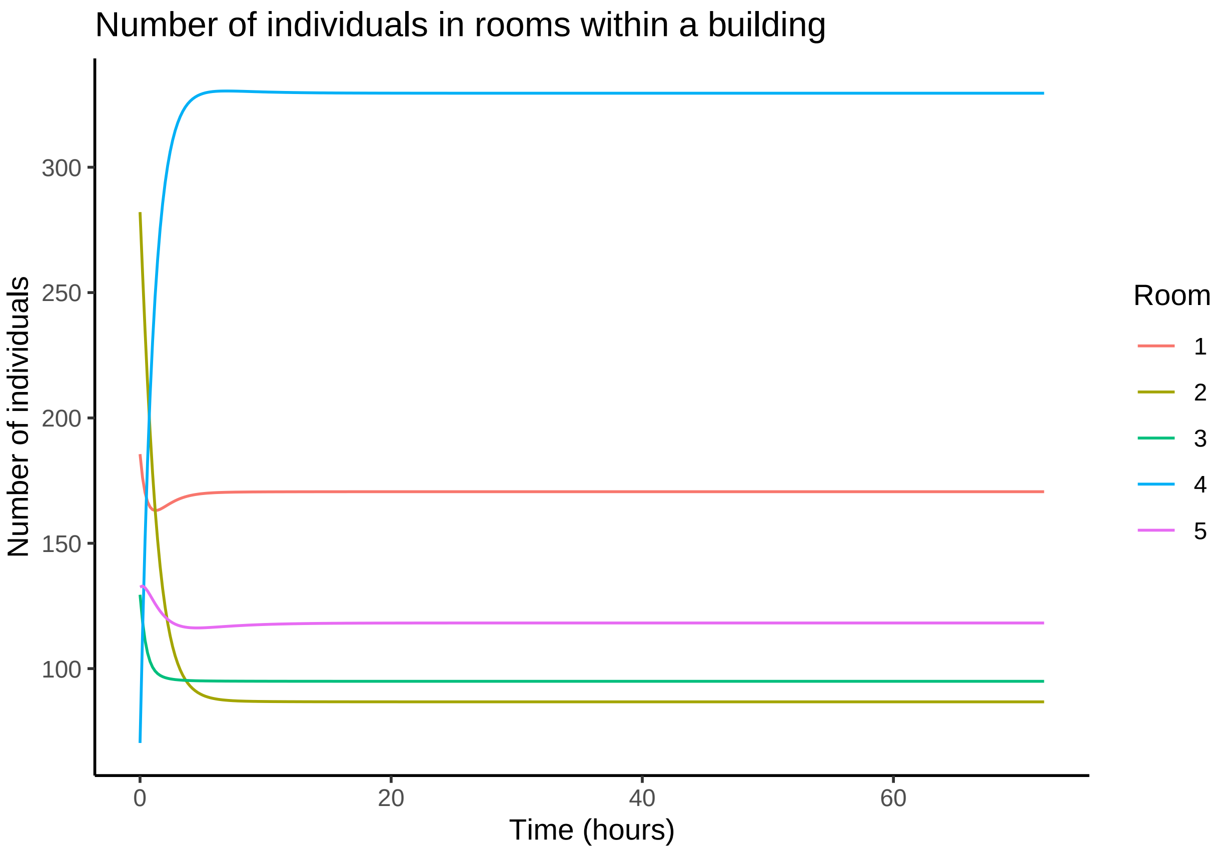

```
Church_data_clean %>% filter(State == "R") %>%
  group_by(State,Room)%>%
  ggplot( aes(x=time, y = Number, group =Room, color = Room))+
  geom_line()+theme_classic()+
  labs(x = "Time (hours)", y= "Proportion of Recovered individuals")+ggtitle("Proportion of Recovered in
```




```
Church_data_clean %>% filter(State == "P") %>%
  group_by(State,Room)%>%
  ggplot( aes(x=time, y = Number, group =Room, color = Room))+
  geom_line()+theme_classic()+
  labs(x = "Time (hours)", y= "Number of infectious particles")+ggtitle("Infectious particles in rooms w
```

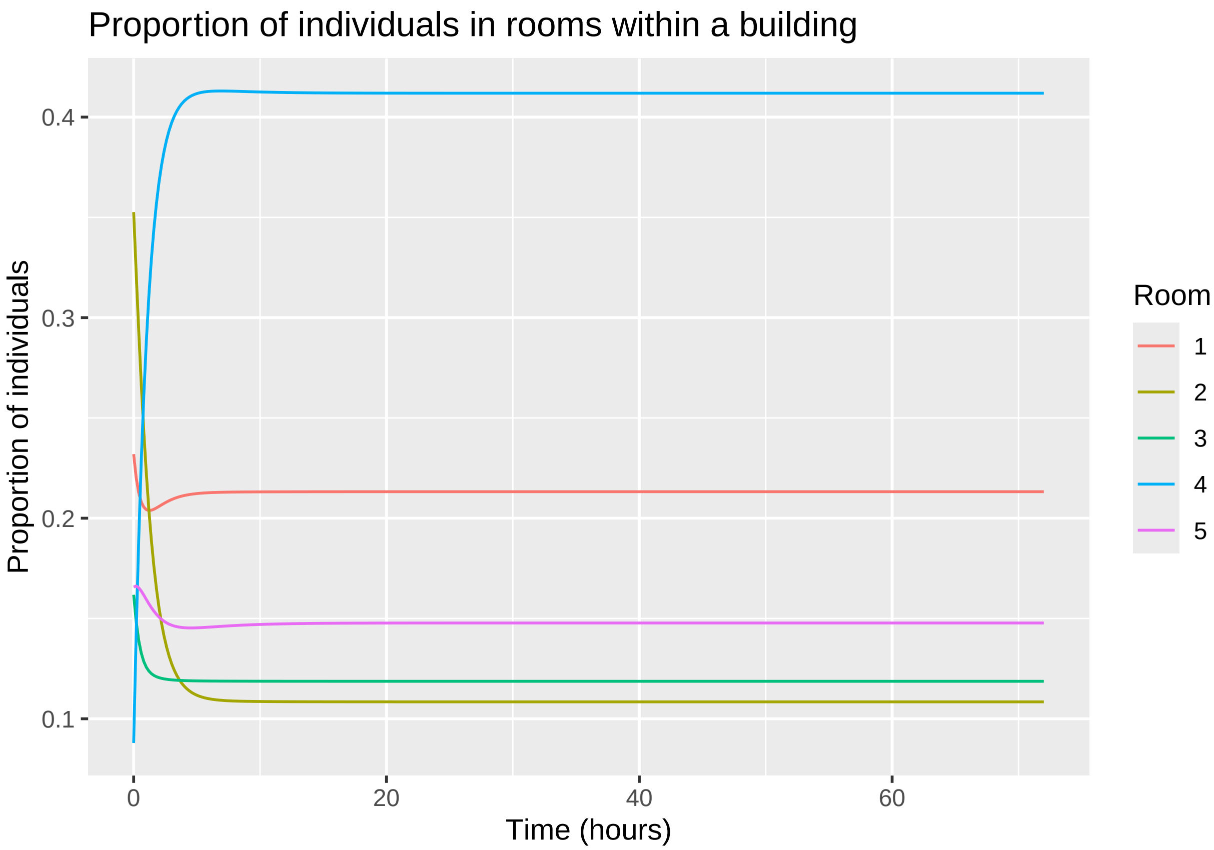



```
Church_data_clean %>% filter(State == "x") %>%
  group_by(State,Room)%>%
  ggplot( aes(x=time, y = Number, group =Room, color = Room))+
  geom_line()+theme_classic()+
  labs(x = "Time (hours)", y= "Number of individuals")+ggtitle("Number of individuals in rooms within a
```

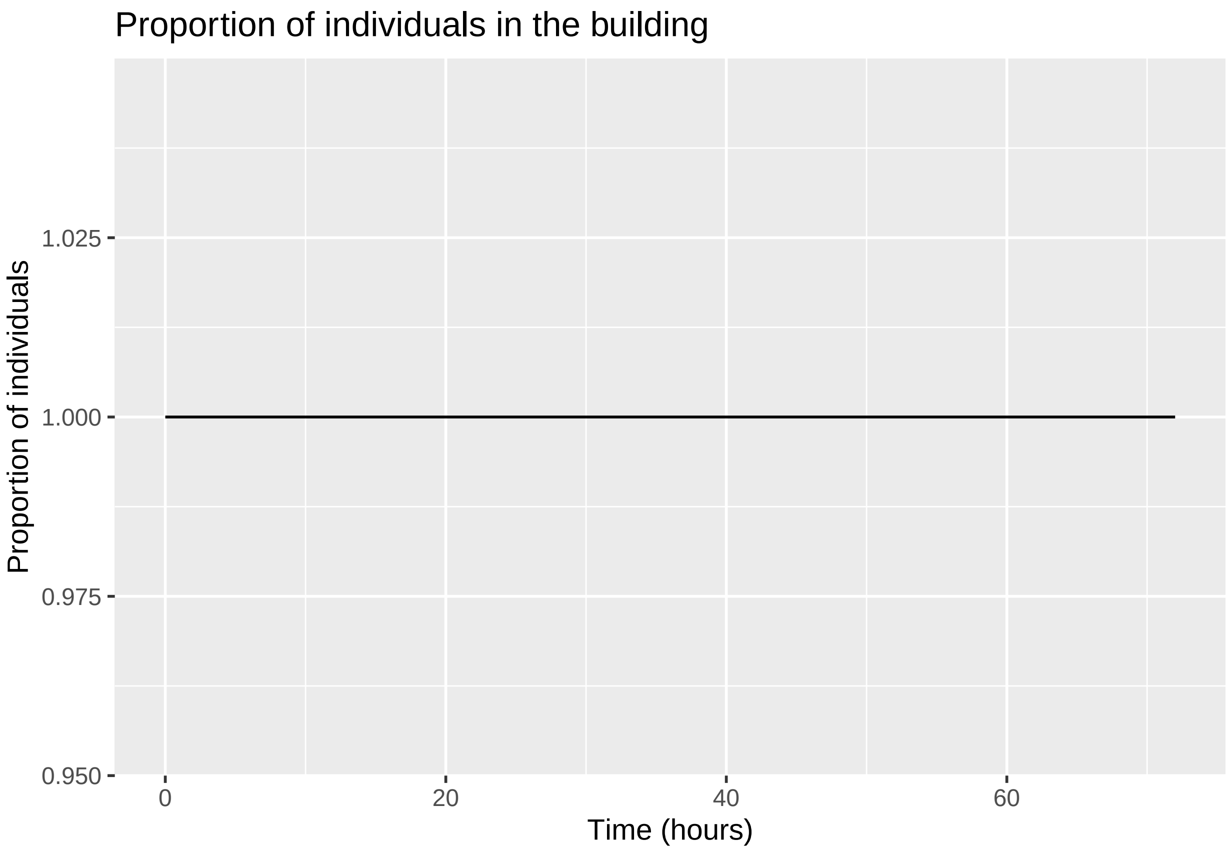



```
# Church_data_clean %>% filter(State != "P") %>%
#   group_by(time,Room)%>% summarise(total_room_pop=sum(Number)) %>% group_by(time) %>% summarise(tot_b
#   ggplot( aes(x=time, y = tot_bld_pop))+
#   geom_line()+theme_classic()+
#   labs(x = "Time (hours)", y= "Number of people")+ggtitle("number of people in building")

Church_data_clean %>% filter(State == "S"| State == "I" | State == "R") %>% ungroup() %>% group_by(time
  ggplot(aes(x=time, y=Total_prop, color=Room))+geom_line()+
  labs(x = "Time (hours)", y= "Proportion of individuals")+ggtitle("Proportion of individuals in rooms w
```

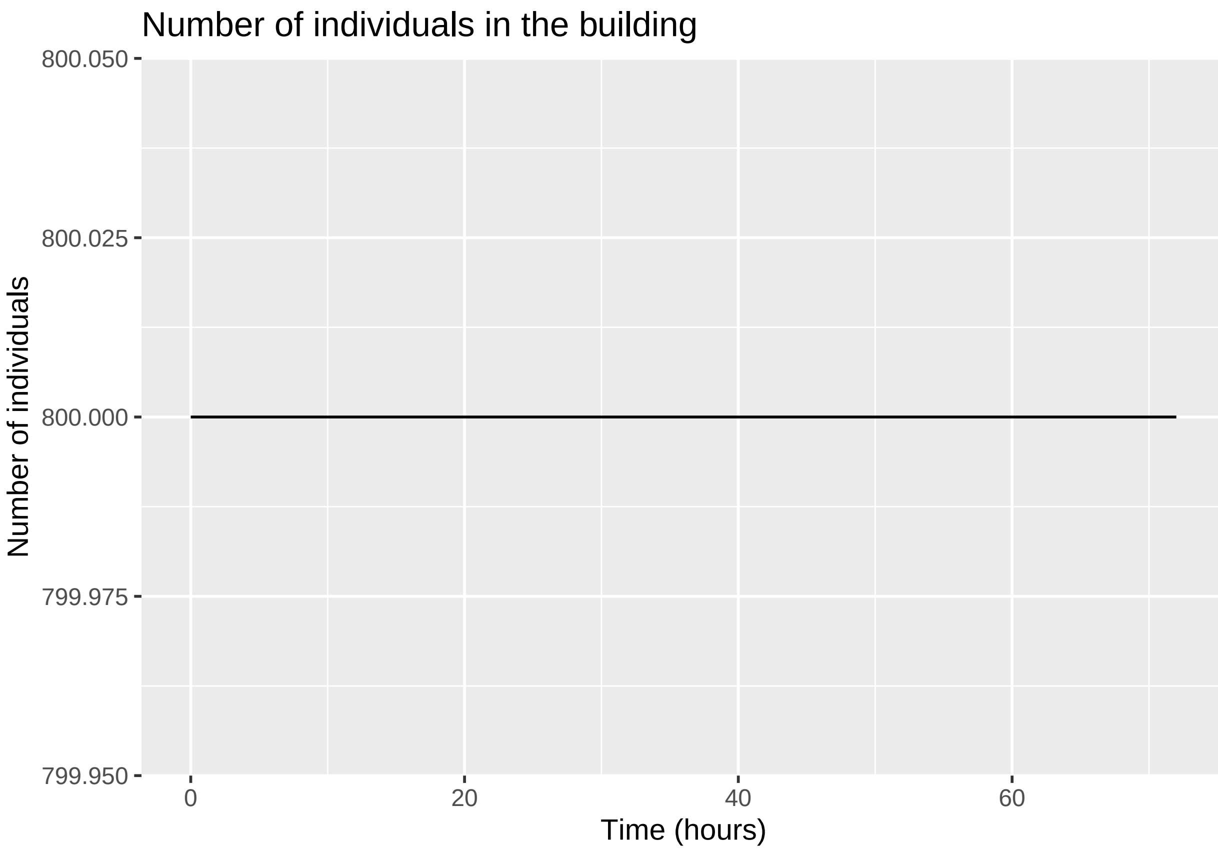
```
## `summarise()` has grouped output by 'time'. You can override using the
## `.groups` argument.
```




```
Church_data_clean %>% filter(State == "S"| State == "I" | State == "R") %>% ungroup() %>% group_by(time
  ggplot(aes(x=time, y=Total_prop))+geom_line()+
  labs(x = "Time (hours)", y= "Proportion of individuals")+ggtitle("Proportion of individuals in the bu
```




```
Church_data_clean %>% filter(State== "x")%>% ungroup() %>% group_by(time) %>% summarise(Total_pop=sum(Nu
  ggplot(aes(x=time, y=Total_pop))+geom_line()+
  labs(x = "Time (hours)", y= "Number of individuals")+ggtitle("Number of individuals in the building")
```



Number of individuals in the building
800.050
800.025
800.000
799.975
799.950
Number of individuals
0
20
40
60
Time (hours)