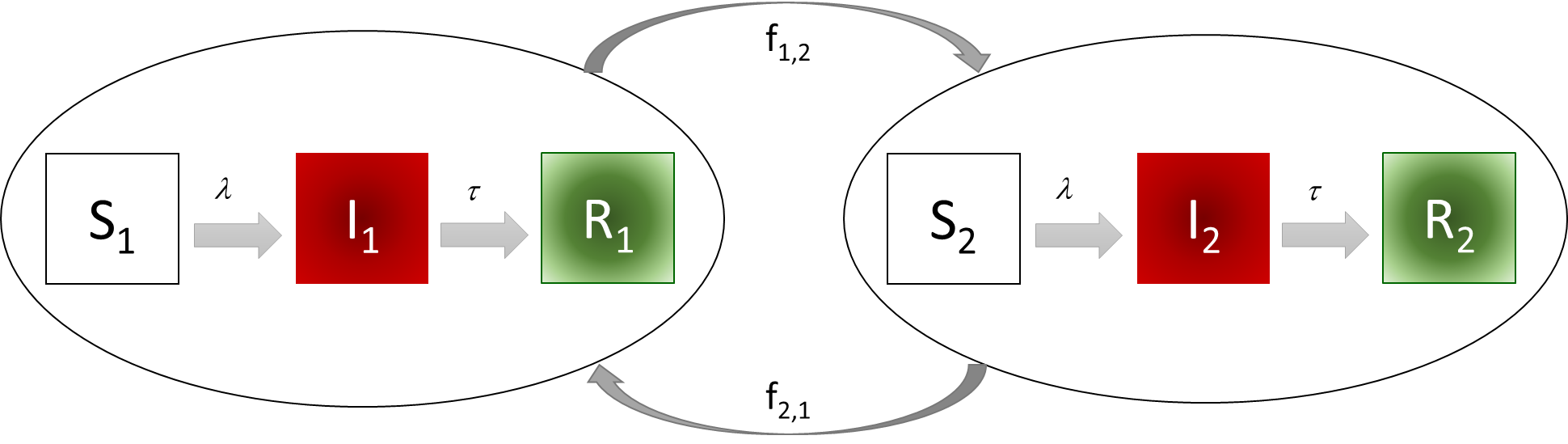
# Spatial models

Let’s consider a setting where two populations are connected by migration rates given by *fij.*



Such a system can be expressed mathematically in R as:

SIR\_patch<-function(t, state, parameters) {

  with(as.list(c(state, parameters)),{

    P1 <- S1 + I1 + R1

    P2 <- S2 + I2 + R2

    lam1 <- beta1 \* I1 / P1

    lam2 <- beta2 \* I2 / P2

    dS1 <- mu\*P1 - mu\*S1- lam1\*S1 - f12\*S1 + f21\*S2

    dI1 <- lam1\*S1 - tau\*I1 - mu\*I1 - f12\*I1 + f21\*I2

    dR1 <- tau\*I1 - mu\*R1 - f12\*R1 + f21\*R2

    dS2 <- mu\*P2 - mu\*S2 - lam2\*S2 + f12\*S1 - f21\*S2

    dI2 <- lam2\*S2 - tau\*I2 - mu\*I2 + f12\*I1 - f21\*I2

    dR2 <- tau\*I2 - mu\*R2 + f12\*R1 - f21\*R2

    N <- P1 + P2

    list(c(dS1, dI1, dR1, dS2, dI2, dR2, N))})

}

Where the parameters and state variables are defined as:

initP1 <- 1000

initI1 <- 1

initR1 <- 0

initS1 <- initP1 - initI1 - initR1

initP2 <- 1000

initR2 <- 0

initI2 <- 0

initS2 <- initP2 - initI2 - initR2

istate <- c(S1 = initS1, I1 = initI1, R1 = initR1,

            S2 = initS2, I2 = initI2, R2 = initR2,

            N = initS1 + initS1 + initR1 + initS2 + initS2 + initR2)

parameters <- c(

  mu = (1/(50\*52\*7)),

  beta1 = 0.75,

  beta2 = 0.45,

  tau = 1/10,

  f12 = 0.001,

  f21 = 0.001

)

For a timeframe of

time\_start <- 0

time\_stop <- 100

deltat<- 0.001

tps <- seq(time\_start , time\_stop , by = deltat)

run the ODE system defined by the *SIR\_patch* function.

out <- deSolve::ode(y = istate, times = tps, func = SIR\_patch, parms = parameters)

plot(out)

1. Check that the population size is constant. In the model function, you’ll find N expressed as the sum of P1 and P2 which are being listed in the list of values returned by the ode function. Why do you think the bottom row panels give those numbers?
2. Use the post-processing approach you have used last week to check the population size in the out object.

> tail(rowSums(out[,2:4]))+tail(rowSums(out[,5:7]))

[1] 2000 2000 2000 2000 2000 2000

1. Interpret what the list function inside the SIR\_patch function does. Interpretation can be aided by adding a counter to the output list:

SIR\_patch<-function(t, state, parameters) {

  with(as.list(c(state, parameters)),{

    P1 <- S1 + I1 + R1

    P2 <- S2 + I2 + R2

    lam1 <- beta1 \* I1 / P1

    lam2 <- beta2 \* I2 / P2

    dS1 <- mu\*P1 - mu\*S1 - lam1\*S1 - f12\*S1 + f21\*S2

    dI1 <- lam1\*S1 - tau\*I1 - mu\*I1 - f12\*I1 + f21\*I2

    dR1 <- tau\*I1 - mu\*R1 - f12\*R1 + f21\*R2

    dS2 <- mu\*P2 - mu\*S2 - lam2\*S2 + f12\*S1 - f21\*S2

    dI2 <- lam2\*S2 - tau\*I2 - mu\*I2 + f12\*I1 - f21\*I2

    dR2 <- tau\*I2 - mu\*R2 + f12\*R1 - f21\*R2

    N <- P1 + P2

    count <- 1

    list(c(dS1, dI1, dR1, dS2, dI2, dR2, N, count))})

}

Remember to include that variable in the initial conditions and set it to zero. Re-run the model and check the value in the count column:

> tail(out[,11],1)

[1] 100

1. Note that movement was set to be symmetrical (same in and out flows from each area/population). What would happen if individuals stopped moving from area 1 to area 2 altogether? As during the previous week, we can just re-set one (or more parameters) and re-run the call to the ode solver:

parameters[“f12”] <- 0.0

out\_no\_migration <- deSolve::ode(y = istate, times = tps, func = SIR\_patch, parms = parameters)

plot(out\_no\_migration)

What do you observe? Relate that to the initial conditions of the system. Change them to ensure you have an outbreak in population 2.

1. Could interventions in one place have a knock-on effect on the other area? Let’s re-set the migration patterns and change the force of infection in region 1. We can achieve this by changing the time a transmission blocking intervention with effectiveness of 20% starts. If it is started at day 10:

parameters["f12"] <- 0.001

parameters["tbit"] <- 10

parameters["tbeff"] <- 0.2

out\_intervention <- deSolve::ode(y = istate, times = tps, func = SIR\_patch, parms = parameters)

par(mfrow=c(2,1))

plot(out[,3], type = "l", col = "orange3", main = "Pop1")

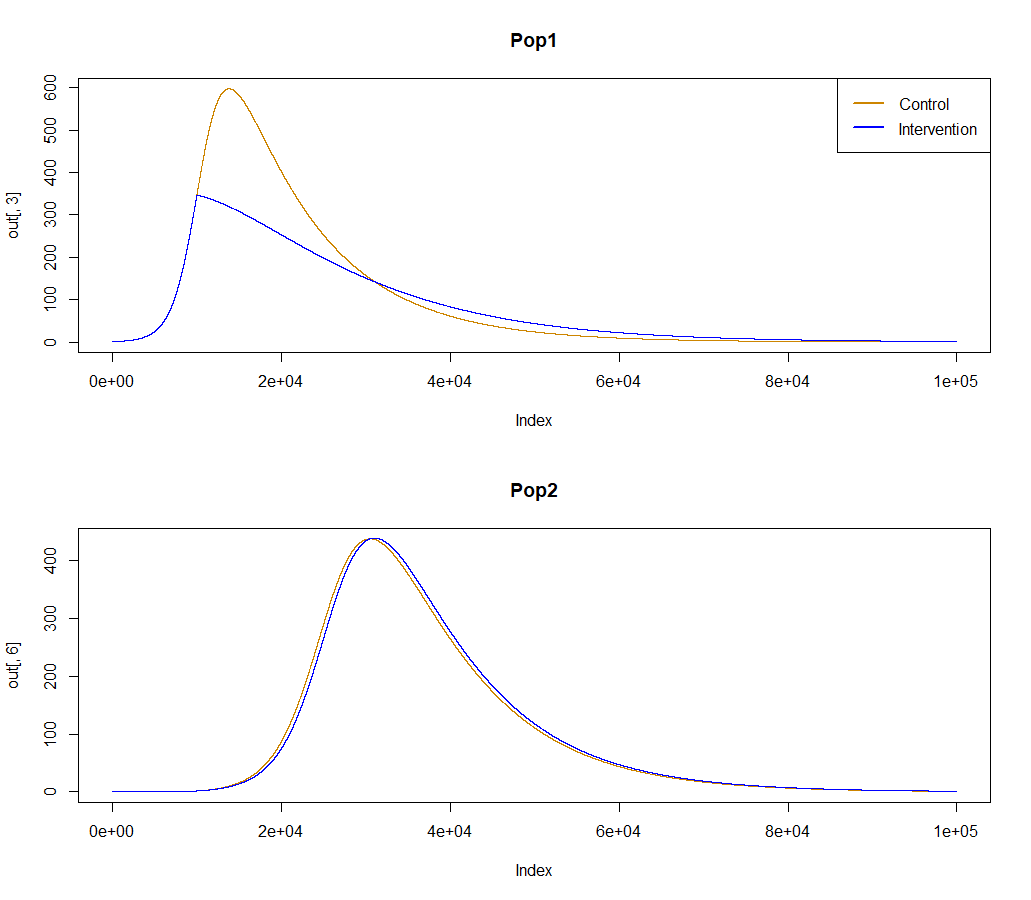
lines(out\_intervention[,3], type = "l", col = "blue")

legend("topright", legend = c("Control", "Intervention"), lwd = 2, col = c("orange3", "blue"))

plot(out[,6], type = "l", col = "orange3", main = "Pop2")

lines(out\_intervention[,6], type = "l", col = "blue")

You should observe something like the figure below.



1. Explore how sensitive these results are to:
   1. the initial conditions of the system
   2. the relative difference in force of infection between the two areas.
   3. The migration rate between Pop 1 and Pop2.

You can start with the following:

parameters["tbit"] <- 1e100

parameters["f12"] <- 0.003

parameters["f21"] <- 0.001

parameters["beta2"] <- 0.25

out\_high\_migr <- deSolve::ode(y = istate, times = tps, func = SIR\_patch, parms = parameters)

parameters["tbit"] <- 10

out\_intervention\_high\_migr <- deSolve::ode(y = istate, times = tps, func = SIR\_patch, parms = parameters)

par(mfrow=c(2,1))

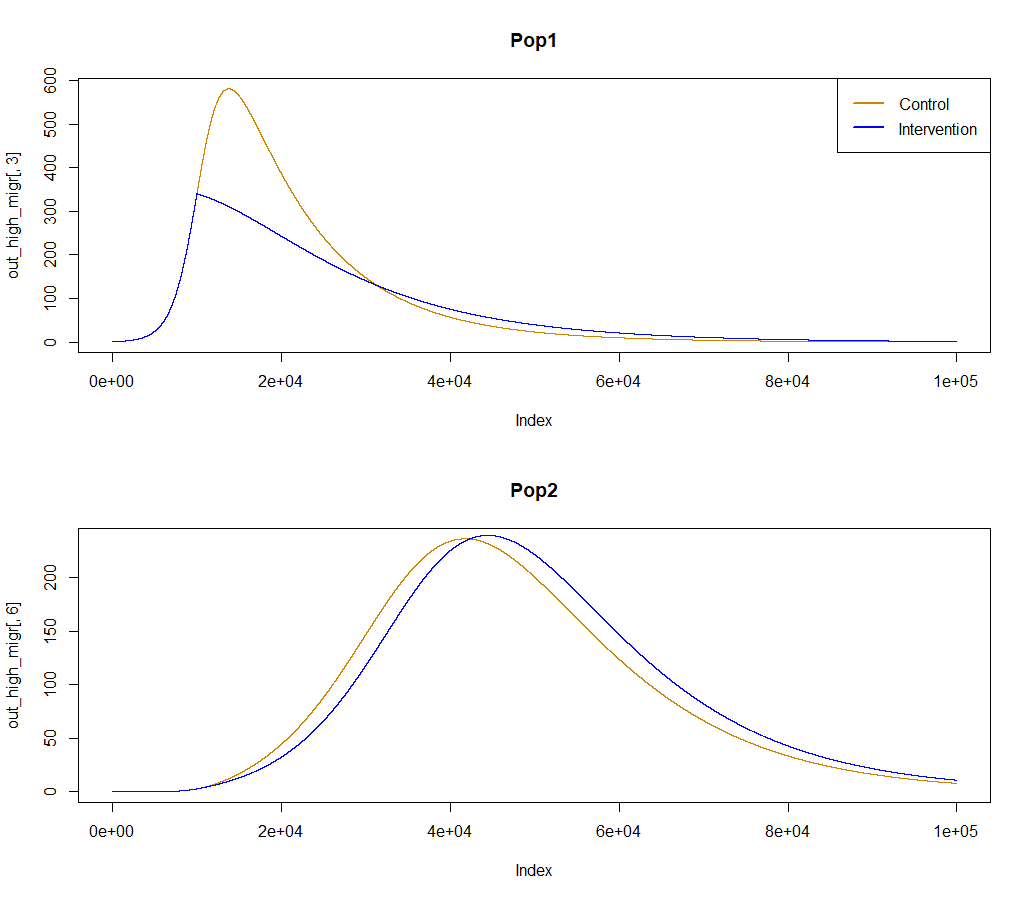
plot(out\_high\_migr[,3], type = "l", col = "orange3", main = "Pop1")

lines(out\_intervention\_high\_migr[,3], type = "l", col = "blue")

legend("topright", legend = c("Control", "Intervention"), lwd = 2, col = c("orange3", "blue"))

plot(out\_high\_migr[,6], type = "l", col = "orange3", main = "Pop2")

lines(out\_intervention\_high\_migr[,6], type = "l", col = "blue")



1. What if R0 in Pop 2 is set to 0.1. Would you still see an outbreak in Pop2 given those migration patterns?

# A scalable framework that takes in mobility data as an input

In some circumstances, we can access mobility data that can inform the network of population migration patterns across large populations. In a new script import the *movement\_spatial* csv file, describing the frequency and direction of population migration events in an area with 16 sub-populations, into R. Plot the matrix using the plot.matrix library.

movement <- as.matrix(read.csv("movement\_spatial.csv", header = FALSE))

movement <- movement \* 0.00002

plot(movement)

Demographic information for each population can be found in the pop\_spatial csv file.

*# Population structure*

popstruc <- as.data.frame(read.csv("pop\_spatial.csv", header = TRUE))

pop\_size <- popstruc$pop

pop\_x <- popstruc$longitude

pop\_y <- popstruc$latitude

areas <- length(pop\_size)

This includes population size for each sub-population, x-y coordinates and the daily incidence per 100 people per day.

*# SPATIAL RISK  DATA*

risk <- popstruc$incidence \* 100

Using the movement object above, we can create a network reflecting the level of mobility between sub-populations. To do so, we will use the *graph.adjacency* function in the *igraph* library. For a better visualisation, we will highlight sub-populations with more than 5 daily cases in red and make the size of the vertices proportional to the population size. Note that the relative position of the modes in this graph are fixed according to the x-y coordinates.

*# VISUALISE NETWORK*

X <- as.matrix(movement)

g2 <- graph.adjacency(X, mode = "undirected",

    weighted = TRUE, diag = FALSE)

*# Assign attributes to the graph*

g2$name <- "SEIR spatial model"

*# Assign attributes to the graph's vertices*

V(g2)$color <- ifelse(risk > 0.5, "red", "green")

V(g2)$x <- pop\_x

V(g2)$y <- pop\_y

V(g2)$size <- pop\_size / 5e4

plot(g2, edge.width = E(g2)$weight \* 8)

We can initialise our metapopulation model with the following parameters

startdate <- as.Date("2023-01-01")

stopdate <- as.Date("2023-12-31")

day\_start <- as.numeric(startdate-startdate)

day\_stop <- as.numeric(stopdate-startdate)

times <- seq(day\_start, day\_stop)

*#MODEL PARAMETERS*

parameters <- c(

  p = 0.047,                *# probability of infection given a contact*

  rho = 0.2,                *# relative infectiousness of incubation phase*

  omega = (1 / (100 \* 365)),*# rate of loss of immunity = 1/(average duration of immunity)*

  gamma = 1 / 4.5,          *# rate of movement from incubation to infectious stage = 1/(average incubation period)*

  nui = 1 / 5,              *# rate of recovery = 1/(average duration of symptomatic infection phase)*

  report = 1 / 8,           *# proportion of all infections that are reported*

  ratem = 1 / 7,            *# 1/ time to death for fatal*

  nus = 1 / 7,              *# rate if recovery for non-fatal severe*

  rhos = 0.1,               *# relative infectiousness\*contacts of severely ill patients*

  csr = 0.1,                *# case severity ratio*

  sfr = 0.2,                *# severe case fatality ratio*

  mort = (1 / (80 \* 365))   *# mortality rate*

)

How can we initialise the state variables of a metapopulation SEIR system? Try doing that using indices, in the same way you saw it done for age-dependent models (more on this in week 6 Monday’s afternoon session).

*# Define the indices for each variable*

Sindex <- 1 : areas

Eindex <- (areas + 1) : (2 \* areas)

Iindex <- (2 \* areas + 1) : (3 \* areas)

Rindex <- (3 \* areas + 1) : (4 \* areas)

Hindex <- (4 \* areas + 1) : (5 \* areas)

Mindex <- (5 \* areas + 1) : (6 \* areas)

Cindex <- (6 \* areas + 1) : (7 \* areas)

CMindex <- (7 \* areas + 1) : (8 \* areas)

*###########################################################################*

*# MODEL INITIAL CONDITIONS*

initI <- round(0.01\* popstruc$pop) *# Infected and symptomatic*

initE <- 0 \* popstruc$pop *# Incubating*

initR <- 0 \* popstruc$pop *# Immune*

initH <- 0 \* popstruc$pop *# hospitalised*

initM <- 0 \* popstruc$pop *# died*

initC <- 0 \* popstruc$pop *# Cumulative cases (true)*

initCM <- 0 \* popstruc$pop *# Cumulative deaths (true)*

initS <- popstruc$pop - initE - initI -

          initR - initH - initM *# Susceptible (non-immune)*

*# initial conditions for the main solution vector*

Y <- c(initS, initI, initE, initR,

       initH, initM, initC, initCM)

Now run your spatial model, taking special care to define the force of infection and migration terms appropriately.

*# set up a function to solve the equations*

spatial\_model <- function(t, Y, parameters) {

  with(as.list(c(Y, parameters)),

    {

      S <- Y[Sindex]

      E <- Y[Eindex]

      I <- Y[Iindex]

      R <- Y[Rindex]

      H <- Y[Hindex]

      M <- Y[Mindex]

      C <- Y[Cindex]

      CM <- Y[CMindex]

*# print(t)*

      contacts <- risk

*#   print(contacts)*

      P <- (S + E + I + R + H + M)

      lam <- p \* contacts \* ((rho \* E + I + rhos \* H) / P)

*#   print(lam)*

*#   print(movement)*

      dSdt <- - S \* lam + omega \* R - mort \* S + mort \* P -

        movement %\*% S + t(movement) %\*% S

      dEdt <- S \* lam - gamma \* E - mort \* E -

        movement %\*% E + t(movement) %\*% E

      dIdt <- gamma \* (1 - csr) \* E - nui \* I - mort \* I -

        movement %\*% I + t(movement) %\*% I

      dRdt <- nui \* I - omega \* R - mort \* R + nus \* H -

        movement %\*% R + t(movement) %\*% R

      dHdt <- - mort \* H + gamma \* csr \* (1 - sfr) \* E - nus \* H

      dMdt <- gamma \* csr \* sfr \* E - ratem \* M - mort \* M

      dCdt <- report \* gamma \* E

      dCMdt <- ratem \* M + mort \* M

*# return the rate of change*

      list(c(dSdt, dEdt, dIdt, dRdt, dHdt, dMdt, dCdt, dCMdt))

    }

  )

}

*# run the model*

out <- ode(y = Y, times = times, func = spatial\_model, parms = parameters)

print("ee")

*# total population*

pop <- out[, (Sindex + 1)] + out[, (Eindex + 1)] + out[, (Iindex + 1)] +

       out[, (Rindex + 1)] + out[, (Hindex + 1)] + out[, (Mindex + 1)]

tpop <- rowSums(pop)

plot(tpop)

1. Uncomment some of the prints above and re-run the code to check that the quantities make sense. How is the force of infection taking population movement into account?
2. Does the total population plot make sense? Explain how we arrive at that number given the code structure and parameter values.
3. Look at how population movement is implemented. What do these terms represent (explain the formula conceptually)

- movement %\*% S + t(movement) %\*% S

1. Report daily case incidence (given the reporting rate), hospital bed requirement over time and cumulative mortality.
2. Plot cumulative mortality for each sub-population.