Lab 9 Storage Solutions for Multiple Simulations; Incorporating Births and Deaths

Partial SOLUTIONS

15 November, 2021

Storage for Multiple Simulations from Stochastic Models

In our previous lab, we modified some of the functions we had written to simulate from deterministic compartmental models to enable us to simulate from stochastic compartmental models. However, each function we modified only resulted in a single realization from a simulation, and we didn't create any storage to accommodate outcomes from multiple simulations.

First, let's think about how we might do this in different way for different situations.

Option 1: Use `rbind`.

The function `rbind` allows for one to row bind vectors, matrices, and data frames. First, let us explore what that means by running the following code chunks:

```
v1 <- c(1:5)
v2 <- c(6:10)
v3 <- rbind(v1, v2)
v3
```

```
##    [,1] [,2] [,3] [,4] [,5]
## v1    1    2    3    4    5
## v2    6    7    8    9   10
```

Briefly describe what `rbind(v1, v2)` does.

```
set.seed(123)
df1 <- data.frame(time=0:5, a=rbinom(n=6, size=10, prob=0.2))
df2 <- data.frame(time=6:10, a=rbinom(n=5, size=10, prob=0.3))
df3 <- rbind(df1, df2)
df3
```

```
##    time a
## 1     0 1
## 2     1 3
## 3     2 2
## 4     3 4
## 5     4 4
## 6     5 0
## 7      6 3
## 8      7 5
## 9      8 3
## 10     9 3
## 11    10 6
```

Briefly describe what `rbind(df1, df2)` does.

```r
set.seed(121)
df4 <- data.frame(time=0:5, a=rbinom(n=6, size=10, prob=0.2))
df5 <- data.frame(time=6:10, b=rbinom(n=5, size=10, prob=0.3))
# df6 <- rbind(df4, df5)
# df6
```

You should get an error when you run the last chunk of code above. What is the problem? What does this tell you about the requirements for using rbind with data frames?

After you answer the above question, comment out the lines that relate to `df6` - if you do not do that, you will not be able to knit your document at the end of the lab.

We can also use `rbind` inside or outside a function. If we want to bind more than two data frames together, we can use the for loop structure again.

```r
SIR_simulation <- function(N, S0, I0, R_0, D, Time, nItr){
  ## Create an empty data frame with the same names
  sim_df <- data.frame(time=character(0),
                       S=character(0),
                       I=character(0),
                       R=character(0),
                       I_SI=character(0),
                       R_IR=character(0),
                       lambda_t=character(0),
                       SimNum=character(0))
  for (j in 1:nItr){
    SIR_df <- data.frame(time=0:Time,
                       S=rep(NA, Time+1),
                       I=rep(NA, Time+1),
                       R=rep(NA, Time+1),
                       I_SI=rep(NA, Time+1),
                       R_IR=rep(NA, Time+1),
                       lambda_t=rep(NA, Time+1),
                       ## Add a column to keep track of simulation number
                       SimNum=rep(j, Time+1))

    SIR_df$S[1] <- S0
    SIR_df$I[1] <- I0
    SIR_df$R[1] <- N-S0-I0

    for (i in 2:(Time+1)){
      SIR_df$lambda_t[i] <- lambda_t_fcn(R_0=R_0, D=D, I_i=SIR_df$I[i-1], N=N)
      SIR_df$I_SI[i] <- rbinom(n=1, size=SIR_df$S[i-1], prob=SIR_df$lambda_t[i])
      SIR_df$R_IR[i] <- rbinom(n=1, size=SIR_df$I[i-1], prob=1/D)
      SIR_df$S[i] <- SIR_df$S[i-1]-SIR_df$I_SI[i]
      SIR_df$I[i] <- SIR_df$I[i-1]+SIR_df$I_SI[i]-SIR_df$R_IR[i]
      SIR_df$R[i] <- SIR_df$R[i-1]+SIR_df$R_IR[i]
    }
    sim_df <- rbind(sim_df, SIR_df)
  }
  return(sim_df)
}
```

Run the SIR_simulation function for N=1000, S0=999, I0=1, R_0=2, D=3, Time=10, nItr=5 and name it test1. What does it look like? What changes to the code allowed for these changes?

Do the same thing, but increase the number of iterations to 500. Name this test2.

Do the same thing, but increase the number of iterations to 500 and the time to 50. Name this test3. You might notice that this takes slightly longer. This is something to note with these methods. Depending on how complex your model is and how many iterations and time points you consider, running simulations can take a lot of time.

Reshaping data

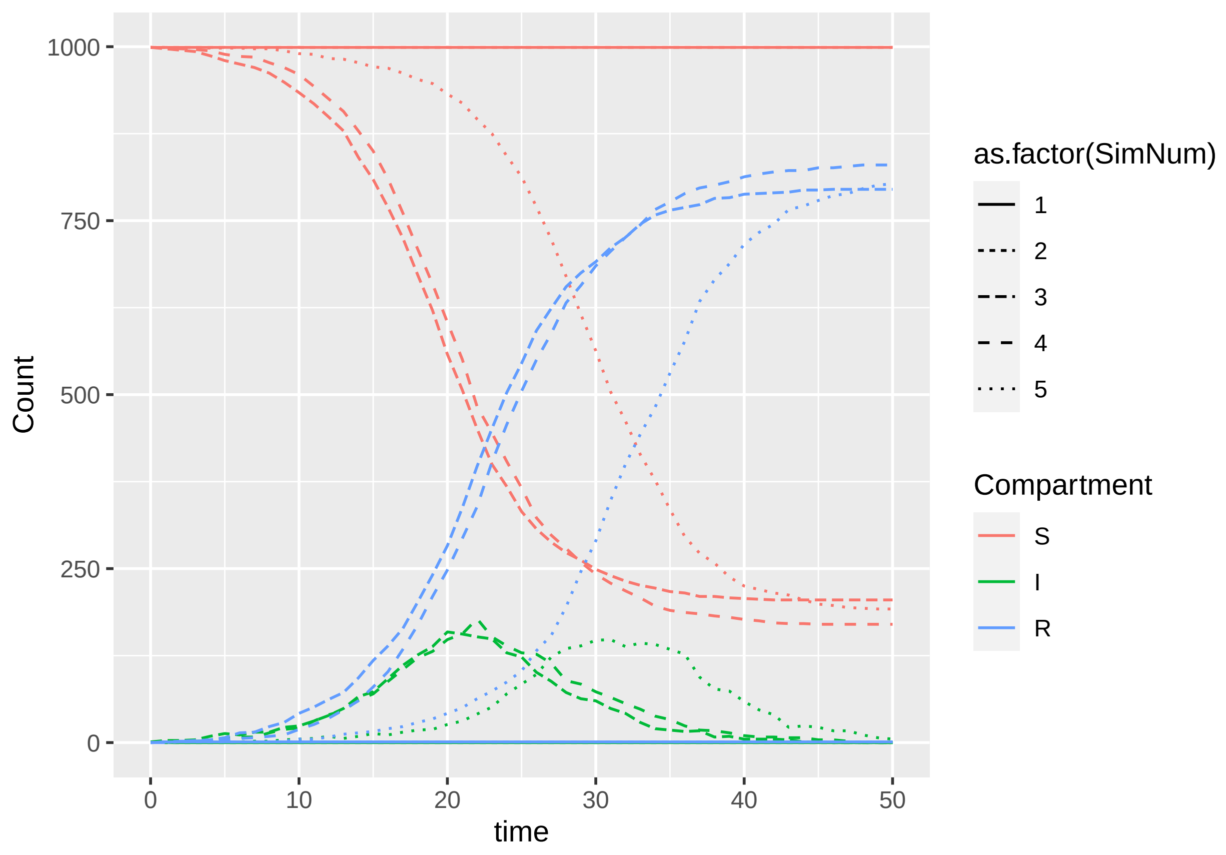
We have been manually converting our data frames for our compartmental models (e.g., `SIR_df` in the function `SIR_simulation`) from a form that has a column for each compartment (imperfectly speaking, "wide form" data) to a form that has a single column that specifies the compartment (called `compartment`) and a single column that specifies the number in the compartment, as well as a time column. There are various packages and functions that streamline this change, including the `melt` function in the reshape2 package, as well as the `gather` function in the tidyr package (http://www.cookbook-r.com/Manipulating_data/Converting_data_between_wide_and_long_format/). Modify the `SIR_simulation` function to return a data frame that you have manipulated using one of these methods.

```
SIR_simulation <- function(N, S0, I0, R_0, D, Time, nItr){
  ## Create an empty data frame with the same names
  sim_df <- data.frame(time=character(0),
                       S=character(0),
                       I=character(0),
                       R=character(0),
                       I_SI=character(0),
                       R_IR=character(0),
                       lambda_t=character(0),
                       SimNum=character(0))
  for (j in 1:nItr){
    SIR_df <- data.frame(time=0:Time,
                       S=rep(NA, Time+1),
                       I=rep(NA, Time+1),
                       R=rep(NA, Time+1),
                       I_SI=rep(NA, Time+1),
                       R_IR=rep(NA, Time+1),
                       lambda_t=rep(NA, Time+1),
                       ## Add a column to keep track of simulation number
                       SimNum=rep(j, Time+1))

    SIR_df$S[1] <- S0
    SIR_df$I[1] <- I0
    SIR_df$R[1] <- N-S0-I0
```

```r
    for (i in 2:(Time+1)){
      SIR_df$lambda_t[i] <- lambda_t_fcn(R_0=R_0, D=D, I_i=SIR_df$I[i-1], N=N)
      SIR_df$I_SI[i] <- rbinom(n=1, size=SIR_df$S[i-1], prob=SIR_df$lambda_t[i])
      SIR_df$R_IR[i] <- rbinom(n=1, size=SIR_df$I[i-1], prob=1/D)
      SIR_df$S[i] <- SIR_df$S[i-1]-SIR_df$I_SI[i]
      SIR_df$I[i] <- SIR_df$I[i-1]+SIR_df$I_SI[i]-SIR_df$R_IR[i]
      SIR_df$R[i] <- SIR_df$R[i-1]+SIR_df$R_IR[i]
    }
    sim_df <- rbind(sim_df, SIR_df)
  }
  ## Using melt from reshape2 package
  sim_df2 <- melt(data=sim_df, id.vars=c("time", "SimNum"),
                  measure.vars=c("S", "I", "R"),
                  variable.name = "Compartment",
                  value.name = "Count")

  return(sim_df2)
}
```

Test your function using one of the sets of arguments from above and plot your results. If you have used either melt or gather correctly when converting your data frame inside your function, then the data frame your function returns will be ready for plotting.

```r
set.seed(123)
test_newsim <- SIR_simulation(N=1000, S0=999, I0=1, R_0=2, D=3, Time=50, nItr=5)

ggplot(data=test_newsim, aes(x=time, y=Count)) +
  geom_line(aes(color=Compartment, linetype=as.factor(SimNum)))
```



Births and Deaths in a Stochastic Compartmental Model

Modify the function named `SIR_simulation_birth_death` below to include births and deaths in the model, drawing on your notes from Tuesday.

- Assume that individuals are only born into the susceptible compartment
- Assume that deaths can occur for individuals in any compartment
- Make the function sufficiently flexible to accommodate different mortality rates for infectious individuals versus other compartments
- Add storage for the births and deaths

As a refresher, the difference equations for a stochastic SIR model with births and deaths would be the following:

$$S_t = S_{t-1} - I_t^{(SI)} + B_t - M_t^S$$

where B_t is a random variable for the number of births at time t and follows either a $Poisson(\beta \times N_t)$ (β is the per capita birth rate) or a $Binomial(N_t, \beta)$. M_t^S is a random variable for the number of deaths from the susceptible compartment at time t and follows a $Binomial(S_{t-1}, \mu)$ distribution (μ is the mortality rate for the population).

$$I_t = I_{t-1} + I_t^{(SI)} - R_t^{(IR)} - M_t^I$$

where M_t^I is a random variable for the number of deaths from the infectious compartment at time t and follows a $Binomial(I_{t-1}, \mu^I)$ distribution (μ^I is the mortality rate for infectious individuals based on the infection).

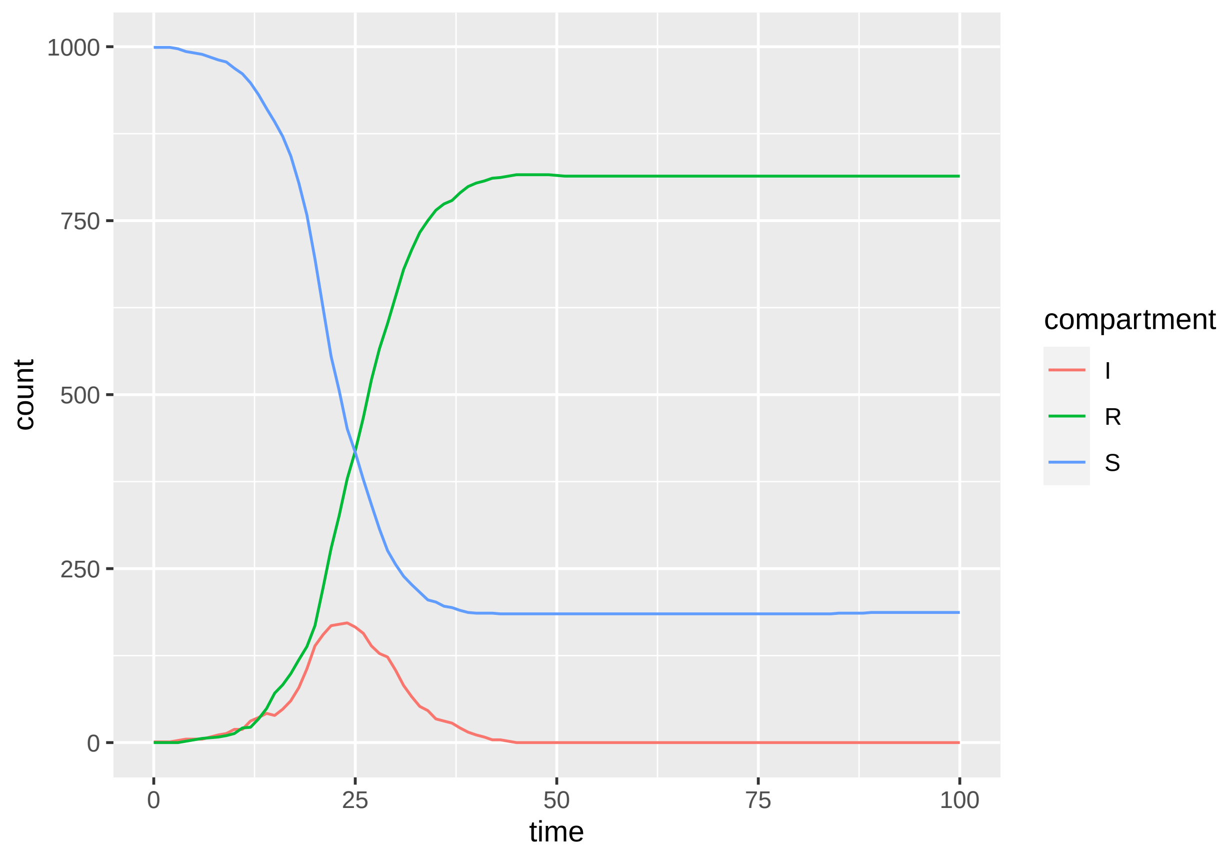
$$R_t = R_{t-1} + R_t^{(IR)} - M_t^R$$

where M_t^R is a random variable for the number of deaths from the susceptible compartment at time t and follows a $Binomial(R_{t-1}, \mu)$ distribution (μ is the mortality rate for the population).

```r
SIR_simulation_birth_death <- function(N0, S0, I0, R_0, D, b, m, mI, Time){
  SIR_df <- data.frame(time=0:Time,
                       S=rep(NA, Time+1),
                       I=rep(NA, Time+1),
                       R=rep(NA, Time+1),
                       I_SI=rep(NA, Time+1),
                       R_IR=rep(NA, Time+1),
                       B=rep(NA, Time+1),
                       M_S=rep(NA, Time+1),
                       M_I=rep(NA, Time+1),
                       M_R=rep(NA, Time+1),
                       lambda_t=rep(NA, Time+1),
                       N=rep(NA, Time+1))

  SIR_df$S[1] <- S0
  SIR_df$I[1] <- I0
  SIR_df$R[1] <- N0-S0-I0
  SIR_df$N[1] <- N0

  for (i in 2:(Time+1)){
    SIR_df$lambda_t[i] <- lambda_t_fcn(R_0=R_0, D=D, I_i=SIR_df$I[i-1], N=SIR_df$N[i-1])
    SIR_df$I_SI[i] <- rbinom(n=1, size=SIR_df$S[i-1], prob=SIR_df$lambda_t[i])
    SIR_df$R_IR[i] <- rbinom(n=1, size=SIR_df$I[i-1], prob=1/D)
    SIR_df$B[i] <- rpois(n=1, lambda=b*SIR_df$N[i-1])
    SIR_df$M_S[i] <- rbinom(n=1, size=SIR_df$S[i-1], prob=m)
    SIR_df$M_I[i] <- rbinom(n=1, size=SIR_df$I[i-1], prob=mI)
    SIR_df$M_R[i] <- rbinom(n=1, size=SIR_df$R[i-1], prob=m)
    SIR_df$S[i] <- SIR_df$S[i-1]-SIR_df$I_SI[i]+SIR_df$B[i]-SIR_df$M_S[i]
    SIR_df$I[i] <- SIR_df$I[i-1]+SIR_df$I_SI[i]-SIR_df$R_IR[i]-SIR_df$M_I[i]
    SIR_df$R[i] <- SIR_df$R[i-1]+SIR_df$R_IR[i]-SIR_df$M_R[i]
    ## Add an update to N to allow for the possibility the population
    ## size grows or shrinks with births/deaths
    SIR_df$N[i] <- SIR_df$S[i]+SIR_df$I[i]+SIR_df$R[i]
  }

  return(data.frame(time=rep(0:Time, 3),
                    compartment=rep(c("S","I", "R"), each=(Time+1)),
                    count=c(SIR_df$S, SIR_df$I, SIR_df$R)))
}
```

Test your function to make sure it works. You will need to choose values for the birth rate and mortality rates. What are reasonable numbers for this?

```r
test_BD <- SIR_simulation_birth_death(N0=1000, S0=999, I0=1, R_0=2, D=3,
                                      b=1/(70*365), m=1/(70*365), mI=1/(68*365),
                                      Time=100)
ggplot(data=test_BD, aes(x=time, y=count)) +
  geom_line(aes(color=compartment))
```



Modify the function above (again) to accommodate multiple simulations. Basically you are combining things that we did for the `SIR_simulation` and `SIR_simulation_birth_death` functions.