Using data.table for simple agent/individual based models

Kaare Græsbøll
Email : kagr@dtu.dk
Department of Applied Mathematics and Computer Science, Technical University of Denmark

The package data.table for R is an extension to the data.frame which is known for its ability to handle large amount of data fast and efficiently. A side effect of the syntax of data.table is that is allows for writing individual based models of disease spread with ease.

data.table syntax

Basic
The general form of data.table syntax is:

```
DT[i, j, by]
```

`i:` handles subsetting by rows similar to data.frame.

`j:` subset AND compute on columns

`by:` group by columns by specifying a list of columns or a character vector of column names or even expressions

Special
Generally all R functions work with data.table. However data.table has a few extra to boost efficiency of code. Two of which we use here:

`.N` holds the number of rows in the current group. A more efficient way of doing `length()`.

`:=` assign by reference. A more memory efficient way of updating inside a data.table. It has a side-effect which correspond to merging results onto original dataset.

Example
When counting number of infected pigs inside each pen:

```
ibm[,NinfPen := sum(infStatus), by = penID]
```

The `:=` operator updates the NinfPen column with the number of infected pigs per pen, for all pigs.

When assigning disease

```
ibm[infStatus==0L , rbinom(.N, 1, prob = ptrans)]
```

The `.N` variable gives the numbers to draw.

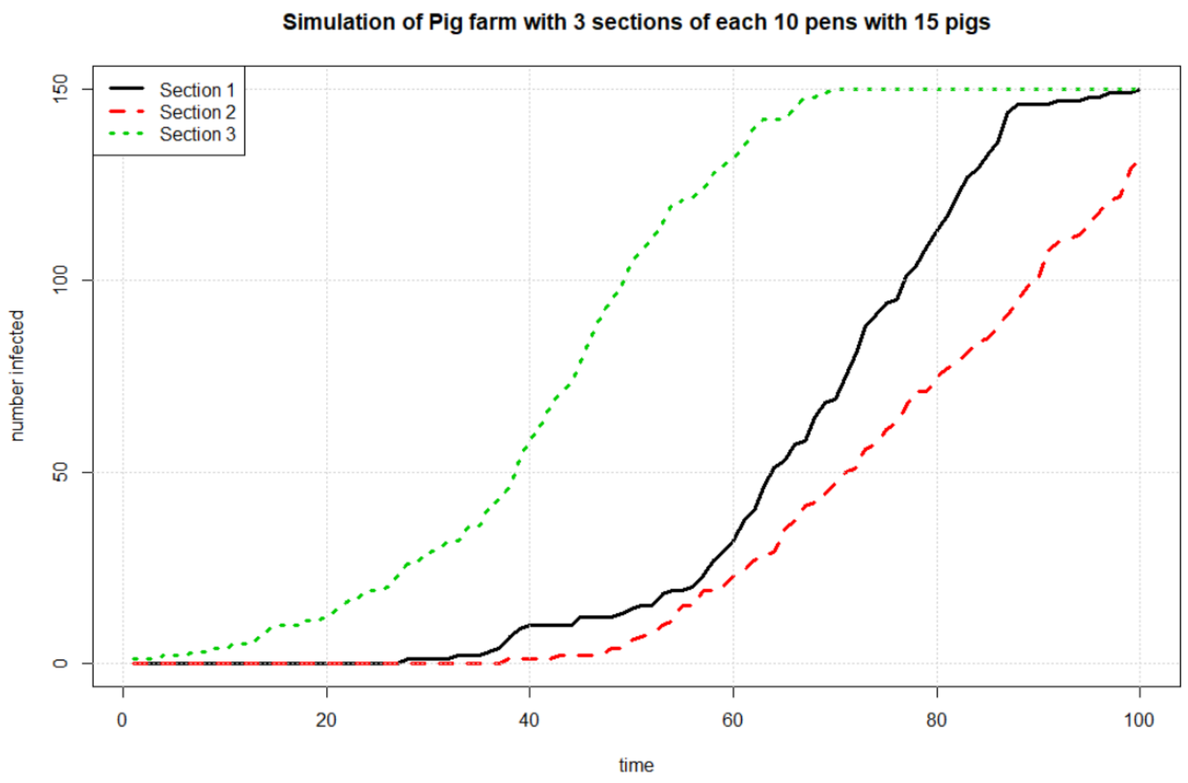
Code
Right: The complete code necessary to model transmission of a disease on a pig farm with individual pigs in pens, belonging to different sections of the farm.

Bottom: Figure showing number of infected pigs per section.

This code could easily be expanded by adding more features to individual pigs to modify i.e. transmission rates, or have individual cure rates etc.

Repetitions could be included by adding an extra group, so that confidence intervals could be estimated.

Code could be easily wrapped into a Shiny for interactive use.




```
nsecs  <- 3  # number sections
npens  <- 10 # pens per section
npigsp <- 15 # pigs per pen

npigs <- nsecs * npens * npigsp # number of pigs

# create ibm

ibm <- data.table(
  ID        = 1:npigs,
  penID     = rep(1:(npens*nsecs), each = npigsp),
  sectionID = rep(1:nsecs, each = npens * npigsp),
  infStatus   = 0
)

# infection rates
irPen  <- 0.2
irSec  <- 0.02
irFarm <- 0.002

# randomly infect some pigs
ibm[sample(ID,1),infStatus := 1]

# wrap in time for loop

ntime <- 100
Ninf <- matrix(0, ncol = nsecs, nrow = ntime)
Ninf[1,] <- ibm[, .( Ninf = sum(infStatus) ), sectionID]$Ninf

for (i in 2:ntime)
{
  # update
  ibm[,NinfPen := sum(infStatus), by = penID]     # No. infected by pen
  ibm[,NinfSec := sum(infStatus), by = sectionID] # No. infected by section

  # transmission probability
  ibm[,ptrans := 1 -
        exp( - irPen * NinfPen/npigsp -
               irSec * (NinfSec-NinfPen)/ ((npens-1) * npigsp) -
               irFarm * (sum(infStatus) - NinfSec) / ((nsecs-1) * npens *
npigsp)  )]

  # transmission
  ibm[infStatus==0L , infStatus := rbinom(.N, 1, prob = ptrans)]

  # count infected pigs per section
  Ninf[i,] <- ibm[, .( Ninf = sum(infStatus) ), sectionID]$Ninf
}
```

Pros
Individual based modelling in data.table is advantageous because of the build in counting by group feature, which makes syntax concise.

Working directly in R using inline testing means fast development.

A good way of making hypothesis generating models.

Functions applied on data.table do not copy data (fewer scoping issues)

Cons
Even though data.table is faster than data.frame it cannot compete with C or Fortran in terms of speed. So for large models this is probably not a good solution.