**AML 610 Fall 2015 Homework #1**

**Submit all files to** [**smtowers@asu.edu**](mailto:smtowers@asu.edu)**.**

**Please submit with name format hwk1\_<first name>\_<initial of last name> Please provide your R files, latex, pdf and bibtex files, and a Word file that gives the output to your screen, plots, etc.**

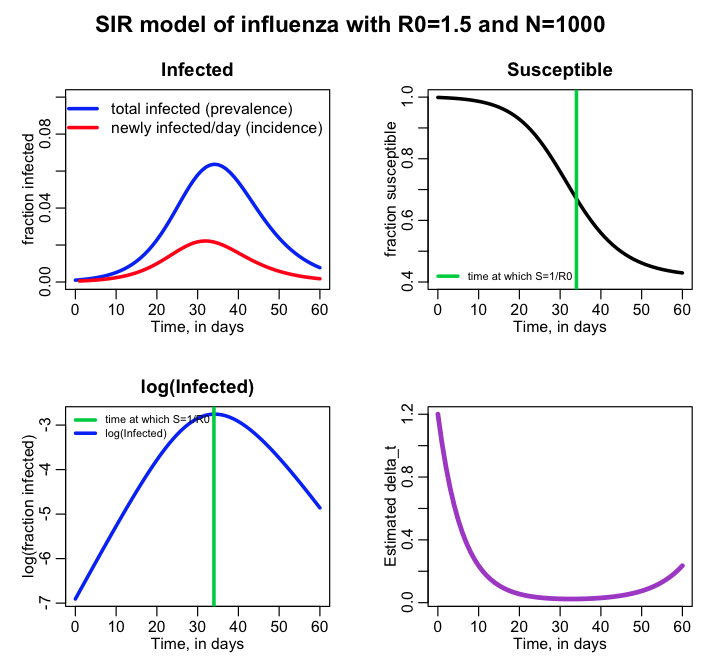
**All code must conform to good coding practices, as described in** [**http://sherrytowers.com/2012/12/14/good-programming-practices-in-any-language/**](http://sherrytowers.com/2012/12/14/good-programming-practices-in-any-language/) **and all plots must conform to good plotting practices, as described in** [**http://sherrytowers.com/2013/01/04/good-practices-in-producing-plots/**](http://sherrytowers.com/2013/01/04/good-practices-in-producing-plots/)

1. Using Google Scholar, find three papers on a topic that interests you that involve some kind of stochastic modeling method(s) in the analyses. Read the entire web post “How to write a good scientific paper (and get your work published as painlessly as possible)” which can be found at <http://sherrytowers.com/?p=1876> In the post, I describe the seven key elements of scientific papers, identified by Lacum et al (2014). Read the Lacum paper (it is linked off of the web page). For each of your papers, provide a compiled latex document, with the papers cited using bibtex, in which you give an itemized list of short sentences summarizing each of the 7 key elements (you can put the summary of all the papers in the same document**). If a paper does not discuss one or more of the key elements, point that out.** Please also provide the PDF of the papers. Also include your annotated bibtex file in the files you submit.
2. A) In this exercise, you are going to estimate the dynamical time step needed to, on average, produce one compartmental transition in an SIR model. This is good practice for you, because calculating these time steps is exactly what you’ll need to do when you code up SDEs, MCMC, or ABM stochastic simulations. Read the module “How to download an R script from the internet and run it” at <http://sherrytowers.com/?p=1902> Follow all the instructions provided to download the R script http://www.sherrytowers.com/sir\_func.R, and the R script http://www.sherrytowers.com/sir.R. The sir.R script uses methods the R deSolve to solve the ODE’s corresponding to the SIR model (in this case, with recovery rate 1/gamma=3 days, and R0=1.5). It uses functions defined in the sir\_func.R script to do this.

Change the population size in the sir.R script to npop=1000, and change the end time of the simulation (in vector vt) to 60 days. Run the script.

Read the module [Calculating the time step when using numerical methods to solve ODE's](http://www.sherrytowers.com/?p=2603)

From the information that is produced by the sir.R script (namely, the time series of the numerical solutions for S and I), for each value of S and I calculate the time step needed to produce, on average, one change in the compartment values at that time. Plot this time step versus time. You should get a plot that looks something like this:



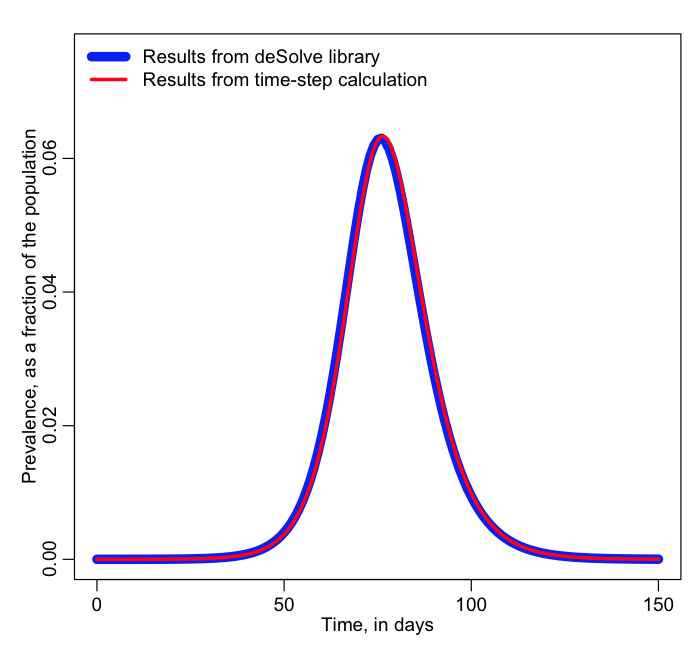
What is the minimum value of the time step? And at what time does it occur? Why is the time step longer near the end and beginning of the epidemic?

B) Now we’ll examine how well a constant time step performs. Go to the module “Compartmental modeling without calculus”

<http://www.sherrytowers.com/2012/12/11/compartmental-modelling-without-calculus>

In that module, there is a link to the R script

<http://www.sherrytowers.com/sir_without_calculus.R>, that uses a for loop with small time steps and Euler’s method to numerically solve the SIR model. Modify this script so that it also solves the model using the R deSolve library, and creates a plot that overlays the two calculated results for the prevalence, I. Like so:

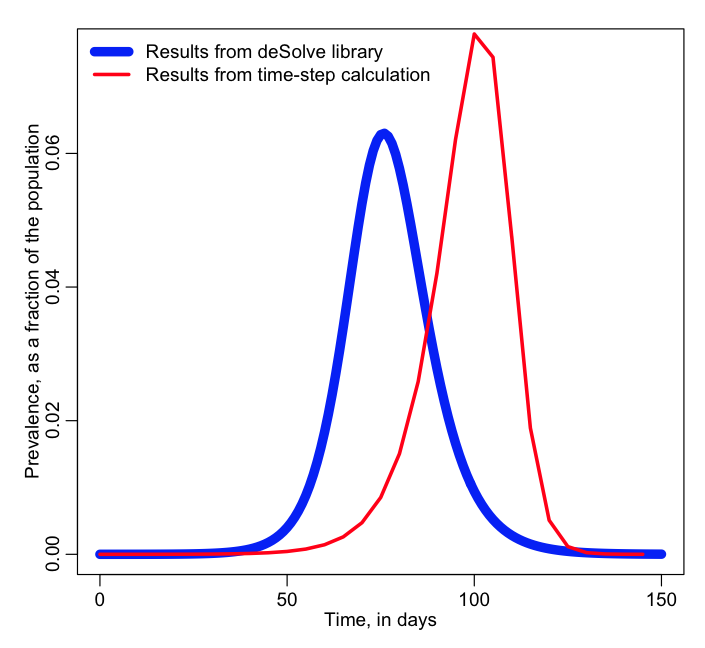


(use npop=1000000, delta\_t=0.1 in your calculations)

Comment on the differences you observe between the two calculations (note: the 4th order Runge-Kutta deSolve calculation is always more exact than Euler’s method)

Does a constant 0.1 days appear to be a sufficiently small time step?

2C) Repeat 2B) except now use a time step of 5 days in the simple time step calculation:



Do you think that 5 days is a sufficiently small time step?

This kind of exercise underlines the importance of a sufficiently small time step.

How can you tell for sure that your time step is small enough? Halve it, and redo the simulation. If the results do not substantially change, then the time step is small enough. If the results do change, then halve it again (and so on) until they don’t.

For stochastic simulations, it is a bit more complicated because you would have to run several realisations of the algorithm to ensure that the collected results look, on average, like the results obtained with a doubled time step.

You should **always** do the exercise of halving your time step to ensure that your results don’t change substantially!

2D) Now change the script to npop=1000 and end the simulation at time=60 days. Read the module [Calculating the time step when using numerical methods to solve ODE's](http://www.sherrytowers.com/?p=2603)

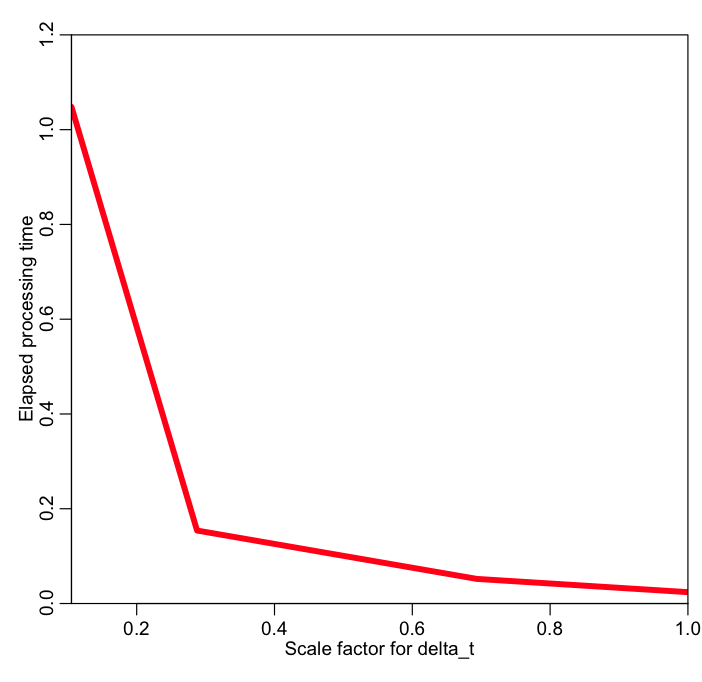
Do the simulation with Euler’s method, using a time step dynamically calculated using the mean of the Exponential distribution with lambda=1/delta\_t. Also do it for the 50th, 25th, and 10th percentiles of the distribution, and overlay all the results, like so (note that I make the foreground lines thinner so you can still see the background lines):



Which dynamic time step calculation results in the best agreement with the more exact results of the deSolve library?

**NOTE: if you are not using a for loop to assess the effect of the different scale factors for delta\_t, you are doing it wrong! The code to calculate the above curves is exactly the same, except for the change of scale factor for delta\_t.**

2E) read the R help related to the proc.time() function in R, which returns the processing time the current R process has taken. At the beginning of the for loop over the delta\_t scale factors in problem 2D, get the elapsed time. Do the same thing at the end of the for loop, and calculate the difference and store it in a vector. This is the elapsed processing time needed to calculate each of the curves in problem 2D. Now plot the elapsed time vs the scale factor for delta\_t, like so:



For the SIR model the processing time for the simulation is not linear in the scale factor for delta\_t. *Explain why this is the case.*

This kind of exercise is important when assessing the scalability (growth rate) of the processing time of your simulations (particularly important for stochastic simulations that can be computationally intensive). Population size can also affect processing time in a non-linear way in compartmental models. <http://en.wikipedia.org/wiki/Big_O_notation>