SEIR Program Suite Description and Instructions

Program Overview

The SEIR suite contains four R programs, one that implements the mathematical model, one that runs a test on that implementation, and two that use it to set up virtual experiments.

seir_sim.R is a function program that accepts input data called from a keyboard command or a script file and produces a table of results for the class counts in each day. The input parameters include β, η, and γ along with parameters that indicate the initial fractions of the infected and removed classes.

SEIR_simtest.R is a script that runs one simulation using seir_sim.R. A single plot shows the susceptible, latent, infectious, and removed fractions of the population for a given scenario. Scenarios are specified using the basic reproductive number, the average durations of the latent and infectious phases, and the initial infectious and removed fractions.

SEIR_simplot.R is a script that uses seir_sim.R to prepare a set of plots that show the time history of the scenario. Two plots show the susceptible and infected populations over time.

SEIR_paramstudy.R is a script that uses seir_sim.R to prepare a set of plots that show how certain key quantities change when one of the parameters is varied:

1. The maximum fraction of infected individuals;
2. The time when the maximum number infected is reached;
3. The final percentage still at risk at the end of the outbreak;
4. The number of days before the total of infected people is below a target threshold.

Getting Started

We assume here that the reader already knows how to use the R editor to create and execute a script file, which is a set of instructions to be executed according to a prescribed flow of control. No significant programming experience is required. Users should start with SEIR_simtest.R, which is designed to ensure that the main program is functioning correctly and to provide an easy entry into R programming.

SEIR_simtest.R is a very simple program, with no programming structures such as loops or conditionals. The file contains the sir_sim function–this description begins after the line marking the start of SEIR_simtest. The program itself contains only 26 lines of code, along with section dividers (lines that start with ##), comments (lines that start with #), and blank lines.

The lines in the SCENARIO DATA, COMMON DATA, and COMPUTATION sections are simple assignment statements. These are statements of the form "name=value" that assign either a specific value or a value computed from a formula or function to the name. The scenario is defined in the SCENARIO DATA section. This is the only section in the program that the user needs to change. This structuring is good programming practice, as it means that all of the lines that need to be

specified for a scenario are grouped together, while other blocks of code contain only lines that do not need to be changed when the scenario changes.

The COMPUTATION section begins with a call to the seir_sim function. The function call uses the name of the function followed by a list of arguments in parentheses. The function file uses these arguments in place of simple assignment statements. Its output consists of a matrix that contains columns for each of the four state variables. These variables are extracted from `results`, and then a list of new infections is calculated as the daily decrease in class S.

The OUTPUT section begins with the construction of a list of times for the plots. Then comes a statement to plot S, followed by statements that use the "lines" function to add curves for E, I, and R to the same plot. The first two arguments of the legend statement identify the x and y coordinates for the lower left corner of the legend. This statement needs to be modified for different scenarios because R lacks built-in coding to find a good legend location.

The remaining lines produce the text output for the program. The "cbind" statement makes a table out of 6 lists of values. The last two lines give the maximum infectious fraction and the final susceptible fraction. Note that R statements do not generally produce output unless they are lines such as `max(I)'` and `S[days+1]` that produce a value without assigning it to a name.

SEIR_simtest.R can be used to run any scenario for the SEIR epidemic model simply by changing the six lines in the scenario data section. Sometimes you will want to tinker with graphics statements to improve the appearance and/or design of a plot.

Using SEIR_simplot and SEIR_paramstudy

The scripts are constructed so that they can be applied to different experiments with a minimum of changes. These changes are identified in the comments of the script files, and are described here in more detail.

It is important to understand the logical structure used to organize these programs. Each is subdivided into sections with a particular name and purpose.

1. The scripts begin with some brief documentation that describes the program and how to use it. Next is the code for sir_sim before the code of sir_sim_test.

2. The first programming section contains default scenario data, which are the values that will be assumed for the parameters to be used as input values for seir_sim if that particular parameter is not the experiment parameter. These default values may need to be changed, depending on the experiment.

3. The next section contains the data for the experiment parameter, described in the program as the "Independent Variable". The program is designed to require minimal changes for different experiments; most of those changes are in this section.

    (a) First, the set of parameter values to be used for the independent variable must be specified with a generic name. The actual parameter you have chosen will be identified elsewhere.

i. The values are listed in SEIR_simplot under the generic name "xvals" and are specified individually using R syntax for a row vector or list. Each of the parameter values in this list will generate its own set of curves in the three-panel plot. Generally a set of 3 to 8 values is best.

ii. In SEIR_paramstudy, the independent variable values are used as the horizontal coordinate on each plot, so a much larger set is necessary. The user specifies the first value, the last value, and the count of values, keeping in mind that the end point values are both counted. For example, if you want the values 0, 0.05, 0.1, and so on up to 1.0, you choose first=0, last=1, and N=21, dividing the interval from 0 to 1 into 20 equal parts.

(b) The set of values is the only thing that needs to be defined in the INDEPENDENT VARIABLE DATA section of SEIR_simplot. SEIR_paramstudy also requires the user to specify the name of the parameter to be used as the horizontal axis label. This specification only applies to the graph; so far the program does not know which of the parameters is to take on the values predefined using first, last, and N.

4. Next comes the INITIALIZATION section, which does basic housekeeping tasks such as setting up plot windows and defining any data structures needed for the data.

5. The programs conclude with sections for computation and output, with the graphs being created in the computation section if they occur inside a loop or in an output section if they are only produced after all the data is collected.

6. The key to the program structure is that each parameter has a default value that will be used in all scenarios, except that one of the parameters will instead use the values specified either as xvals or using first, last, and N. The function call requires one value to be passed in for each of the parameters. The first line inside the main program loop identifies the name of the parameter whose values were defined in the Independent Variable section. This line overwrites the previous value each time through the loop; meanwhile, the other parameters have only been given default values, so these are also used. The default value for the independent variable parameter is never used, but having it specified in the data section allows for a minimum of changes as the program is repurposed for a new experiment.

7. Under some circumstances, the user might want to tinker with the details of the plot specifications. R lacks the sophisticated automatic graph layout statements of Matlab, so it is common to have to redesign graphs after running a new scenario. Your instructor should be able to help you do this.

SPUR modifications

Addressing questions about the impact of isolation of individuals with symptoms requires a more sophisticated model, such as the SPUR model described in the Student Notes. These changes are needed to convert the SEIR programs to SPUR. Start by saving copies of SEIR_simtest.R and SEIR_paramstudy.R as SPUR_simtest.R and SPUR_paramstudy.R. Then make the indicated changes.

For spur_sim within the program SPUR_simtest:

1. Change the function argument list to `(beta,sigma,gamma,I0,q,V,target)`.
2. In INITIALIZATION, add these lines to calculate `P0` and `U0`:

   ```
   b = beta-sigma+gamma
   c = -(1-q)
   rho = (sqrt(b.\^2-4*beta*c)-b)/(2*beta)
   P0 = I0/(1+rho)
   U0 = rho*P0
   ```

   Also delete `-E0` from the formula for `S0` and change the formula for `Y` to `Y = c(S0,P0,U0)`
3. In OUTPUT, add these lines before the `return` statement:

   (a) `S = results[,1]`

   (b) `I = results[,2]+results[,3]`

   (c) `R = results[,4]`

   (d) `results = cbind(S,I,R)`
4. Change the FUNCTION FOR THE DIFFERENTIAL EQUATION to match the model given in P1-2, Student Notes. This will require you to change the splitting of components to reflect the variable order S-P-U, add a line `I=P+U`, replace the formulas for `Ep` and `Ip` with two formulas for `Pp` and `Up`, and change the assembly line to use the derivatives `Pp`, and `Up` instead of `Ep` and `Ip`.

For SPUR_simtest:

1. In SCENARIO DATA, set `R0=5`, delete the `eta`, `gamma`, and `E0` lines, and add assignments `T=10`, `Tp=2`, and `q=0.5`.
2. In INITIALIZATION, add `sigma=1/Tp` and `gamma=1/(T-Tp)` before the formula for `beta`.
3. In COMPUTATION, change the formula for `results` to match the function name and argument list of spur_sim. Also delete the `E` line and change the column numbers in the `I` and `R` lines from 3 and 4 to 2 and 3.
4. In OUTPUT, delete the statement that plots `E` and delete "`E`", and "`orange`", from the `legend` statement.

5. Run SPUR_simtest to make sure everything works. Error messages referring to the computation line indicate errors in spur_sim. After you fix these, rerun SPUR_simtest until the program works.

6. It is a good idea to update the comments in the programs. This makes it easier to understand programs you haven't looked at in a long time.

For SPUR_paramstudy:

1. Replace the entire spur_sim code with a copy of the modified code from SPUR_simtest.

2. Modify DEFAULT SCENARIO DATA and COMPUTATION in the same way as in SPUR_simtest.

3. Set the experiment to vary the parameter `q` from 0 to 1.