Modelling Infectious Diseases with Kendrick

<Put the authors here>

Version of 2015-04-24

Contents

Chapter 1

Introduction

Some examples from this book are coming from the book of M. Keeling & P. Rohani "Modeling Infectious Diseases in Humans and Animals". There is a website with on-line material for the book, where you can find the programs and the background of each program in C++, FORTRAN and Matlab.

Kendrick is a domain-specific modelling language that provide tools in order to design, explore and visualize your epidemics models. Kendrick is an embedded DSL and use the Pharo programming language as its host language.

In order to use Kendrick, you need first to install the last version of Pharo in your computer.

1.1 How to install Pharo

Pharo is available as a free download from http://pharo.org/download. Click the button for your operating system to download the appropriate `.zip` file. For example, the full Pharo 4.0 distribution for Mac OS X will be available at http://files.pharo.org/platform/Pharo4.0-mac.zip.

Once that file is unzipped, it will contain everything you need to run Pharo (this includes the VM, the image, and the sources, as explained below).

1.2 Pharo components

Like many Smalltalk-derived systems, Pharo currently consists of three main components. Although you do not need to deal with them directly for the purposes of this book, it is important to understand the roles that they play.

1. The **image** is a current snapshot of a running Pharo system, frozen in time. It consists of two files: an *.image* file, which contains the state of all of the objects in the system (including classes and methods, since they are objects too), and a *.changes* file, which contains the log of all of the changes to the source code of the system. For Pharo 4.0, these files are named `Pharo4.0.image` and `Pharo4.0.changes`. These files are portable across operating systems, and can be copied and run on any appropriate virtual machine.

2. The **virtual machine** (VM) is the only part of the system that is different for each operating system. Pre-compiled virtual machines are available for all major computing environments. (For example, on Windows, the VM file is named `Pharo.exe`).

3. The **sources** file contains the source code for all of the parts of Pharo that don't change very frequently. For Pharo 4.0, this file is named `PharoV40.sources`.

As you work in Pharo, the *.image* and *.changes* files are modified (so you need to make sure that they are writable). Always keep these two files together. Never edit them directly with a text editor, as Pharo uses them to store the objects you work with and to log the changes you make to the source code. It is a good idea to keep a backup copy of the downloaded *.image* and *.changes* files so you can always start from a fresh image and reload your code.

The *.sources* file and the VM can be read-only, and can be shared between different users. All of these files can be placed in the same directory, but it is also possible to put the Virtual Machine and sources file in separate directory where everyone has access to them. Do whatever works best for your style of working and your operating system.

1.3 Launching Pharo

To start Pharo, double click on the Pharo executable (or, for more advanced users, drag and drop the `.image` file onto the VM, or use the command line).

On **Mac OS X**, double click the `Pharo4.0.app` bundle in the unzipped download.

On **Linux**, double click (or invoke from the command line) the `pharo` executable bash script from the unzipped Pharo folder.

On **Windows**, enter the unzipped Pharo folder and double click `Pharo.exe`.

In general, Pharo tries to "do the right thing". If you double click on the VM, it looks for an image file in the default location. If you double click on

an `.image` file, it tries to find the nearest VM to launch it with.

Once you have multiple VMs (or multiple images) installed on your machine, the operating system may no longer be able to guess the right one. In this case, it is safer to specify exactly which ones you meant to launch, either by dragging and dropping the image file onto the VM, or specifying the image on the command line (see the next section).

Launching Pharo via the command line

The general pattern for launching Pharo from a terminal is:

```
<Pharo executable> <path to Pharo image>
```

Linux command line

For Linux, assuming that you're in the unzipped `pharo4.0` folder:

```
./pharo shared/Pharo4.0.image
```

Mac OS X command line

For Mac OS X, assuming that you're in the directory with the unzipped `Pharo4.0.app` bundle:

```
Pharo4.0.app/Contents/MacOS/Pharo Pharo4.0.app/Contents/Resources/Pharo4.0.
    image
```

Incidentally, to drag-and-drop images on Mac OS in Finder, you need to right-click on `Pharo4.0.app` and select 'Show Package Contents'.

Windows command line

For Windows, assuming that you're in the unzipped `Pharo4.0` folder:

```
Pharo.exe Pharo4.0.image
```

1.4 The World Menu

Once Pharo is running, you should see a single large window, possibly containing some open workspace windows (see Figure 1.1). You might notice a menu bar, but Pharo mainly makes use of context-dependent pop-up menus.

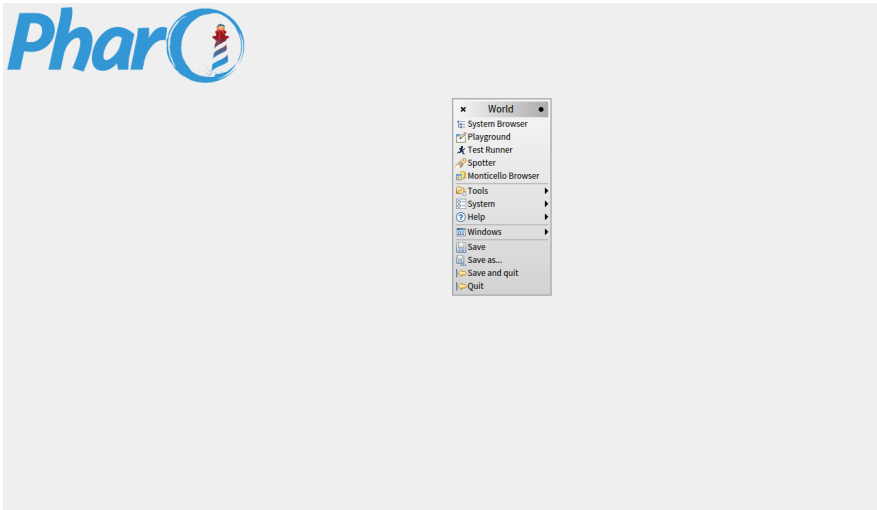

Figure 1.1: Pharo 4.0 window with World Menu activated.

Clicking anywhere on the background of the Pharo window will display the World Menu. World Menu contains many of the Pharo tools, utilities and settings.

You will see a list of several core tools in Pharo, including the class browser and the workspace.

Chapter 2

Introduction to Simple Epidemic Model

2.1 Simple SIR (without births and deaths)

Program 2.1 is a simple SIR model (page 19 of the book). These are the equations and the code of the model:

Equations

$$\frac{dS}{dt} = -\beta * S * I \tag{2.1}$$

$$\frac{dI}{dt} = \beta * S * I - \gamma * I \tag{2.2}$$

$$\frac{dR}{dt} = \gamma * I \tag{2.3}$$

Pharo code

```
|solver system dt beta gamma values stepper diag colors maxTime|
dt := 1.0.
beta := 1.4247.
gamma := 0.14286.
maxTime := 70.0.
system := ExplicitSystem block: [ :x :t| |c|
    c := Array new: 3.
    c at: 1 put: (beta negated) * (x at: 1) * (x at: 2).
    c at: 2 put: (beta * (x at: 1) * (x at: 2)) - (gamma * (x at: 2)).
```

```
    c at: 3 put: gamma * (x at: 2).
    c
    ].
stepper := RungeKuttaStepper onSystem: system.
solver := (ExplicitSolver new) stepper: stepper; system: system; dt: dt.
state := { 1-1e-6. 1e-6. 0}.
values := (0.0 to: maxTime by: dt) collect: [ :t| |state| state := stepper doStep: state
                                                time: t stepSize: dt ].
diag := OrderedCollection new.
colors := Array with: Color blue with: Color red with: Color green.
1 to: 3 do: [ :i|
   diag add:
      ((GETLineDiagram new)
         models: (1 to: maxTime+1 by: 1);
         y: [ :x| (values at: x) at: i ];
         color: (colors at: i))
    ].
builder := (GETDiagramBuilder new).
builder compositeDiagram
   xAxisLabel: 'Time in days';
   yAxisLabel: 'Number of Individuals';
   regularAxis;
   diagrams: diag.
builder open.
```

2.2 SIR model with births and deaths

Equations

$$\frac{dS}{dt} = \mu - \beta * S * I - \mu * S \tag{2.4}$$

$$\frac{dI}{dt} = \beta * S * I - \gamma * I - \mu * I \tag{2.5}$$

$$\frac{dR}{dt} = \gamma * I - \mu * R \tag{2.6}$$

Pharo code

```
|solver system dt beta gamma values stepper diag mu colors maxTime|
dt := 1.0.
mu := 1/(70*365.0).
beta := 520/365.0.
gamma := 1/7.0.
maxTime := 60*365.
```

```
system := ExplicitSystem block: [ :x :t| |c|
    c := Array new: 3.
    c at: 1 put: mu − (beta ∗ (x at: 1) ∗ (x at: 2)) − (mu ∗ (x at:1)).
    c at: 2 put: (beta ∗ (x at: 1) ∗ (x at: 2)) − (gamma ∗ (x at: 2)) − (mu ∗ (x at:2)).
    c at: 3 put: (gamma ∗ (x at: 2)) − (mu ∗ (x at: 2)).
    c
    ].

stepper := RungeKuttaStepper onSystem: system.
solver := (ExplicitSolver new) stepper: stepper; system: system; dt: dt.
state := { 0.1. 1e−4. 1−0.1−1e−4}.
values := (0.0 to: maxTime by: dt) collect: [ :t| |state| state := stepper doStep: state
                                                time: t stepSize: dt ].
diag := OrderedCollection new.
colors := Array with: Color blue with: Color red with: Color green.
1 to: 3 do: [ :i|
   diag add:
      ((GETLineDiagram new)
         models: (1 to: maxTime+1 by: 1);
         y: [ :x| (values at: x) at: i ];
         color: (colors at: i))
    ].
builder := (GETDiagramBuilder new).
builder compositeDiagram
   xAxisLabel: 'Time in days';
   yAxisLabel: 'Number of Individuals';
   regularAxis;
   diagrams: diag.
builder open.
```

2.3 SIR model with disease induced mortality and density dependent transmission

Equations

$$\frac{dX}{dt} = \nu - \beta * X * Y - \mu * X \tag{2.7}$$

$$\frac{dY}{dt} = \beta * X * Y - \frac{\gamma + \mu}{1 - \rho} * Y \tag{2.8}$$

$$\frac{dZ}{dt} = \gamma * Y - \mu * Z \tag{2.9}$$

2.4 SIR model, disease induced mortality and frequency dependent transmission

Equations

$$\frac{dX}{dt} = \nu - \beta * X * Y/N - \mu * X \tag{2.10}$$

$$\frac{dY}{dt} = \beta * X * Y/N - \frac{\gamma + \mu}{1 - \rho} * Y \tag{2.11}$$

$$\frac{dZ}{dt} = \gamma * Y - \mu * Z \tag{2.12}$$

Pharo code

2.5 SIS model without births or deaths

Equations

$$\frac{dS}{dt} = \gamma * I - \beta * S * I \tag{2.13}$$

$$\frac{dI}{dt} = \beta * S * I - \gamma * I \tag{2.14}$$

2.6 SEIR model with births and deaths

2.7 SIR with a carrier state

Chapter 3

Host Heterogeneities

Chapter 4

Multi-Pathogen/Multi-Host Models