Part 3: Modelling epidemics in continuous time and using stochastic processes

Darren Wilkinson

darrenjw.github.io

Abstracting model structure from simulation approach

The structure on an SIR model

Blah

```
import smfsb._
import breeze.linalg._
import breeze.numerics._

val dMod = SpnModels.sir[IntState]()
// dMod: Spn[IntState] = UnmarkedSpn(
//   List("S", "I", "R"),
//   1  1  0
// 0  1  0  ,
//   0  2  0
// 0  0  1  ,
//   smfsb.SpnModels$$$Lambda$5366/1227579254@5671178df
// )
val stepSIRds = Step.gillespie(dMod)
// stepSIRds: (IntState, Time, Time) => IntState = smfsb.S
val tsSIRds = Sim.ts(DenseVector(100,5,0), 0.0, 10.0, 0.05
// tsSIRds: Ts[IntState] = List(
```

Plot

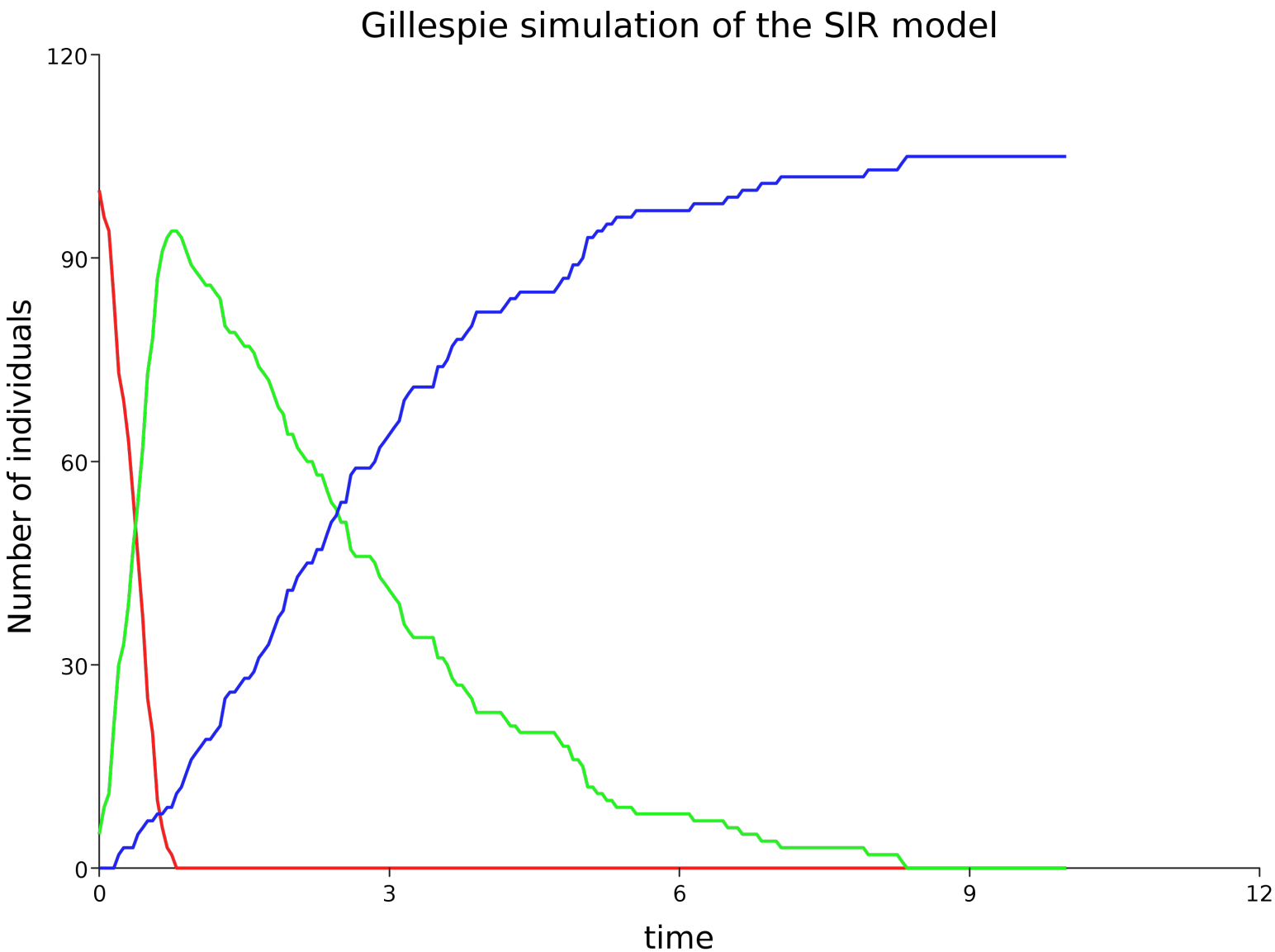

Figure 1:

SEIR

```
val stepSEIR = Step.gillespie(SpnModels.seir[IntState]())
// stepSEIR: (IntState, Time, Time) => IntState = smfsb.St
val tsSEIR = Sim.ts(DenseVector(100,5,0,0), 0.0, 20.0, 0.05
// tsSEIR: Ts[IntState] = List(
//   (0.0, DenseVector(100, 5, 0, 0)),
//   (0.05, DenseVector(100, 5, 0, 0)),
//   (0.1, DenseVector(100, 5, 0, 0)),
//   (0.1500000000000002, DenseVector(100, 5, 0, 0)),
//   (0.2, DenseVector(100, 5, 0, 0)),
//   (0.25, DenseVector(100, 5, 0, 0)),
//   (0.3, DenseVector(100, 5, 0, 0)),
//   (0.35, DenseVector(100, 5, 0, 0)),
//   (0.3999999999999997, DenseVector(100, 5, 0, 0)),
//   (0.4499999999999996, DenseVector(100, 5, 0, 0)),
//   (0.4999999999999994, DenseVector(100, 5, 0, 0)),
//   (0.5499999999999999, DenseVector(100, 5, 0, 0)),
//   (0.6, DenseVector(100, 5, 0, 0)),
//   (0.65, DenseVector(100, 5, 0, 0)),
```