Chapter 2

2.3 Numerical integration of the SIR model

```
using OrdinaryDiffEq
using LabelledArrays
using DataFrames
using Plots;
```

Step 1: define the function.

```
function sirmod(u, p, t)
    S,I,R = u
      = p.
      = p.
      = p.
    N = p.N
    dS =  *(N-S) -  *S*I/N
    dI =  *S*I/N - ( + )*I
    dR =  *I -  *R
    [dS, dI, dR]
end;
```

Steps 2-4: define the time, the parameters, and the initial conditions.

```
t0 = 0.0
t1 = 0.5
 t = 1.0/365
p = LVector( =0.0, N=1.0, R =4.0,  =365.0/14)
p = [p; LVector( =p.R *p.  + p. )]
u0 = [0.999, 0.001, 0.0] .* p.N;
```

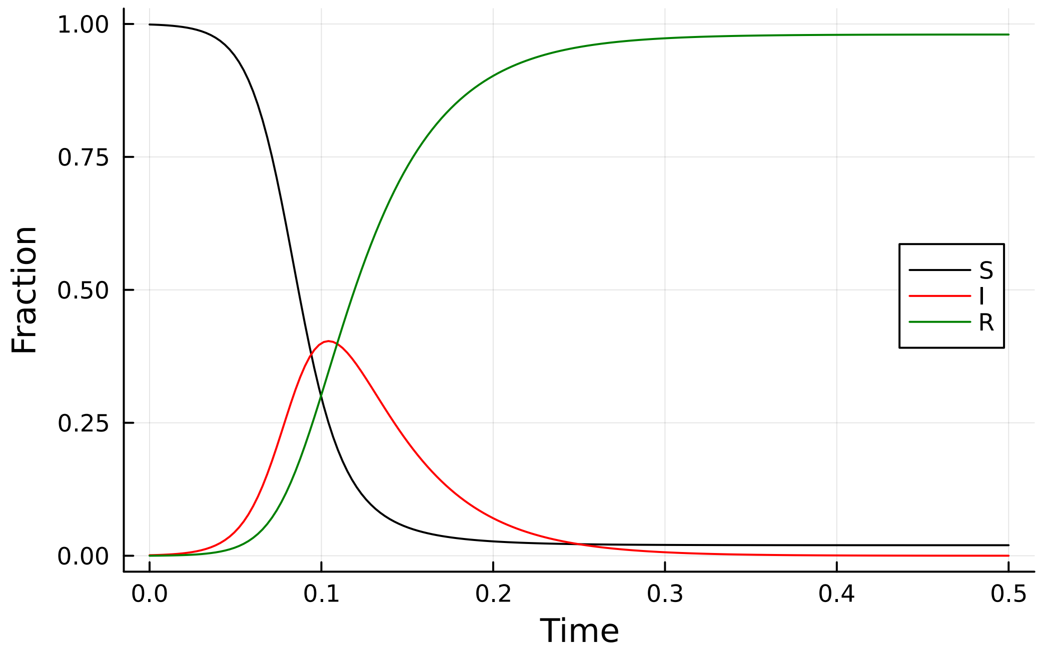
Step 5: solve the model.

```
prob = ODEProblem(sirmod, u0, (t0, t1), p)
sol = solve(prob, Rodas5P(); saveat= t);
```

```
out = DataFrame(sol)
rename!(out, [:time, :S, :I, :R])
first(round.(out, digits = 3), 6)
```

	time	S	I	R
	Float64	Float64	Float64	Float64
1	0.0	0.999	0.001	0.0
2	0.003	0.999	0.001	0.0
3	0.005	0.998	0.002	0.0
4	0.008	0.998	0.002	0.0
5	0.011	0.997	0.002	0.0
6	0.014	0.996	0.003	0.001

```
plot(out.time, out.S, ylabel="Fraction", xlabel="Time", color=:black, label="S", legend=:r
plot!(out.time, out.I, color=:red, label="I")
plot!(out.time, out.R, color=:green, label="R")
```



2.4 Final epidemic size

Find final epidemic size by running to steady state.

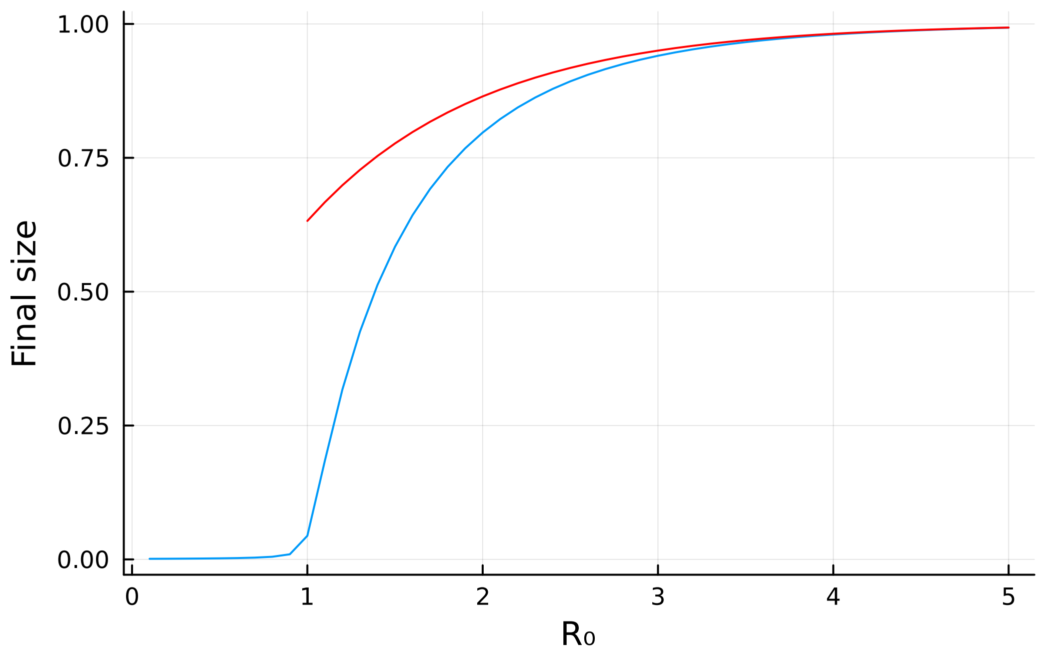
```
using SteadyStateDiffEq
ssprob = SteadyStateProblem(sirmod, u0, p)
sssol = solve(ssprob, DynamicSS(Rodas5P()))
round.(sssol, digits=2)
```

```
3-element Vector{Float64}:
  0.02
 -0.0
  0.98
```

Calculate final size over a range of values of R .

```
nsims = 50
R  = range(0.1, 5, nsims)
betas = R  .* p.  .+ p.
fs = Array{Float64}(undef, nsims)
for i in 1:nsims
    sp = remake(ssprob, p=LVector( =0.0, N=1.0,  =365.0/14,  =betas[i]))
    ss = solve(sp, DynamicSS(Rodas5P()))
    fs[i] = ss[3]
end
```

```
plot(R , fs, xlabel="R ", ylabel="Final size", legend=false)
x = 1:0.1:5
plot!(x, 1 .- exp.(-x), color=:red)
```



Use root-finding to calculate the final size.

```
using NonlinearSolve
fn(u, p) = exp(-(p[1]*(1-u[1]))) - u[1]
rprob = IntervalNonlinearProblem(fn, (0.0, 1.0 - 1e-9), [2.0])
rsol = solve(rprob, Falsi())
1.0 - rsol.u[1]
```

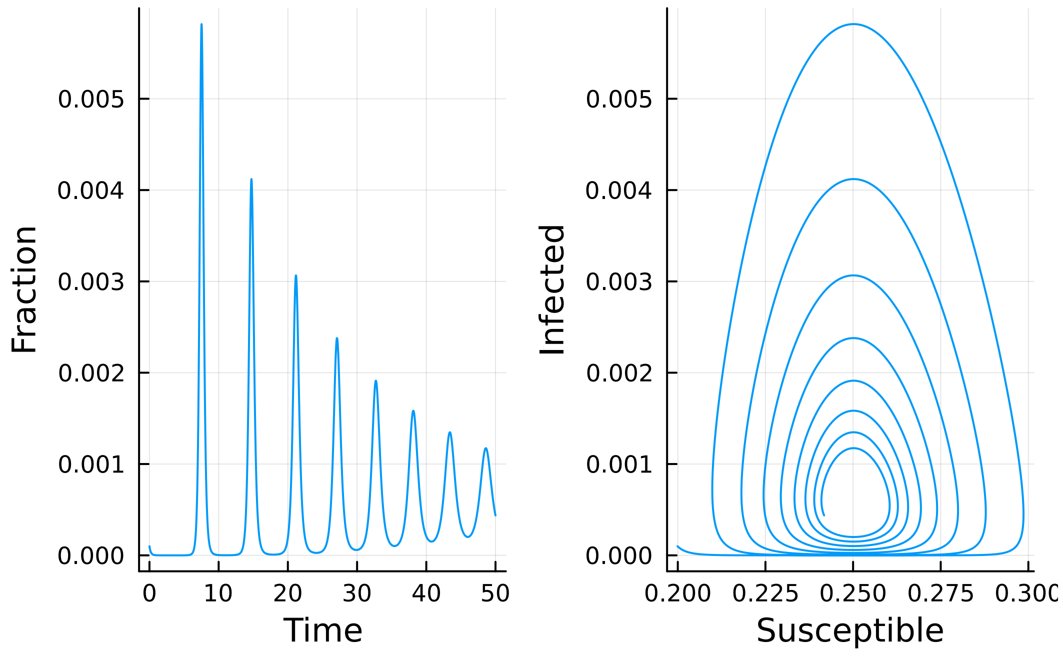
```
0.7968121300200202
```

2.5 The open epidemic

```
t0 = 0.0
t1 = 50.0
 t = 1.0/365
p = LVector( =1.0/50, N=1.0, R =4.0,  =365.0/14)
p = [p; LVector( =p.R *p.  + p. )]
u0 = [0.1999, 0.0001, 0.8] .* p.N
prob = ODEProblem(sirmod, u0, (t0, t1), p)
sol = solve(prob, Rodas5P(); saveat= t)
```

```
out = DataFrame(sol)
rename!(out, [:time, :S, :I, :R]);
```

```
l = @layout [a b]
p1 = plot(t0: t:t1, out.I, xlabel="Time", ylabel="Fraction", legend=false)
p2 = plot(out.S, out.I, xlabel="Susceptible", ylabel="Infected", legend=false)
plot(p1, p2, layout=l)
```


Fraction
Time
Infected
Susceptible