Discrete event simulation using SimJulia

Simon Frost (@sdwfrost), 2020-04-27

Libraries

```
using ResumableFunctions
```

```
Error: ArgumentError: Package ResumableFunctions not found in current path:
- Run `import Pkg; Pkg.add("ResumableFunctions")` to install the ResumableF
unctions package.
```

```
using SimJulia
using Distributions
using DataFrames
using Random
using StatsPlots
using BenchmarkTools
```

Utility functions

```
function increment!(a::Array{Int64})
    push!(a,a[length(a)]+1)
end

function decrement!(a::Array{Int64})
    push!(a,a[length(a)]-1)
end

function carryover!(a::Array{Int64})
    push!(a,a[length(a)])
end;
```

```
carryover! (generic function with 1 method)
```

Transitions

```
mutable struct SIRPerson
    id::Int64 # numeric ID
    status::Symbol # :S,I,R
end;

mutable struct SIRModel
    sim::Simulation
    β::Float64
    c::Float64
    γ::Float64
    ta::Array{Float64}
    Sa::Array{Int64}
    Ia::Array{Int64}
    Ra::Array{Int64}
    allIndividuals::Array{SIRPerson}
end
```

These functions update the state of the 'world' when either an infection or recovery occurs.

```
function infection_update!(sim::Simulation,m::SIRModel)
    push!(m.ta,now(sim))
    decrement!(m.Sa)
    increment!(m.Ia)
    carryover!(m.Ra)
end;
```

```
infection_update! (generic function with 1 method)
```

```
function recovery_update!(sim::Simulation,m::SIRModel)
    push!(m.ta,now(sim))
    carryover!(m.Sa)
    decrement!(m.Ia)
    increment!(m.Ra)
end;
```

```
recovery_update! (generic function with 1 method)
```

The following is the main simulation function. It's not efficient, as it involves activating a process for all susceptibles; a more efficient algorithm would involve just considering infected individuals, and activating each susceptible individual when infection occurs. This however requires more bookkeeping and detracts from the ability to easily compare between implementations.

```
@resumable function live(sim::Simulation, individual::SIRPerson, m::SIRModel)
  while individual.status==:S
      # Wait until next contact
      @yield timeout(sim,rand(Distributions.Exponential(1/m.c)))
      # Choose random alter
      alter=individual
      while alter==individual
          N=length(m.allIndividuals)
          index=rand(Distributions.DiscreteUniform(1,N))
          alter=m.allIndividuals[index]
      end
      # If alter is infected
      if alter.status==:I
          infect = rand(Distributions.Uniform(0,1))
          if infect < m.β
              individual.status=:I
              infection_update!(sim,m)
          end
      end
  end
  if individual.status==:I
      # Wait until recovery
      @yield timeout(sim,rand(Distributions.Exponential(1/m.γ)))
      individual.status=:R
      recovery_update!(sim,m)
  end
end;
```

```
Error: LoadError: ArgumentError: Package ResumableFunctions not found in cu
rrent path:
- Run `import Pkg; Pkg.add("ResumableFunctions")` to install the ResumableF
unctions package.

in expression starting at none:1
```

```
function MakeSIRModel(u0,p)
    (S,I,R) = u0
    N = S+I+R
    (β,c,γ) = p
    sim = Simulation()
    allIndividuals=Array{SIRPerson,1}(undef,N)
    for i in 1:S
        p=SIRPerson(i,:S)
        allIndividuals[i]=p
    end
    for i in (S+1):(S+I)
        p=SIRPerson(i,:I)
        allIndividuals[i]=p
    end
    for i  in (S+I+1):N
        p=SIRPerson(i,:R)
        allIndividuals[i]=p
    end
    ta=Array{Float64,1}(undef,0)
    push!(ta,0.0)
    Sa=Array{Int64,1}(undef,0)
    push!(Sa,S)
    Ia=Array{Int64,1}(undef,0)
    push!(Ia,I)
    Ra=Array{Int64,1}(undef,0)
    push!(Ra,R)
    SIRModel(sim,β,c,γ,ta,Sa,Ia,Ra,allIndividuals)
end;
```

```
MakeSIRModel (generic function with 1 method)
```

```
function activate(m::SIRModel)
     [@process live(m.sim,individual,m) for individual in m.allIndividuals]
end;
```

```
activate (generic function with 1 method)
```

```
function sir_run(m::SIRModel,tf::Float64)
    SimJulia.run(m.sim,tf)
end;
```

```
sir_run (generic function with 1 method)
```

```
function out(m::SIRModel)
    result = DataFrame()
    result[!,:t] = m.ta
    result[!,:S] = m.Sa
    result[!,:I] = m.Ia
    result[!,:R] = m.Ra
    result
end;
```

```
out (generic function with 1 method)
```

Time domain

```
tmax = 40.0;
```

```
40.0
```

Initial conditions

```
u0 = [990,10,0];
```

```
3-element Array{Int64,1}:
 990
  10
   0
```

Parameter values

```
p = [0.05,10.0,0.25];
```

```
3-element Array{Float64,1}:
  0.05
 10.0
  0.25
```

Random number seed

```
Random.seed!(1234);
```

```
MersenneTwister(UInt32[0x000004d2], Random.DSFMT.DSFMT_state(Int32[-1393240
018, 1073611148, 45497681, 1072875908, 436273599, 1073674613, -2043716458,
1073445557, -254908435, 1072827086  ...  -599655111, 1073144102, 367655457, 1
072985259, -1278750689, 1018350124, -597141475, 249849711, 382, 0]), [0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0  ...  0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0], UInt128[0x00000000000000000000000000000000, 0x00000
000000000000000000000000000, 0x00000000000000000000000000000000, 0x00000000
000000000000000000000000, 0x00000000000000000000000000000000, 0x00000000000
000000000000000000000, 0x00000000000000000000000000000000, 0x00000000000000
000000000000000000, 0x00000000000000000000000000000000, 0x0000000000000000
000000000000000  ...  0x00000000000000000000000000000000, 0x0000000000000000
000000000000000, 0x00000000000000000000000000000000, 0x0000000000000000000
000000000000, 0x00000000000000000000000000000000, 0x0000000000000000000000
000000000, 0x00000000000000000000000000000000, 0x0000000000000000000000000
000000, 0x00000000000000000000000000000000, 0x0000000000000000000000000000
000], 1002, 0)
```

Running the model

```
des_model = MakeSIRModel(u0,p)
activate(des_model)
```

```
Error: UndefVarError: live not defined
```

```
sir_run(des_model,tmax)
```

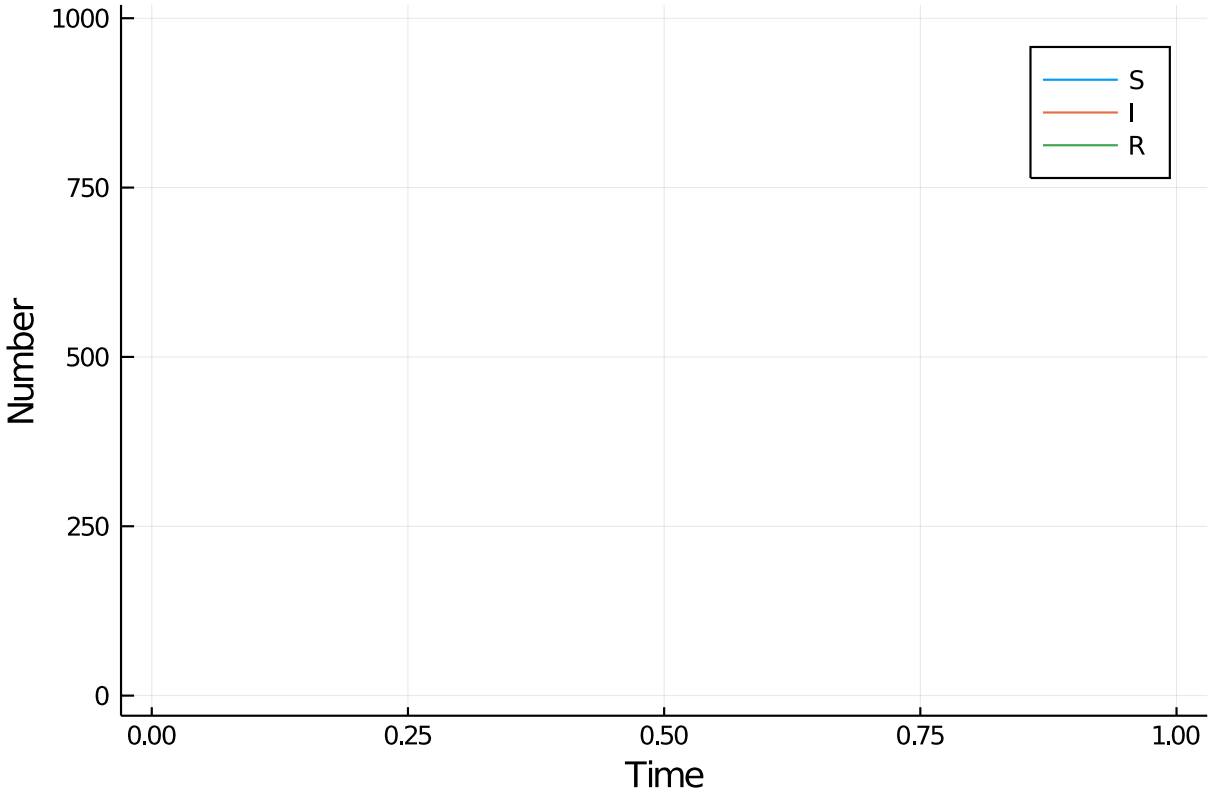
Postprocessing

```
data_des=out(des_model);
```

	t	S	I	R
	Float64	Int64	Int64	Int64
1	0.0	990	10	0

Plotting

```
@df data_des plot(:t,
    [:S :I :R],
    labels = ["S" "I" "R"],
    xlab="Time",
    ylab="Number")
```



Benchmarking

```
@benchmark begin
    des_model = MakeSIRModel(u0,p)
    activate(des_model)
    sir_run(des_model,tmax)
end
```

```
Error: UndefVarError: live not defined
```

Appendix

Computer Information

```
Julia Version 1.4.1
Commit 381693d3df* (2020-04-14 17:20 UTC)
Platform Info:
  OS: Linux (x86_64-pc-linux-gnu)
  CPU: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-8.0.1 (ORCJIT, icelake-client)
Environment:
```

```
  JULIA_NUM_THREADS = 4
```

Package Information

```
Status `~/.julia/environments/v1.4/Project.toml`
[46ada45e-f475-11e8-01d0-f70cc89e6671] Agents 3.1.0
[c52e3926-4ff0-5f6e-af25-54175e0327b1] Atom 0.12.11
[6e4b80f9-dd63-53aa-95a3-0cdb28fa8baf] BenchmarkTools 0.5.0
[a134a8b2-14d6-55f6-9291-3336d3ab0209] BlackBoxOptim 0.5.0
[2445eb08-9709-466a-b3fc-47e12bd697a2] DataDrivenDiffEq 0.2.0
[a93c6f00-e57d-5684-b7b6-d8193f3e46c0] DataFrames 0.21.0
[ebbdde9d-f333-5424-9be2-dbf1e9acfb5e] DiffEqBayes 2.14.0
[459566f4-90b8-5000-8ac3-15dfb0a30def] DiffEqCallbacks 2.13.2
[c894b116-72e5-5b58-be3c-e6d8d4ac2b12] DiffEqJump 6.7.5
[1130ab10-4a5a-5621-a13d-e4788d82bd4c] DiffEqParamEstim 1.14.1
[0c46a032-eb83-5123-abaf-570d42b7fbaa] DifferentialEquations 6.14.0
[31c24e10-a181-5473-b8eb-7969acd0382f] Distributions 0.23.2
[634d3b9d-ee7a-5ddf-bec9-22491ea816e1] DrWatson 1.11.0
[587475ba-b771-5e3f-ad9e-33799f191a9c] Flux 0.8.3
[28b8d3ca-fb5f-59d9-8090-bfdbd6d07a71] GR 0.49.1
[523d8e89-b243-5607-941c-87d699ea6713] Gillespie 0.1.0
[7073ff75-c697-5162-941a-fcdaad2a7d2a] IJulia 1.21.2
[4076af6c-e467-56ae-b986-b466b2749572] JuMP 0.21.2
[e5e0dc1b-0480-54bc-9374-aad01c23163d] Juno 0.8.2
[093fc24a-ae57-5d10-9952-331d41423f4d] LightGraphs 1.3.3
[1914dd2f-81c6-5fcd-8719-6d5c9610ff09] MacroTools 0.5.5
[ee78f7c6-11fb-53f2-987a-cfe4a2b5a57a] Makie 0.9.5
[961ee093-0014-501f-94e3-6117800e7a78] ModelingToolkit 3.6.0
[76087f3c-5699-56af-9a33-bf431cd00edd] NLopt 0.6.0
[429524aa-4258-5aef-a3af-852621145aeb] Optim 0.21.0
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.38.1
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 1.3.1
[428bdadb-6287-5aa5-874b-9969638295fd] SimJulia 0.8.0
[05bca326-078c-5bf0-a5bf-ce7c7982d7fd] SimpleDiffEq 1.1.0
[f3b207a7-027a-5e70-b257-86293d7955fd] StatsPlots 0.14.6
[789caeaf-c7a9-5a7d-9973-96adeb23e2a0] StochasticDiffEq 6.23.0
[fce5fe82-541a-59a6-adf8-730c64b5f9a0] Turing 0.12.0
[44d3d7a6-8a23-5bf8-98c5-b353f8df5ec9] Weave 0.10.0
```