SIR Model with Demographic Stochasticity

K. Bodie Weedop

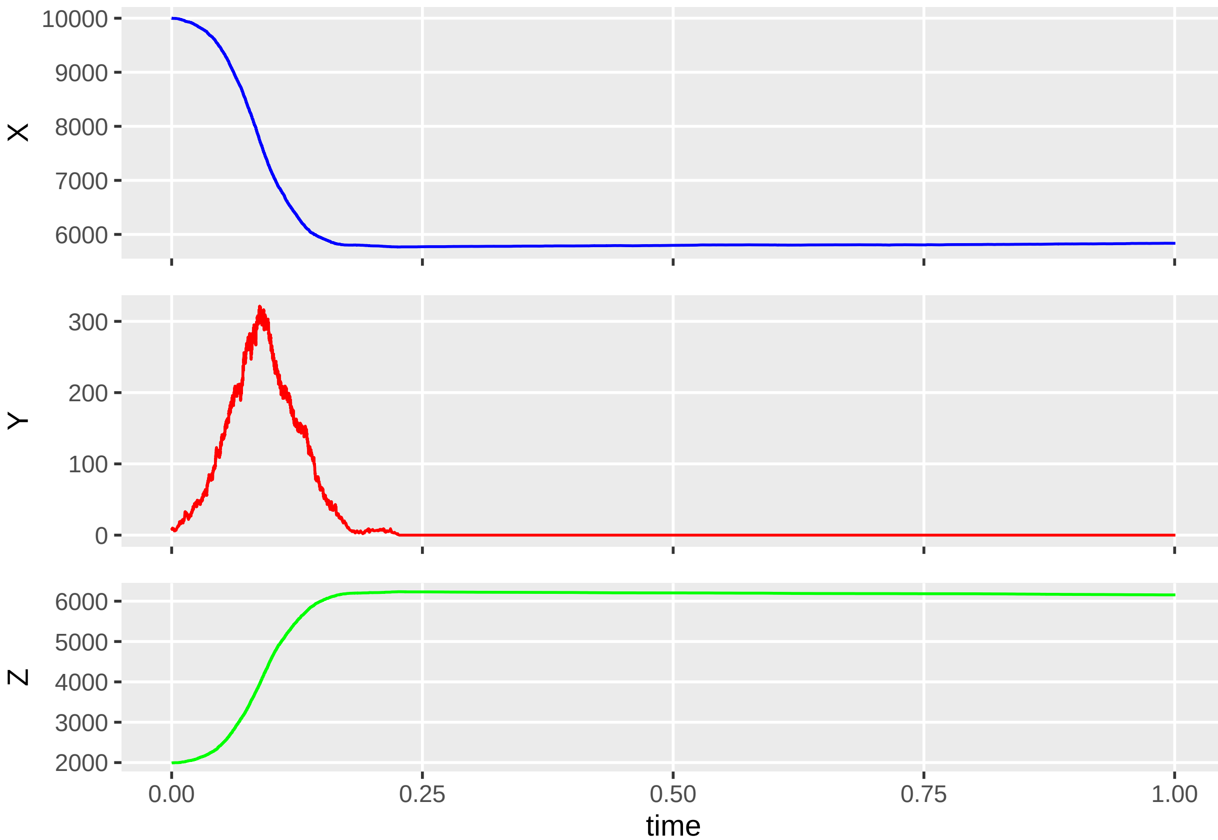
10/23/2019

Using Gillespie's Direct Algorithm

```
set.seed(123)
# Parameter values
gamma <- 1/2
mu <- 1/(70*365)
beta <- 1.5 * (gamma + mu)
params <- c(beta, gamma, mu)

# Initial conditions
N0 <- 12000
X0 <- 10000
Y0 <- 10
Z0 <- N0 - X0 - Y0
vars <- c(X0, Y0, Z0)

t0 <- 0
years <- 1
tn <- years * 365

direct.results <- data.frame(time = t0,
                             X = X0,
                             Y = Y0,
                             Z = Z0)

direct.sir <- function(vars, params) {
  beta <- params[1]
  gamma <- params[2]
  mu <- params[3]

  x0 <- vars[1]
  y0 <- vars[2]
  z0 <- vars[3]
  n0 <- sum(vars)

  R.n <- rep(0, 6)
  change <- matrix(nrow = 6, ncol = 3)

  #Events
  # Transmission
  R.n[1] <- beta * x0 * y0 / n0
  change[1,] <- c(-1, 1, 0)
  # Recovery
  R.n[2] <- gamma * y0
  change[2,] <- c(0, -1, 1)
  # Birth
```

```r
  R.n[3] <- mu * n0
  change[3,] <- c(1, 0, 0)
  # Death in each class
  R.n[4] <- mu * x0
  change[4,] <- c(-1, 0, 0)
  R.n[5] <- mu * y0
  change[5,] <- c(0, -1, 0)
  R.n[6] <- mu * z0
  change[6,] <- c(0, 0, -1)

  rand1 <- runif(n=1)
  rand2 <- runif(n=1)

  d.t <- (-1/sum(R.n))*log(rand1)
  p <- min(which(cumsum(R.n) >= rand2 * sum(R.n)))
  vars <- vars + change[p,]
  return(c(d.t, vars))
}

while (t0 <= tn) {
  tmp <- direct.sir(vars = vars, params = params)
  t0 <- t0 + tmp[1]
  tmp[1] <- t0
  direct.results <- rbind(direct.results, tmp)
  vars <- tmp[2:4]
}
```

```r
direct.results$time <- direct.results$time / 365
x.plot <- ggplot(data=direct.results, aes(x=time, y=X)) + geom_line(color="blue")
x.plot <- x.plot + theme(axis.text.x=element_blank(),
                         axis.title.x=element_blank(),
                         legend.position="none",
                         panel.border=element_blank(),
                         panel.grid.minor=element_blank())
y.plot <- ggplot(data=direct.results, aes(x=time, y=Y)) + geom_line(color="red")
y.plot <- y.plot + theme(axis.text.x=element_blank(),
                         axis.title.x=element_blank(),
                         legend.position="none",
                         panel.border=element_blank(),
                         panel.grid.minor=element_blank())
z.plot <- ggplot(data=direct.results, aes(x=time, y=Z)) + geom_line(color="green")
z.plot <- z.plot + theme(legend.position="none",
                         panel.border=element_blank(),
                         panel.grid.minor=element_blank())
plot_grid(x.plot, y.plot, z.plot, nrow=3, align="v")
```



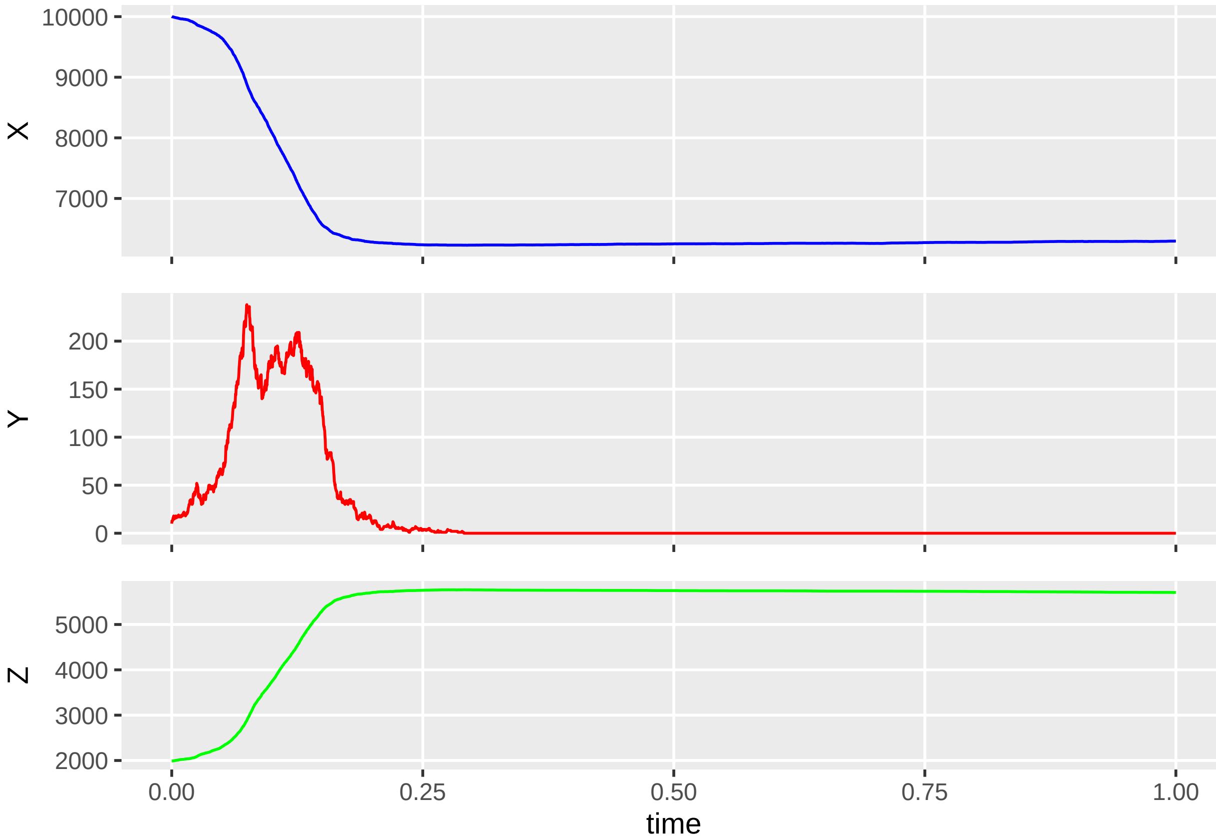
Using the "τ-leap" Method – from last week

```
# Reset initial values for variables;
vars <- c(X0, Y0, Z0)

# Time to run model simulation
years <- 1
tau <- 1/10
t_range <- seq(0, years*365, tau)

# SIR model using tau-leap method for stochastics
tau.sir <- function(vars, params) {
  beta <- params[1]
  gamma <- params[2]
  mu <- params[3]

  x0 <- vars[1]
  y0 <- vars[2]
  z0 <- vars[3]
  n0 <- sum(vars)

  # Initializing data frame for event rates and the direction in which the variables change
  rate <- rep(0, 6)
  change <- matrix(nrow = 6, ncol = 3)
```

```r
  # Transmission Event
  rate[1] <- (beta * x0 * y0) / n0
  change[1,] <- c(-1, 1, 0)
  # Recovery Event
  rate[2] <- gamma * y0
  change[2,] <- c(0, -1, 1)
  # Birth Event
  rate[3] <- mu * n0
  change[3,] <- c(1, 0, 0)
  # Death Events
  rate[4] <- mu * x0
  change[4,] <- c(-1, 0, 0)
  rate[5] <- mu * y0
  change[5,] <- c(0, -1, 0)
  rate[6] <- mu * z0
  change[6,] <- c(0, 0, -1)

  # Update variables
  for (i in 1:length(rate)) {
    # Number of times events occur assumed to be Poisson
    tmp <- rpois(n=1, lambda=rate[i]*tau)
    # Ensuring that values do not go negative -- use whichever value is the minimum
    non.neg <- min(c(tmp, vars[which(change[i,] < 0)]))
    # Update using change values (e.g. [+1, -1, 0]) multiplied by the rate of that event
    vars = vars + change[i,] * non.neg
  }
  return(vars)
}

tau.results <- data.frame(time=t_range, X=NA, Y=NA, Z=NA)
tau.results[1, 2:ncol(tau.results)] <- vars

for (t in 1:(length(t_range)-1)) {
  t.vars <- as.numeric(tau.results[t, 2:ncol(tau.results)])
  t <- t + 1
  tmp <- tau.sir(t.vars, params)
  tau.results[t, 2:ncol(tau.results)] <- tmp
}
```

```r
tau.results$time <- tau.results$time / 365
# Plotting
# Each variable (S, I, R) will be plotted individually over total time
#    simulation allowed to run.
x.plot <- ggplot(data=tau.results, aes(x=time, y=X)) + geom_line(color="blue")
x.plot <- x.plot + theme(axis.text.x=element_blank(),
                         axis.title.x=element_blank(),
                         legend.position="none",
                         panel.border=element_blank(),
                         panel.grid.minor=element_blank())
y.plot <- ggplot(data=tau.results, aes(x=time, y=Y)) + geom_line(color="red")
y.plot <- y.plot + theme(axis.text.x=element_blank(),
                         axis.title.x=element_blank(),
                         legend.position="none",
                         panel.border=element_blank(),
```

```
                          panel.grid.minor=element_blank())
z.plot <- ggplot(data=tau.results, aes(x=time, y=Z)) + geom_line(color="green")
z.plot <- z.plot + theme(legend.position="none",
                          panel.border=element_blank(),
                          panel.grid.minor=element_blank())
plot_grid(x.plot, y.plot, z.plot, nrow=3, align="v")
```



Using the τ leap method with multinomial distributed events

```
# Reset initial conditions
vars <- c(X0, Y0, Z0)

# Time to run model simulation
years <- 1
tau <- 1/100
t_range <- seq(0, years*365, tau)

multinom.events <- function(var, events, tau) {
  null.prob <- exp(-sum(events)*tau) # Probability that no events take place
  event.prob <- rep(NA, length(events)) # Initialize vector to hold event probabilities
  for (i in 1:length(events)) {
    # Calculate probabilities for each event
    event.prob[i] <- (1 - null.prob) * (events[i]/sum(events))
  }
```

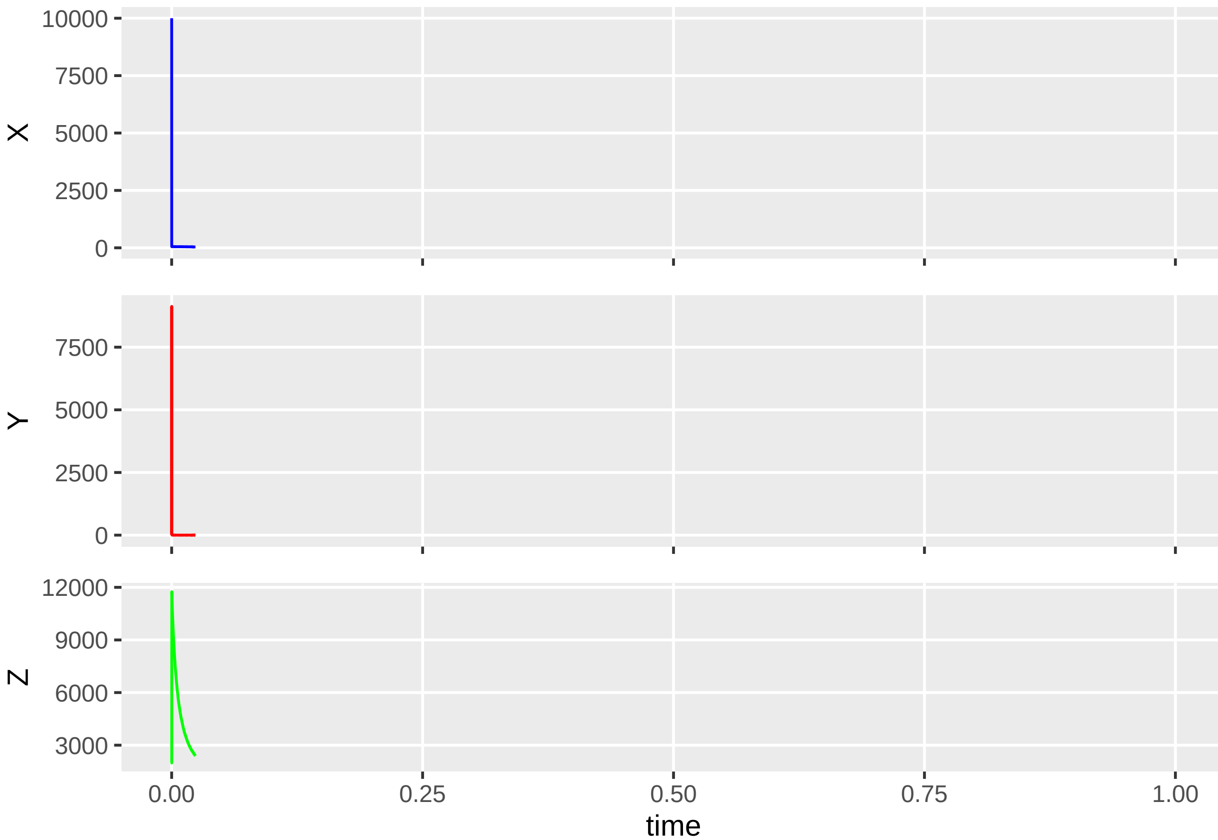
```r
  # Concatenate probability of nothing happening with other prob. for events
  event.prob <- c(null.prob, event.prob)
  # Number of occurrences;
  event.occ <- rmultinom(n = 1,
                         size = var,
                         prob = event.prob)
  return(event.occ)
}

# SIR model using tau-leap method for stochastics
multinomtau.sir <- function(vars, params) {
  beta <- params[1]
  gamma <- params[2]
  mu <- params[3]

  x0 <- vars[1]
  y0 <- vars[2]
  z0 <- vars[3]
  n0 <- sum(vars)

  # Initializing data frame for event rates and the direction in which the variables change
  rate <- rep(0, 6)
  occur <- rep(0, 6)
  change <- matrix(nrow = 6, ncol = 3)

  # Transmission Event
  rate[1] <- (beta * x0 * y0) / n0
  change[1,] <- c(-1, 1, 0)
  # Recovery Event
  rate[2] <- gamma * y0
  change[2,] <- c(0, -1, 1)
  # Birth Event
  rate[3] <- mu * n0
  change[3,] <- c(1, 0, 0)
  # Death Events
  rate[4] <- mu * x0
  change[4,] <- c(-1, 0, 0)
  rate[5] <- mu * y0
  change[5,] <- c(0, -1, 0)
  rate[6] <- mu * z0
  change[6,] <- c(0, 0, -1)

  # Births assumed to be Poisson
  occur[3] <- rpois(n = 1, lambda = rate[3]*tau)
  occur[c(1, 4)] <- multinom.events(x0, rate[c(1, 4)], tau = tau)[-1,] # remove nothing happened event
  occur[c(2, 5)] <- multinom.events(y0, rate[c(2, 5)], tau = tau)[-1,]
  occur[6] <- multinom.events(z0, rate[6], tau = tau)[-1,]

  # Update variables
  for (i in 1:length(occur)) {
    # Update using change values (e.g. [+1, -1, 0]) multiplied by the rate of that event
    vars = vars + change[i,] * occur[i]
  }
```

```
  return(vars)
}

multinomtau.results <- data.frame(time=t_range, X=NA, Y=NA, Z=NA)
multinomtau.results[1, 2:ncol(multinomtau.results)] <- vars

for (t in 1:(length(t_range)-1)) {
  t.vars <- as.numeric(multinomtau.results[t, 2:ncol(multinomtau.results)])
  t <- t + 1
  tmp <- multinomtau.sir(t.vars, params)
  multinomtau.results[t, 2:ncol(multinomtau.results)] <- tmp
}
```

```
## Error in rmultinom(n = 1, size = var, prob = event.prob): NA in probability vector
```

```
multinomtau.results$time <- multinomtau.results$time / 365
# Plotting
# Each variable (S, I, R) will be plotted individually over total time
#    simulation allowed to run.
x.plot <- ggplot(data=multinomtau.results, aes(x=time, y=X)) + geom_line(color="blue")
x.plot <- x.plot + theme(axis.text.x=element_blank(),
                         axis.title.x=element_blank(),
                         legend.position="none",
                         panel.border=element_blank(),
                         panel.grid.minor=element_blank())
y.plot <- ggplot(data=multinomtau.results, aes(x=time, y=Y)) + geom_line(color="red")
y.plot <- y.plot + theme(axis.text.x=element_blank(),
                         axis.title.x=element_blank(),
                         legend.position="none",
                         panel.border=element_blank(),
                         panel.grid.minor=element_blank())
z.plot <- ggplot(data=multinomtau.results, aes(x=time, y=Z)) + geom_line(color="green")
z.plot <- z.plot + theme(legend.position="none",
                         panel.border=element_blank(),
                         panel.grid.minor=element_blank())
plot_grid(x.plot, y.plot, z.plot, nrow=3, align="v")
```

```
## Warning: Removed 35636 rows containing missing values (geom_path).

## Warning: Removed 35636 rows containing missing values (geom_path).

## Warning: Removed 35636 rows containing missing values (geom_path).
```


10000
7500
5000
2500
0
X
7500
5000
2500
0
Y
12000
9000
6000
3000
Z
0.00
0.25
0.50
0.75
1.00
time