Animated Plots with `knitr`

Samuel M. Jenness
sjenness@uw.edu

November 13, 2012

`knitr` is an update to `Sweave`, which is software that integrates data analysis in `R` and word-processing in LaTeX. One nice feature of `knitr` is easy animated plots. Rstudio can directly evaluate and compile `knitr` documents with one click. This provides three examples. **Note: you must open this PDF in Adobe Acrobat to see the animations.**

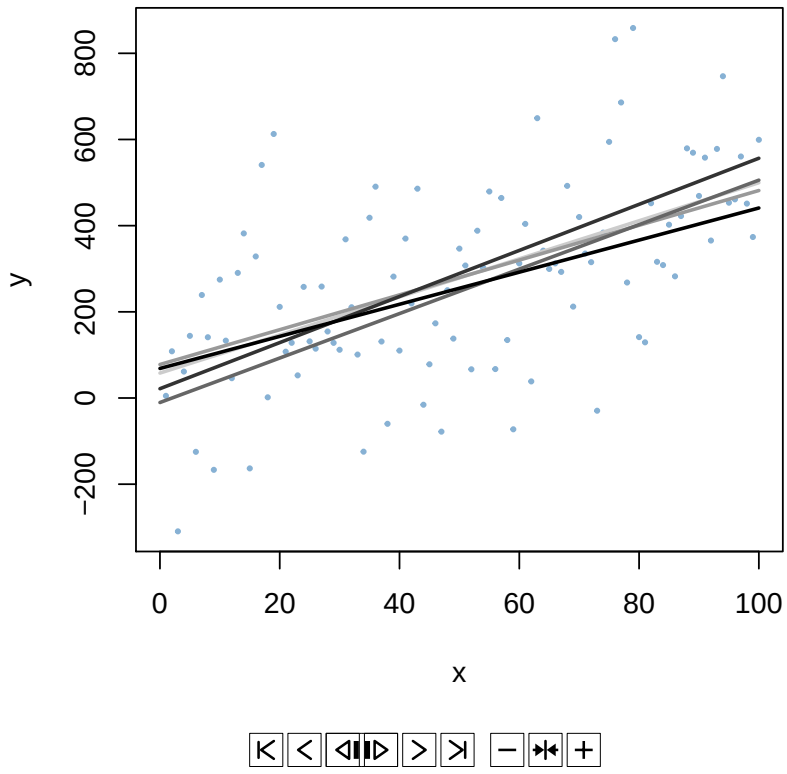
1 Linear Regression

This defines the values of x and y, estimates the coefficents in a linear regression model, and simulates coefficients given the model using Gelman's `sim` function in the `arm` package. This is a nice way to see inferential uncertainty in the coefficients.

```
x <- 1:100
y <- 5*x + rnorm(length(x), 0, 200)
mod <- lm(y ~ x)
mod.sim <- sim(mod, n.sims=50)
b <- coef(mod.sim)
```

An animated plot is constructed like this, with everything is wrapped in a loop. I used the multiple lines of code for the curve function to create a fade-out effect. The animation options (not shown here) are: a frame rate of 10 frames per second, automatically play upon opening the PDF, and loop the animation.

```
for (i in 5:nrow(b)) {
    plot(x, y, pch = 20, cex=0.5,
         xlim=c(0,max(x)), ylim=range(y), col=pal[2])
    curve(b[i-4,1] + b[i-4,2]*x, lwd=2, add=T, col='grey80')
    curve(b[i-3,1] + b[i-3,2]*x, lwd=2, add=T, col='grey60')
    curve(b[i-2,1] + b[i-2,2]*x, lwd=2, add=T, col='grey40')
    curve(b[i-1,1] + b[i-1,2]*x, lwd=2, add=T, col='grey20')
    curve(b[i,1] + b[i,2]*x, lwd=2, add=T)
}
```



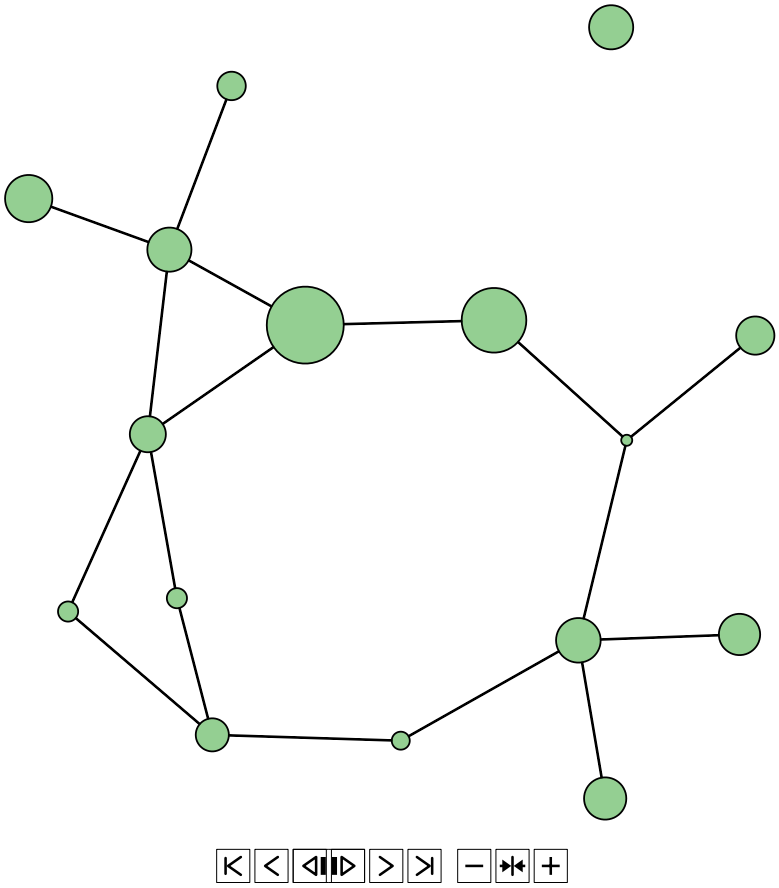
2 Network Simulation

Here is the flomarriage network estimated and simulated, using code featured in the ERGM workshop.

```
data(florentine)
wealth <- flomarriage %v% 'wealth'
flomodel.03 <- ergm(flomarriage~edges+nodecov('wealth'))
flomodel.03.sim <- simulate(flomodel.03, nsim=25)
```

The animated plot shows all 20 simulations of the graph with static node coordinates (code not shown). The animation options here are: a frame rate of 2 frames per second, no autoplay (you must press the play button), and no looping.

```
par(mar=c(0,0,0,0))
for (i in 1:length(flomodel.03.sim)){
  plot(flomodel.03.sim[[i]], coord=floplot,
       vertex.cex=sqrt(wealth)/2.5, vertex.col=pal[3])
}
```



3 SIR Model

Here I set up a deterministic SIR model with births and deaths, solved with ordinary differential equations.

```
library(deSolve)
S <- 999
I <- 1
R <- 0
N <- S+I+R

cont <- 6
beta <- 0.2
lambda <- cont*beta/N

v <- 0.3 # Recovery rate
f <- 0.011 # Birth rate
ms <- 0.01 # Death rate for S
mi <- 0.02 # Death rate for I
mr <- 0.01 # Death rate for R

p <- c(beta=beta, cont=cont, v=v, f=f, ms=ms, mi=mi, mr=mr)
```

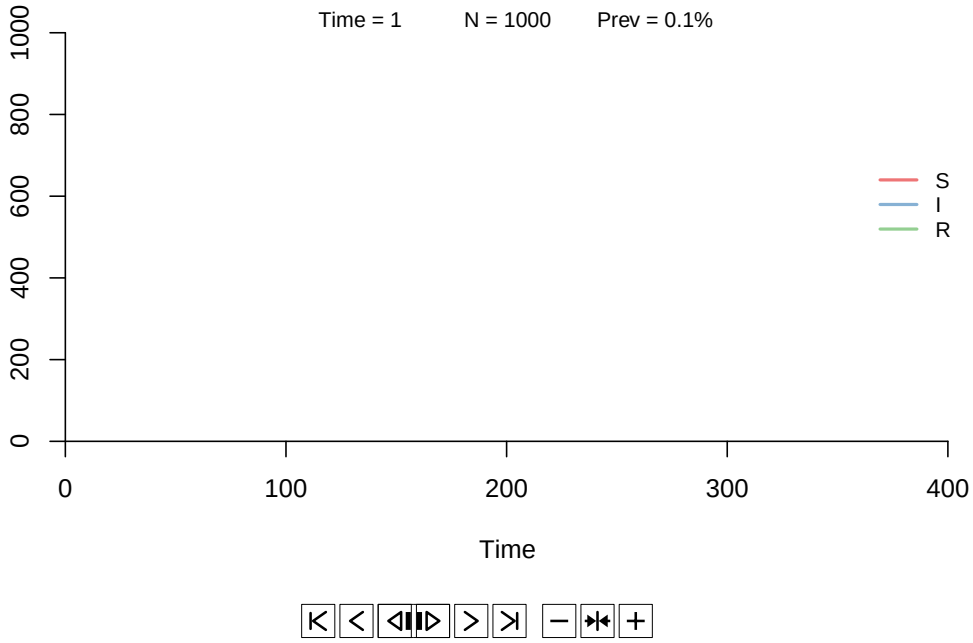
```
t0 <- c(S=S, I=I, R=R, N=N)

SIR <- function(t, t0, params) {
  with(as.list(c(t0, p)), {
dS <- -(cont*beta/N)*S*I + f*N - ms*S
dI <- (cont*beta/N)*S*I - v*I - mi*I
dR <- v*I - mr*R
dN <- f*N - ms*S - mi*I - mr*R
list(c(dS, dI, dR, dN))
})
}

times <- seq(from=0, to=400, by=1)
out <- data.frame(ode(y=t0, times=times, func=SIR, parms=p))
times <- times+1
```

For the plot, it is necessary to index the time cumulatively to see the dynamics. The animation options are: a frame rate of 10 frames per second, no autoplay or looping, and wider dimensions for the plot. It is possible to step through the times manually with the plot buttons.

```
for (i in 1:max(times)){
  plot(out$I[1:times[i]],type='l',col=pal[2], lwd=2,
       ylab='' , xlab='Time',
       ylim=c(0,round(max(out[,2:4]), -2)), xlim=c(0,max(times)),
       xaxs='i', yaxs='i', bty='n')
  lines(out$S[1:times[i]], col=pal[1], lwd=2)
  lines(out$R[1:times[i]], col=pal[3], lwd=2)
  mtext(paste('Time =', times[i], sep=' '), 3,
       at=max(times)/3, cex=0.8)
  mtext(paste(paste('Prev =',
                    100*round(out$I[times[i]]/(out$N[times[i]]), 4), sep=' '),
                    '%', sep=''), 3,
       at=max(times)/3*2, cex=0.8)
  mtext(paste('N =', round(out$N[times[i]], 0), sep=' '),
       at=max(times)/2, cex=0.8)
  legend(max(times)-(0.10*max(times)), 700, legend=c('S','I','R'),
        bty='n', lty=1, lwd=2, col=c(pal[1:3]), cex=0.8)
}
```


Time = 1
N = 1000
Prev = 0.1%
0
200
400
600
800
1000
0
100
200
300
400
Time
S
I
R