plant SIR model

The model

This is a discrete-time model for disease spread in a experimental grid of plants. As you described in your document, there are three states: susceptible, challenged, and infected. Since the model deal with discrete time, we can think of each plant having a status in each time step. We talked about the *transition matrix*, which lists the probability of going from some status in time period t to its status in period $t+1$. Here is an example, where each row represents the status at time t and each column represents the status at time $t+1$. The value in the cell is the probability of transitioning from the rowstate to the column state. Each row must therefore sum to one:

	S	C	I
S	$e^{-\beta I}$	0	$1 - e^{-\beta I}$
C	$a \times (1 - e^{-\delta})$	$a \times e^{-\delta - \alpha - \beta \times I}$	$a \times (1 - e^{-\alpha - \beta I)})$
I	0	0	1

Turning observations into data

The strategy is to take each transition in each plant as an observation. Here I mean transition as the period between two observations (because the initial state is not random). So if there are seven time steps and two plants, there are twelve total observations. Let's say the state of plant 1 evolves in time like `S -> S -> S -> I -> I -> I -> I` and plant 2 evolves like `C -> C -> I -> I -> I -> I -> I`. I would think of each arrow as an observation, where you look up the probability from the transition probability matrix where the state to the left of the arrow is the row index and the state to the arrow's right is the column index.

One way to think of each observation is with the same transition matrix, where the row index is the state at t and the column index is the state at $t+1$. Then the obseved data for plant one are like:

$t = 0$	S	C	I
S	1	0	0
C	0	0	0
I	0	0	0

$t = 1$	S	C	I
S	1	0	0
C	0	0	0
I	0	0	0

$t = 2$	S	C	I
S	0	0	1

$t = 2$	S	C	I
C	0	0	0
I	0	0	0

$t = 3$	S	C	I
S	0	0	0
C	0	0	0
I	0	0	1

$t = 4$	S	C	I
S	0	0	0
C	0	0	0
I	0	0	1

$t = 5$	S	C	I
S	0	0	0
C	0	0	0
I	0	0	1

$t = 6$	S	C	I
S	0	0	0
C	0	0	0
I	0	0	1

I think it is natural to "unwrap" these matrices into vectors, so the $t = 0$ step would look like:

$y_0 = (1, 0, 0, 0, 0, 0, 0, 0, 0)$

This is an observation of a multinomial random variable where the probabilities were those of the first row of the transition probability matrix (and zeroes elsewhere because there is zero probabilit of going, for instance, `C -> I` when the starting sate was `S`.) Thus, the probability governing the multinomial random varaible y_0 is:

$p_0 = (1 - (1 - e^{-\beta I}), 0, 1 - e^{-\beta I}, 0, 0, 0, 0, 0, 0)$

Implementing the model

You may have noticed that the pattern of zeroes in this vector of probabilities p_t will change based on the status at time t. That turns out to stress my knowledge of the `rethinking` package. If the pattern of zeroes weren't changing, you'd simply specify this random variable in `rethinking` as:

```
y ~ categorical( p )
```

Surely there are ways to do it, but my work here is only meant to be quick and educational, not optimal. So what I decided was to break the data into three pieces: one for each starting state (since these correspond to rows of the transition probability matrix.)

Code

Here I create the data that were descibed before and estimate parameters of the model. Running the models below generates errors because the probability C -> C is outside of (0, 1). In fact, it is negative. I suspect that one of us has made a math error in the probabilities - probably the error is in the calculation of $P(C->I)$ (my opinion).

```
# transition indicators for subject one
subj1 <- matrix( c( rep(c(1, 0, 0, 0, 0, 0, 0, 0, 0), 2),
                    c(0, 0, 1, 0, 0, 0, 0, 0, 0),
                    rep(c(0, 0, 0, 0, 0, 0, 0, 0, 1), 3)
                  ), nrow=6, ncol=9, byrow = TRUE)

#transition indicators for subjct two
subj2 <- matrix( c( c(0, 0, 0, 0, 1, 0, 0, 0, 0),
                    c(0, 0, 0, 0, 0, 1, 0, 0, 0),
                    rep(c(0, 0, 0, 0, 0, 0, 0, 0, 1), 4)
                  ), nrow=6, ncol=9, byrow = TRUE)

# combine the data
raw <- rbind(subj1, subj2)

# change from indicators to labels:
dat <- apply(raw, 1, function(row) which(row > 0) )

# isolate the C, S, and I data
S_obs <- dat[ dat >= 1 & dat <= 3 ] - 0
C_obs <- dat[ dat >= 4 & dat <= 6 ] - 3
I_obs <- dat[ dat >= 7 & dat <= 9 ] - 6
```

```
require(rstan)
```

```
## Loading required package: rstan
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.19.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
require(rethinking)
```

```
## Loading required package: rethinking
```

```
## Loading required package: parallel
```

```
## rethinking (Version 1.59)
```

```
SCI_code <- " data{
      int n_S; // number of rows for each starting state
      int n_C;
      int n_I;

      int S[n_S]; // the observed data for each starting state
```

```
      int C[n_C];
      int I[n_I];

      int I_count;

    }
    parameters{
      real<lower=0> alpha;
      real<lower=0> beta;
      real<lower=0> delta;
    }
    model{
      vector[3] pS;
      vector[3] pC;
      vector[3] pI;
      real sum_pC;

      // prior distributions for the parameters (total guesswork, no thought went into these)
      alpha ~ normal(1, 1);
      beta ~ normal(1, 1);
      delta ~ normal(1, 1);

      // poopulate the transition probabilities.

      // transition probability from state S
      pS[1] = exp(-beta * I_count);
      pS[2] = 0;
      pS[3] = 1 - exp(-beta * I_count);

      // transition probability from state C
      pC[1] = 1 - exp(-delta);
      pC[2] = exp(-delta - alpha - beta * I_count);
      pC[3] = 1 - exp(-alpha - beta * I_count);

      sum_pC = pC[1] + pC[2] + pC[3];

      pC[1] = pC[1] / sum_pC;
      pC[2] = pC[2] / sum_pC;
      pC[3] = pC[3] / sum_pC;

      // transition probability from state I
      pI[1] = 0;
      pI[2] = 0;
      pI[3] = 1;

      // define the distribution lf the observed data
      S ~ categorical( pS );
      C ~ categorical( pC );
      I ~ categorical( pI );
    } "

SCI_model <- stan_model(model_code = SCI_code)
```

```
data_list <- list(
  n_S = length(S_obs),
  n_C = length(C_obs),
  n_I = length(I_obs),
  S = S_obs,
  C = C_obs,
  I = I_obs,
  I_count = 7
  )
myfit <- sampling( SCI_model, data=data_list , chains=4 )
precis( myfit , 2 )
```

This model is only a prototype, it needs to be adjusted for things like: calculating I_count from the data, and letting it be different for each observation. I also haven't begun to implement random effects for the different trays.