State Space Reporting Delay

Terms

$N_{t,\infty}$ = total observed cases from time t onward (cases that occurred at time t and were observed at some point?)

$N_{t,T}$ = cases that occurred on time t and were observed by time T

$n_{t,d}$ = cases that occured on time t and were observed exactly d days later

We see that

$$N_{t,T} = \sum_{i=0}^{min(T-t,D)} n_{t,i}$$

$$N_{t,\infty} = \sum_{i=0}^{\infty} n_{t,i} = \sum_{i=0}^{D} n_{t,i}$$

where D is the maximum observed delay value

State Space Reporting Delay

$$X_t \sim N(X_{t-1}, \sigma_X^2)$$
$$\lambda_t \sim N(X_t, \sigma_\lambda^2)$$

$$N_{t,\infty}|\lambda_t \sim Pois(exp(\lambda_t))$$
$$N_{t,T}|N_{t,\infty}, \lambda_t \sim Binom(N_{t,\infty}, q_{T-t})$$

(OR?:

$$(n_{t,0}, \ldots, n_{t,T})|N_{t,\infty}, \lambda_t \sim Multinomial(N_{t,\infty}, p)$$

)

where

$$q_{T-t} = \sum_{i=1}^{T-t} p_{t,d}$$

and

$$(p_{t,1}, p_{t,2}, ..., p_{t,D}) \sim Dirichlet(\alpha_1, \alpha_2, ..., \alpha_D)$$

```r
library(ggplot2)
library(Metrics)
library(MCMCpack)
```

```
## Loading required package: coda
```

```
## Loading required package: MASS
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
## ## Copyright (C) 2003-2018 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
## ##
## ## Support provided by the U.S. National Science Foundation
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

```r
require(rbiips)
```

```
## Loading required package: rbiips
```

```r
## custom factorization fo joint density
dMN_dim <- function(x,lam,n_t_inf,a1,a2) {
    # Check dimensions of the input and return dimension of the output of
    # distribution dMN
    5
}
dMN_sample <- function(x,lam,n_t_inf,a1,a2,a3) {
    # Draw a sample of distribution dMN
    x1 <- rnorm(1,x,1)
    lam1 <- rnorm(1,x1,1)
    n_t_inf1 <- rpois(1,exp(lam1))
    qt <- rdirichlet(1,c(a1,a2))
    return (c(x1,lam1,n_t_inf1,qt))

}
biips_add_distribution('dcustom', 5, dMN_dim, dMN_sample)
```

```
## * Added distribution dcustom
```

```r
# custom dirichlet distribution
ddirichlet_dim <- function(a1,a2,a3) {
  # Check dimensions of the input and return dimension of the output of
  # distribution dMN
  3
}
ddirichlet_sample <- function(a1,a2,a3) {
  # Draw a sample of distribution dMN
  return (rdirichlet(1,c(a1,a2,a3)))

}
biips_add_distribution('ddirch', 3, ddirichlet_dim, ddirichlet_sample)
```

```
## * Added distribution ddirch
```

```r
## define data, constants, and initial values
## here we take T = 4  and D = 3 for illustration so the
## reporting trapezoid should look like
## n_{4,0}, 0      , 0      , 0
## n_{3,0}, n_{3,1}, 0      , 0
## n_{2,0}, n_{2,1}, n_{2,2}, 0
## n_{1,0}, n_{1,1}, n_{1,2}, 0
T_max = 4
d = 3
n_t_d = matrix(c(
  3, 0,0,0,
  2,2,0,0,
  4,3,4,0,
  1,2,3,0
), nrow =T_max,ncol=T_max,byrow = TRUE)

## compute N_{t,Ts}

N_t_T = c(
  sum(n_t_d[4,1:3]),sum(n_t_d[3,1:3]),sum(n_t_d[2,1:3]),sum(n_t_d[1,1:3])
)







model_file = '/Users/gcgibson/reporting_delay/delay.bug' # BUGS model filename
cat(readLines(model_file), sep = "\n")
```

```
## var x[5,t_max],y[t_max]
##
## model
## {
##
##   x[,1] ~ dcustom(mean_x_init[1],mean_x_init[2],mean_x_init[3],mean_x_init[4],mean_x_init[5])
##   y[1] ~ dbinom(x[4,1],x[3,1] + 100)
##   for (t in 2:t_max)
##   {
##     x[,t] ~ dcustom(x[1,t-1],x[2,t-1],x[3,t-1],x[4,t-1],x[5,t-1])
##     y[t] ~ dbinom(x[4,1],x[3,t]+100)
##   }
## }
```

```r
t_max = length(N_t_T)
data = list(t_max=t_max, y = N_t_T,mean_x_init=c(1,1,1,1,1))
sample_data = FALSE # Boolean
model = biips_model(model_file, data, sample_data=sample_data) # Create Biips model and sample data
```

```
## * Parsing model in: /Users/gcgibson/reporting_delay/delay.bug
## * Compiling model graph
##   Declaring variables
##   Resolving undeclared variables
##   Allocating nodes
##   Graph size: 36
```

```r
n_part = 1000 # Number of particles
variables = c('x') # Variables to be monitored
mn_type = 'fs'; rs_type = 'stratified'; rs_thres = .5 # Optional parameters
out_smc = biips_smc_samples(model, variables, n_part,
                            type=mn_type, rs_type=rs_type, rs_thres=rs_thres)
```

```
## * Assigning node samplers
## * Running SMC forward sampler with 1000 particles
##   |--------------------------------------------------| 100%
##   |**************************************************| 4 iterations in 0.70 s
```

```
diag_smc = biips_diagnosis(out_smc)
```

```
## * Diagnosis of variable: x[1:5,1:4]
##   Filtering: GOOD
##   Smoothing: GOOD
```

```
summ_smc = biips_summary(out_smc, probs=c(.025, .975))
print (summ_smc$x$f$mean)
```

```
##            [,1]       [,2]       [,3]       [,4]
## [1,] 1.00934457 0.98651468 0.70061883 0.53501056
## [2,] 1.07994981 0.99718664 0.53144343 0.43001177
## [3,] 6.46166429 9.66301900 5.79825395 6.50593798
## [4,] 0.06658286 0.08789479 0.08428532 0.07369302
## [5,] 0.93341714 0.91210521 0.91571468 0.92630698
```