This is the example I tried to modify:
https://kingaa.github.io/sbied/stochsim/notes.pdf
There are some exercises, solutions can be found here:
https://kingaa.github.io/sbied/stochsim/exercises.html
Similar example modified here:
http://kingaa.github.io/short-course/stochsim/stochsim.html

```
load("worked_example_data.RData")
```

```
# loading observed data
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4
```

```
## Warning: package 'readr' was built under R version 4.1.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::map()    masks pomp::map()
```

```
read_csv(paste0("https://kingaa.github.io/sbied/stochsim/", "Measles_Consett_1948.csv")) %>% select(week
```

```
## Rows: 53 Columns: 2
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (2): week, cases
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
meas %>% as.data.frame() %>% print.data.frame()
```

```
##    week reports
## 1     1       0
## 2     2       0
## 3     3       2
## 4     4       0
## 5     5       3
## 6     6       0
## 7     7       1
## 8     8       0
## 9     9       2
## 10   10       4
## 11   11       2
## 12   12       4
## 13   13       7
## 14   14      34
## 15   15      35
## 16   16      22
## 17   17      18
## 18   18      75
## 19   19      43
```

```
## 20   20      47
## 21   21      44
## 22   22      63
## 23   23      49
## 24   24      17
## 25   25      19
## 26   26      16
## 27   27       1
## 28   28       2
## 29   29       0
## 30   30       1
## 31   31       1
## 32   32       1
## 33   33       1
## 34   34       1
## 35   35       4
## 36   36       1
## 37   37       0
## 38   38       1
## 39   39       0
## 40   40       0
## 41   41       0
## 42   42       0
## 43   43       0
## 44   44       0
## 45   45       0
## 46   46       0
## 47   47       0
## 48   48       0
## 49   49       0
## 50   50       0
## 51   51       0
## 52   52       0
## 53   53       0
```

SIR model

```
sir_step <- function (S, I, R, N, Beta, mu_IR, delta.t, ...) {
  dN_SI <- rbinom(n=1,size=S,prob=1-exp(-Beta*I/N*delta.t))
  dN_IR <- rbinom(n=1,size=I,prob=1-exp(-mu_IR*delta.t))
  S <- S - dN_SI
  I <- I + dN_SI - dN_IR
  R <- R + dN_IR
  c(S = S, I = I, R = R) # Return S I R at the end of each step
}
```

```
# initialise the model using N (population size) and eta, some fraction of susceptibles
sir_rinit <- function (N, eta, ...) {
  c(S = round(N * eta),
    I = 1,
    R = round(N * (1 - eta)))
}
```

```
library(pomp)
meas %>%
  pomp(
    times = "week",
    t0 = 0,
    rprocess = euler(sir_step, delta.t = 1 / 7),
    rinit = sir_rinit
  ) -> measSIR
```

SIRH

H is an accumulator variable. It is reset every week and "observes" the unreported cases (or so I understand). I don't quite understand where/why we would introduce this in our (Tom's) model. Besides, the code won't run with H, so I've completely excluded it. Which then messes up the meaning of k and p (see dmeas and rmeas).

```
sir_step <- function (S, I, R, H, N, Beta, mu_IR, delta.t, ...) {
  dN_SI <- rbinom(n=1,size=S,prob=1-exp(-Beta*I/N*delta.t))
  dN_IR <- rbinom(n=1,size=I,prob=1-exp(-mu_IR*delta.t))
  S <- S - dN_SI
  I <- I + dN_SI - dN_IR
  R <- R + dN_IR
  H <- H + dN_IR;
c(S = S, I = I, R = R, H = H)
}

sir_rinit <- function (N, eta, ...) {
  c(
    S = round(N * eta),
    I = 1,
    R = round(N * (1 - eta)),
    H = 0
  )
}
```

```
measSIR %>%
  pomp(
    rprocess = euler(sir_step, delta.t = 1 / 7),
    rinit = sir_rinit,
    accumvars = "H"
  ) -> measSIR
```

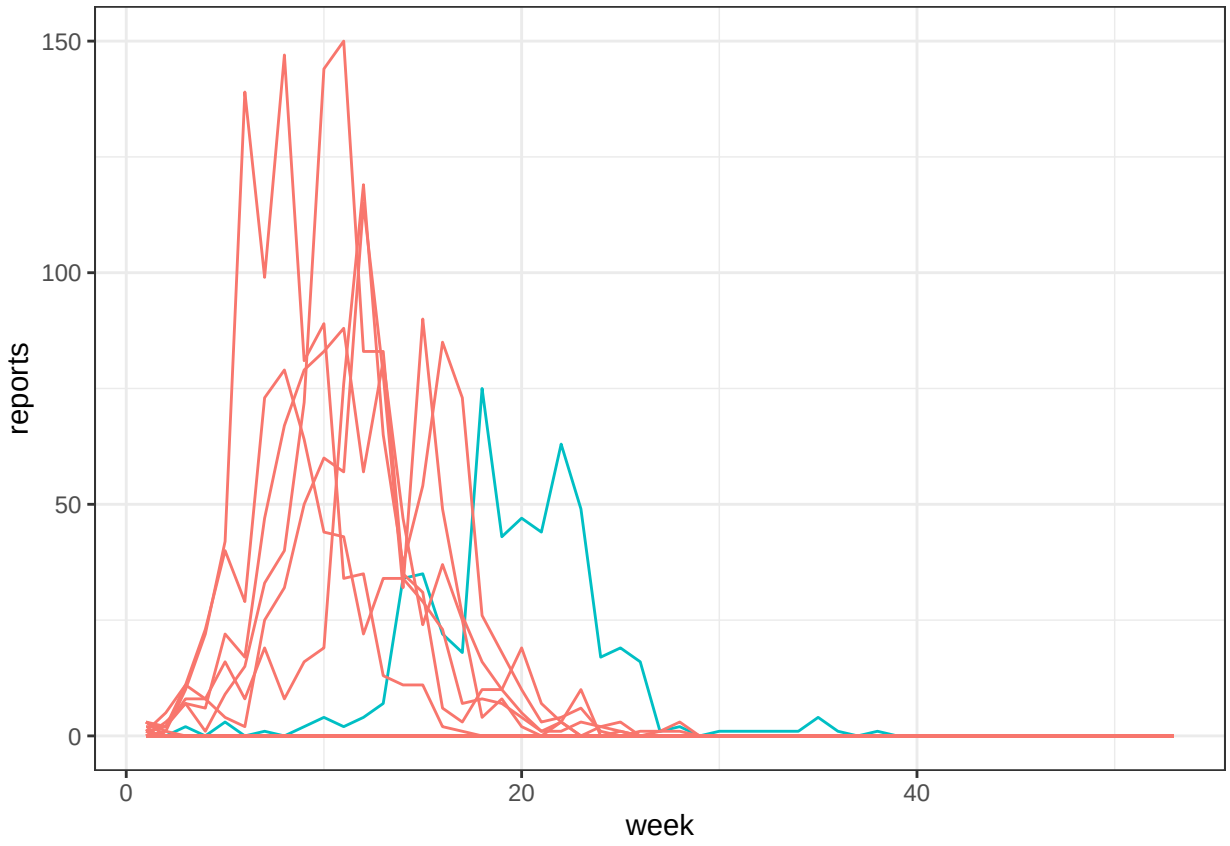
dmeas and rmeas functions

```
# dmeas will be used for likelihood estimation
sir_dmeas <- function(reports, H, rho, k, log, ...) {
  dnbinom(
    x = reports,
    size = k,
    mu = rho * H,
    log = log
  )
}
# rmeas will be used for simulation
```

```
# effectively, probability of a case (ie the data) ~ NegBin (pH, k)
# where p = probability of reporting/diagnosing/observing a case
# H = true incidence, an accumulator variable which is reset after every observation(?)
sir_rmeas <- function (H, rho, k, ...) {
  c(reports = rnbinom(n = 1, size = k, mu = rho * H))
}
```

```
measSIR %>%
  pomp(rmeasure = sir_rmeas,
       dmeasure = sir_dmeas) -> measSIR
```

```
## Running some simulations and plotting the output.
measSIR %>%
  simulate(params = c(Beta = 30, mu_EI = 0.8, mu_IR = 1.3,
                      rho = 0.5, k = 10, eta = 0.06, N = 38000),nsim = 20,
           format = "data.frame", include.data = TRUE) %>%
  ggplot(aes(x = week, y = reports, group = .id, color = .id == "data")) +
  geom_line() + guides(color = "none") + theme_bw()
```



C snippets which achieve the exact same end. Not used bc Li said Csnippets are impossible to debug.

```
seir_step <- Csnippet("
  double dN_SE = rbinom(S,1-exp(-Beta*I/N*dt));
  double dN_EI = rbinom(E,1-exp(-mu_EI*dt));
  double dN_IR = rbinom(I,1-exp(-mu_IR*dt));
  S -= dN_SE;
  E += dN_SE - dN_EI;
```

```
  I += dN_EI - dN_IR;
  R += dN_IR;
  H += dN_IR;
")


seir_init <- Csnippet("
  S = nearbyint(eta*N);
  E = 0;
  I = 1;
  R = nearbyint((1-eta)*N);
  H = 0;
")

measSIR %>%
  pomp(
    rprocess=euler(seir_step,delta.t=1/7),
    rinit=seir_init,
    paramnames=c("N","Beta","mu_EI","mu_IR","rho","eta"),
    statenames=c("S","E","I","R","H")
  ) -> measSEIR
```