discrete/binomial pomp analysis

2025-05-22

f_{proc} = discrete, f_{meas} = binomial

We will first present the one-unit (London case) discrete + binomial scenario, and then show how to match it to the WWR model.

Preparation for the pomp

```
## ----packages,incluxde=F,echo=F,cache=F----------------------------------------
library("pomp")
library("spatPomp")
library("ggplot2")
library("tidyverse")
library("knitr")
library("doRNG")
library("doParallel")

set.seed(12345)

measles_cases <- read.csv("case1.csv")
measles_covar <- read.csv("covar2.csv")

measles_cases<- measles_cases[measles_cases$city == "LONDON", ]
measles_covar <- measles_covar[measles_covar$city == "LONDON", ]


measles_cases <-  measles_cases[,-1]
measles_covar <-  measles_covar[,-1]



colnames(measles_cases) <- c("time","cases1")
colnames(measles_covar) <- c("time",
                             "lag_birthrate1","pop1")



basic_params <- c(
  alpha     = 1,
  iota      = 0,
  betabar   = 10,
  c         = 0.1,
  a         = 0.3,
  rho       = 0.1,
  gamma     = 0.1,
  delta     = 0.02/(26*4),  # Here is a time scale transform.
```

```
  sigma_xi   =  2,
  qmean      = 0.7,
  qvar       = 0.2,
  g          = 0,
  S_0        = 0.015,
  E_0        = 0.0002,
  I_0        = 0.0002
)
```

f_{proc} = discrete

```
rproc <- Csnippet("
  double t_mod = fmod(t, 364.0);
  double br1;
  double beta1, seas1;
  double foi1;
  double xi1;
  double betafinal1;
  static double betafinal1_prev = 0.0;

  int trans_S1[2], trans_E1[2], trans_I1[2];
  double prob_S1[2], prob_E1[2], prob_I1[2];

  if ((t_mod >= 6 && t_mod < 99) ||
      (t_mod >= 115 && t_mod < 198) ||
      (t_mod >= 252 && t_mod < 299) ||
      (t_mod >= 308 && t_mod < 355)) {
    seas1 = 1.0 + a * 2 * (1 - 0.759);
  } else {
    seas1 = 1.0 - 2 * a * 0.759;
  }

  beta1 = betabar * seas1;

  if (fabs(t_mod - 248.5) < 0.5) {
    br1 = c * lag_birthrate1;
  } else {
    br1 = (1.0 - c) * lag_birthrate1 / 103.0;
  }

  double I_ratio1 = I1 / pop1;

  foi1 = pow((I1 + iota) / pop1, alpha);

  stepCount += 1.0;

  if (fabs(fmod(stepCount, 4.0)) < 1e-8) {
    xi_current = rgamma(sigma_xi, 1 / sigma_xi);
    betafinal1 = beta1 * I_ratio1 * xi_current;
  } else {
    betafinal1 = betafinal1_prev;
  }
```

```
  betafinal1_prev = betafinal1;

  int SD1 = rbinom(S1, delta);
  int ED1 = rbinom(E1, delta);
  int ID1 = rbinom(I1, delta);
  int RD1 = rbinom(R1, delta);

  S1 -= SD1;
  E1 -= ED1;
  I1 -= ID1;
  R1 -= RD1;

  prob_S1[0] = exp(-dt * betafinal1);
  prob_S1[1] = 1 - exp(-dt * betafinal1);

  prob_E1[0] = exp(-dt * rho);
  prob_E1[1] = 1 - exp(-dt * rho);

  prob_I1[0] = exp(-dt * gamma);
  prob_I1[1] = 1 - exp(-dt * gamma);

  rmultinom(S1, prob_S1, 2, trans_S1);
  rmultinom(E1, prob_E1, 2, trans_E1);
  rmultinom(I1, prob_I1, 2, trans_I1);

  S1 = trans_S1[0] + rpois(br1);
  E1 = trans_E1[0] + trans_S1[1];
  I1 = trans_I1[0] + trans_E1[1];
  R1 += trans_I1[1];
  C1 += trans_I1[1];

  q1 = -1;
  while (q1 < 0 || q1 > 1) {
    q1 = rnorm(qmean, qvar);
  }
");
```

f_{meas} = binomial

```
# --- 5.1 dmeasure ---
dmeas <- Csnippet("
 lik =  dbinom(cases1, C1, q1, 1);
");

# --- 5.2 rmeasure ---
rmeas <- Csnippet("
 cases1 = rbinom(C1,q1);
")
```

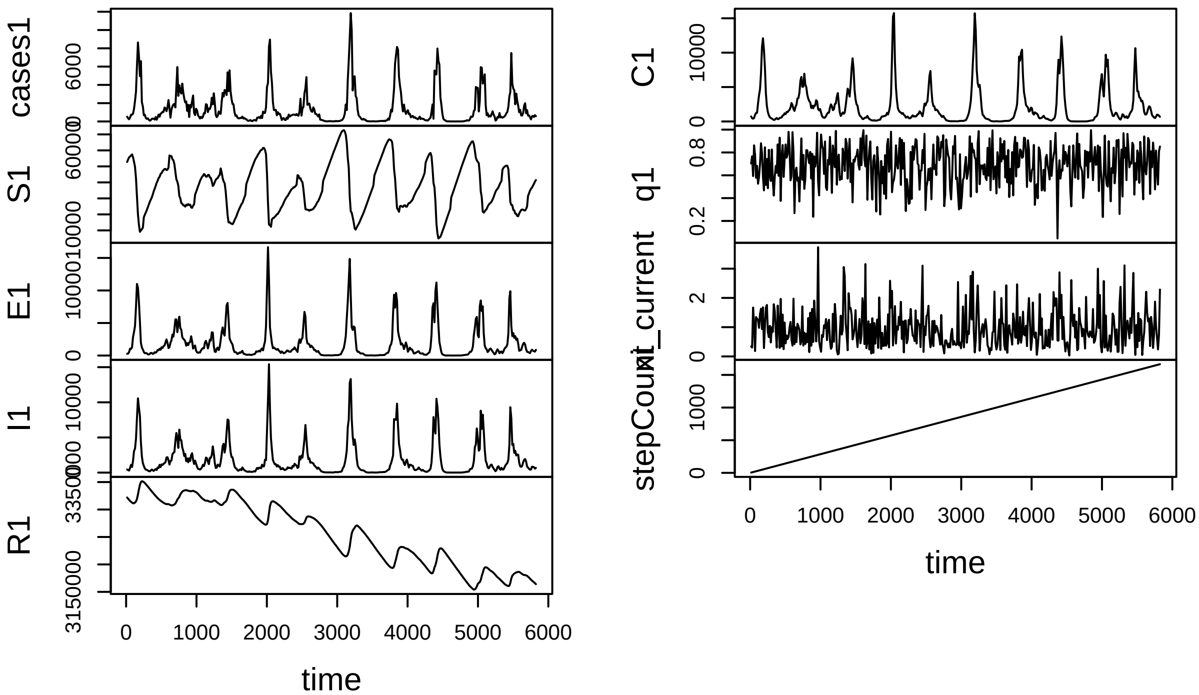
Build the rinit

```
rinit <- Csnippet("
  double probs1[4];
  probs1[0] = S_0;
  probs1[1] = E_0;
  probs1[2] = I_0;
  probs1[3] = 1.0 - probs1[0] - probs1[1] - probs1[2];

  int counts1[4];
  rmultinom(pop1, probs1, 4, counts1);

  S1 = counts1[0];
  E1 = counts1[1];
  I1 = counts1[2];
  R1 = counts1[3];
  C1 = 0;
  xi_current = 1;
  stepCount = 0;
")
```

Construct the POMP

```
basic_log_names   <- c("rho", "gamma", "sigma_xi", "betabar", "g", "iota", "delta")
basic_logit_names <- c("a", "alpha", "c", "qmean", "S_0", "E_0", "I_0", "qvar")
log_names   <- basic_log_names
logit_names <- basic_logit_names
measles_partrans <- parameter_trans(
  log   = log_names,
  logit = logit_names
)

one_city_pomp <- pomp(
  data       = measles_cases,
  times      = "time",
  t0         = 0,
  rprocess   = discrete_time(rproc, delta.t = 3.5),
  rinit      = rinit,
  dmeasure   = dmeas,
  rmeasure   = rmeas,
  statenames = c("S1","E1","I1","R1","C1","q1","xi_current","stepCount"),
  paramnames = c("alpha","iota","betabar","c","a","rho","gamma",
                 "delta","sigma_xi","g","qmean","qvar",
                 "S_0","E_0","I_0"),
  covar      = covariate_table(measles_covar,times = "time"),
  covarnames = c("lag_birthrate1","pop1"),
  accumvars  = c("C1")
)

coef(one_city_pomp) <- basic_params

sim <- simulate(one_city_pomp, params =  basic_params)
```

```r
plot(sim)
```



Make the simulation

```r
# Number of simulations
n_simulations <- 1000

# Placeholder to store the results
results <- data.frame(
  mean = numeric(n_simulations),
  median = numeric(n_simulations),
  variance = numeric(n_simulations)
)

# Loop through the simulations
for (i in 1:n_simulations) {

  # Simulate the system
  sim <- simulate(one_city_pomp, params = basic_params)
  sim_data <- as.data.frame(sim)
  simlondon <- sim_data$cases1


  # Calculate the mean, median, and variance of the simulation data
  results$mean[i] <- mean(simlondon, na.rm = TRUE)
  results$median[i] <- median(simlondon, na.rm = TRUE)
  results$variance[i] <- var(simlondon, na.rm = TRUE)
}
```

Python code for simulation

```
import os
os.environ["PYTHONHASHSEED"] = "12345"

import random
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow_probability as tfp
import matplotlib.pyplot as plt
from scipy.optimize import minimize

import sys
sys.path.append('Scripts/')
from measles_simulator_KH import *

SEED = 12345
random.seed(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)

UKbirths_array = np.load("Data/UKbirths_array.npy")
UKpop_array = np.load("Data/UKpop_array.npy")
measles_distance_matrix_array = np.load("Data/measles_distance_matrix_array.npy")
UKmeasles_array = np.load("Data/UKmeasles_array.npy")
modelA_array = np.load("Data/Parameter/final_parameters_lookahead_A.npy")

UKbirths = tf.convert_to_tensor(UKbirths_array, dtype = tf.float32)
UKpop = tf.convert_to_tensor(UKpop_array, dtype = tf.float32)
measles_distance_matrix = tf.convert_to_tensor(measles_distance_matrix_array,
dtype = tf.float32)
UKmeasles = tf.convert_to_tensor(UKmeasles_array, dtype = tf.float32)

df = pd.read_csv("Data/londonbirth.csv")
data_array = df.values
UKbirths = tf.convert_to_tensor(data_array, dtype=tf.float32)

df1 = pd.read_csv("Data/londonpop.csv")
data_array1 = df1.values
UKpop = tf.convert_to_tensor(data_array1, dtype=tf.float32)

term   = tf.convert_to_tensor([6, 99, 115, 198, 252, 299, 308, 355, 366],
dtype = tf.float32)
school = tf.convert_to_tensor([0, 1, 0, 1, 0, 1, 0, 1, 0], dtype =
tf.float32)

n_cities = tf.constant(40, dtype = tf.int64)

initial_pop = UKpop[:,0]

T = UKmeasles.shape[1]
intermediate_steps = 4
h = tf.constant(14/tf.cast(intermediate_steps, dtype = tf.float32), dtype =
```

```
tf.float32)
is_school_term_array, is_start_school_year_array, times_total, times_obs =
school_term_and_school_year(T, intermediate_steps, term, school)

is_school_term_array = tf.convert_to_tensor(is_school_term_array, dtype =
tf.float32)
is_start_school_year_array = tf.convert_to_tensor(is_start_school_year_array,
dtype = tf.float32)

pi_0_1 = 0.02545
pi_0_2 = 0.00422
pi_0_3 = 0.000061
pi_0 = tf.convert_to_tensor([[pi_0_1, pi_0_2, pi_0_3, 1 - pi_0_1 - pi_0_2 -
pi_0_3]], dtype = tf.float32)*tf.ones((n_cities, 4), dtype = tf.float32)

initial_pop = UKpop[:,0]

beta_bar  = tf.convert_to_tensor( [[6.32]], dtype =
tf.float32)*tf.ones((n_cities, 1), dtype = tf.float32)
rho   = tf.convert_to_tensor([[0.142]], dtype =
tf.float32)*tf.ones((n_cities, 1), dtype = tf.float32)
gamma = tf.convert_to_tensor([[0.0473]], dtype =
tf.float32)*tf.ones((n_cities, 1), dtype = tf.float32)

g = tf.convert_to_tensor([[0]], dtype = tf.float32)*tf.ones((n_cities, 1),
dtype = tf.float32)
p = tf.constant(0.759, dtype = tf.float32)
a = tf.constant(0.1476,   dtype = tf.float32)
c = tf.constant(0.219,   dtype = tf.float32)

Xi = tfp.distributions.Gamma(concentration = 0.318, rate = 0.318)
Q  = tfp.distributions.TruncatedNormal( 0.7, 0.306, 0, 1)

delta_year = tf.convert_to_tensor([[1/50]], dtype =
tf.float32)*tf.ones((n_cities, 4), dtype = tf.float32)

T_small = tf.constant(415, dtype = tf.float32)

# Initialize result lists
means = np.zeros((40, 25))
variances = np.zeros((40, 25))
medians = np.zeros((40, 25))

# Perform 1000 simulations
for i in range(25):
    X_t, Y_t, Xi_t, Q_t = run(T_small, intermediate_steps, UKbirths, UKpop,
    g, measles_distance_matrix,
                              initial_pop, pi_0, beta_bar, p, a, is_school_term_array,
                              is_start_school_year_array, h, rho, gamma, Xi, Q, c, n_cities, delta_year



    max_time = 415
```

```python
    # Calculate the log values for each city Y_t_log
    for city in range(40):
        Y_t_log = Y_t[1:(max_time + 1), city, 0]
        # Calculate mean, variance, median
        means[city, i] = np.mean(Y_t_log)
        variances[city, i] = np.var(Y_t_log)
        medians[city, i] = np.median(Y_t_log)

# Initialize an empty list to store the results for each city
all_results = []

for city_index in range(40):
    # Create a DataFrame for the current city
    results_df_city = pd.DataFrame({
        'Simulation': np.arange(25),
        'Mean': means[city_index, :],
        'Variance': variances[city_index, :],
        'Median': medians[city_index, :]
    })

    # Add city column
    results_df_city['City'] = city_index

    # Add the current city's results to the total list
    all_results.append(results_df_city)

# Combine results from all cities
combined_results_df = pd.concat(all_results, ignore_index=True)

# Save to a new CSV file
combined_results_df.to_csv("/Users/mac/Desktop/PAL_measles/combined_simulation_results_for_1000_times.c
```

```r
combined_simulation_results_for_1000_times <-
  read.csv("combined_simulation_results_for_1000_times.csv")

t.test(results$mean,combined_simulation_results_for_1000_times$Mean)
```

```
##
##  Welch Two Sample t-test
##
## data:  results$mean and combined_simulation_results_for_1000_times$Mean
## t = 0.24729, df = 1984.4, p-value = 0.8047
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -8.482432 10.930195
## sample estimates:
## mean of x mean of y
##  1416.620  1415.396
```

```r
t.test(results$median,combined_simulation_results_for_1000_times$Median)
```

```
##
##  Welch Two Sample t-test
##
```

```
## data:  results$median and combined_simulation_results_for_1000_times$Median
## t = 0.22122, df = 1996.4, p-value = 0.8249
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -13.56325  17.01225
## sample estimates:
## mean of x mean of y
##  783.0875  781.3630
```

```
t.test(results$variance,combined_simulation_results_for_1000_times$Variance)
```

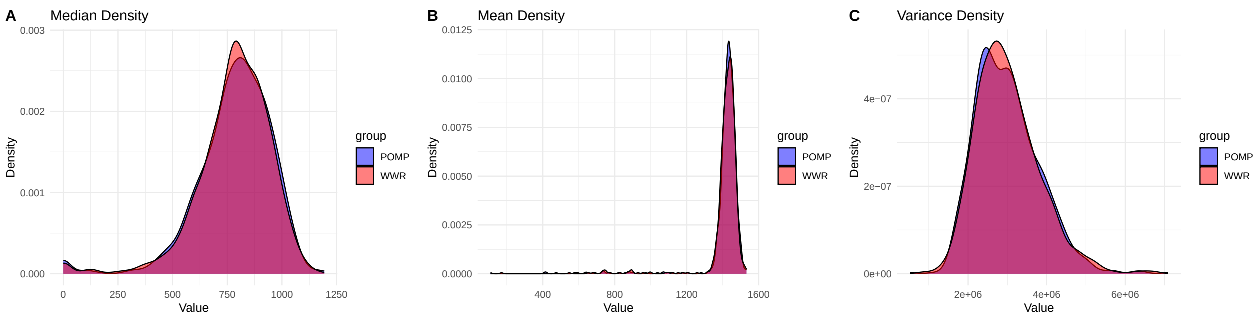
```
##
##  Welch Two Sample t-test
##
## data:  results$variance and combined_simulation_results_for_1000_times$Variance
## t = -0.29231, df = 1997.7, p-value = 0.7701
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -81435.56  60308.73
## sample estimates:
## mean of x mean of y
##   2999552   3010115
```

```
data <- data.frame(
  value = c(results$median, combined_simulation_results_for_1000_times$Median),
  group = rep(c("POMP", "WWR"),
              c(length(results$median), length(combined_simulation_results_for_1000_times$Median)))
)

p_median <- ggplot(
  data.frame(
    value = c(results$median, combined_simulation_results_for_1000_times$Median),
    group = rep(c("POMP", "WWR"),
                c(length(results$median),
                  length(combined_simulation_results_for_1000_times$Median)))
  ),
  aes(x = value, fill = group)
) +
  geom_density(alpha = 0.5) +
  labs(title = "Median Density", x = "Value", y = "Density") +
  theme_minimal() +
  scale_fill_manual(values = c("blue", "red"))

p_mean <- ggplot(
  data.frame(
    value = c(results$mean, combined_simulation_results_for_1000_times$Mean),
    group = rep(c("POMP", "WWR"),
                c(length(results$mean),
                  length(combined_simulation_results_for_1000_times$Mean)))
  ),
  aes(x = value, fill = group)
) +
  geom_density(alpha = 0.5) +
  labs(title = "Mean Density", x = "Value", y = "Density") +
  theme_minimal() +
```

```r
  scale_fill_manual(values = c("blue", "red"))

p_variance <- ggplot(
  data.frame(
    value = c(results$variance, combined_simulation_results_for_1000_times$Variance),
    group = rep(c("POMP", "WWR"),
                c(length(results$variance),
                  length(combined_simulation_results_for_1000_times$Variance)))
  ),
  aes(x = value, fill = group)
) +
  geom_density(alpha = 0.5) +
  labs(title = "Variance Density", x = "Value", y = "Density") +
  theme_minimal() +
  scale_fill_manual(values = c("blue", "red"))


library(cowplot)
```

```
##
## Attaching package: 'cowplot'

## The following object is masked from 'package:lubridate':
##
##     stamp
```

```r
plot_grid(
  p_median, p_mean, p_variance,
  labels = c("A", "B", "C"),
  ncol   = 3,
  align  = "h"
)
```



f_{proc} = discrete, f_{meas} = gaussian

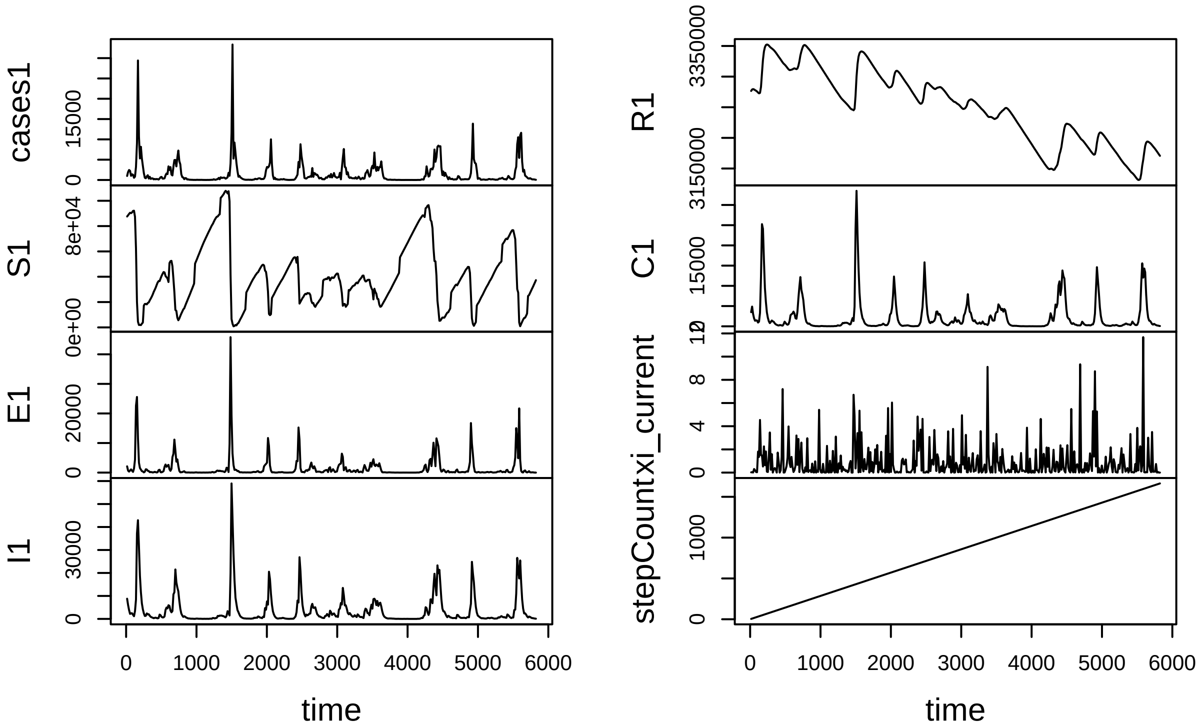
```r
basic_params <- c(
  alpha        = 1,
  iota         = 0,
  betabar      = 7.32,
  c            = 0.219,
  a            = 0.1476,
  rho          = 0.142,
  gamma        = 0.0473,
```

```
  delta       = 0.02/(26*4),  # timescale transform
  sigma_xi    = 0.318,
  gaussianrho = 0.7,
  psi         = 0.306,
  g           = 0,
  S_0         = 0.02545,
  E_0         = 0.00422,
  I_0         = 0.000061
)

rproc <- Csnippet("
  double t_mod = fmod(t, 364.0);
  double br1;
  double beta1, seas1;
  double foi1;
  double xi1;
  double betafinal1;
  static double betafinal1_prev = 0.0;

  int trans_S1[2], trans_E1[2], trans_I1[2];
  double prob_S1[2], prob_E1[2], prob_I1[2];

  if ((t_mod >= 6 && t_mod < 99) ||
      (t_mod >= 115 && t_mod < 198) ||
      (t_mod >= 252 && t_mod < 299) ||
      (t_mod >= 308 && t_mod < 355)) {
    seas1 = 1.0 + a * 2 * (1 - 0.759);
  } else {
    seas1 = 1.0 - 2 * a * 0.759;
  }

  beta1 = betabar * seas1;

  if (fabs(t_mod - 248.5) < 0.5) {
    br1 = c * lag_birthrate1;
  } else {
    br1 = (1.0 - c) * lag_birthrate1 / 103.0;
  }

  double I_ratio1 = I1 / pop1;

  foi1 = pow((I1 + iota) / pop1, alpha);

  stepCount += 1.0;

  if (fabs(fmod(stepCount, 4.0)) < 1e-8) {
    xi_current = rgamma(sigma_xi, 1 / sigma_xi);
    betafinal1 = beta1 * I_ratio1 * xi_current;
  } else {
    betafinal1 = betafinal1_prev;
  }

  betafinal1_prev = betafinal1;
```

```
  int SD1 = rbinom(S1, delta);
  int ED1 = rbinom(E1, delta);
  int ID1 = rbinom(I1, delta);
  int RD1 = rbinom(R1, delta);

  S1 -= SD1;
  E1 -= ED1;
  I1 -= ID1;
  R1 -= RD1;

  prob_S1[0] = exp(-dt * betafinal1);
  prob_S1[1] = 1 - exp(-dt * betafinal1);

  prob_E1[0] = exp(-dt * rho);
  prob_E1[1] = 1 - exp(-dt * rho);

  prob_I1[0] = exp(-dt * gamma);
  prob_I1[1] = 1 - exp(-dt * gamma);

  rmultinom(S1, prob_S1, 2, trans_S1);
  rmultinom(E1, prob_E1, 2, trans_E1);
  rmultinom(I1, prob_I1, 2, trans_I1);

  S1 = trans_S1[0] + rpois(br1);
  E1 = trans_E1[0] + trans_S1[1];
  I1 = trans_I1[0] + trans_E1[1];
  R1 += trans_I1[1];
  C1 += trans_I1[1];
");




## ----dmeasure-------------------------------------------------
dmeas <- Csnippet("
  double m = gaussianrho*C1;
  double v = m*(1.0-gaussianrho+psi*psi*m);
  double tol = 0.0;
  if (cases1 > 0.0) {
    lik = pnorm(cases1+0.5,m,sqrt(v)+tol,1,0)
           - pnorm(cases1-0.5,m,sqrt(v)+tol,1,0) + tol;
  } else {
    lik = pnorm(cases1+0.5,m,sqrt(v)+tol,1,0) + tol;
  }
  if (give_log) lik = log(lik);
")

## ----rmeasure-------------------------------------------------
rmeas <- Csnippet("
  double m = gaussianrho*C1;
  double v = m*(1.0-gaussianrho+psi*psi*m);
  double tol = 0.0;
  cases1 = rnorm(m,sqrt(v)+tol);
```

```
  if (cases1 > 0.0) {
    cases1 = nearbyint(cases1);
  } else {
    cases1 = 0.0;
  }
")

rinit <- Csnippet("
  double probs1[4];
  probs1[0] = S_0;
  probs1[1] = E_0;
  probs1[2] = I_0;
  probs1[3] = 1.0 - probs1[0] - probs1[1] - probs1[2];

  int counts1[4];
  rmultinom(pop1, probs1, 4, counts1);

  S1 = counts1[0];
  E1 = counts1[1];
  I1 = counts1[2];
  R1 = counts1[3];
  C1 = 0;
  xi_current = 1;
  stepCount = 0;
");

basic_log_names   <- c("rho", "gamma", "sigma_xi", "betabar", "g", "iota", "delta")
basic_logit_names <- c("a", "alpha", "c", "gaussianrho", "S_0", "E_0", "I_0", "psi")
log_names   <- basic_log_names
logit_names <- basic_logit_names
measles_partrans <- parameter_trans(
  log   = log_names,
  logit = logit_names
)

one_city_pomp <- pomp(
  data       = measles_cases,
  times      = "time",
  t0         = 0,
  rprocess   = discrete_time(rproc, delta.t = 3.5),
  rinit      = rinit,
  dmeasure   = dmeas,
  rmeasure   = rmeas,
  statenames = c("S1","E1","I1","R1","C1","xi_current","stepCount"),
  paramnames = c("alpha","iota","betabar","c","a","rho","gamma",
                 "delta","sigma_xi","g","gaussianrho","psi",
                 "S_0","E_0","I_0"),
  covar      = covariate_table(measles_covar,times = "time"),
  covarnames = c("lag_birthrate1","pop1"),
  accumvars  = c("C1")
)

coef(one_city_pomp) <- basic_params
```

```r
sim <- simulate(one_city_pomp, params =  basic_params, seed   = 154234)

plot(sim)
```



Compute the loglik

```r
tmp <- pfilter(sim,Np = 10000)
tmp@loglik
```

```
## [1] -Inf
```

```r
tmp@cond.logLik
```

```
##   [1]  -9.494812 -10.787597 -12.623799 -12.869915 -14.154716 -13.166800
##   [7] -11.890800 -11.898447  -9.724651 -14.525150       -Inf       -Inf
##  [13]       -Inf       -Inf       -Inf       -Inf       -Inf       -Inf
##  [19] -15.574410  -6.203311  -7.839913  -8.636693  -8.524084  -6.560429
##  [25]  -7.990728  -6.046247  -6.200163  -5.509151  -6.782622  -5.313677
##  [31]  -5.461666  -5.526466  -5.001368  -9.374375  -8.340343  -6.653450
##  [37] -10.211281 -10.172066 -10.683442  -8.429386 -10.086998 -10.381961
##  [43]  -8.003158  -9.678449  -8.957706  -9.927371  -9.131922  -7.986091
##  [49] -10.933448  -9.764193  -8.283301 -10.676415  -9.373010  -8.283960
##  [55]  -8.483701  -9.064544  -9.223916  -7.667776  -8.976317  -6.104454
##  [61]  -7.001441  -5.829780  -5.660075  -4.677345  -5.606578  -8.928879
##  [67]  -8.801762  -9.496964  -8.679481  -7.226033  -6.627025  -6.544387
##  [73] -10.150877 -10.716541 -12.067512 -12.159810 -12.521240 -12.771102
##  [79] -13.253962 -13.011790 -12.704195 -12.341805 -12.042625 -11.912367
##  [85] -12.261770 -12.614841 -12.155046 -11.722138  -8.954579  -9.237954
##  [91] -10.812210 -10.812315  -9.858397  -6.843497 -10.262388  -6.584894
##  [97]  -6.960908  -6.278151  -6.995651  -6.929249  -6.045439  -6.049717
```

```
## [103]  -6.885912  -7.322437  -7.239320 -12.347520 -14.527406 -39.419183
## [109]  -9.190179 -14.230910 -17.667550 -12.657901 -10.469883  -6.443920
## [115] -11.291150  -7.356799  -6.881090  -5.828546  -5.281814  -4.694546
## [121]  -4.525057  -3.928056  -3.763716  -6.958931  -4.168525  -4.751806
## [127]  -5.571073  -4.903937  -4.414798  -5.415225  -9.203962  -5.498849
## [133]  -6.049403  -6.793574  -6.969053  -6.492889  -5.789654  -6.282615
## [139]  -5.790524  -7.447009  -9.736032 -10.781326  -8.463121  -8.038825
## [145]  -8.080430  -8.220134 -12.115628  -8.521985  -9.836525 -11.335662
## [151]  -8.152501  -9.232015  -7.399179 -10.886390  -9.983441  -9.407088
## [157]  -6.270485  -5.794193  -9.434924 -11.725079 -12.243685 -12.440399
## [163] -12.886440 -13.392187 -13.280457 -13.188842 -13.123382 -12.639208
## [169] -11.841136 -11.284614 -10.363523  -6.824513  -7.039945 -13.711477
## [175]       -Inf -22.188582       -Inf -20.891491  -8.869295  -7.878109
## [181] -10.975056 -10.545610 -10.338271  -8.539032  -8.995616  -8.059989
## [187]  -6.629300  -6.843510 -15.950768  -7.044640 -12.808038 -11.968285
## [193] -11.050674 -10.817601  -8.163371  -6.064380  -6.972258  -8.898154
## [199]  -5.499066  -5.976116  -5.360646 -11.075399  -9.747479  -6.708617
## [205]  -9.383891  -8.339791  -6.641379 -10.433120  -7.118554  -8.108637
## [211]  -9.299171  -7.940837  -7.978988  -7.554455  -7.253804 -10.271083
## [217]  -8.026840 -11.581623 -10.593898 -13.460809 -12.627035  -8.167955
## [223]  -7.817385  -7.224118 -10.904526  -6.456491  -6.021622  -9.039236
## [229]  -6.406198  -6.155642  -6.378456  -6.786419  -7.407197  -9.901753
## [235] -11.487860  -9.563749  -9.098364 -10.253943  -9.029760  -6.753109
## [241]  -6.845429  -6.115681 -12.435378  -9.485075 -10.903124  -7.299587
## [247]  -6.798547  -7.567400 -10.789711 -11.799660  -9.142057 -29.472400
## [253]  -9.870247  -7.645866  -8.693364  -7.699742 -11.037124 -10.272696
## [259] -12.924845  -7.618623  -7.653184  -9.936229  -8.402320  -7.366042
## [265]  -7.650931  -6.375779  -5.288048  -5.644899  -4.210291  -3.857056
## [271]  -3.747775  -6.516558  -4.091262  -5.645950  -8.079773  -6.597516
## [277]  -4.660914  -6.675632  -6.786107  -8.357611  -9.230719 -10.689745
## [283] -10.247963 -10.523685 -11.488987 -12.517847 -12.430746 -12.538126
## [289] -12.519664 -12.621444  -7.507670 -12.996070 -12.739996 -12.443434
## [295] -12.129728 -11.524708 -10.859228 -10.258693  -7.682879  -5.503378
## [301]  -5.006935  -5.025682 -15.372720  -7.164158 -15.071909  -8.444321
## [307]  -9.275007 -10.198774  -8.087587  -8.001889 -10.108868  -8.496824
## [313] -11.763425  -8.448551  -8.841450 -11.242043 -10.573214 -16.537121
## [319] -31.266148 -10.585179  -6.866646 -22.319301  -8.057369 -20.479017
## [325]  -9.459029  -8.716533  -6.241053 -14.248407  -5.536567  -5.189233
## [331]  -7.902018  -7.334638  -9.742111  -7.946377  -8.983412  -6.522082
## [337]  -6.713917  -6.615762  -8.393757 -10.866425 -11.239891 -12.525449
## [343] -13.164110 -12.922968 -12.349203 -11.470876 -10.860984  -9.445333
## [349]  -6.836141 -11.682941 -32.153281       -Inf -19.510412  -9.791880
## [355] -19.190413 -10.364079  -8.854363  -6.116026  -5.926019  -8.957666
## [361]  -8.441679  -8.748358  -9.418163  -9.394571  -9.780470  -8.888477
## [367]  -7.884632  -6.326217  -5.896049  -6.809144  -9.148813  -9.173524
## [373] -10.068208  -9.475608  -9.655064  -8.927148 -10.779910  -9.091148
## [379]  -7.887435  -8.165862  -9.567952 -11.970060 -12.632294 -13.394350
## [385] -13.451130 -14.078334 -13.143768 -10.791114  -9.680781  -9.337206
## [391]  -8.315138  -7.183322  -5.294721 -11.722474       -Inf       -Inf
## [397]       -Inf       -Inf       -Inf       -Inf       -Inf -43.239091
## [403]  -7.334466  -7.735198  -7.627852  -7.244880  -6.554480  -6.326436
## [409]  -6.112101  -6.741891  -5.936376  -8.057731 -10.332788 -11.106709
## [415] -12.076986 -12.410494
```