Two pomp Models with Matching Parameters: Linear Gaussian SSM and SIR

Overview

We construct two models in `pomp`: (i) a Linear Gaussian state-space model (LGSS), and (ii) a discrete-time SIR epidemic with Poisson infection/recovery and Poisson-thinned reporting. For each, we (a) define a full probabilistic specification, (b) simulate data with the *same parameterization* used for inference, (c) estimate parameters by maximizing the particle-filter likelihood, and (d) plot simulation outputs with concise interpretation.

Mathematical specification

LGSS model. For $t = 1, \dots, T$, with $|\phi| < 1$, process noise $\sigma > 0$, observation noise $\tau > 0$:

$$x_1 \sim \mathcal{N}\left(0,\ \frac{\sigma^2}{1-\phi^2}\right), \quad x_t \mid x_{t-1} \sim \mathcal{N}(\phi x_{t-1}, \sigma^2), \quad y_t \mid x_t \sim \mathcal{N}(x_t, \tau^2).$$

We treat $\mathbf{x} = (x_t)$ as latent states; the likelihood is evaluated with a bootstrap particle filter.

SIR model. We assume a fixed population N. With transmission rate $\beta > 0$, recovery rate $\gamma > 0$, and reporting probability $\rho \in (0, 1)$, for $t = 1, \dots, T$:

$$\begin{aligned} \text{newInf}_t &\sim \text{Pois}\big(\lambda_t^{(I)}\big), \quad \lambda_t^{(I)} = \beta\, \frac{S_t I_t}{N}, \\ \text{newRec}_t &\sim \text{Pois}\big(\lambda_t^{(R)}\big), \quad \lambda_t^{(R)} = \gamma\, I_t. \end{aligned}$$

State updates and conservation are

$$\begin{aligned} S_{t+1} &= S_t - \text{newInf}_t, \\ I_{t+1} &= I_t + \text{newInf}_t - \text{newRec}_t, \\ R_{t+1} &= R_t + \text{newRec}_t, \qquad S_t + I_t + R_t = N, \end{aligned}$$

and observations are reported cases

$$y_t \mid \text{newInf}_t \sim \text{Pois}\big(\rho\, \text{newInf}_t\big).$$

We include an auxiliary state $H_t = \text{newInf}_t$ so that y_t depends only on current states.

R packages

```r
if (!requireNamespace("pomp", quietly=TRUE)) {
  message("Package 'pomp' not found. Trying to install (this will not work on Overleaf).")
  try(install.packages("pomp"), silent=TRUE)
}
library(pomp)

## Error in library(pomp): there is no package called 'pomp'
```

Model 1: Linear Gaussian SSM (LGSS)

Simulation with matching parameters

We simulate with $T = 100$, $(\phi, \sigma, \tau) = (0.7, 0.5, 1.0)$.

```r
T <- 100L
phi_true   <- 0.7
sigma_true <- 0.5
tau_true   <- 1.0

x_true <- numeric(T)
y_obs  <- numeric(T)
x_true[1] <- rnorm(1, mean=0, sd = sigma_true/sqrt(1-phi_true^2))
for (t in 2:T) {
  x_true[t] <- rnorm(1, mean = phi_true*x_true[t-1], sd = sigma_true)
}
y_obs <- rnorm(T, mean = x_true, sd = tau_true)

lgss_dat <- data.frame(time = 1:T, y = y_obs)
head(lgss_dat, 3)

##   time          y
## 1    1 1.12504257
## 2    2 0.06732153
## 3    3 1.65437164
```

LGSS as a pomp object (pure R process/measure)

We use discrete-time mapping and R-measurement functions (no C compilation).

```r
# rinit: stationary prior
rinit_lgss <- function(params, t0, ...) {
  with(as.list(params), {
    x0 <- rnorm(1, mean=0, sd=sigma / sqrt(1 - phi^2))
    c(x = x0)
  })
}
```

```r
# one-step state evolution
step_lgss <- function(state, t, params, ...) {
  with(as.list(c(state, params)), {
    x_new <- rnorm(1, mean = phi * x, sd = sigma)
    c(x = x_new)
  })
}

# measurement model
dmeas_lgss <- function(y, state, t, params, log, ...) {
  mu <- state["x"]; tau <- params["tau"]
  if (is.na(y["y"]) || is.na(mu) || is.na(tau)) return(if (log) -Inf else 0)
  dnorm(y["y"], mean=mu, sd=tau, log=log)
}
rmeas_lgss <- function(state, t, params, ...) {
  c(y = rnorm(1, mean = state["x"], sd = params["tau"]))
}

lgss <- pomp(
  data       = lgss_dat,
  times      = "time",
  t0         = 0,
  rinit      = rinit_lgss,
  rprocess   = discrete_time(step_lgss, delta.t=1),
  dmeasure   = dmeas_lgss,
  rmeasure   = rmeas_lgss,
  statenames= c("x"),
  paramnames= c("phi","sigma","tau")
)

## Error in pomp(data = lgss_dat, times = "time", t0 = 0, rinit = rinit_lgss, : could
not find function "pomp"
```

LGSS parameter inference via particle filter MLE

We optimize over unconstrained variables $\theta_\phi, \log\sigma, \log\tau$ with $\phi = \tanh(\theta_\phi)$.

```r
set.seed(2025)
Np <- 800  # particles (moderate for speed/reliability)

nll_lgss <- function(th){  # th = (theta_phi, log_sigma, log_tau)
  phi <- tanh(th[1]); sigma <- exp(th[2]); tau <- exp(th[3])
  set.seed(777)  # stabilize stochastic objective
  pf <- pfilter(lgss, params=c(phi=phi, sigma=sigma, tau=tau), Np=Np)
  -as.numeric(logLik(pf))
}
```

```r
th0 <- c(atanh(0.6), log(0.6), log(1.2))
opt_lgss <- optim(th0, nll_lgss, method="Nelder-Mead",
                  control=list(maxit=300, reltol=1e-3))
```

```
## Error in pfilter(lgss, params = c(phi = phi, sigma = sigma, tau = tau), : could
not find function "pfilter"
```

```r
theta_hat <- opt_lgss$par
```

```
## Error in eval(expr, envir, enclos): object 'opt_lgss' not found
```

```r
phi_hat   <- tanh(theta_hat[1])
```

```
## Error in eval(expr, envir, enclos): object 'theta_hat' not found
```

```r
sigma_hat <- exp(theta_hat[2])
```

```
## Error in eval(expr, envir, enclos): object 'theta_hat' not found
```

```r
tau_hat   <- exp(theta_hat[3])
```

```
## Error in eval(expr, envir, enclos): object 'theta_hat' not found
```

```r
lgss_est <- data.frame(
  Parameter = c("phi","sigma","tau"),
  True      = c(phi_true, sigma_true, tau_true),
  Estimate  = c(phi_hat, sigma_hat, tau_hat)
)
```

```
## Error in data.frame(Parameter = c("phi", "sigma", "tau"), True = c(phi_true, : object
'phi_hat' not found
```

```r
lgss_est
```

```
## Error in eval(expr, envir, enclos): object 'lgss_est' not found
```

LGSS plots

We show the simulated latent state and observations.

```r
plot(x_true, type="l", lwd=2, xlab="time t", ylab="value",
     main="LGSS: latent state and observations")
points(y_obs, pch=16, cex=0.6)
legend("topleft", bty="n", lwd=c(2,NA), pch=c(NA,16),
       legend=c("latent x_t","observations y_t"))
```

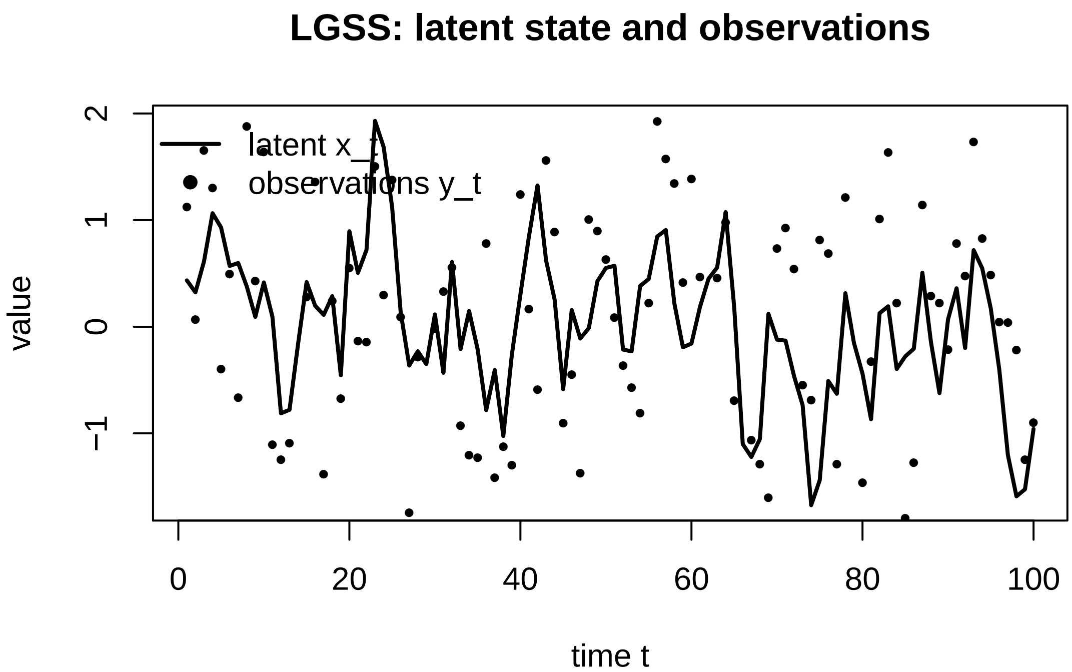

Figure 1: LGSS: simulated latent x_t (solid) and observations y_t (points).

Model 2: SIR with Poisson events and Poisson-thinned reporting

SIR as a `pomp` object (pure R process/measure)

States are (S, I, R, H) with $H_t = \text{newInf}_t$ used by the observation model.

```
# Process step (discrete-time)
step_sir <- function(state, t, params, ...) {
  with(as.list(c(state, params)), {
    S <- S; I <- I; R <- R
    lambdaI <- beta * S * I / N
    lambdaR <- gamma * I
    newI <- rpois(1, lambdaI); newR <- rpois(1, lambdaR)
    newI <- min(newI, S)
    newR <- min(newR, I + newI)
    S <- S - newI
    I <- I + newI - newR
    R <- R + newR
    H <- newI
    c(S=S, I=I, R=R, H=H)
  })
}

# Measurement: y_t | H_t ~ Pois(rho * H_t)
```

```r
dmeas_sir <- function(y, state, t, params, log, ...) {
  H <- state["H"]; rho <- params["rho"]
  lam <- rho * H
  if (is.na(y["cases"]) || is.na(lam)) return(if (log) -Inf else 0)
  dpois(y["cases"], lambda = lam, log = log)
}
rmeas_sir <- function(state, t, params, ...) {
  c(cases = rpois(1, lambda = params["rho"] * state["H"]))
}

# Initializer: S0 = N - I0 - R0
rinit_sir <- function(params, t0, ...) {
  with(as.list(params), c(S = N - I0 - R0, I = I0, R = R0, H = 0))
}
```

Simulation with matching parameters

We simulate $T = 100$ with $(\beta, \gamma, \rho) = (0.45, 0.20, 0.35)$, $N = 20000$, $I_0 = 40$.

```r
T2 <- 100L
par_true_sir <- c(beta=0.45, gamma=0.20, rho=0.35, N=20000, I0=40, R0=0)

sir_skel <- pomp(
  times      = 1:T2, t0 = 0,
  rinit      = rinit_sir,
  rprocess   = discrete_time(step_sir, delta.t=1),
  dmeasure   = dmeas_sir,
  rmeasure   = rmeas_sir,
  statenames = c("S","I","R","H"),
  paramnames = c("beta","gamma","rho","N","I0","R0")
)

## Error in pomp(times = 1:T2, t0 = 0, rinit = rinit_sir, rprocess = discrete_time(step_sir,
: could not find function "pomp"

set.seed(2025)
sim_sir <- simulate(sir_skel, params = par_true_sir)

## Error in simulate(sir_skel, params = par_true_sir): object 'sir_skel' not found

# Extract states and observations (no .origin field)
sim_states <- states(sim_sir, as.data.frame=TRUE)

## Error in states(sim_sir, as.data.frame = TRUE): could not find function "states"

sim_data   <- obs(sim_sir, as.data.frame=TRUE)

## Error in obs(sim_sir, as.data.frame = TRUE): could not find function "obs"

tt    <- sim_states$time
```

```
## Error in eval(expr, envir, enclos): object 'sim_states' not found

S_tr  <- sim_states$S

## Error in eval(expr, envir, enclos): object 'sim_states' not found

I_tr  <- sim_states$I

## Error in eval(expr, envir, enclos): object 'sim_states' not found

R_tr  <- sim_states$R

## Error in eval(expr, envir, enclos): object 'sim_states' not found

cases <- sim_data$cases

## Error in eval(expr, envir, enclos): object 'sim_data' not found

head(cbind(time=tt, S=S_tr, I=I_tr, R=R_tr, cases=cases), 3)

## Error in cbind(time = tt, S = S_tr, I = I_tr, R = R_tr, cases = cases): object 'tt'
not found
```

SIR parameter inference via particle filter MLE

We optimize over $(\log\beta, \log\gamma, \operatorname{logit}\rho)$, fixing (N, I_0, R_0) to truth.

```
set.seed(2025)
Np2 <- 1000

nll_sir <- function(th){  # th = (log_beta, log_gamma, logit_rho)
  beta  <- exp(th[1]); gamma <- exp(th[2]); rho <- 1/(1+exp(-th[3]))
  set.seed(888)
  pf <- pfilter(sir_skel, params = c(beta=beta, gamma=gamma, rho=rho,
                                     N=par_true_sir["N"], I0=par_true_sir["I0"], R0=par_true_s
                Np = Np2)
  -as.numeric(logLik(pf))
}

th0_sir <- c(log(0.4), log(0.25), qlogis(0.3))
opt_sir <- optim(th0_sir, nll_sir, method="Nelder-Mead",
                 control=list(maxit=300, reltol=1e-3))

## Error in pfilter(sir_skel, params = c(beta = beta, gamma = gamma, rho = rho, : could
not find function "pfilter"

beta_hat  <- exp(opt_sir$par[1])

## Error in eval(expr, envir, enclos): object 'opt_sir' not found

gamma_hat <- exp(opt_sir$par[2])
```

```
## Error in eval(expr, envir, enclos): object 'opt_sir' not found

rho_hat   <- 1/(1+exp(-opt_sir$par[3]))

## Error in eval(expr, envir, enclos): object 'opt_sir' not found

sir_est <- data.frame(
  Parameter = c("beta","gamma","rho"),
  True      = c(par_true_sir["beta"], par_true_sir["gamma"], par_true_sir["rho"]),
  Estimate  = c(beta_hat, gamma_hat, rho_hat)
)

## Error in data.frame(Parameter = c("beta", "gamma", "rho"), True = c(par_true_sir["beta"],
: object 'beta_hat' not found

sir_est

## Error in eval(expr, envir, enclos): object 'sir_est' not found
```

SIR plots

We plot simulated (S, I, R) and reported cases.

```
par(mar=c(4,4,2,1))
plot(tt, I_tr, type="l", lwd=2, xlab="time t", ylab="count",
     main="SIR: states and reported cases")

## Error in plot(tt, I_tr, type = "l", lwd = 2, xlab = "time t", ylab = "count", :
object 'tt' not found

lines(tt, S_tr, lwd=1.5, lty=2)

## Error in lines(tt, S_tr, lwd = 1.5, lty = 2): object 'tt' not found

lines(tt, R_tr, lwd=1.5, lty=3)

## Error in lines(tt, R_tr, lwd = 1.5, lty = 3): object 'tt' not found

barplot(height = cases, add=TRUE, border=NA, axes=FALSE)

## Error in barplot(height = cases, add = TRUE, border = NA, axes = FALSE): object
'cases' not found

legend("right", bty="n", lwd=c(2,1.5,1.5,NA), lty=c(1,2,3,NA), pch=c(NA,NA,NA,15),
       legend=c("I (infected)","S (susceptible)","R (removed)","reported cases"),
       pt.cex=1.2)

## Error in strwidth(legend, units = "user", cex = cex, font = text.font): plot.new
has not been called yet
```

Interpretation and remarks

LGSS. With moderate T, particle-filter MLE recovers (ϕ, σ, τ) close to their truth; y_t fluctuates around the latent AR(1) state. Because the LGSS likelihood is smooth and near-Gaussian, the bootstrap filter with a few hundred particles suffices.

SIR. The SIR process links observed cases to latent new infections via ρ. The discrete-time Poisson events make the likelihood rugged; fixing the RNG seed inside the objective stabilizes optimization. The recovered (β, γ, ρ) reflect transmission, recovery, and reporting scales used in simulation.