13 Time series models with covariates, and a case study of polio

Edward Ionides

04/08/2018

Contents

Produced with R version 3.4.3 and **pomp** version 1.16.

Objectives

1. Discuss covariates in POMP models as a generalization of regression with ARMA errors.
2. Demonstrate the use of covariates in **pomp** to add demographic data (birth rates and total population) and seasonality to an epidemiological model.
3. Present a case study, developing and fitting a POMP model with covariates.

Covariates in time series analysis

- Suppose our time series of primary interest is $y^*_{1:N}$.
- A **covariate** time series is an additional time series $z_{1:N}$ which is used to help explain $y^*_{1:N}$.
- When we talk about covariates, it is often implicit that we think of $z_{1:N}$ as a measure of an **external forcing** to the system producing $y^*_{1:N}$. This means that the process generating the data $z_{1:N}$ affects the process generating $y^*_{1:N}$, but not vice versa.
  - For example, the weather might affect human health, but human health has negligible effect on weather: weather is an external forcing to human health processes.
- When we make an assumption of external forcing, we should try to make it explicit.
- In regression analysis, we usually **condition** on covariates. Equivalently, we model them as fixed numbers, rather than modeling them as the outcome of random variables.

- When the process leading to $z_{1:N}$ is not external to the system generating it, we must be alert to the possibility of **reverse causation** and **confounding variables**. Further discussion of this issue, in the context of a time series analysis, is given in the case study on An association between unemployment and mortality?.
- Isses involved in inferring causation from fitting statistical models are essentially the same whether the model is linear and Gaussian or not.

Covariates in linear time series analysis

- The main tool we have seen previously for investigating dependence on covariates is regression with ARMA errors.
- This tool can also be used to identify lag relationships, where y_n^* depends on z_{n-L}.
  - An example of identifying lag in the midterm project: The relationship between S&P500 and CPI.
- Another way to investigate associations at different lags is by computing the sample correlation between y_n^* and z_{n-L}, for $n \in L+1:N$, and plotting this against L.
- This is called the **cross-correlation function** and can be computed with the R function `ccf`.
  - An example of the use of the cross-correlation function in a midterm project: The Association between Recent Cholera Epidemics and Rainfall in Haiti.

Covariates in nonlinear POMP models

- Time series modeling of regression errors is only one of many ways in which covariates could be used to explain a dynamic system.
- In general, it is nice if scientific considerations allow you to propose sensible ways to model the relationship.
  - In an epidemiological model for malaria, rainfall might affect the number of mosquitoes (and hence the disease transmission rate) but not the duration of infection.
  - In an economic model, geopolitical shocks to the oil supply might have direct influence on energy prices, secondary direct effects on inflation and investment, and indirect consequences for unemployment.
  - In a hydrology model, precipitation is a covariate explaining river flow, but the exact nature of the relationship is a question of interest.
- The general POMP modeling framework allows essentially arbitrary modeling of covariates.
- Recall that a POMP model is specified by defining the following:

$$
\begin{aligned}
&f_{X_0}(x_0\,;\,\theta),\\
&f_{X_n|X_{n-1}}(x_n\,|\,x_{n-1}\,;\,\theta),\\
&f_{Y_n|X_n}(y_n\,|\,x_n\,;\,\theta),
\end{aligned}
$$

  for $n = 1:N$

- The possibility of a general dependence on n includes the possibility that there is some covariate time series $z_{0:N}$ such that

$$\begin{array}{lcl} f_{X_0}(x_0\,;\,\theta) & = & f_{X_0}(x_0\,;\,\theta, z_0) \\ f_{X_n|X_{n-1}}(x_n\,|\,x_{n-1}\,;\,\theta) & = & f_{X_n|X_{n-1}}(x_n\,|\,x_{n-1}\,;\,\theta, z_n), \\ f_{Y_n|X_n}(y_n\,|\,x_n\,;\,\theta) & = & f_{Y_n|X_n}(y_n\,|\,x_n\,;\,\theta, z_n), \end{array}$$

  for $n = 1 : N$

- One specific choice of covariates is to construct $z_{0:N}$ so that it fluctuates periodically, once per year. This allows **seasonality** enter the POMP model in whatever way is appropriate for the system under investigation.
- All that remains is to hypothesize what is a reasonable way to include covariates for your system, and to implement the resulting data analysis.
- The **pomp** package provides facilities for including covariates in a pomp object, and making sure that the covariates are accessible to `rprocess`, `dprocess`, `rmeasure`, `dmeasure`, and the state initialization at time t_0.
- Named covariate time series entered via the `covar` argument to `pomp` are automatically defined within Csnippets used for the `rprocess`, `dprocess`, `rmeasure`, `dmeasure` and `initializer` arguments.
- Let's see this in practice, using population census, birth data and seasonality as covariates in an epidemiological model.

Case study: polio in Wisconsin

- The massive global polio eradication initiative (GPEI) has brought polio from a major global disease to the brink of extinction.
- Finishing this task is proving hard, and improved understanding polio ecology might assist.
- Martinez-Bakker et al. (2015) investigated this using extensive state level pre-vaccination era data in USA.
- We will follow the approach of Martinez-Bakker et al. (2015) for one state (Wisconsin). In the context of their model, we can quantify seasonality of transmission, the role of the birth rate in explaining the transmission dynamics, and the persistence mechanism of polio.
- Martinez-Bakker et al. (2015) carrried out this analysis for all 48 contigous states and District of Columbia, and their data and code are publicly available. The data we study, in polio_wisconsin.csv, consist of `cases`, the monthly reported polio cases; `births`, the monthly recorded births; `pop`, the annual census; `time`, date in years.

```
polio_data <- read.table("polio_wisconsin.csv")
colnames(polio_data)
```

```
## [1] "time"   "cases"  "births" "pop"
```

- We implement the compartment model of Martinez-Bakker et al. (2015), having compartments representing susceptible babies in each of six one-month birth cohorts (S^B_1,...,S^B_6), susceptible older individuals (S^O), infected babies (I^B), infected older individuals (I^O), and recovered with lifelong immunity (R).

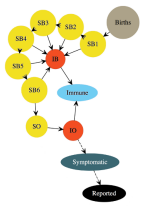


Figure 1: Polio model diagram

- The state vector of the disease transmission model consists of numbers of individuals in each compartment at each time,
$$X(t) = \big(S_1^B(t), ..., S_6^B(t), I^B(t), I^O(t), R(t)\big).$$
- Babies under six months are modeled as fully protected from symptomatic poliomyelitis; older infections lead to reported cases (usually paralysis) at a rate ρ.
- The flows through the compartments are graphically represented as follows (Figure 1A of Martinez-Bakker et al. (2015)):
- Since duration of infection is comparable to the one-month reporting aggregation, a discrete time model may be appropriate. Martinez-Bakker et al. (2015) fitted monthly observations from May 1932 through January 1953, so we define $t_n = 1932 + (4 + n)/12$ for $n = 0, \ldots, N$, and we write
$$X_n = X(t_n) = \big(S_{1,n}^B, ..., S_{6,n}^B, I_n^B, I_n^O, R_n\big).$$
- The mean force of infection, in units of yr^{-1}, is modeled as
$$\bar{\lambda}_n = \left(\beta_n \frac{I_n^O + I_n^B}{P_n} + \psi\right)$$
where P_n is census population interpolated to time t_n and seasonality of transmission is modeled as
$$\beta_n = \exp\left\{\sum_{k=1}^{K} b_k \xi_k(t_n)\right\},$$
with $\{\xi_k(t), k = 1, \ldots, K\}$ being a periodic B-spline basis. We set $K = 6$. The force of infection has a stochastic perturbation,
$$\lambda_n = \bar{\lambda}_n \epsilon_n,$$
where ϵ_n is a Gamma random variable with mean 1 and variance $\sigma_{\text{env}}^2 + \sigma_{\text{dem}}^2/\bar{\lambda}_n$. These two terms capture variation on the environmental and demographic scales, respectively. All compartments suffer a mortality rate, set at $\delta = 1/60\text{yr}^{-1}$.
- Within each month, all susceptible individuals are modeled as having exposure to constant competing hazards of mortality and polio infection. The chance of remaining in the susceptible population when exposed to these hazards for one month is therefore
$$p_n = \exp\big\{-(\delta + \lambda_n)/12\big\},$$
with the chance of polio infection being
$$q_n = (1 - p_n)\lambda_n/(\lambda_n + \delta).$$
- We employ a continuous population model, with no demographic stochasticity (in some sense, the demographic-scale stochasticity in λ_n is in fact environmental stochasticity since it modifies a rate that

affects all compartments equally). Writing B_n for births in month n, we obtain the dynamic model of Martinez-Bakker et al. (2015):

$$\begin{aligned} S^B_{1,n+1} &= B_{n+1} \\ S^B_{k,n+1} &= p_n S^B_{k-1,n} \quad \text{for } k = 2,\dots,6 \\ S^O_{n+1} &= p_n (S^O_n + S^B_{6,n}) \\ I^B_{n+1} &= q_n \textstyle\sum_{k=1}^{6} S^B_{k,n} \\ I^O_{n+1} &= q_n S^O_n \end{aligned}$$

The model for the reported observations, conditional on the state, is a discretized normal distribution truncated at zero, with both environmental and Poisson-scale contributions to the variance:

$$Y_n = \max\{\text{round}(Z_n), 0\}, \quad Z_n \sim \text{normal}\left(\rho I^O_n, (\tau I^O_n)^2 + \rho I^O_n\right).$$

Additional parameters are used to specify initial state values at time $t_0 = 1932 + 4/12$. We will suppose there are parameters $(\tilde{S}^B_{1,0}, ..., \tilde{S}^B_{6,0}, \tilde{I}^B_0, \tilde{I}^O_0, \tilde{S}^O_0)$ that specify the population in each compartment at time t_0 via

$$S^B_{1,0} = \tilde{S}^B_{1,0}, ..., S^B_{6,0} = \tilde{S}^B_{6,0}, \quad I^B_0 = P_0 \tilde{I}^B_0, \quad S^O_0 = P_0 \tilde{S}^O_0, \quad I^O_0 = P_0 \tilde{I}^O_0.$$

Following Martinez-Bakker et al. (2015), we make an approximation for the initial conditions of ignoring infant infections at time t_0. Thus, we set $\tilde{I}^B_0 = 0$ and use monthly births in the preceding months (ignoring infant mortality) to fix $\tilde{S}^B_{k,0} = B_{1-k}$ for $k = 1,\dots,6$. The estimated initial conditions are then defined by the two parameters $\tilde{I}^O_0$ and $\tilde{S}^O_0$, since the initial recovered population, R_0, is specified by subtraction of all the other compartments from the total initial population, P_0. Note that it is convenient to parameterize the estimated initial states as fractions of the population, whereas the initial states fixed at births are parameterized directly as a count.

Building a pomp object for the polio model

Observations are monthly case reports, $y^*_{1:N}$, occurring at times $t_{1:N}$. Since our model is in discrete time, we only really need to consider the discrete time state process,. However, the model and POMP methods extend naturally to the possibility of a continuous-time model specification. We code the state and observation variables, and the choice of t_0, as

```
polio_statenames <- c("SB1","SB2","SB3","SB4","SB5","SB6","IB","SO","IO")
polio_obsnames <- "cases"
polio_t0 <- 1932+4/12
```

We do not explictly code R, since it is defined implicitly as the total population minus the sum of the other compartments. Due to lifelong immunity, individuals in R play no role in the dynamics. Even occasional negative values of R (due to a discrepancy between the census and the mortality model) would not be a fatal flaw.

Now, let's define the covariates. `time` gives the time at which the covariates are defined. `P` is a smoothed interpolation of the annual census. `B` is monthly births. The B-spline basis is coded as `xi1,...,xi6`

```
polio_K <- 6
polio_tcovar <- polio_data$time
polio_bspline_basis <- periodic.bspline.basis(polio_tcovar,nbasis=polio_K,degree=3,period=1)
colnames(polio_bspline_basis)<- paste("xi",1:polio_K,sep="")
covartable <- data.frame(
  time=polio_tcovar,
```

```
  polio_bspline_basis,
  B=polio_data$births,
  P=predict(smooth.spline(x=1931:1954,y=polio_data$pop[12*(1:24)]),
            x=polio_tcovar)$y
)
```

The parameters $b_1, \ldots, b_{\mathrm{K}}, \psi, \rho, \tau, \sigma_{\mathrm{dem}}, \sigma_{\mathrm{env}}$ in the model above are *regular parameters* (RPs), meaning that they are real-valued parameters that affect the dynamics and/or the measurement of the process. These regular parameters are coded as

```
polio_rp_names <- c("b1","b2","b3","b4","b5","b6","psi","rho","tau","sigma_dem","sigma_env")
```

The *initial value parameters* (IVPs), $\tilde{I}_0^O$ and $\tilde{S}_0^O$, are coded for each state named by adding `_0` to the state name:

```
polio_ivp_names <- c("SO_0","IO_0")
polio_paramnames <- c(polio_rp_names,polio_ivp_names)
```

Finally, there are two quantities in the dynamic model specification, $\delta = 1/60\mathrm{yr}^{-1}$ and $\mathrm{K} = 6$, that we are not estimating. In addition, there are six other initial value quantities, $\{\tilde{S}^B_{1,0}, \ldots, \tilde{S}^B_{6,0}\}$, which we are treating as *fixed parameters* (FPs).

```
polio_fp_names <- c("delta","K","SB1_0","SB2_0","SB3_0","SB4_0","SB5_0","SB6_0")
polio_paramnames <- c(polio_rp_names,polio_ivp_names,polio_fp_names)
```

Alternatively, these fixed quantities could be passed as constants using the `globals` argument of `pomp`. We can check how the initial birth parameters are set up:

```
covar_index_t0 <- which(abs(covartable$time-polio_t0)<0.01)
polio_initial_births <- as.numeric(covartable$B[covar_index_t0-0:5])
names(polio_initial_births) <- c("SB1_0","SB2_0","SB3_0","SB4_0","SB5_0","SB6_0")
polio_fixed_params <- c(delta=1/60,K=polio_K,polio_initial_births)
```

We read in a table of previous parameter search results from `polio_params.csv`, and take the one with highest likelihood as our current estimate of an MLE. We can inspect that the fixed parameters are indeed set to their proper values.

```
polio_params <- data.matrix(read.table("polio_params.csv",row.names=NULL,header=TRUE))
polio_mle <- polio_params[which.max(polio_params[,"logLik"]),][polio_paramnames]
polio_mle[polio_fp_names]
```

```
##        delta            K        SB1_0        SB2_0        SB3_0
## 1.666667e-02 6.000000e+00 4.069000e+03 4.565000e+03 4.410000e+03
##        SB4_0        SB5_0        SB6_0
## 4.616000e+03 4.305000e+03 4.032000e+03
```

```
polio_fixed_params
```

```
##        delta            K        SB1_0        SB2_0        SB3_0
## 1.666667e-02 6.000000e+00 4.069000e+03 4.565000e+03 4.410000e+03
##        SB4_0        SB5_0        SB6_0
## 4.616000e+03 4.305000e+03 4.032000e+03
```

The process model is

```
polio_rprocess <- Csnippet("
  double lambda, beta, var_epsilon, p, q;

  beta = exp(dot_product( (int) K, &xi1, &b1));
```

```
  lambda = (beta * (IO+IB) / P + psi);
  var_epsilon = pow(sigma_dem,2)/ lambda +  pow(sigma_env,2);
  lambda *= (var_epsilon < 1.0e-6) ? 1 : rgamma(1/var_epsilon,var_epsilon);
  p = exp(- (delta+lambda)/12);
  q = (1-p)*lambda/(delta+lambda);
  SB1 = B;
  SB2= SB1*p;
  SB3=SB2*p;
  SB4=SB3*p;
  SB5=SB4*p;
  SB6=SB5*p;
  SO= (SB6+SO)*p;
  IB=(SB1+SB2+SB3+SB4+SB5+SB6)*q;
  IO=SO*q;
")
```

The measurement model is

```
polio_dmeasure <- Csnippet("
  double tol = 1.0e-25;
  double mean_cases = rho*IO;
  double sd_cases = sqrt(pow(tau*IO,2) + mean_cases);
  if(cases > 0.0){
    lik = pnorm(cases+0.5,mean_cases,sd_cases,1,0) - pnorm(cases-0.5,mean_cases,sd_cases,1,0) + tol;
  } else{
    lik = pnorm(cases+0.5,mean_cases,sd_cases,1,0) + tol;
  }
  if (give_log) lik = log(lik);
")

polio_rmeasure <- Csnippet("
  cases = rnorm(rho*IO, sqrt( pow(tau*IO,2) + rho*IO ) );
  if (cases > 0.0) {
    cases = nearbyint(cases);
  } else {
    cases = 0.0;
  }
")
```

The map from the initial value parameters to the initial value of the states at time t_0 is coded by the initializer function:

```
polio_initializer <- Csnippet("
  SB1 = SB1_0;
  SB2 = SB2_0;
  SB3 = SB3_0;
  SB4 = SB4_0;
  SB5 = SB5_0;
  SB6 = SB6_0;
  IB = 0;
  IO = IO_0 * P;
  SO = SO_0 * P;
")
```

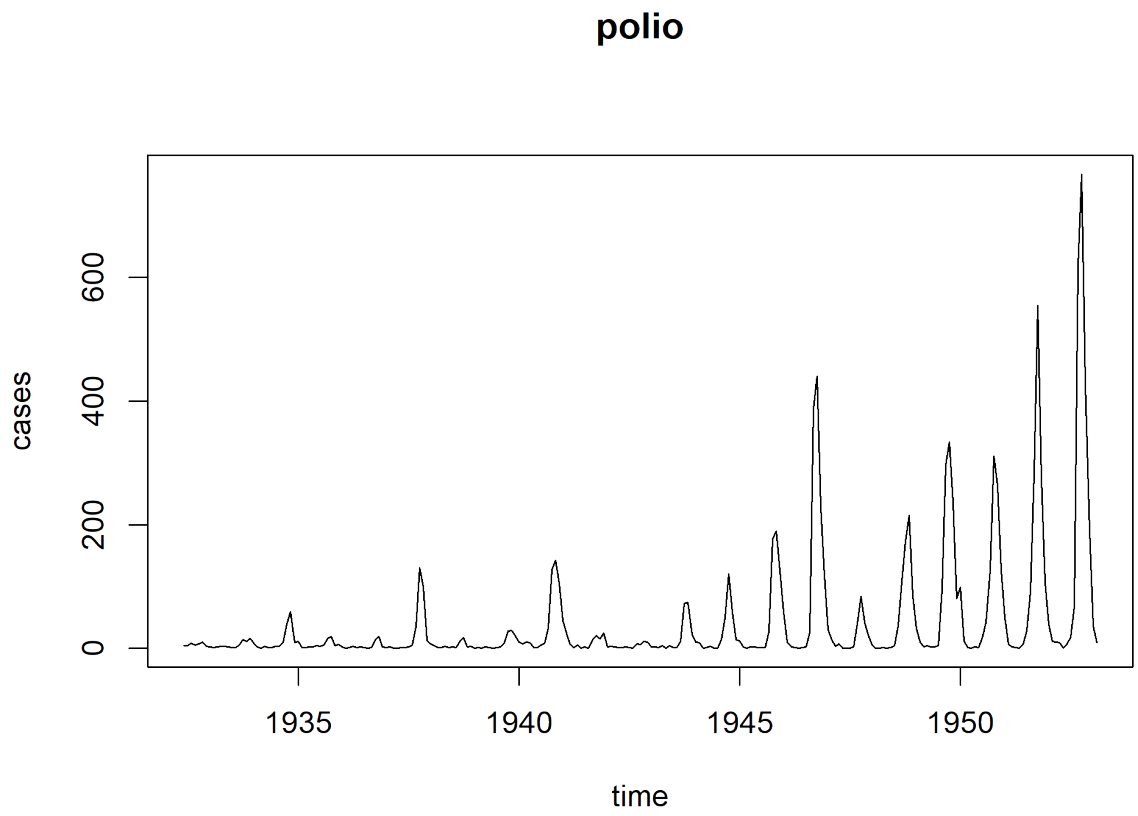
To carry out parameter estimation, it is also helpful to have transformations that map each parameter into

the whole real line:

```
polio_toEstimationScale <- Csnippet("
 Tpsi = log(psi);
 Trho = logit(rho);
 Ttau = log(tau);
 Tsigma_dem = log(sigma_dem);
 Tsigma_env = log(sigma_env);
 TSO_0 =  logit(SO_0);
 TIO_0 = logit(IO_0);
")

polio_fromEstimationScale <- Csnippet("
 Tpsi = exp(psi);
 Trho = expit(rho);
 Ttau = exp(tau);
 Tsigma_dem = exp(sigma_dem);
 Tsigma_env = exp(sigma_env);
 TSO_0 =  expit(SO_0);
 TIO_0 = expit(IO_0);
")
```

We can now put these pieces together into a pomp object.

```
polio <- pomp(
  data=subset(polio_data,
              (time > polio_t0 + 0.01) & (time < 1953+1/12+0.01),
              select=c("cases","time")),
  times="time",
  t0=polio_t0,
  params=polio_mle,
  rprocess = euler.sim(step.fun = polio_rprocess, delta.t=1/12),
  rmeasure= polio_rmeasure,
  dmeasure = polio_dmeasure,
  covar=covartable,
  tcovar="time",
  obsnames = polio_obsnames,
  statenames = polio_statenames,
  paramnames = polio_paramnames,
  covarnames = c("xi1","B","P"),
  initializer=polio_initializer,
  toEstimationScale=polio_toEstimationScale,
  fromEstimationScale=polio_fromEstimationScale
)
plot(polio)
```



Setting run levels to control computation time

- To develop and debug code, it is nice to have a version that runs extra quickly.
- To facilitate switching quickly between versions of the document that have different run times, we set up `run_level` options.
- `run_level=1` will set all the algorithmic parameters to the first column of values in the following code.
- Here, `Np` is the number of particles (i.e., sequential Monte Carlo sample size), and `Nmif` is the number of iterations of the optimization procedure carried out below.
- Empirically, `Np=5000` and `Nmif=200` are around the minimum required to get stable results with an error in the likelihood of order 1 log unit for this example; this is implemented by setting `run_level=2`.
- One can then ramp up to larger values for more refined computations, implemented here by `run_level=3`.

```
run_level=3
polio_Np <-          c(100,5e3,1e4)
polio_Nmif <-        c(10, 200,400)
polio_Nreps_eval <-  c(2,  10,  20)
polio_Nreps_local <- c(10, 20, 40)
```

```
polio_Nreps_global <-c(10, 20, 100)
polio_Nsim <-        c(50,100, 500)
```

- Here, `run_level` is coded differently from the previous case studies. It is functionally equivalent. Which way you prefer is up to you, but seeing different alternatives may help to clarify the goal of the code.
- `run_level` is a facility that is convenient for when you are editing the source code. It plays no fundamental role in the final results. If you are not editing the source code, or using the code as a template for developing your own analysis, it has no function.
- When you edit a document with different `run_level` options, you can debug your code by editing `run_level=1`. Then, you can get preliminary assessment of whether your results are sensible with `run_level=2` and get finalized results, with reduced Monte Carlo error, by editing `run_level=3`.
- In practice, you probably want `run_level=1` to run in minutes, `run_level=2` to run in tens of minutes, and `run_level=3` to run in hours.
- You can increase or decrease the numbers of particles, or the number of mif2 iterations, or the number of global searches carried out, to make sure this procedure is practical on your machine.
- Appropriate values of the algorithmic parameters for each run-level are context dependent.

Exercise: Choosing algorithmic parameters

- Discuss how you choose the algorithmic parameters for each run level when building a new likelihood-based data analysis using `pfilter()` and `mif2()` in **pomp**.

Likelihood evaluation at an estimated MLE

- Let's carry out a likelihood evaluation at the reported MLE.
- Since most modern machines have multiple cores, it is convenient to do some parallelization to generate replicated calls to `pfilter`. Notice that the replications are averaged using the `logmeanexp` function.

```
require(doParallel)
registerDoParallel()
```

```
stew(file=sprintf("pf1-%d.rda",run_level),{
  t1 <- system.time(
    pf1 <- foreach(i=1:20,.packages='pomp',
                   .options.multicore=list(set.seed=TRUE)) %dopar% try(
                     pfilter(polio,Np=polio_Np[run_level])
                   )
  )
},seed=493536993,kind="L'Ecuyer")
(L1 <- logmeanexp(sapply(pf1,logLik),se=TRUE))
```

```
##                       se
## -794.5299165    0.1076427
```

- In 5.7 seconds, we obtain an unbiased likelihood estimate of -794.53 with a Monte standard error of 0.11.

Comparison of our implementation with Martinez-Bakker et al. (2015)

This setup has minor differences in notation, model construction and code compared to Martinez-Bakker et al. (2015). The MLE reported for these data by Martinez-Bakker et al. (2015) is -794.34 (with Monte Carlo evaluation error of 0.18) which is similar to the log likelihood at the MLE for our model (-794.53 with Monte Carlo evaluation error 0.11). This suggests that the differences do not substantially improve or decrease the fit of our model compared to Martinez-Bakker et al. (2015). When different calculations match reasonably closely, it demonstrates some reproducibility of both results.

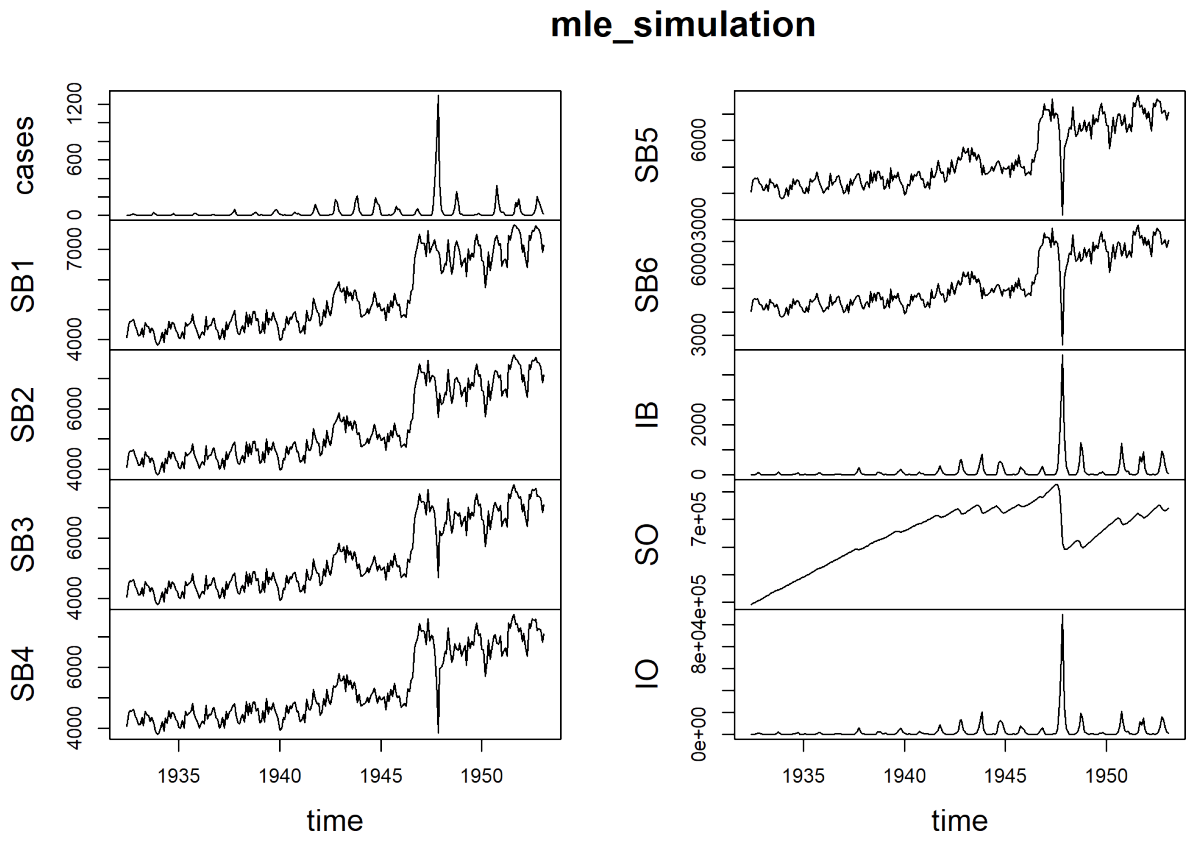
Simulation to investigate the fitted model: Local persistence

The scientific purpose of fitting a model typically involves analyzing properties of the fitted model, often investigated using simulation. Following Martinez-Bakker et al. (2015), we are interested in how often months with no reported cases ($Y_n = 0$) correspond to months without any local asymptomatic cases, defined for our continuous state model as $I_n^B + I_n^O < 1/2$. For Wisconsin, using our model at the estimated MLE, we compute as follows:

```
stew(sprintf("persistence-%d.rda",run_level),{
  t_sim <- system.time(
    sim <- foreach(i=1:polio_Nsim[run_level],.packages='pomp',
                   .options.multicore=list(set.seed=TRUE)) %dopar%
      simulate(polio)
  )
},seed=493536993,kind="L'Ecuyer")

no_cases_data <- sum(obs(polio)==0)
no_cases_sim <- sum(sapply(sim,obs)==0)/length(sim)
fadeout1_sim <- sum(sapply(sim,function(po)states(po)["IB",]+states(po)["IO",]<1))/length(sim)
fadeout100_sim <- sum(sapply(sim,function(po)states(po)["IB",]+states(po)["IO",]<100))/length(sim)
imports_sim <- coef(polio)["psi"]*mean(sapply(sim,function(po) mean(states(po)["SO",]+states(po)["SB1",]
```

- For the data, there were 26 months with no reported cases, in reasonable accordance with the mean of 37.6 for simulations from the fitted model. Months with no asyptomatic infections for the simulations were rare, on average 0.5 months per simulation. Months with fewer than 100 infections averaged 53.1 per simulation, which in the context of a reporting rate of 0.0124 can explain the absences of case reports. For this model, the mean monthly infections due to importations (more specifically, due to the term ψ, whatever its biological interpretation) is 137.1. This does not give much opportunity for local elimination of poliovirus. One could profile over ψ to investigate how sensitive this conclusion is to values of ψ consistent with the data.
- It is also good practice to look at simulations from the fitted model:

```
mle_simulation <- simulate(polio,seed=127)
plot(mle_simulation)
```



- We see from this simulation that the fitted model can generate report histories that look qualitatively similar to the data. However, there are things to notice in the reconstructed latent states. Specifically, the pool of older susceptibles, $S^O(t)$, is mostly increasing. The reduced case burden in the data in the time interval 1932–1945 is explained by a large initial recovered (R) population, which implies much higher levels of polio before 1932. There were large epidemics of polio in the USA early in the 20th century, so this is not implausible.
- A liklihood profile over the parameter $\tilde{S}_0^O$ could help to clarify to what extent this is a critical feature of how the model explains the data.

Local likelihood maximization

- Let's see if we can improve on the previous MLE. We use the iterated filtering algorithm IF2 of Ionides et al. (2015), which uses a random walk in parameter space to approach the MLE. We set a constant random walk standard deviation for each of the regular parameters and a larger constant for each of the initial value parameters.

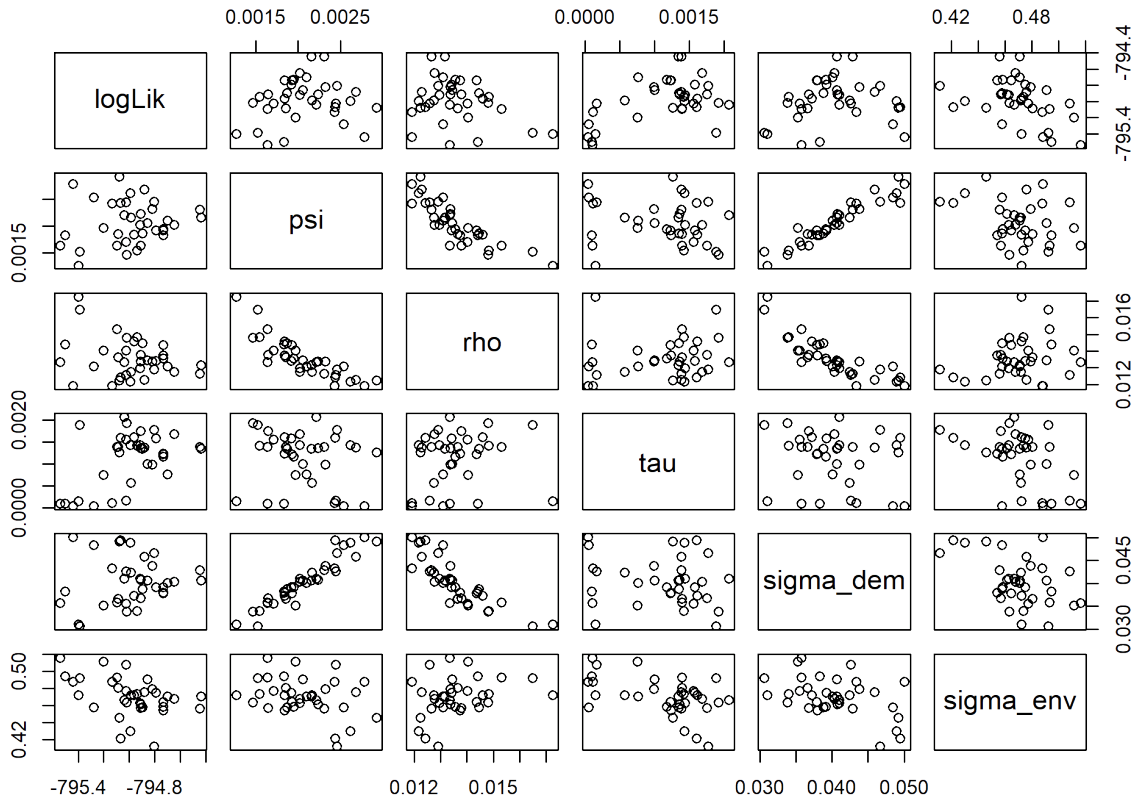
```
polio_rw.sd_rp <- 0.02
polio_rw.sd_ivp <- 0.2
polio_cooling.fraction.50 <- 0.5
```

```
stew(sprintf("mif-%d.rda",run_level),{
  t2 <- system.time({
    m2 <- foreach(i=1:polio_Nreps_local[run_level],
                  .packages='pomp', .combine=c,
                  .options.multicore=list(set.seed=TRUE)) %dopar% try(
                    mif2(polio,
                         Np=polio_Np[run_level],
                         Nmif=polio_Nmif[run_level],
                         cooling.type="geometric",
                         cooling.fraction.50=polio_cooling.fraction.50,
                         transform=TRUE,
                         rw.sd=rw.sd(
                           b1=polio_rw.sd_rp,
                           b2=polio_rw.sd_rp,
                           b3=polio_rw.sd_rp,
                           b4=polio_rw.sd_rp,
                           b5=polio_rw.sd_rp,
                           b6=polio_rw.sd_rp,
                           psi=polio_rw.sd_rp,
                           rho=polio_rw.sd_rp,
                           tau=polio_rw.sd_rp,
                           sigma_dem=polio_rw.sd_rp,
                           sigma_env=polio_rw.sd_rp,
                           IO_0=ivp(polio_rw.sd_ivp),
                           SO_0=ivp(polio_rw.sd_ivp)
                         )
                    )
                  )

    lik_m2 <- foreach(i=1:polio_Nreps_local[run_level],.packages='pomp',
                      .combine=rbind,.options.multicore=list(set.seed=TRUE)) %dopar%
                      {
                        logmeanexp(
                          replicate(polio_Nreps_eval[run_level],
                                    logLik(pfilter(polio,params=coef(m2[[i]]),Np=polio_Np[run_level]))
                          ),
                          se=TRUE)
                      }
  })
},seed=318817883,kind="L'Ecuyer")

r2 <- data.frame(logLik=lik_m2[,1],logLik_se=lik_m2[,2],t(sapply(m2,coef)))
if (run_level>1)
  write.table(r2,file="polio_params.csv",append=TRUE,col.names=FALSE,row.names=FALSE)
summary(r2$logLik,digits=5)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1026.7  -795.2  -795.0  -813.4  -794.9  -794.4
```

- This investigation took 185.3 minutes. These repeated stochastic maximizations can also show us the geometry of the likelihood surface in a neighborhood of this point estimate:

```
pairs(~logLik+psi+rho+tau+sigma_dem+sigma_env,data=subset(r2,logLik>max(logLik)-20))
```



- We see strong tradeoffs between ψ, ρ and σ_{dem}. By itself, in the absence of other assumptions, the pathogen immigration rate ψ is fairly weakly identified. However, the reporting rate ρ is essentially the fraction of poliovirus infections leading to acute flaccid paralysis, which is known to be around 1%. This plot suggests that fixing an assumed value of ρ might lead to much more precise inference on ψ; the rate of pathogen immigration presumably being important for understanding disease persistence. These hypotheses could be investigated more formally by construction of profile likelihood plots and likelihood ratio tests.

Global likelihood maximization

When carrying out parameter estimation for dynamic systems, we need to specify beginning values for both the dynamic system (in the state space) and the parameters (in the parameter space). By convention, we use *initial values* for the initialization of the dynamic system and *starting values* for initialization of the parameter search.

Practical parameter estimation involves trying many starting values for the parameters. One can specify a large box in parameter space that contains all parameter vectors which seem remotely sensible. If an estimation method gives stable conclusions with starting values drawn randomly from this box, this gives some confidence that an adequate global search has been carried out.

For our polio model, a box containing reasonable parameter values might be

```
polio_box <- rbind(
  b1=c(-2,8),
  b2=c(-2,8),
  b3=c(-2,8),
  b4=c(-2,8),
  b5=c(-2,8),
  b6=c(-2,8),
  psi=c(0,0.1),
  rho=c(0,0.1),
  tau=c(0,0.1),
  sigma_dem=c(0,0.5),
  sigma_env=c(0,1),
  S0_0=c(0,1),
  I0_0=c(0,0.01)
)
```

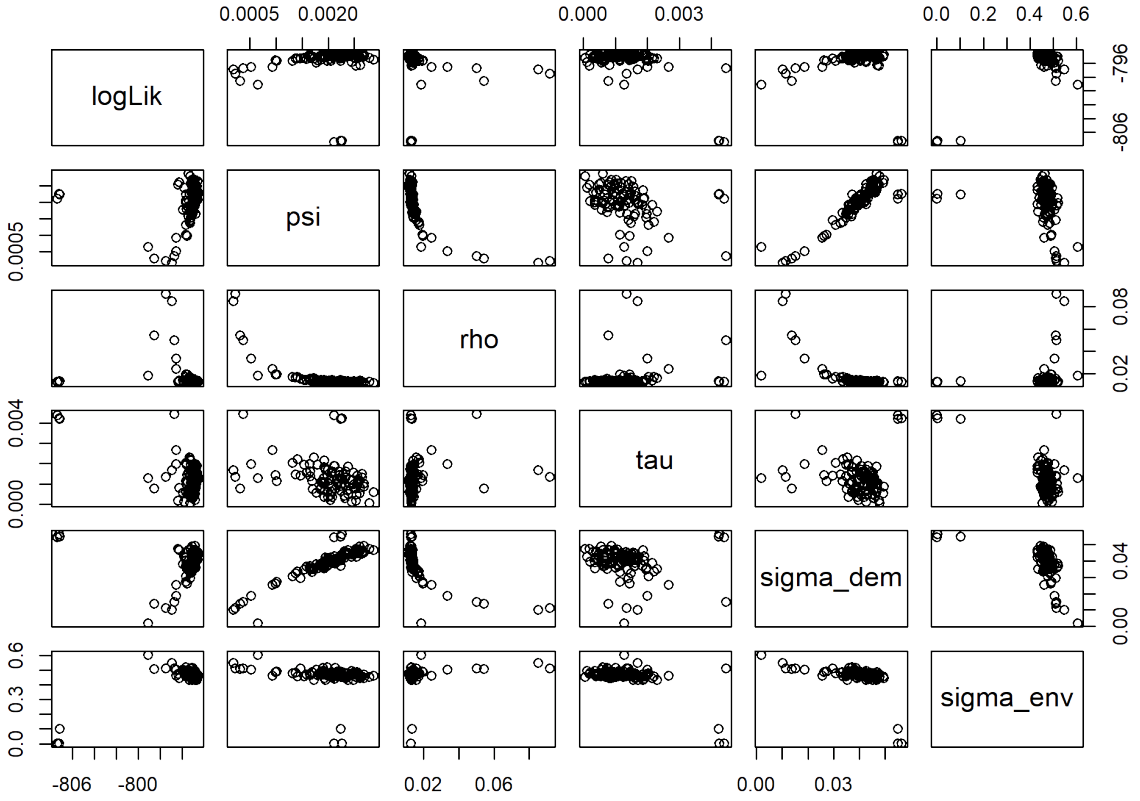
We then carry out a search identical to the local one except for the starting parameter values. This can be succinctly coded by calling `mif2` on the previously constructed object, `m2[[1]]`, with a reset starting value:

```
stew(file=sprintf("box_eval-%d.rda",run_level),{
  t3 <- system.time({
    m3 <- foreach(i=1:polio_Nreps_global[run_level],.packages='pomp',.combine=c,
                  .options.multicore=list(set.seed=TRUE)) %dopar%
      mif2(
        m2[[1]],
        start=c(apply(polio_box,1,function(x)runif(1,x[1],x[2])),polio_fixed_params)
      )

    lik_m3 <- foreach(i=1:polio_Nreps_global[run_level],.packages='pomp',.combine=rbind,
                      .options.multicore=list(set.seed=TRUE)) %dopar% {
                        set.seed(87932+i)
                        logmeanexp(
                          replicate(polio_Nreps_eval[run_level],
                                    logLik(pfilter(polio,params=coef(m3[[i]]),Np=polio_Np[run_level]))
                          ),
                          se=TRUE)
                      }
  })
},seed=290860873,kind="L'Ecuyer")


r3 <- data.frame(logLik=lik_m3[,1],logLik_se=lik_m3[,2],t(sapply(m3,coef)))
if(run_level>1) write.table(r3,file="polio_params.csv",append=TRUE,col.names=FALSE,row.names=FALSE)
summary(r3$logLik,digits=5)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1012.5  -795.6  -795.1  -805.7  -794.9  -794.5
```

- Evaluation of the best result of this search gives a likelihood of -794.5 with a standard error of 0.1. We see that optimization attempts from diverse remote starting points can approach our MLE, but do not exceed it. This gives us some reasonable confidence in our MLE.
- Plotting these diverse parameter estimates can help to give a feel for the global geometry of the likelihood surface

```
pairs(~logLik+psi+rho+tau+sigma_dem+sigma_env,data=subset(r3,logLik>max(logLik)-20))
```



- To understand these global searches, many of which may correspond to parameter values having no meaningful scientific interpretation, it is helpful to put the log likelihoods in the context of some non-mechanistic benchmarks.

Benchmark likelihoods for non-mechanistic models

- The most basic statistical model for data is independent, identically distributed (IID). Picking a negative binomial model,

```
nb_lik <- function(theta) -sum(dnbinom(as.vector(obs(polio)),size=exp(theta[1]),prob=exp(theta[2]),log='
nb_mle <- optim(c(0,-5),nb_lik)
-nb_mle$value
```

```
## [1] -1036.227
```

- We see that a model with likelihood below -1036.2 is unreasonable. This explains a cutoff around this value in the global searches: in these cases, the model is finding essentially IID explanations for the data.
- Linear, Gaussian auto-regressive moving-average (ARMA) models provide non-mechansitic fits to the data including flexible dependence relationships. We fit to $\log(y_n^* + 1)$ and correct the likelihood back to the scale appropriate for the untransformed data:

```
log_y <- log(as.vector(obs(polio))+1)
arma_fit <- arima(log_y,order=c(2,0,2),seasonal=list(order=c(1,0,1),period=12))
arma_fit$loglik-sum(log_y)
```

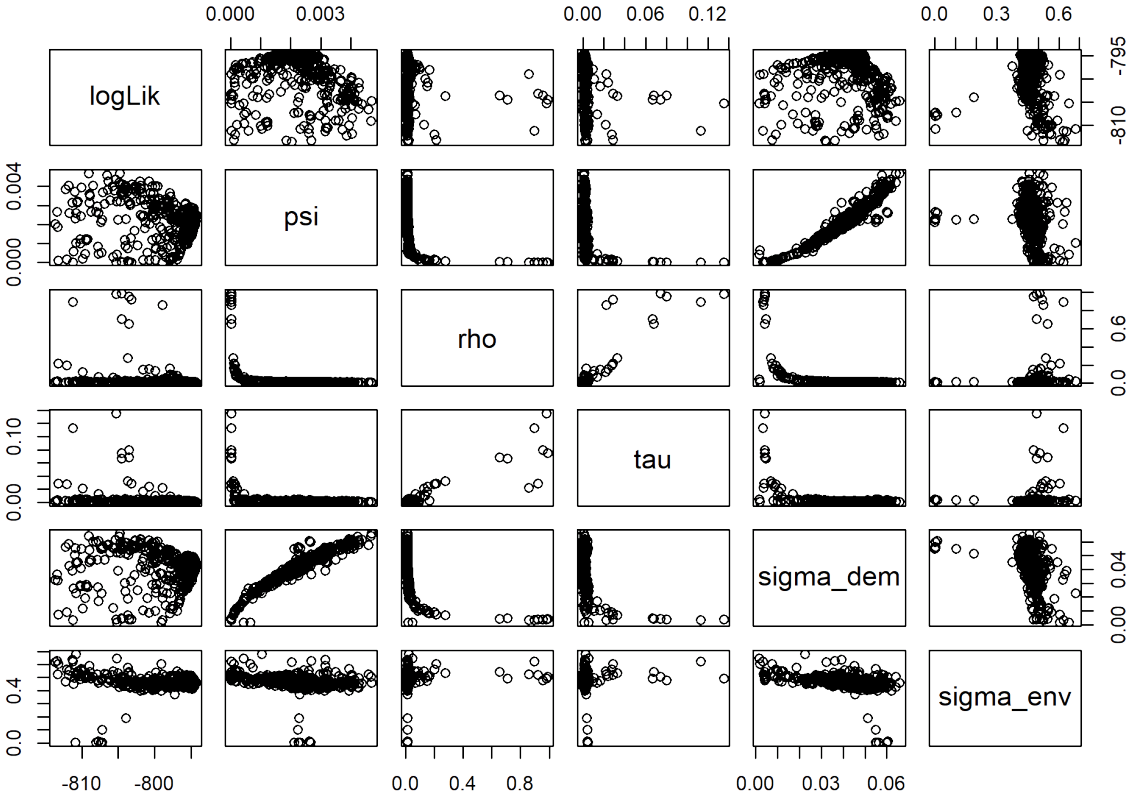
```
## [1] -822.0827
```

- This 7-parameter model, which knows nothing of susceptible depletion, attains a likelihood of -822.1. Although our goal is not to beat non-mechanstic models, it is comforting that we're competitive with them.

Mining previous investigations of the likelihood

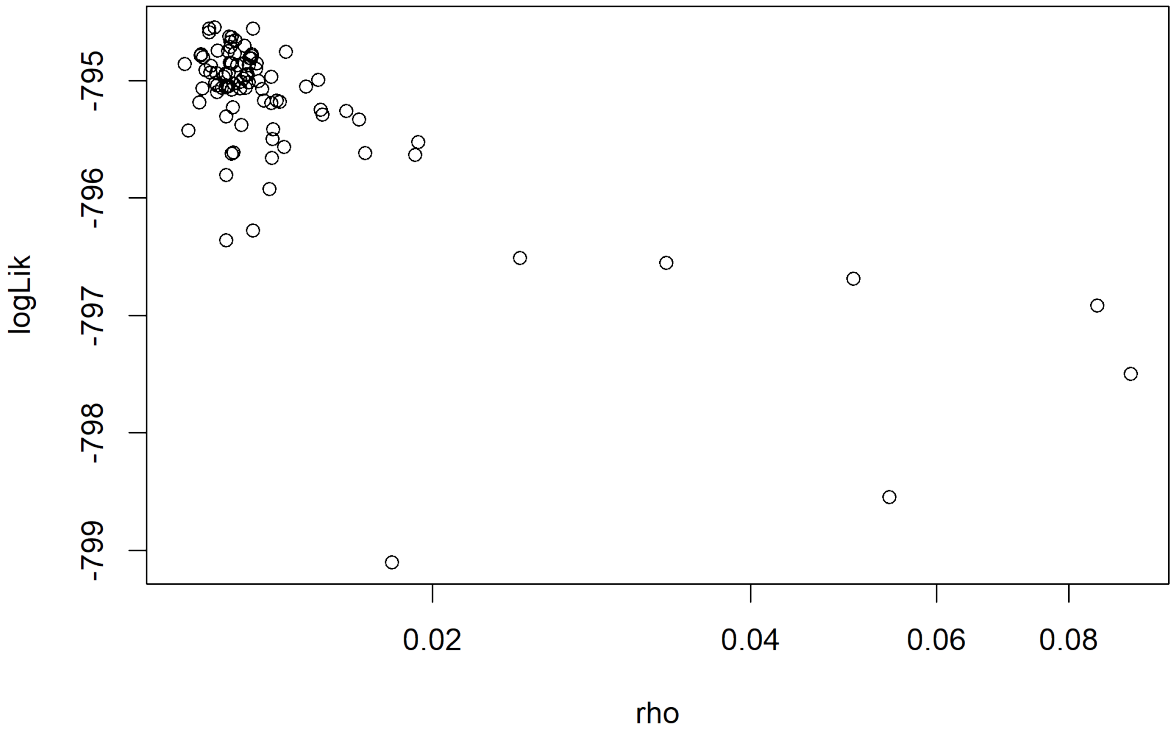
- Saving the results of previous searches, with likelihoods that have been repeatedly evaluated by particle filters, gives a resource for building up knowledge about the likelihood surface. Above, we have added our new results to the file `polio_params.csv`, which we now investigate.

```
polio_params <- read.table("polio_params.csv",row.names=NULL,header=TRUE)
pairs(~logLik+psi+rho+tau+sigma_dem+sigma_env,data=subset(polio_params,logLik>max(logLik)-20))
```



- Here, we see that the most successful searches have always led to models with reporting rate around 1%. This impression can be reinforced by looking at results from the global searches:

```
plot(logLik~rho,data=subset(r3,logLik>max(r3$logLik)-10),log="x")
```



- We see that reporting rates close to 1% seem to provide a small but clear (several units of log likelihood) advantage in explaining the data. These are the reporting rates for which depletion of susceptibles can help to explain the dynamics.

Exercises

Exercise: Initial values

When carrying out parameter estimation for dynamic systems, we need to specify beginning values for both the dynamic system (in the state space) and the parameters (in the parameter space). By convention, we use *initial values* for the initialization of the dynamic system and *starting values* for initialization of the parameter search.

Discuss issues in specifying and inferring initial conditions, with particular reference to this polio example.

Suggest a possible improvement in the treatment of initial conditions here, code it up and make some preliminary assessment of its effectiveness. How will you decide if it is a substantial improvement?

Exercise: Parameter estimation using randomized starting values.

Comment on the computations above, for parameter estimation using randomized starting values. Propose and try out at least one modification of the procedure. How could one make a formal statement quantifying the error of the optimization procedure?

Exercise: Demography and discrete time

- It can be surprisingly hard to include birth, death, immigration, emmigration and aging into a disease model in satisfactory ways. Consider the strengths and weaknesses of the analysis presented. For example, how does it compare to a continuous-time model? In an imperfect world, it is nice to check the extent to which the conclusions are insensitive to alternative modeling decisions. If you have some ideas to change the treatmentof demography (or an other aspect of the model) you could have a go at coding it up to see if it makes a difference.

Technical exercise: Diagnosing filtering and maximization convergence

Are there outliers in the data (i.e., observations that do not fit well with our model)? Are we using unnecessarily large amounts of computer time to get our results? Are there indications that we would should run our computations for longer? Or maybe with different choices of algorithmic settings?

In particular, `cooling.fraction.50` gives the fraction by which the random walk standard deviation is decreased ("cooled") in 50 iterations. If `cooling.fraction.50` is too small, the search will "freeze" too soon, evidenced by flat parallel lines in the convergence diagnostics. If `cooling.fraction.50` is too large, the researcher may run of of time, patience or computing budget (or all three) before the parameter trajectories approach an MLE.

Interpret the diagnostic plots below. Carry out some numerical experiments to test your interpretations.

One could look at filtering diagnostics at the MLE, for example, `plot(pf1[[1]])` but the diagnostic plots for iterated filtering include filtering diagnostics for the last iteration anyhow, so let's just consider the `mif` diagnostic plot. Looking at several simultaneously permits assessment of Monte Carlo variability. `plot` applied to a `mifList` object does this: here, `m3` is of class `mifList` since that is the class resulting from concatenation of `mif2d.pomp` objects using `c()`:

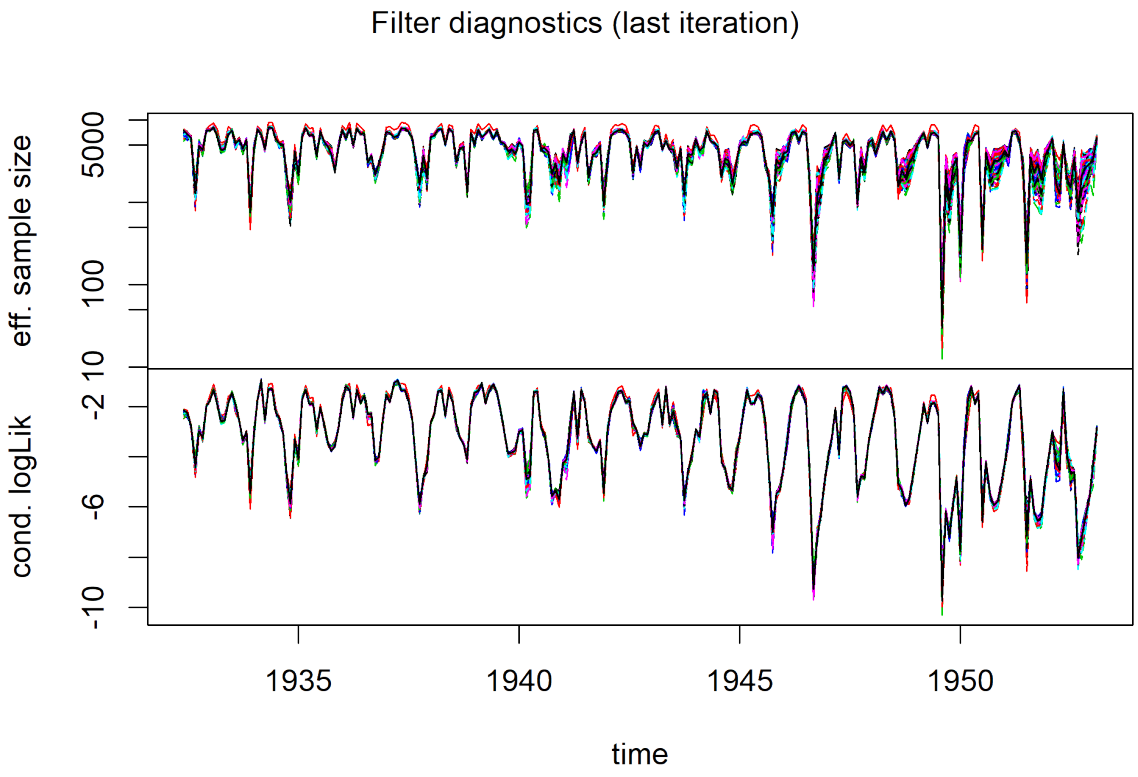
```
class(m3)
```

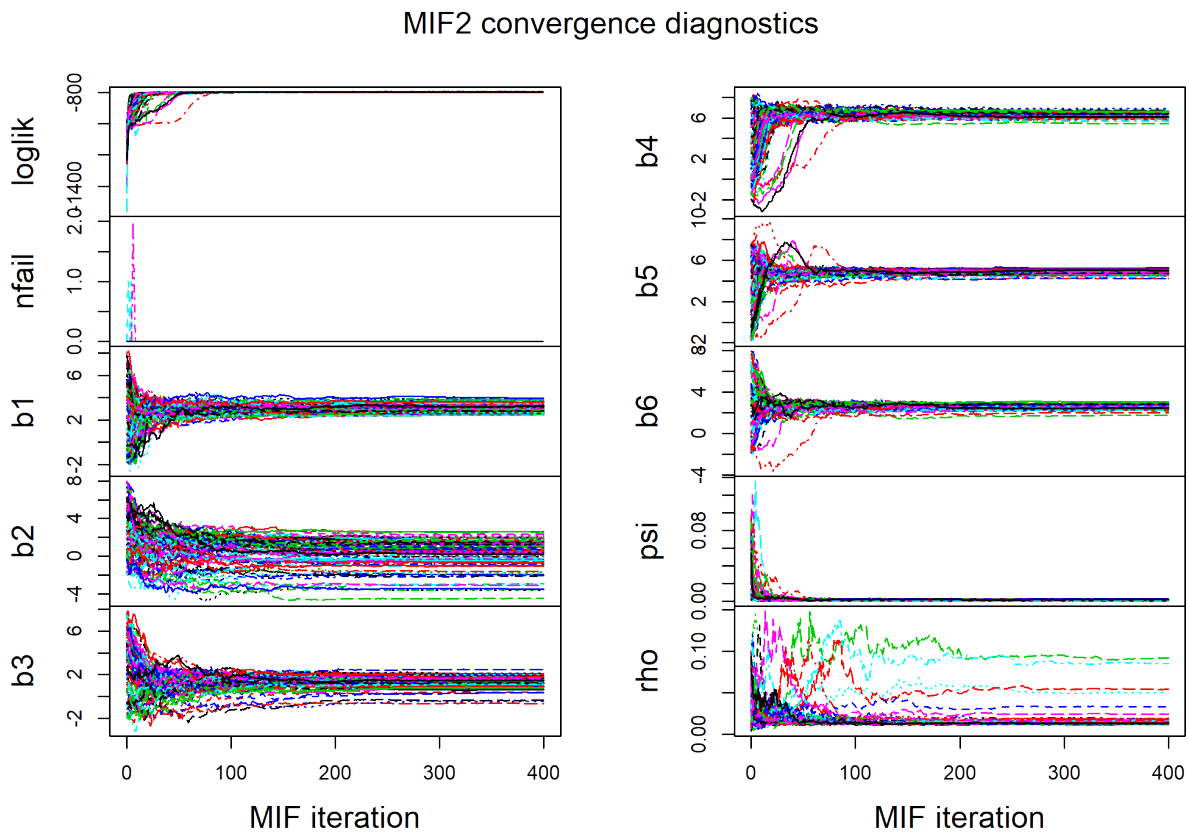
```
## [1] "mif2List"
## attr(,"package")
## [1] "pomp"
```

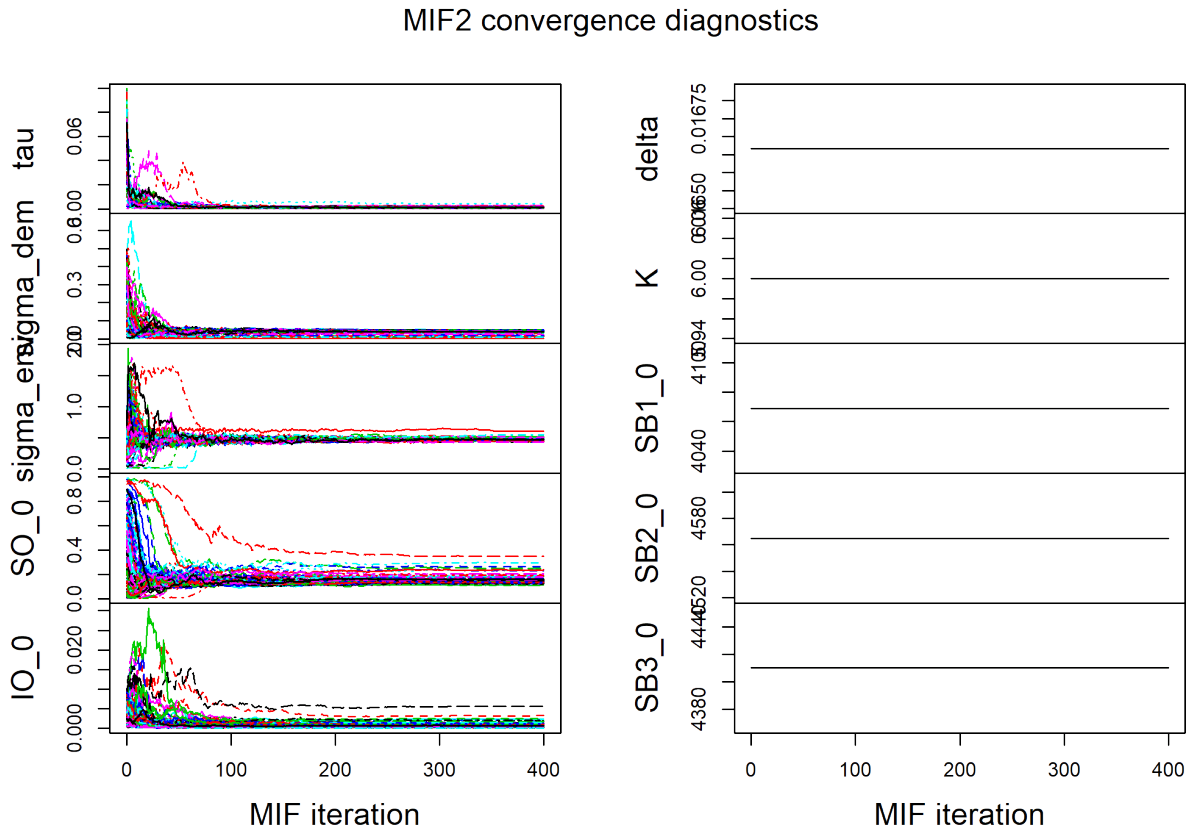
```
class(m3[[1]])
```

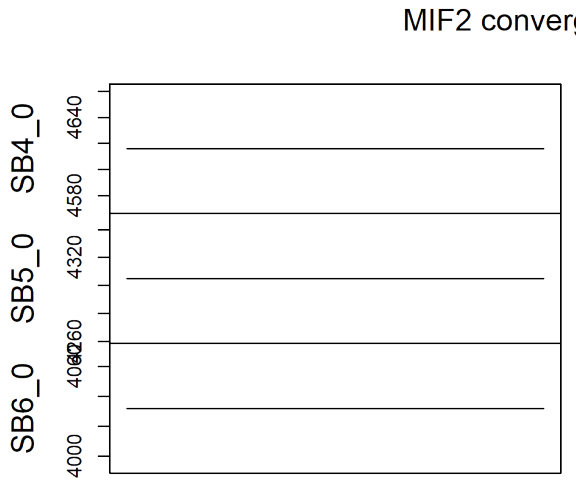
```
## [1] "mif2d.pomp"
## attr(,"package")
## [1] "pomp"
```

```
plot(m3[r3$logLik>max(r3$logLik)-10])
```



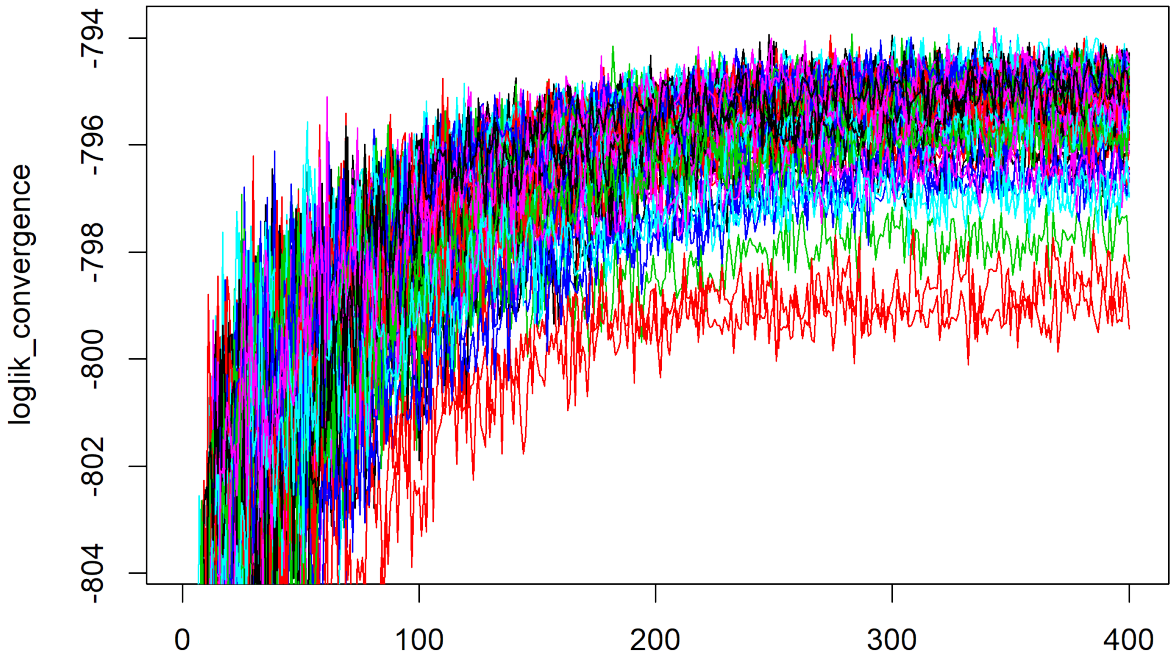

MIF2 convergence diagnostics
loglik
nfail
b1
b2
b3
b4
b5
b6
psi
rho
MIF iteration
MIF iteration


MIF2 convergence diagnostics
tau
sigma_dem
sigma_env
SO_0
IO_0
delta
K
SB1_0
SB2_0
SB3_0
MIF iteration
MIF iteration



The likelihood is particularly important to keep in mind. If parameter estimates are numerically unstable, that could be a consequence of a weakly identified parameter subspace. The presence of some weakly identified combinations of parameters is not fundamentally a scientific flaw; rather, our scientific inquiry looks to investigate which questions can and cannot be answered in the context of a set of data and modeling assumptions. Thus, as long as the search is demonstrably approaching the maximum likelihood region we should not necessarily be worried about the stability of parameter values (at least, from the point of diagnosing successful maximization). So, let's zoom in on the likelihood convergence:

```
loglik_convergence <- do.call(cbind,conv.rec(m3[r3$logLik>max(r3$logLik)-10],"loglik"))
matplot(loglik_convergence,type="l",lty=1,ylim=max(loglik_convergence,na.rm=T)+c(-10,0))
```



Acknowledgment

These notes draw on material developed for a short course on Simulation-based Inference for Epidemiological Dynamics by Aaron King and Edward Ionides, taught at the University of Washington Summer Institute in Statistics and Modeling in Infectious Diseases, 2015, 2016 and 2017.

References

Ionides, E. L., D. Nguyen, Y. Atchadé, S. Stoev, and A. A. King. 2015. Inference for dynamic and latent variable models via iterated, perturbed Bayes maps. Proceedings of the National Academy of Sciences of USA 112:719–724.

Martinez-Bakker, M., A. A. King, and P. Rohani. 2015. Unraveling the transmission ecology of polio. PLoS Biology 13:e1002172.