

## Lesson 6: Iterated filtering in pomp

Spencer J. Fox    Qianying (Ruby) Lin    Jesse Wheeler

## Applying IF2 to the Consett measles outbreak

We are going to look at an example of Iterated Filtering (IF2) in practice. We will also discuss parameter uncertainty and confidence interval estimation.

As an example, we will the Consett measles outbreak we began to examine in past lessons.

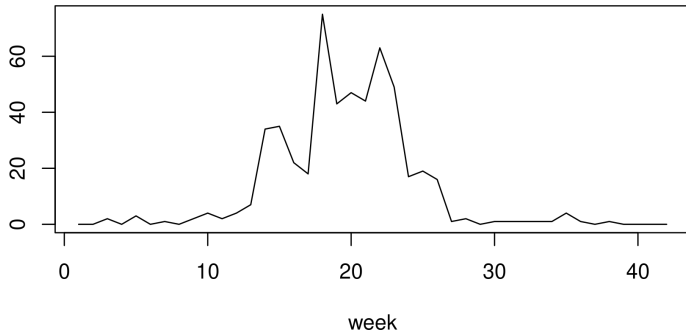
The following loads the data, pomp, and the stochastic SIR model we constructed there.

## Rebuilding Model with transformations

```
source("scripts/model_measSIR.R")

# Transformations to stay "in-bounds":
#   - log: parameters must be positive.
#   - logit: parameters in (0, 1)
measSIR <- measSIR |> pomp(
  partrans = parameter_trans(
    log=c("Beta", "Gamma", "k"),logit=c("Rho","Eta")
  ),
  paramnames = c("Beta","Gamma","Eta","Rho","k","N")
)
```

## Measles in Consett



## Workflow Overview

As we have seen, IF2 (`mif2` in `pomp`) is stochastic, meaning we don't get the same results every time. How do we know when we have found an MLE?

The basic MIF2 workflow may include some work on our local machines (laptops), and some on larger computing clusters (if possible).

- ▶ Ensure we can simulate and `pfilter` our model (local).
- ▶ Pick reasonable start values for our parameters (local).
- ▶ Perform a small local search for parameters (local).
- ▶ Increase search area and computational effort (local/cluster).
- ▶ Once we have a good sense of the likelihood surface, compute profiles (cluster).

A cluster is not always needed, but some computations take a long time.

## Setting up the estimation problem

Let's assume that the population size,  $N$ , is known accurately. We'll fix that parameter.

Let's revisit the assumption that the infectious period is 2 weeks, imagining that we have access to the results of household and clinical studies that have concluded that infected patients shed the virus for 3–4~da. We'll use these results to constrain the infectious period in our model to 3.5~da, i.e.,  $\gamma = 2 \text{ wk}^{-1}$ . We also fix  $k = 10$ . Later, we can relax our assumptions.

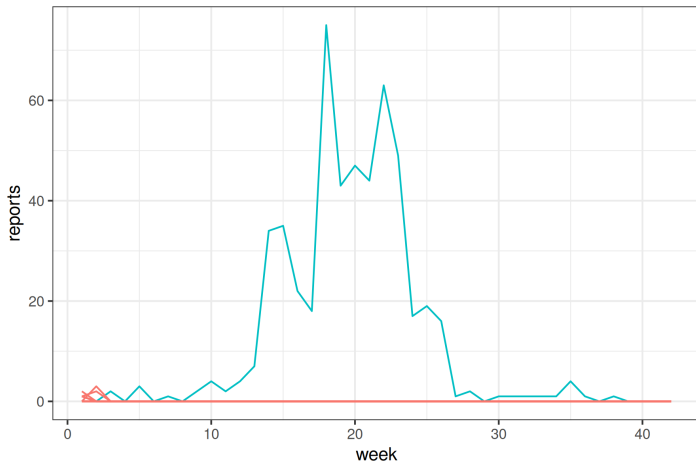
```
fixed_params <- c(N=38000, Gamma=2, k=10)
coef(measSIR, names(fixed_params)) <- fixed_params
coef(measSIR)
```

| Beta    | Gamma   | Rho     | k       | Eta     | N       |
|---------|---------|---------|---------|---------|---------|
| 1.5e+01 | 2.0e+00 | 5.0e-01 | 1.0e+01 | 6.0e-02 | 3.8e+04 |

We proceed to estimate  $\beta$ ,  $\eta$ , and  $\rho$ .

## Sanity check

In Lesson 3, we have introduced how to test the codes and the particle filter from three aspects. Now we can compare the simulations with the raw data using the proposed parameters:



## Parallel computing

It will be helpful to parallelize most of the computations.

```
library(foreach)
library(doParallel)
library(doRNG)

cores <- parallel::detectCores()
registerDoParallel(cores)

# Windows:
# cl <- makePSOCKcluster(cores) # Windows
# registerDoParallel(cl) # Windows
```



## Running a particle filter I

We proceed to carry out replicated particle filters at an initial guess of  $\beta = 15$ ,  $\eta = 0.06$ , and  $\rho = 0.5$ .

```
registerDoRNG(123)
foreach(i=1:10,.combine=c) %dopar% {
  measSIR |> pfilter(Np=5000)
} -> pf
pf |> logLik() |> logmeanexp(se=TRUE) -> L_pf
L_pf
```

| est         | se       |
|-------------|----------|
| -289.927238 | 1.653861 |

In 1.15 seconds, using 10 cores, we obtain an unbiased likelihood estimate of -289.9 with a Monte Carlo standard error of 1.7.

## Building up a picture of the likelihood surface I

- ▶ Given a model and a set of data, the likelihood surface is well defined, though it may be difficult to visualize.
- ▶ We can develop a progressively more complete picture of this surface by storing likelihood estimates whenever we compute them.
- ▶ **It is a very good idea** to set up a database within which to store the likelihood of every point for which we have an estimated likelihood.
- ▶ This will become larger and more complete as our parameter-space search goes on and will be a basis for a variety of explorations.

At this point, we've computed the likelihood at a single point. Let's store this point, together with the estimated likelihood and our estimate of the standard error on that likelihood, in a CSV file:

## Building up a picture of the likelihood surface II

```
pf[[1]] |> # 10 pfilterd objects, all have same parameters
  coef() |> # Extract parameters
  bind_rows() |> # Convert to data.frame
  bind_cols(loglik=L_pf[1],loglik.se=L_pf[2]) |> # Add evaluation.
  write_csv("measles_params.csv") # Save to "database"
```

## A local search of the likelihood surface I

Let's carry out a local search using `mif2` around this point in parameter space.

- ▶ We need to choose the `rw.sd` and `cooling.fraction.50` algorithmic parameters.
- ▶ Since  $\beta$  and  $\gamma$  will be estimated on the log scale, and we expect that multiplicative perturbations of these parameters will have roughly similar effects on the likelihood, we'll use a perturbation size of 0.02, which we imagine will have a small but non-negligible effect.
- ▶ For simplicity, we'll use the same perturbation size on  $\rho$ .
- ▶ We fix `cooling.fraction.50`=0.5, so that after 50 `mif2` iterations, the perturbations are reduced to half their original magnitudes.

## A local search of the likelihood surface II

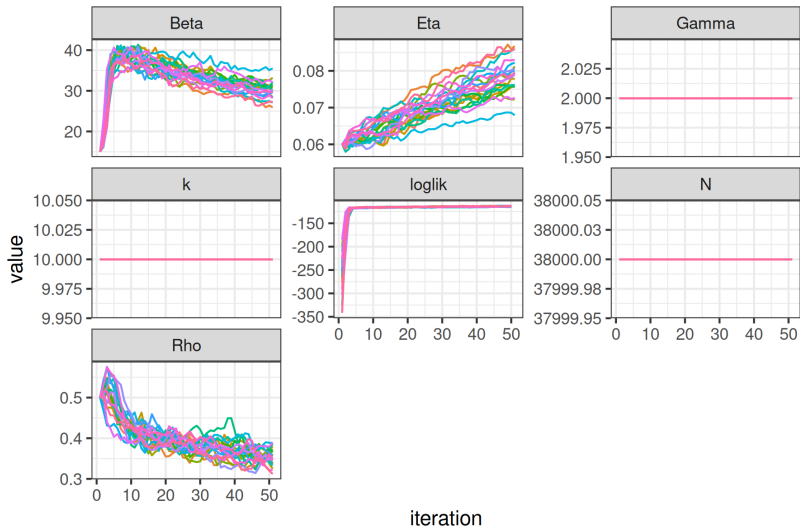
```
registerDoRNG(829479)
foreach(i=1:20,.combine=c) %dopar% {
  measSIR |>
    mif2(
      Np=2000, Nmif=50, cooling.fraction.50=0.5,
      rw.sd=rw_sd(Beta=0.02, Rho=0.02, Eta=ivp(0.02))
    )
} -> mifs_local
```

## Iterated filtering diagnostics I

We obtain some diagnostic plots with the `plot` command applied to `mifs_local`. Here is a way to get a prettier version:

```
mifs_local |>
  traces() |>
  melt() |>
  ggplot(aes(x=iteration,y=value,group=.L1,color=factor(.L1)))+
  geom_line()+
  guides(color="none")+
  facet_wrap(~name,scales="free_y")
```

## Iterated filtering diagnostics II



## Iterated filtering diagnostics III

- ▶ We see that the likelihood increases as the iterations proceed, though there is considerable variability due to
  1. the pooriness of our starting guess and
  2. the stochastic nature of this Monte Carlo algorithm.
- ▶ We see movement in the parameters, though variability remains.



## Estimating the likelihood I

Although the filtering carried out by `mif2` in the final filtering iteration generates an approximation to the likelihood at the resulting point estimate, this is not good enough for reliable inference.

- ▶ Partly, this is because parameter perturbations are applied in the last filtering iteration, so that the likelihood reported by `mif2` is not identical to that of the model of interest.
- ▶ Partly, this is because `mif2` is usually carried out with fewer particles than are needed for a good likelihood evaluation.

## Estimating the likelihood II

Therefore, we evaluate the likelihood, together with a standard error, using replicated particle filters at each point estimate.

```
registerDoRNG(900242)
foreach(mf=mifs_local,.combine=rbind) %dopar% {
  evals <- replicate(10, logLik(pfilter(mf,Np=5000)))
  ll <- logmeanexp(evals,se=TRUE)
  mf |> coef() |> bind_rows() |>
    bind_cols(loglik=ll[1],loglik.se=ll[2])
} -> results
```

On 24 processors, this local investigation took 16~sec for the maximization and 6~sec for the likelihood evaluation.

## Estimating the likelihood III

```
results |> filter(loglik==max(loglik))
```

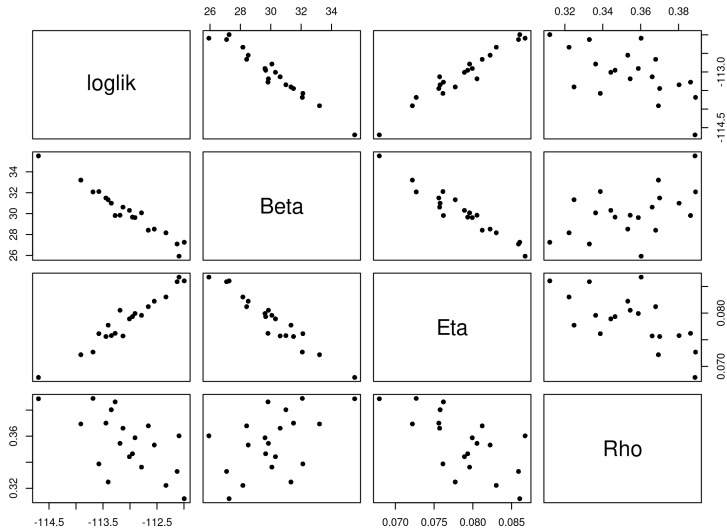
```
# A tibble: 1 x 8
```

|   | Beta  | Gamma | Rho   | k     | Eta    | N     | loglik | loglik.se |
|---|-------|-------|-------|-------|--------|-------|--------|-----------|
|   | <dbl> | <dbl> | <dbl> | <dbl> | <dbl>  | <dbl> | <dbl>  | <dbl>     |
| 1 | 27.3  | 2     | 0.312 | 10    | 0.0861 | 38000 | -112.  | 0.0388    |

These repeated stochastic maximizations can also show us the geometry of the likelihood surface in a neighborhood of this point estimate:

```
pairs(~loglik+Beta+Eta+Rho,data=results,pch=16)
```

## Estimating the likelihood IV



## Building up a picture of the likelihood surface

This plot shows a hint of a ridge in the likelihood surface (cf.~the  $\beta$ - $\eta$  panel). However, the sampling is as yet too sparse to give a clear picture. We add these newly explored points to our database,

```
read_csv("measles_params.csv") |>
  bind_rows(results) |>
  arrange(-loglik) |>
  write_csv("measles_params.csv")
```

and move on to a more thorough exploration of the likelihood surface.

## A global search of the likelihood surface I

- ▶ 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).
- ▶ Practical parameter estimation involves trying many starting values for the parameters.
- ▶ One way to approach this is to choose a large box in parameter space that contains all remotely sensible parameter vectors.
- ▶ 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.

## A global search of the likelihood surface II

- ▶ Large searches can take a long time to finish.
- ▶ Fortunately, we can easily estimate how long we will expect our search to take. - For a fixed data set and model, IF2 has computational complexity that scales linearly with the number of iterations ( $M$ ), and the number of particles ( $J$ ).

Here's a basic example. Suppose that we did a small local search with  $M = 50$ ,  $J = 2000$  replicated 16 times, using 10 cores, and that it took 3 minutes. From this, we can conclude each individual search takes approximately  $3 \text{ min} / \text{ceiling}(16/10) = 1.5 \text{ min}$ .

Then, if I want to use  $M = 75$  iterations,  $J = 2500$  particles, with 87 replicates on the same computer, then the total computing time would be approximately:

$$1.5 \text{ min} \times (75/50) \times (2500/2000) \times \text{ceiling}(87/10) \approx 25.3 \text{ min}$$

## A global search of the likelihood surface III

For our measles model, a box containing reasonable parameter values might be  $\beta \in (5, 80)$ ,  $\rho \in (0.2, 0.9)$ ,  $\eta \in (0, 1)$ .

We are now ready to carry out likelihood maximizations from diverse starting points.

```
set.seed(2062379496)

runif_design(
  lower=c(Beta=5,Rho=0.2,Eta=0),
  upper=c(Beta=80,Rho=0.9,Eta=1),
  nseq=400
) -> guesses

mf1 <- mifs_local[[1]]
```



## A global search of the likelihood surface IV

```
registerDoRNG(127040)
foreach(guess=iter(guesses,"row"), .combine=rbind) %dopar% {
  mf1 |>
    mif2(params=c(guess,fixed_params)) |>
    mif2(Nmif=100) -> mf
  replicate(
    10,
    mf |> pfilter(Np=5000) |> logLik()
  ) |>
    logmeanexp(se=TRUE) -> ll
  mf |> coef() |> bind_rows() |>
    bind_cols(loglik=ll[1],loglik.se=ll[2])
} -> results
```

## A global search of the likelihood surface $V$

- ▶ The above codes run one search from each of 400 starting values.
- ▶ Each search consists of an initial run of 50 IF2 iterations, followed by another 100 iterations.
- ▶ These codes exhibit a general pomp behavior:
  - ▶ Re-running a command on an object (i.e., `mif2` on `mf1`) created by the same command preserves the algorithmic arguments.
  - ▶ In particular, running `mif2` on the result of a `mif2` computation re-runs IF2 from the endpoint of the first run.
  - ▶ In the second computation, by default, all algorithmic parameters are preserved; here we overrode the default choice of `Nmif`.
- ▶ Following the `mif2` computations, the particle filter is used to evaluate the likelihood, as before.

## A global search of the likelihood surface VI

```
results |> filter(loglik==max(loglik))
```

```
# A tibble: 1 x 8
```

|   | Beta  | Rho    | Eta   | N     | Gamma | k     | loglik | loglik.se |
|---|-------|--------|-------|-------|-------|-------|--------|-----------|
|   | <dbl> | <dbl>  | <dbl> | <dbl> | <dbl> | <dbl> | <dbl>  | <dbl>     |
| 1 | 3.91  | 0.0607 | 0.566 | 38000 | 2     | 10    | -104.  | 0.0352    |

- ▶ In contrast to the local-search codes above, here we return only the endpoint of the search, together with the likelihood estimate and its standard error in a named vector.
- ▶ The best result of this search had a likelihood of -104.3 with a standard error of 0.04.
- ▶ This took 13.7 minutes altogether using 24 processors.

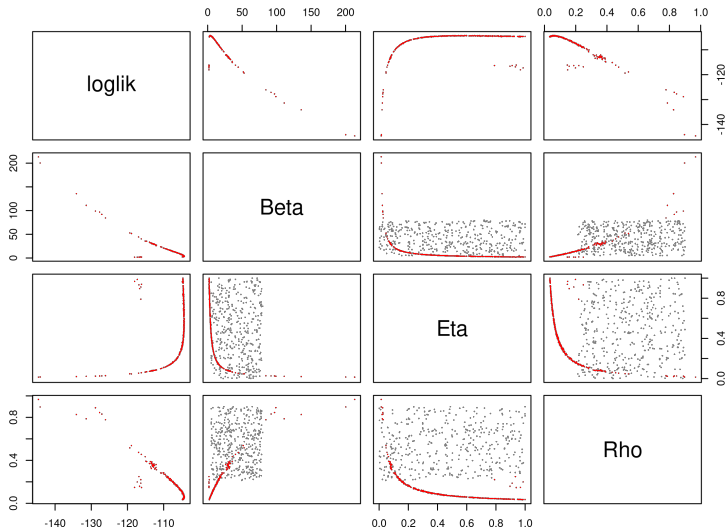
## A global search of the likelihood surface VII

Again, we attempt to visualize the global geometry of the likelihood surface using a scatterplot matrix. In particular, here we plot both the starting values (grey) and the IF2 estimates (red).

```
read_csv("measles_params.csv") |>
  filter(loglik>max(loglik)-50) |>
  bind_rows(guesses) |>
  mutate(type=if_else(is.na(loglik),"guess","result")) |>
  arrange(type) -> all

pairs(~loglik+Beta+Eta+Rho, data=all, pch=16, cex=0.3,
  col=ifelse(all$type=="guess",grey(0.5),"red"))
```

# A global search of the likelihood surface VIII



## A global search of the likelihood surface IX

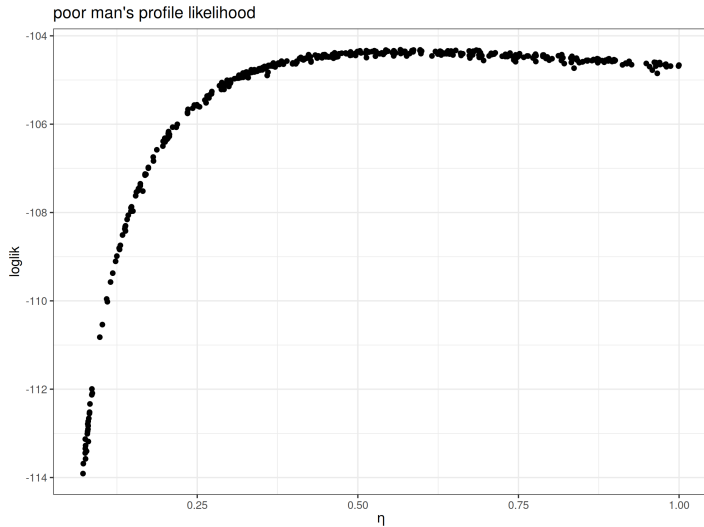
- ▶ We see that optimization attempts from diverse remote starting points converge on a particular region in parameter space.
- ▶ The estimates have comparable likelihoods, despite their considerable variability.
- ▶ This gives us some confidence in our maximization procedure.

## A global search of the likelihood surface X

The projections of the estimates give us 'poor mans profiles':

```
all |>
  filter(type=="result") |>
  filter(loglik>max(loglik)-10) |>
  ggplot(aes(x=Eta,y=loglik))+
  geom_point()+
  labs(
    x=latex2exp::TeX('$\\eta$'),
    title="poor man's profile likelihood"
  )
```

# A global search of the likelihood surface XI





## Likelihood Geometry: implications

- ▶ Parameter estimates are not very useful without some measure of their uncertainty.
- ▶ Traditionally, this is done in the form of a(n) (approximate) confidence interval.
- ▶ We can use the curvature in the likelihood surface to express uncertainty: parameters with likelihood near the maximum are more favorable than those who have much lower likelihood.
- ▶ Question: How much lower than the maximum is acceptable?

## Standard errors for the MLE

There are three main approaches to estimating the statistical uncertainty in an MLE.

1. The Fisher information.
2. **Profile likelihood estimation.**
3. A simulation study, also known as a bootstrap.

## Fisher information

- ▶ A computationally quick approach when one has access to satisfactory numerical second derivatives of the log-likelihood.
- ▶ The approximation is satisfactory only when  $\hat{\theta}$  is well approximated by a normal distribution.
- ▶ **Neither of the two requirements above are typically met for POMP models.**

## Profile likelihood estimation

This approach is generally preferable to the Fisher information for POMP models. We will explain this method below and put it into practice.

## The bootstrap

- ▶ If done carefully and well, this can be the best approach.
- ▶ A confidence interval is a claim about reproducibility. You claim, so far as your model is correct, that on 95% of realizations from the model, a 95% confidence interval you have constructed will cover the true value of the parameter.
- ▶ A simulation study can check this claim fairly directly, but requires the most effort.
- ▶ The simulation study takes time for you to develop and debug, time for you to explain, and time for the reader to understand and check what you have done. We usually carry out simulation studies to check our main conclusions only.

## Confidence intervals via the profile likelihood I

- ▶ Let's consider the problem of obtaining a confidence interval for the first component of  $\theta$ . We'll write

$$\theta = (\phi, \psi).$$

- ▶ The **profile log-likelihood function** of  $\phi$  is defined to be

$$\ell^{\text{profile}}(\phi) = \max_{\psi} \ell(\phi, \psi).$$

In general, the profile likelihood of one parameter is constructed by maximizing the likelihood function over all other parameters.

- ▶ Note that,  $\max_{\phi} \ell^{\text{profile}}(\phi) = \max_{\theta} \ell(\theta)$  and that maximizing the profile likelihood  $\ell^{\text{profile}}(\phi)$  gives the MLE,  $\hat{\theta}$ . Why?
- ▶ An approximate 95% confidence interval for  $\phi$  is given by

$$\{\phi : \ell(\hat{\theta}) - \ell^{\text{profile}}(\phi) < 1.92\}.$$

## Confidence intervals via the profile likelihood II

- ▶ This is known as a profile likelihood confidence interval. The cutoff 1.92 is derived using Wilks' theorem.
- ▶ Although Wilks' theorem also uses asymptotic justification, profile likelihood confidence intervals tend to work better than Fisher information confidence intervals when  $N$  is finite—particularly when the log-likelihood function is not quadratic near its maximum.

## Profile likelihood over $\eta$ I

- ▶ The curvature displayed in the upper envelope of the above plot suggests that there is indeed information in the data with respect to the susceptible fraction,  $\eta$ .
- ▶ To solidify this evidence, we compute a profile likelihood over this parameter.
- ▶ Recall that this means determining, for each value of  $\eta$ , the best likelihood that the model can achieve.
- ▶ To do this, we'll first bound the uncertainty by putting a box around the highest-likelihood estimates we've found so far.
- ▶ Within this box, we'll choose some random starting points, for each of several values of  $\eta$ .



## Profile likelihood over $\eta$ II

```
read_csv("measles_params.csv") |>
  filter(loglik>max(loglik)-20,loglik.se<2) |>
  sapply(range) -> box
box
```

|      | Beta      | Gamma | Rho        | k  | Eta        | N     |
|------|-----------|-------|------------|----|------------|-------|
| [1,] | 1.777527  | 2     | 0.03367763 | 10 | 0.04667726 | 38000 |
| [2,] | 52.917785 | 2     | 0.53726724 | 10 | 0.99981184 | 38000 |

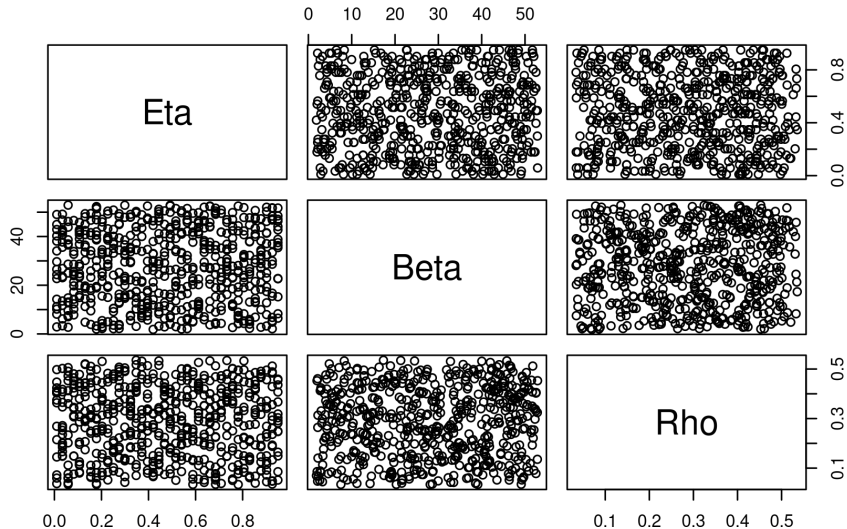
  

|      | loglik    | loglik.se  |
|------|-----------|------------|
| [1,] | -119.3687 | 0.01436077 |
| [2,] | -104.3148 | 0.22958738 |

## Profile likelihood over $\eta$ III

```
freeze(seed=1196696958,  
  profile_design(  
    Eta=seq(0.01,0.95,length=40),  
    lower=box[1,c("Beta","Rho")],  
    upper=box[2,c("Beta","Rho")],  
    nprof=15, type="runif"  
  )) -> guesses  
plot(guesses)
```

## Profile likelihood over $\eta$ IV



## Profile likelihood over $\eta$ $\vee$

- ▶ Now, we'll start one independent sequence of iterated filtering operations from each of these points.
- ▶ We'll be careful to keep  $\eta$  fixed.
- ▶ This is accomplished by not giving this parameter a random perturbation in the `mif2` call.

## Profile likelihood over $\eta$ VI

```
registerDoRNG(830007)
foreach(guess=iter(guesses,"row"), .combine=rbind) %dopar% {
  mf1 |>
    mif2(params=c(guess,fixed_params),
          rw.sd=rw_sd(Beta=0.02,Rho=0.02)) |>
    mif2(Nmif=100,cooling.fraction.50=0.3) -> mf
  replicate(
    10,
    mf |> pfilter(Np=5000) |> logLik()) |>
    logmeanexp(se=TRUE) -> ll
  mf |> coef() |> bind_rows() |>
    bind_cols(loglik=ll[1],loglik.se=ll[2])
} -> results
```

## Visualizing profile likelihood I

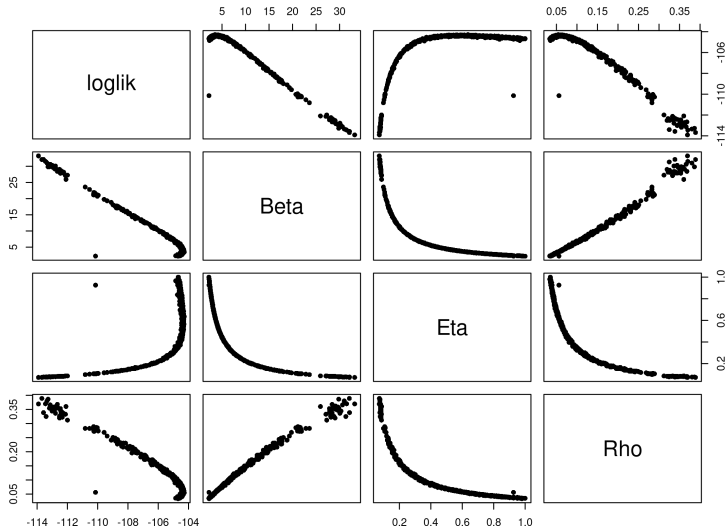
As always, we save the results in our global database and plot the results.

```
read_csv("measles_params.csv") |>
  bind_rows(results) |>
  filter(is.finite(loglik)) |>
  arrange(-loglik) |>
  write_csv("measles_params.csv")
```

```
read_csv("measles_params.csv") |>
  filter(loglik>max(loglik)-10) -> all

pairs(~loglik+Beta+Eta+Rho,data=all,pch=16)
```

## Visualizing profile likelihood II



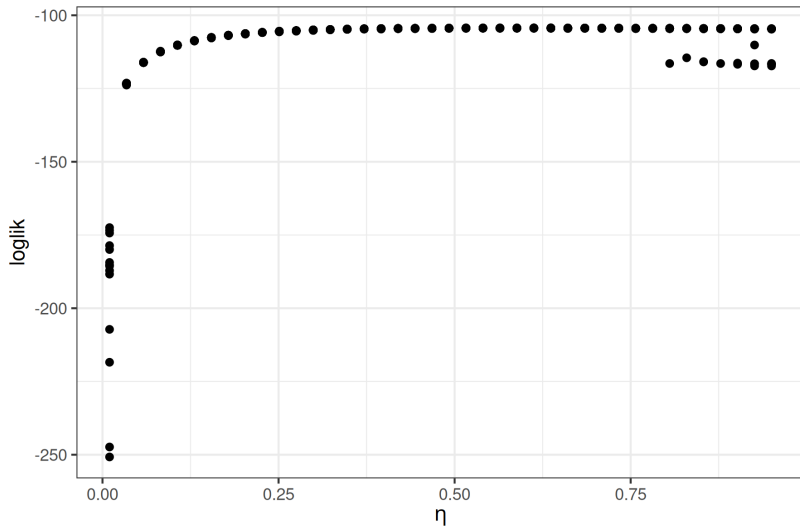
## Visualizing profile likelihood III

Plotting just the results of the profile calculation reveals that, while some of the IF2 runs either become “stuck” on local minima or run out of opportunity to reach the heights of the likelihood surface, many of the runs converge on high likelihoods.

```
results |>  
  ggplot(aes(x=Eta,y=loglik))+  
  geom_point() + xlab(expression(eta))
```



## Visualizing profile likelihood IV

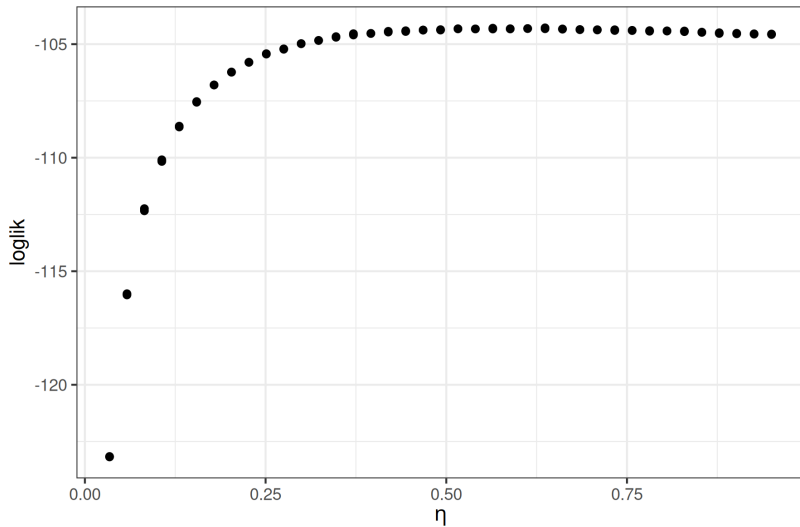


## Visualizing profile likelihood V

A closer look shows what at first appears to be quite a flat surface over much of the explored range of  $\eta$ . Note that this appearance is due to the vertical scale, which is driven by the very low likelihoods associated with the smallest values of  $\eta$ .

```
results |>
  filter(is.finite(loglik)) |>
  group_by(round(Eta,5)) |>
  filter(rank(-loglik)<3) |>
  ungroup() |>
  filter(loglik>max(loglik)-20) |>
  ggplot(aes(x=Eta,y=loglik))+
  geom_point() + xlab(expression(eta))
```

## Visualizing profile likelihood VI



## Visualizing profile likelihood VII

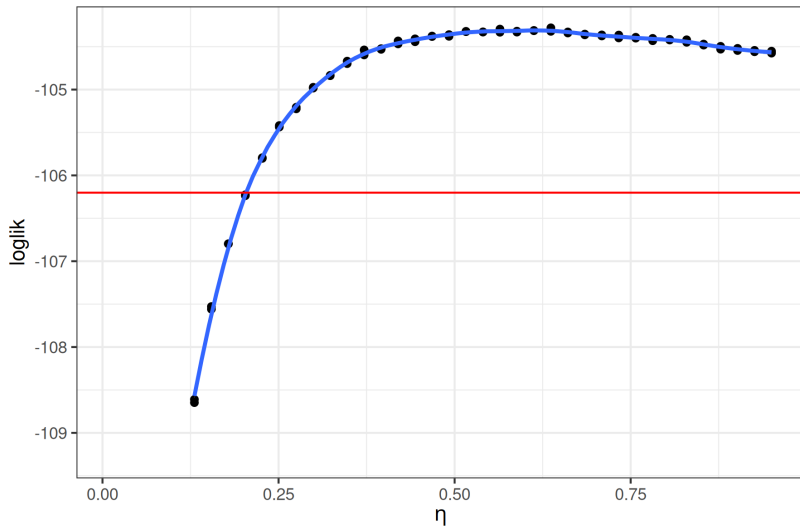
Focusing on just the top of the surface shows that, in fact, one is able to estimate  $\eta$  using these data. In the following plot, the cutoff for the 95% confidence interval (CI) is shown.

## Visualizing profile likelihood VIII

```
maxloglik <- max(results$loglik, na.rm=TRUE)
ci.cutoff <- maxloglik - 0.5 * qchisq(df=1, p=0.95)

results |>
  filter(is.finite(loglik)) |>
  group_by(round(Eta, 5)) |>
  filter(rank(-loglik) < 3) |>
  ungroup() |>
  ggplot(aes(x=Eta, y=loglik)) +
  geom_point() + xlab(expression(eta)) +
  geom_smooth(method="loess", span=0.25) +
  geom_hline(color="red", yintercept=ci.cutoff) +
  lims(y=maxloglik - c(5, 0))
```

## Visualizing profile likelihood IX



## Visualizing profile likelihood X

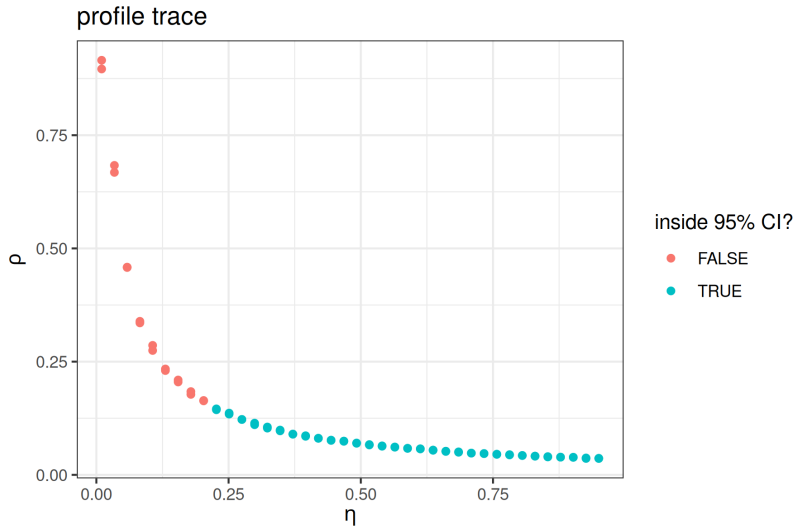
- ▶ As one varies  $\eta$  across the profile, the model compensates by adjusting the other parameters.
- ▶ It can be very instructive to understand how the model does this.
- ▶ For example, how does the reporting efficiency,  $\rho$ , change as  $\eta$  is varied?
- ▶ We can plot  $\rho$  vs  $\eta$  across the profile.
- ▶ This is called a *profile trace*.

## Visualizing profile likelihood XI

```
results |>
  filter(is.finite(loglik)) |>
  group_by(round(Eta,5)) |>
  filter(rank(-loglik)<3) |>
  ungroup() |>
  mutate(in_ci=loglik>max(loglik)-1.92) |>
  ggplot(aes(x=Eta,y=Rho,color=in_ci))+
  geom_point()+
  labs(
    color="inside 95% CI?",
    x=expression(eta),
    y=expression(rho),
    title="profile trace"
  )
```



## Visualizing profile likelihood XII



## Profile over $\rho$

While the above profile trace is suggestive that the 95% CI for  $\rho$  must be between roughly 4% and 10%, to confirm this, we should construct a proper profile likelihood over  $\rho$ . We do so now.

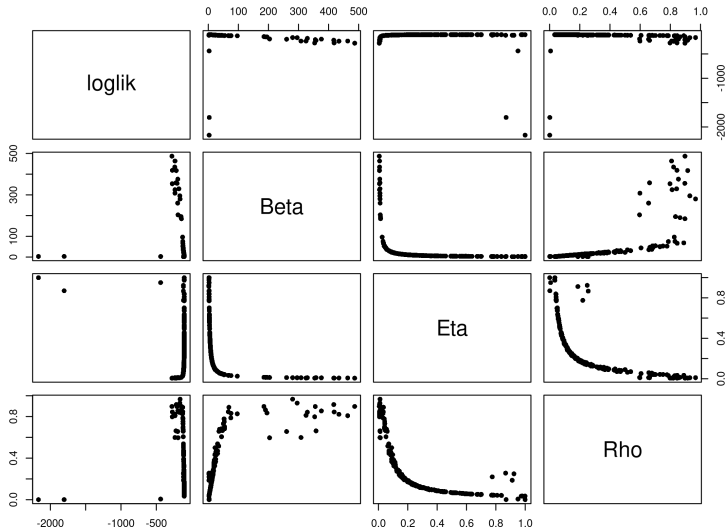
This time, we will initialize the IF2 computations at points we have already established have high likelihoods.

```
read_csv("measles_params.csv") |>
  group_by(cut=round(Rho,2)) |>
  filter(rank(-loglik)<=10) |>
  ungroup() |>
  arrange(-loglik) |>
  select(-cut,-loglik,-loglik.se) -> guesses
```

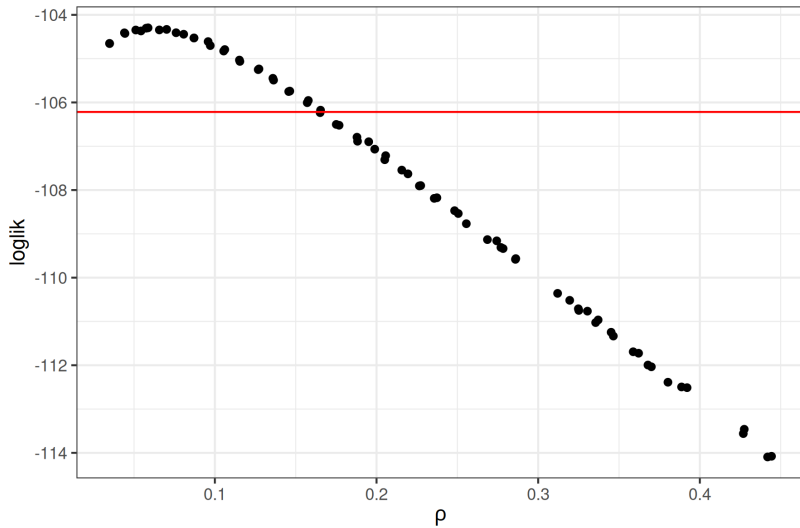
## Profile over $\rho$ II

```
registerDoRNG(210568)
foreach(guess=iter(guesses,"row"), .combine=rbind) %dopar% {
  mf1 |>
    mif2(params=guess,
          rw.sd=rw_sd(Beta=0.02,Eta=ivp(0.02))) |>
    mif2(Nmif=100,cooling.fraction.50=0.3) |>
    mif2() -> mf
  replicate(
    10,
    mf |> pfilter(Np=5000) |> logLik()) |>
    logmeanexp(se=TRUE) -> ll
  mf |> coef() |> bind_rows() |>
    bind_cols(loglik=ll[1],loglik.se=ll[2])
} -> results
```

## Profile over $\rho$ : results I



## Profile over $\rho$ : results II



## Profile over $\rho$ : results III

```
results |>  
  filter(loglik>max(loglik)-0.5*qchisq(df=1,p=0.95)) |>  
  summarize(min=min(Rho),max=max(Rho)) -> rho_ci
```

The data appear to be consistent with reporting efficiencies in the 3.4–17% range (95% CI).

The investigation continues....

## Parameter estimates as model predictions I

- ▶ The estimated parameters are one kind of model prediction.
- ▶ When we can estimate parameters using other data, we can test these predictions.
- ▶ In the case of a highly contagious, immunizing childhood infection such as measles, we can obtain an estimate of the reporting efficiency,  $\rho$  by simply regressing cumulative cases on cumulative births (Anderson and May 1991) over many years.
- ▶ When we do this for Consett, we see that the reporting efficiency is roughly 60%.
- ▶ Since such a value makes the outbreak data quite unlikely, the prediction does not appear to be borne out.
- ▶ We can conclude that one or more of our model assumptions is inconsistent with the data.
- ▶ Let's revisit our assumption that the infectious period is known to be 0.5~wk.
- ▶ Indeed, it would not be surprising were we to find that the *effective* infectious period, at the population scale, were somewhat shorter than the *clinical* infectious period.



## Parameter estimates as model predictions II

- ▶ For example, confinement of patients should reduce contact rates, and might therefore curtail the effective infectious period.
- ▶ To investigate this, we'll relax our assumption about the value of  $\gamma$ .

## Another global search I

We will estimate the model under the assumption that  $\rho = 0.6$ , but without making assumptions about the duration of the infectious period. As before, we'll construct a random design of starting parameters.

```
freeze(seed=55266255,  
  runif_design(  
    lower=c(Beta=5, Gamma=0.2, Eta=0),  
    upper=c(Beta=80, Gamma=5, Eta=0.99),  
    nseq=1000  
  )) |>  
mutate(  
  Rho=0.6, k=10, N=38000  
) -> guesses
```

## Another global search II

- ▶ For each of these starting points, we'll run a series of IF2 computations.
- ▶ Since we have gained some experience applying `mif2` to this model and these data, we have some expectation about how much computation is required.
- ▶ In the following, we'll use a lot more computational power than we have so far.

## Another global search III

For each of the starting points, we'll first perform 100 IF2 iterations:

```
measSIR |>  
  mif2(params=guess, Np=2000, Nmif=100,  
        cooling.fraction.50=0.5,  
        rw.sd=rw_sd(Beta=0.02, Gamma=0.02, Eta=ivp(0.02))) -> mf
```

We use random perturbations of the same magnitude as before, taking care to transform the parameters we are estimating.

## Another global search IV

We adopt a *simulated tempering* approach (following a metallurgical analogy), in which we increase the size of the random perturbations some amount (i.e., “reheat”), and then continue cooling.

```
mf |>
  mif2(
    Nmif=100, rw.sd=rw_sd(Beta=0.01, Gamma=0.01, Eta=ivp(0.01))
  ) |>
  mif2(
    Nmif=100,
    rw.sd=rw_sd(Beta=0.005, Gamma=0.005, Eta=ivp(0.005))
  ) -> mf
```

## Another global search V

We wrap the above in a `foreach` loop as before and take care to evaluate the likelihood at each end-point using `pfilter`.

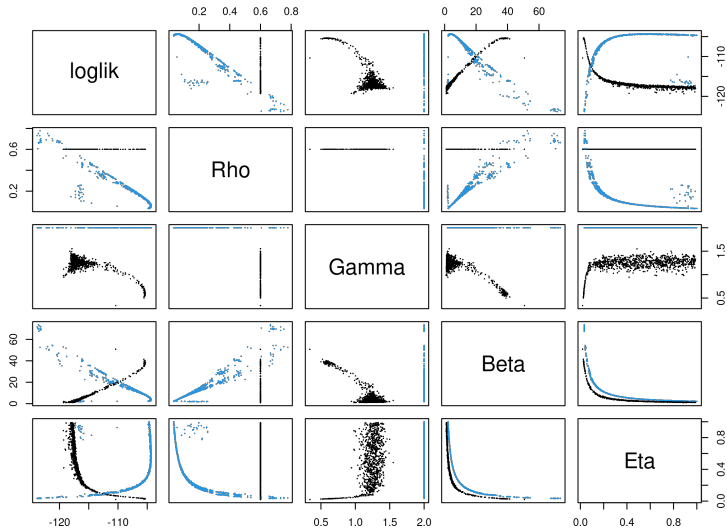
See the R code for this lesson to see exactly how this is done.

The computations above required 62.8 minutes on 24 processors.

```
read_csv("measles_params.csv") |>
  filter(loglik>max(loglik)-20) -> all

pairs(~loglik+Rho+Gamma+Beta+Eta,data=all,pch=16,cex=0.3,
      col=if_else(round(all$Rho,3)==0.6,1,4))
```

## Another global search VI

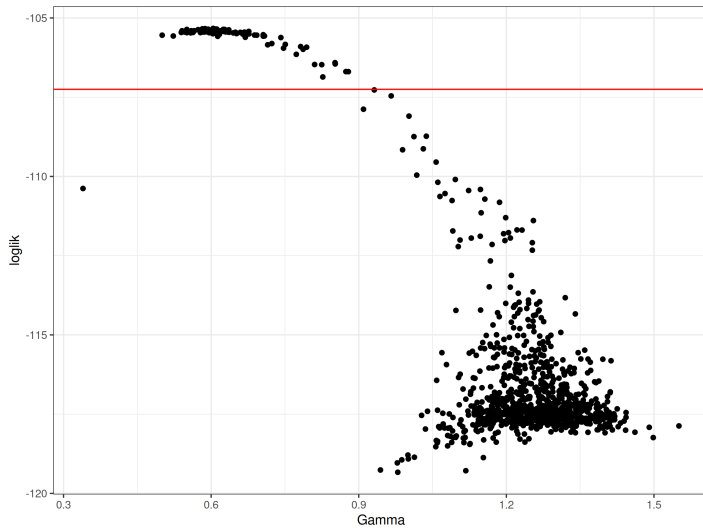


## Another global search VII

```
results |>
  filter(loglik>max(loglik)-20,loglik.se<1) |>
  ggplot(aes(x=Gamma,y=loglik))+
  geom_point()+
  geom_hline(
    color="red",
    yintercept=max(results$loglik)-0.5*qchisq(df=1,p=0.95)
  )
```



## Another global search VIII



## Profile over infectious period I

To make inferences about  $\gamma$ , we can again compute a profile likelihood. As before, we bound the region we will search:

```
read_csv("measles_params.csv") |>
  filter(
    loglik > max(loglik) - 20,
    loglik.se < 2,
    abs(Rho - 0.6) < 0.01
  ) |>
  sapply(range) -> box
```

## Profile over infectious period II

```
freeze(seed=610408798,  
  profile_design(  
    Gamma=seq(0.2,2,by=0.1),  
    lower=box[1,c("Beta","Eta")],  
    upper=box[2,c("Beta","Eta")],  
    nprof=100, type="runif"  
  )) |>  
mutate(  
  N=38000,  
  Rho=0.6,  
  k=10  
) -> guesses
```

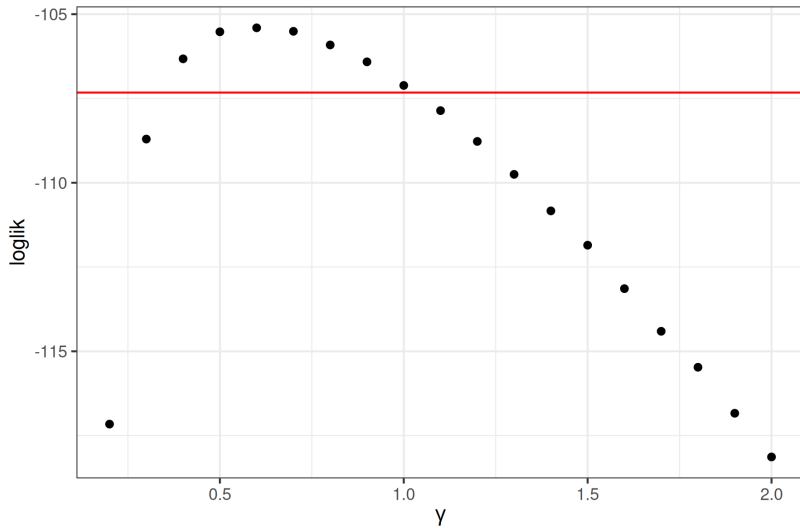
## Profile over infectious period III

```
foreach(guess=iter(guesses,"row"), .combine=rbind
) %dopar% {
  measSIR |>
    mif2(params=guess, Np=2000, Nmif=100, cooling.fraction.50=0.5,
        rw.sd=rw_sd(Beta=0.02,Eta=ivp(0.02))
    ) |>
    mif2(Nmif=100) |>
    mif2(Nmif=100,rw.sd=rw_sd(Beta=0.01,Eta=ivp(0.01))) |>
    mif2(Nmif=100,rw.sd=rw_sd(Beta=0.005,Eta=ivp(0.005))) -> mf
  replicate(10,mf |> pfilter(Np=5000) |> logLik()) |>
    logmeanexp(se=TRUE) -> ll
  mf |> coef() |> bind_rows() |>
    bind_cols(loglik=ll[1],loglik.se=ll[2])
} -> results
```

## Infectious period profile I

```
results |>
  group_by(round(Gamma,2)) |>
  filter(rank(-loglik)<=1) |>
  ungroup() |>
  ggplot(aes(x=Gamma,y=loglik))+
  geom_point() + xlab(expression(gamma)) +
  geom_hline(
    color="red",
    yintercept=max(results$loglik)-0.5*qchisq(df=1,p=0.95)
  )
```

## Infectious period profile II



## Infectious period profile III

- ▶ This suggests that  $\rho = 0.6$  is consistent only with smaller values of  $\gamma$ , and hence *longer* infectious periods than are possible if the duration of shedding is actually less than one week.
- ▶ Thus the model is incapable of reconciling both an infectious period of less than one week and a reporting rate of 60%.
- ▶ *What structural changes to the model might we make to improve its ability to explain the data?*

## Visualizing predictions I

After all these analyses, we would like to visualize how exactly the model with the MLEs matches the data. We can do it by plotting the simulations with 95% the prediction interval.

```
read_csv("measles_params.csv") |>
  filter(loglik == max(loglik)) |>
  select(-loglik, -loglik.se) -> best.params

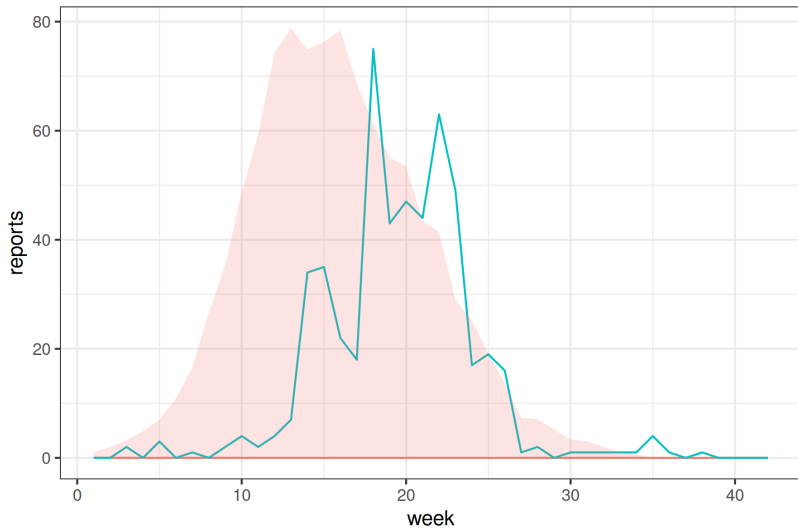
measSIR |>
  simulate(
    params=unlist(best.params),
    nsim=1000, format="data.frame", include.data=TRUE
  ) -> sims
```



## Visualizing predictions II

```
sims |>
  mutate(data=.id=="data") |>
  group_by(week,data) |>
  reframe(
    p=c(0.025,0.5,0.975),
    value=wquant(reports,probs=p),
    name=c("lo","med","up")
  ) |>
  select(-p) |> pivot_wider() |> ungroup() |>
  ggplot(aes(x=week,y=med,color=data,fill=data,ymin=lo,ymax=up))+
  geom_line()+ geom_ribbon(alpha=0.2,color=NA) +
  labs(y="reports")+
  theme_bw() + guides(color="none",fill="none")
```

## Visualizing predictions III



## Exercises

There are three main exercises for trying to use `mif2` to fit a model. The fourth exercise is included to give you some practice at understanding and debugging POMP models.

- ▶ Exercise 1: Fitting SEIR Model.
- ▶ Exercise 2: Fitting all parameters of SIR model (e.g.,  $k$ ,  $\gamma$ ,  $\eta$ )
- ▶ Exercise 3: Construct a profile.

More details below.

## Exercise: Fitting the SEIR model I

In this exercise, you will estimate the parameters and likelihood of the SEIR model you implemented in the earlier lessons by following the template above. Purely for the sake of simplicity, you may assume that the values of  $\gamma$  and  $k$  are known.

1. First, conduct a local search and compute the likelihood at the end of each `mif2` run. Use only as many parallel `mif2` computations as you have processors on your computer (or perhaps somewhat fewer). Track the time used and compute the amount of time used per cpu per IF2 iteration per 1000 particles. (Recall that one particle filter computation is roughly equal to a IF2 iteration in computational complexity if they both use the same number of particles.)
2. Using the expense estimates you generated in `Step~(@ref(it:zero))`, choose a number of IF2 iterations so that you can do a very crude “global search” that will complete in two or three minutes. Do not reduce `Np` drastically, as we don't want to degrade the performance of the individual IF2 computations. Run your global search with these settings. This serves to debug your global search code.

## Exercise: Fitting the SEIR model II

3. Now we want a computation that gives us some results we can begin to interpret, but that is still as quick as possible. Choose `Nmif` and the number of random starts so that you can obtain the beginnings of a global search of the parameter space in one hour or less. Run your global search with these settings and plot the results.
4. (At Home) Now we want to increase computing for final or near-final results. You may want to tune your settings (`Nmif`, `Np`, `rw.sd`, `cooling.fraction.50`) at this point, based on what you found at run-level 2. Decide how much time in the next 18 hours is available to you for a computation. Choose the number of starting guesses so that you can obtain as thorough a global search as possible within this period. Run your global search and identify a maximum likelihood estimate.
5. How does the SEIR model compare with the SIR model? Discuss the overall quality of the fit as well as important differences in how the two models are explaining the data.

## Exercise: Fitting all parameters of the SIR model

In all of the foregoing, we have assumed a fixed value of the dispersion parameter,  $k$ , of the negative binomial measurement model. We've also fixed one or the other of  $\gamma$ ,  $\eta$ . Now attempt to estimate all the parameters simultaneously. To accomplish this, use the same system of run-levels as in the previous Exercise. How much is the fit improved? How has the model's explanation of the data changed?

## Exercise: Construct a profile likelihood

How strong is the evidence about the contact rate,  $\beta$ , given this model and data? Use `mif2` to construct a profile likelihood. Due to time constraints, you may be able to compute only a preliminary version.

It is also possible to profile over the basic reproduction number,  $R_0 = \beta/\gamma$ . Is this more or less well determined than  $\beta$  for this model and data?

## Exercise: Checking the source code I

Check the source code for the `measSIR` pomp object, using the `spy` command.

1. Does the code implement the model described?

It can be surprisingly hard to make sure that the written equations and the code are perfectly matched. Papers should be written to be readable, and therefore people rarely choose to clutter papers with numerical details which they hope and believe are scientifically irrelevant.


2. What problems can arise due to the conflict between readability and reproducibility?
3. What solutions are available?



## References I

Anderson, R. M., and R. M. May. 1991. *Infectious Diseases of Humans*. Oxford: Oxford University Press.

## License, acknowledgments, and links

- ▶ This lesson is prepared for the Simulation-based Inference for Epidemiological Dynamics module at the Summer Institute in Statistics and Modeling in Infectious Diseases, SIS MID.
- ▶ The materials build on previous versions of this course and related courses.
- ▶ Licensed under the Creative Commons Attribution-NonCommercial license. Please share and remix non-commercially, mentioning its origin. 
- ▶ Produced with R version 4.5.0 and pomp version 6.3.
- ▶ Compiled on 2025-07-21.

[Back to Lesson](#)

[R code for this lesson](#)