Simulated Data Work

Jonathan Waring

July 15, 2016

Simulated Data Work for REU Summer 2016 Project

Project Description

During an outbreak of an emerging infection, there are many epidemiologically important quantities that we need to estimate in order to get a better picture of how severe the epidemic may turn out to be. These include quantities such as the mean infectious period and the transmission potential of the pathogen. Fitting transmission models to incidence reports has become a standard way of achieving quick estimates of these parameters. Too often, practitioners use cumulative data (total number of infections to date) rather than raw incidence (number of new cases in a defined reporting period). There is evidence to suggest this choice can severely affect our perception of the variability in parameters and hence the certainty in predictions. This project is focused on further elaborating on this problem using simulated epidemic data. The aim is to use both deterministic and stochastic models to fit to both raw and cumulative data in order to systematically assess the likely biases and errors that result from the choice of data.

POMP Model

pomp [1] is a package in the R progamming language that encodes a partially-observed Makrov process model with a time series. One implements the model by specififfying different components written as R functions or C code snippets. The code below demonstrates how the epidemic time-series data used in this project was created (Simulator.R), trajectory matched, and filtered (IF2 algorithm). The pomp model described below is used in most of the scripts to describe the data.

The continuous-time process model for raw (simulation of state process given parameters) was specified as:

```
#Continuous-time process model (rprocess)
step.fun<- Csnippet("
                    double dQ = rgammawn(beta_sd,dt);
                    double lambda = Beta*I/N*dQ/dt;
                    double births = rpois(mu*N);

                    S += (births - lambda*S - mu*S)*dt;
                    I += (lambda*S - gamma*I - mu*I)*dt;
                    RealCases += gamma*I*dt;
                    ")
```

The continuous-time process model for cumulative (simulation of state process given parameters) was specified as:

```
#Continuous-time process model (rprocess)
step.funCum<- Csnippet("
                    double dQ = rgammawn(beta_sd,dt);
                    double lambda = Beta*I/N*dQ/dt;
                    double births = rpois(mu*N);
```

```
                    S += (births - lambda*S - mu*S)*dt;
                    I += (lambda*S - gamma*I - mu*I)*dt;
                    RealCases += gamma*I*dt;
                    CumulativeCases += (RealCases - (RealCases * gamma))*dt;
                    ")
```

The raw model was initialized as follows:

```
#Initializer
init <- Csnippet("
                  S = N-1;
                  I= 1;
                  RealCases = 1;
                  ")
```

The cumulative model was initialized as follows:

```
#Initializer
initCum <- Csnippet("
                  S = N-1;
                  I= 1;
                  RealCases = 1;
                  CumulativeCases = 1;
                  ")
```

The measurement process was modeled as the following rmeasure (simulation of observation process given state and parameters) and dmeasure (evaluation of the likelihood of a set of observations given the states and parameters):

```
#Rmeasure model
rmeas <- Csnippet("
                  Cases = rpois(RealCases * rho);
                  ")

#Dmeasure model
dmeas <- Csnippet("
                  lik = dpois(Cases, RealCases * rho, give_log);
                  ")
```

The raw deterministic skeleton as a point in state space, given parameters was evaluated by:

```
#Deterministic model skeleton
skel <- Csnippet("
                  double lambda = Beta*I/N;
                  double births = mu*N;

                  DS = births - lambda*S -mu*S;
                  DI = lambda*S -gamma*I -mu*I;
                  DRealCases = gamma*I;
                  ")
```

The cumulative deterministic skeleton as a point in state space, given parameters was evaluated by:

```
#Deterministic model skeleton
skelCum <- Csnippet("
                  double lambda = Beta*I/N;
                  double births = mu*N;

                  DS = births - lambda*S -mu*S;
                  DI = lambda*S -gamma*I -mu*I;
                  DRealCases = gamma*I;
                  DCumulativeCases = (RealCases - (RealCases * gamma));
                  ")
```

Lastly, the paramters were transformed using:

```
#Paramter transformations
logtrans <- Csnippet("
                     Tbeta_sd = log(beta_sd);
                     TBeta = log(Beta);
                     Tmu = log(mu);
                     Tgamma = log(gamma);
                     Trho = log(rho);
                     TN = log(N);
                     ")

exptrans <- Csnippet("
                     Tbeta_sd = exp(beta_sd);
                     TBeta = exp(Beta);
                     Tmu = exp(mu);
                     Tgamma = exp(gamma);
                     Trho = exp(rho);
                     TN = exp(N);
                     ")
```

To construct the raw pomp model, you use the constructor like such:

```
sampleData <- read.csv(file ="sample_data.csv")
sampleData <- subset(sampleData, select=c("time", "Cases"))
rawModel <- pomp(sampleData, time="time", t0=0,
               rprocess=euler.sim(step.fun, delta.t=0.1), initializer = init,
               statenames=c("S", "I", "RealCases"), paramnames=c("Beta", "gamma", "mu",
               "rho", "N","beta_sd"), obsnames = "Cases", zeronames = "RealCases",
               rmeasure = rmeas, dmeasure = dmeas, skeleton = vectorfield(skel),
               toEstimationScale=logtrans, fromEstimationScale = exptrans)
```

To construct the cumulative pomp model, you use the constructor like such:

```
sampleData <- read.csv(file ="sample_data.csv")
gamma <- 0.6
cumulativeSampleData <- sampleData
actualCumulative <- trunc(cumsum(cumulativeSampleData$Cases-(gamma*cumulativeSampleData$Cases)))
cumulativeSampleData <- cbind(cumulativeSampleData,actualCumulative)
cumulativeSampleData <- subset(cumulativeSampleData, select=c("time", "Cases", "actualCumulative"))
cumModel <- pomp(cumulativeSampleData, time="time", t0=0,
```

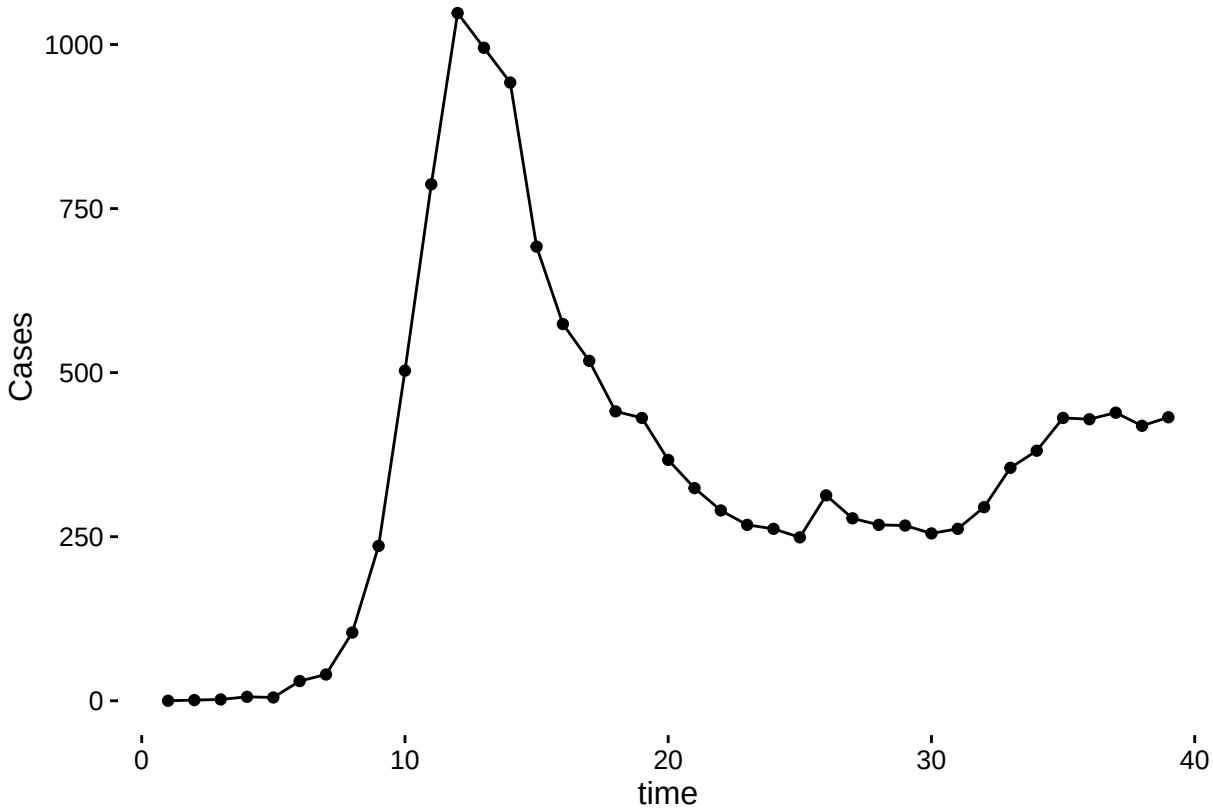
```
              rprocess=euler.sim(step.funCum, delta.t=0.1), initializer = initCum,
              statenames=c("S", "I", "RealCases", "CumulativeCases"),
              paramnames=c("Beta", "gamma", "mu", "rho", "N","beta_sd"),
              obsnames = "Cases", zeronames = "RealCases",
              rmeasure = rmeas, dmeasure = dmeas, skeleton = vectorfield(skelCum),
              toEstimationScale=logtrans, fromEstimationScale = exptrans)
```

Simulator

25 simulated epidemic time-series data sets were generated using the pomp model above. Parameters for the model were set to the following levels: beta_sd = 0.1, Beta=1.5, mu=0.1, gamma=0.6, rho=0.8, N=10000. The example code is below.

```
S.0 <- 1e4-1
I.0 <- 1
N <- S.0+I.0
newSimulation <- simulate(rawModel,nsim=1,params=c(beta_sd= 0.1, Beta = 1.5, mu=0.1, gamma=0.6,
                                                   rho= 0.8, N=N, S.0= S.0, I.0 = I.0,
                                                   RealCases.0 = 1),as.data.frame=T)
```

This will produce something data that looks similar to this:



The Simulator.R script will generate a list of allSims, which contains 24 simulated data frames for epidemic data generated from the pomp model. It will be saved as a file called: simulations.Rda. The first simulation is in a file called sample_data.csv.

Deterministic Likelihood Estimation

We can use the function dmeasure to evaluate the log likelihood of the data given the states, the model, and the parameters. To get a maximum likelihood estimate of two important parameter values, beta and gamma, I've constructed two vectors (betaVec and gammaVec) that contain 100 estimates of the parameter values. Then I ran 500 simulations of the pomp model at the 10,000 possible combination of parameter values and got the maximum likelihood estimate from the set of 500 simulations. These are then stored in a 100X100 matrix that is plotted as a contour map to show where the maximum likelihood estimates tend to aggregate at possible parameter values. The code for this is shown below.

Raw Contour Map

```
rawBetaVector <- seq(from=1.0, to=2.0, by=((2.0-1.0)/99))
rawGammaVector <- seq(from=0.25, to=1.0, by=((1.0-.25)/99))
rawLogLiks <- matrix(nrow = 100, ncol = 100)

#Filling logLiks matrix at different Beta and Gamma values
if(file.exists("RawLogLiks.Rda")) {
  load("RawLogLiks.Rda")
} else {
  for(i in 1:100) {
    for(j in 1:100) {
      sims <- simulate(rawModel,params=c(beta_sd=0.1, Beta=rawBetaVector[i], mu=0.1, gamma=rawGammaVecto
                       nsim=500, states=TRUE)
      ll <- dmeasure(model, y=obs(rawModel), x=sims, times=time(rawModel), log=TRUE,
                     params=c(beta_sd=0.1, Beta=rawBetaVector[i], mu=0.1, gamma=rawGammaVector[j], rho=
      ll <- apply(ll, 1, sum)
      ll <- max(ll)
      rawLogLiks[i,j] <- ll
    }
  }
  save(rawLogLiks, file="RawLogLiks.Rda")
}

#Plotting logLiks matrix as a Contour Map
my_palette <- colorRampPalette(c("red", "yellow", "green", "blue", "purple"))(n = 1999)
rawLogLikliehood <- matrix(nrow=100, ncol=100)
for(i in 1:100) {
  for(j in 1:100) {
    if(logLiks[i,j] >= -1000) {
      rawLogLikliehood[i,j] = logLiks[i,j]
    }else{
      rawLogLikliehood[i,j] = -1000
    }
  }
}
x <- list(
  title = "Beta",
  autotick = FALSE,
  ticks = "outside",
  tick0 = 0,
  dtick = 0.25
```

```
)
y <- list(
  title = "Gamma",
  autotick = FALSE,
  ticks = "outside",
  tick0 = 0,
  dtick = 0.25
)
plot_ly(x=rawBetaVector, y=rawGammaVector, z=rawLogLikliehood, type="contour", colors=my_palette) %>%
  layout(xaxis=x, yaxis=y)
```

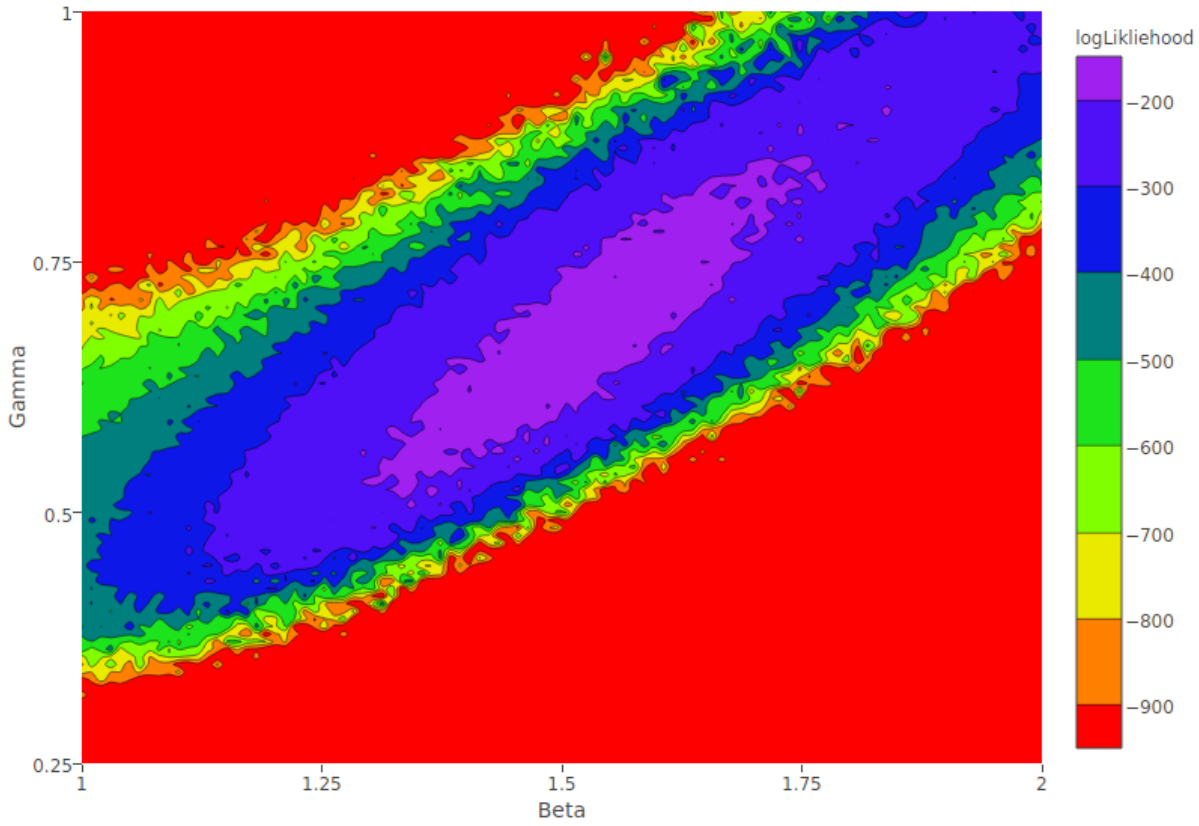


Figure 1:

Cumulative Contour Map

```
cumBetaVector <- seq(from=1.0, to=2.0, by=((2.0-1.0)/99))
cumGammaVector <- seq(from=0.25, to=1.0, by=((1.0-.25)/99))
cumLogLiks <- matrix(nrow = 100, ncol = 100)

#Filling logLiks matrix at different Beta and Gamma values
if(file.exists("CumLogLiks.Rda")) {
  load("CumLogLiks.Rda")
```

```
} else {
  for(i in 1:100) {
    for(j in 1:100) {
      sims <- simulate(cumModel,params=c(beta_sd=0.1, Beta=cumBetaVector[i], mu=0.1, gamma=cumGammaVecto
                       nsim=500, states=TRUE)
      ll <- dmeasure(model, y=obs(cumModel), x=sims, times=time(cumModel), log=TRUE,
                     params=c(beta_sd=0.1, Beta=cumBetaVector[i], mu=0.1, gamma=cumGammaVector[j], rho=
      ll <- apply(ll, 1, sum)
      ll <- max(ll)
      cumLogLiks[i,j] <- ll
    }
  }
  save(cumLogLiks, file="RawLogLiks.Rda")
}

#Plotting logLiks matrix as a Contour Map
my_palette <- colorRampPalette(c("red", "yellow", "green", "blue", "purple"))(n = 1999)
cumLogLikliehood <- matrix(nrow=100, ncol=100)
for(i in 1:100) {
  for(j in 1:100) {
    if(logLiks[i,j] >= -1000) {
      cumLogLikliehood[i,j] = logLiks[i,j]
    }else{
      cumLogLikliehood[i,j] = -1000
    }
  }
}
x <- list(
  title = "Beta",
  autotick = FALSE,
  ticks = "outside",
  tick0 = 0,
  dtick = 0.25
)
y <- list(
  title = "Gamma",
  autotick = FALSE,
  ticks = "outside",
  tick0 = 0,
  dtick = 0.25
)
plot_ly(x=cumBetaVector, y=cumGammaVector, z=cumLogLikliehood, type="contour", colors=my_palette) %>%
  layout(xaxis=x, yaxis=y)
```

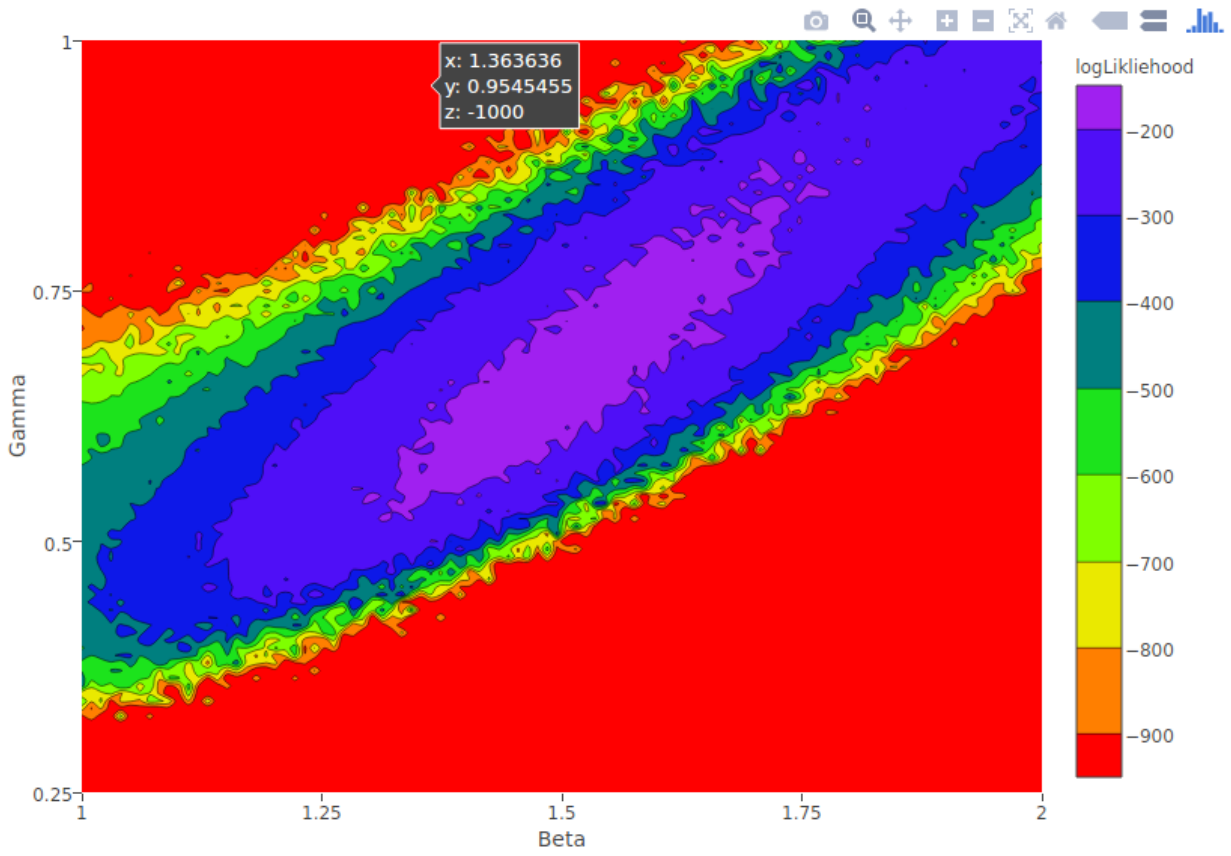


Figure 2:

The raw and cumulative contour maps appear very similar and are hard to compare. Thus it is useful to plot beta and gamma as functions of log likelihood and see which data set (raw or cumulative) provides wider confidence intervals, and thus better quantifies uncertainity.

Raw Confidence Interval Mappings

TODO

Cumulative Confidence Interval Mappings

TODO

Trajectory Matching

Trajectory matching is the method of fitting a deterministic model to data assuming independent errors. In pomp, the function traj.match searches parameter space to find parameters under which the likelihood of the data, given a trajectory of the deterministic skeleton, is maximized.

For each of the 25 simulations, the RawSimDataTrajMatch.R and CumSimDataTrajMatch.R scripts generates 38 trajectory matches for the 25 simulations. The 38 different trajectory matches each uses a different amount of time points, in order to see if the amount of points used changes which data choice (raw or cumulative) is more effective.

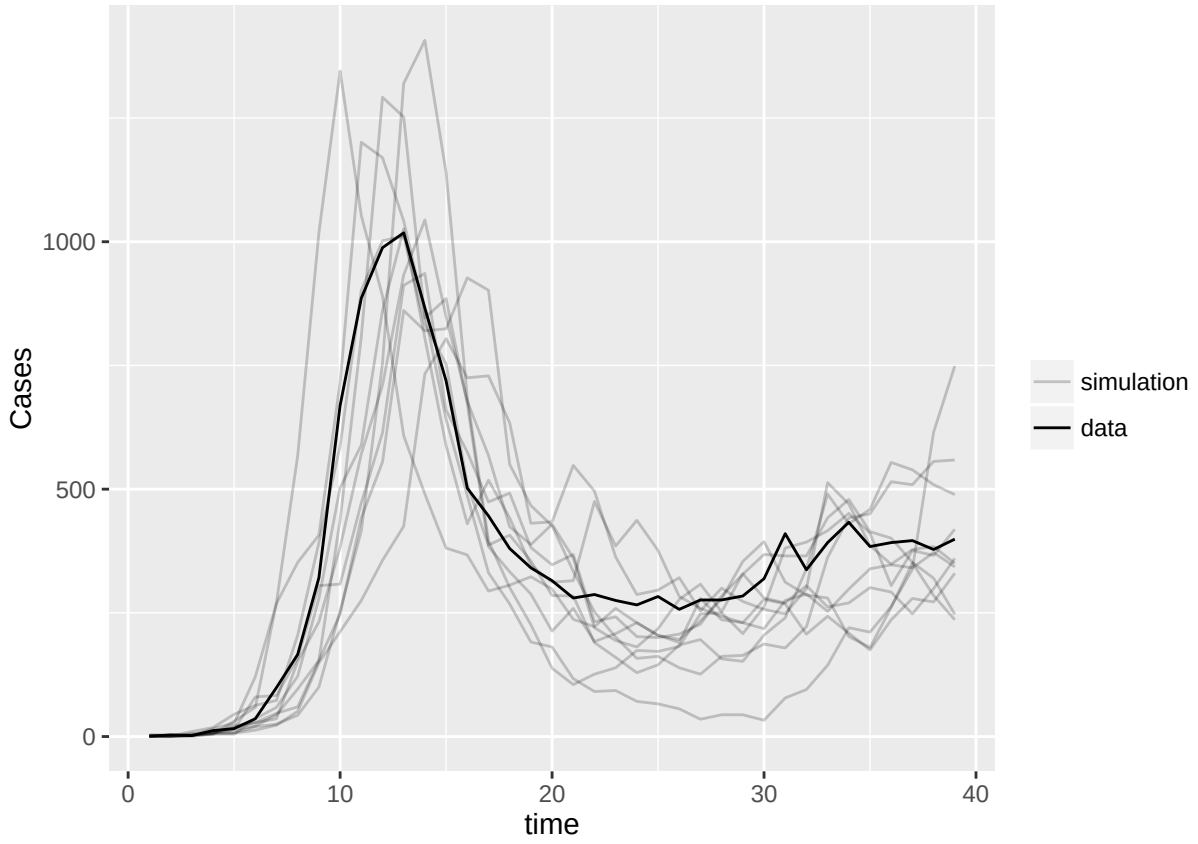
The results are saved in allCumTrajMatches.Rda and allRawTrajMatches.Rda files as ggplots. These scripts can be edited to save them as something other than the plots.

Using the above pomp model, you can do a raw trajectory match with the following code.

```
tm <- traj.match(rawModel, start=c(beta_sd=0.12, Beta=1.8, mu=0.15, gamma=0.65, rho=0.75, N=10000),
                      est=c("beta_sd", "Beta", "mu", "gamma", "rho", "N"), transform = TRUE)
```

Then you can run simulations to see how the fitted model compares to the data.

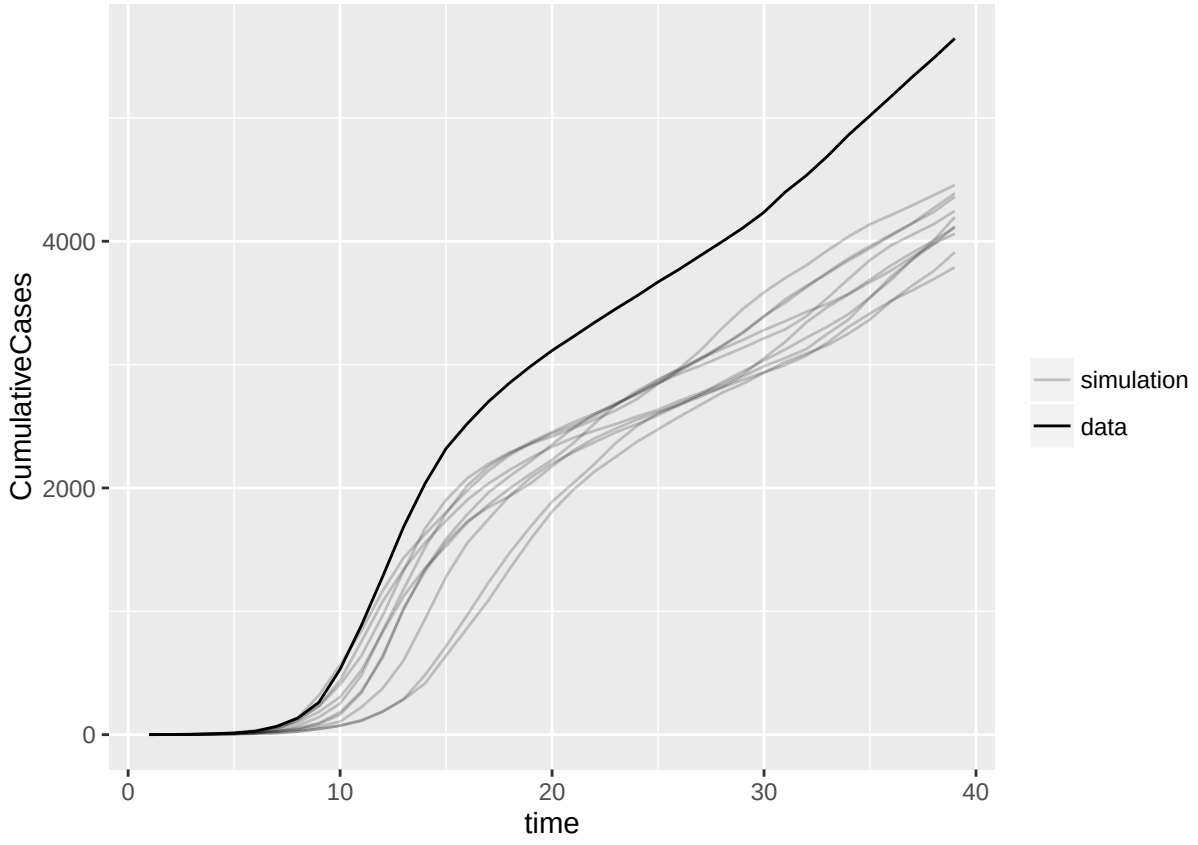
```
p <- simulate(tm,nsim=10,as.data.frame=TRUE,include.data=TRUE)
p <- subset(p, select=c("time", "Cases", "sim"))
p <- subset(p, !is.na(p$Cases))
ggplot(data=p,aes(x=time,y=Cases,group=sim,alpha=(sim=="data")))+
  scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                     labels=c(`FALSE`="simulation",`TRUE`="data"))+
  geom_line()
```



Cumulative trajectory matching can be done in a similar manner, as shown here:

```
cumulativeTM <- traj.match(cumModel, start=c(beta_sd=0.12, Beta=1.8, mu=0.15, gamma=0.65, rho=0.75, N=1
                      est=c("beta_sd", "Beta", "mu", "gamma", "rho", "N"), transform = TRUE)
p <- simulate(cumulativeTM,nsim=10,as.data.frame=TRUE,include.data=TRUE)
p$CumulativeCases[1:39] <- p$actualCumulative[1:39]
p <- subset(p, select=c("time", "CumulativeCases", "sim"))
p <- subset(p, !is.na(p$CumulativeCases))
ggplot(data=p,aes(x=time,y=CumulativeCases,group=sim,alpha=(sim=="data")))+
```

```
scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                   labels=c(`FALSE`="simulation",`TRUE`="data"))+
geom_line()
```



Traj Match Analysis

38 trajectory matches for Raw and Cumulative data was done for each of the 25 realizations using a different amount of time points (increment by 1 each time). All trajectory matches were done with starting estimates as follows: beta_sd=0.12, Beta=1.8, mu=0.15, gamma=0.65, rho=0.75, N=10000. Analysis of the trajectory matches was done in the Visuals.R script which plotted each of the 25 simulations together at the 38 different timestamps for both raw and cumulative data. An example is shown here with raw data and cumulative data with 23 points.

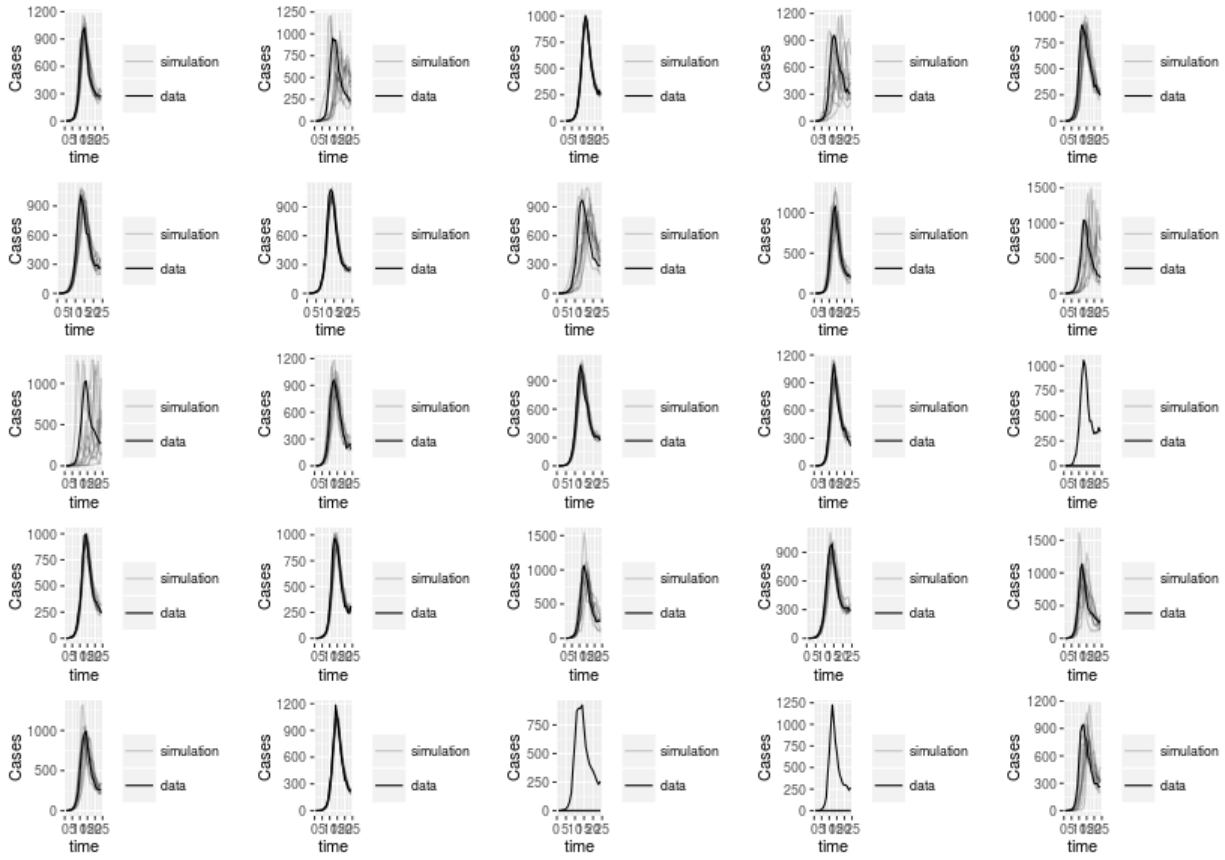


Figure 3: 25 Raw Simulated Epidemic trajectory matches at 23 timestamps

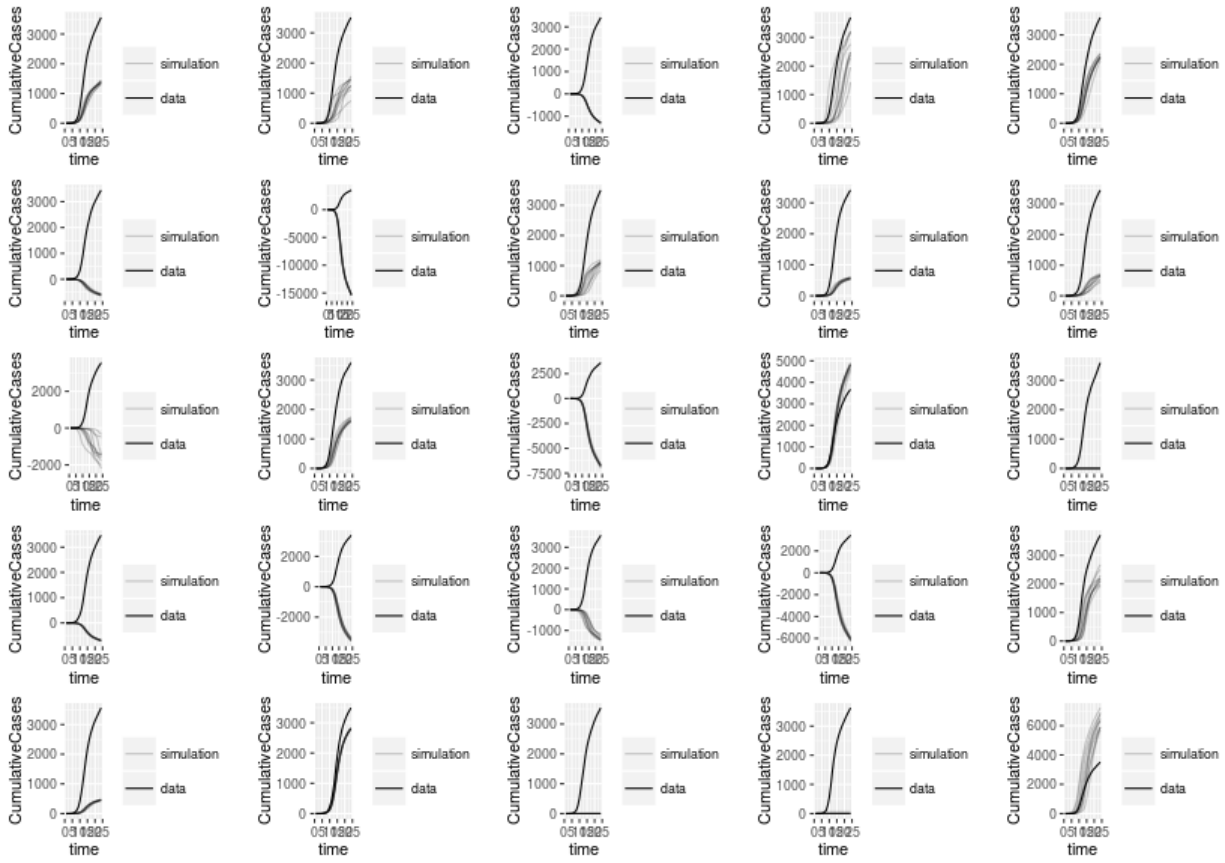


Figure 4: 25 Cumulative Simulated Epidemic trajectory matches at 23 timestamps

Trajectory matching showed similar results to that of the "Avoidable errors in the modelling of outbreaks of emerging pathogens, with special references to Ebola" paper [2]. Both the raw and cumulative models fail to fit to the data well until 6+ time series points are added. When fitting the raw and cumulative incidence data to a deterministic model, the cumulative incidence curve superficially appears to do better during the epidemic takeoff. However, this is known to not be a robust estimate, and the uncertainity is not accurately quantified. Fitting cumulative data given the assumptions of the statistical error model has been shown to be problematic in past modeling research [3]. When you reach the point where the epidemic begins to slow down, which would be the peak of the raw epidemic curve or the point of inflection of a cumulative incidence curve, the trend seems to switch. As more points are added to the data, the trajectory matching starts to fit better with raw incidence data. The cumulative data fits start to either underpredict or diverges from data completely. Raw fits are all very similar, whereas the cumulative incidence fits are much more variable. This would seem to suggest that the cumulative curves better quantify uncertainity than raw trajectory matching fits. However, raw incidence fits may still be more accurate themselves despite its lower variance. To see this, run the Visuals.R script. It seems that with trajectory matching, raw data still seems to be the appropriate choice for the whole course of an epidemic if you want a close fit, but cumulative data would be appropriate for better quantification of uncertanity. However, it should be noted that if cumulative data is used, that the predicted simulations are likely under predicting the "true" incidence.

Iterated Filtering

Another method for maximizing likelihood of parameters of a partially-observed Markov process is iterated filtering. The iterated filtering algorithm [4] runs a particle filter (sequential Monte Carlo algorithm) at each iteration on a perturbed version of the model, which effectively smooths the likelihood surface. Using our

stochastic model, the RawIteratedFiltering.R and CumIteratedFiltering.R scripts run the iterated filtering algorithm over one instance of the 25 simulated epidemic time-series data for both Raw and Cumulative data.

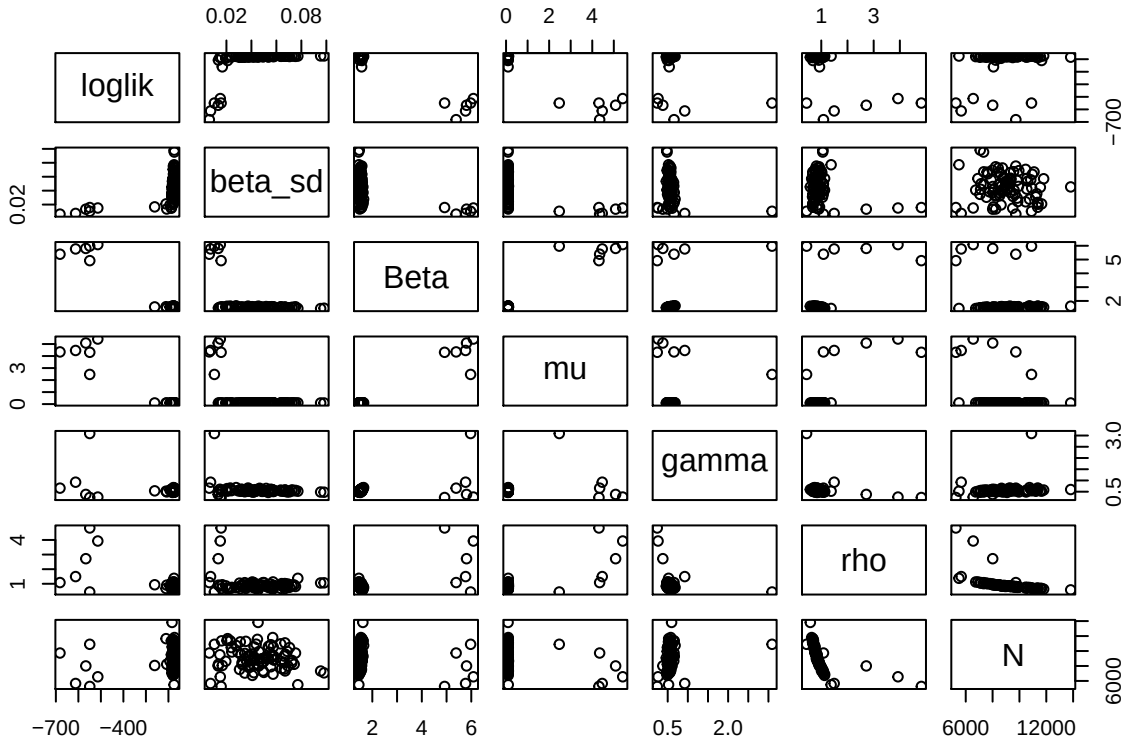
Raw Iterated Filtering

The raw iterated filtering algorithm is defined here:

```
registerDoParallel()
bake(file="RawFakeData-mif.rds", {
  guesses <- sobolDesign(lower=c(beta_sd=0.05, Beta=1.0, mu=0.01, gamma=0.1, rho=0.05, N=10000),
                         upper=c(beta_sd=0.5, Beta=3.0, mu=1.0, gamma=0.99, rho=0.99, N=10000),
                         nseq=100)
  foreach (guess=iter(guesses,"row"),.combine=rbind,
          .options.mpi=list(seed=334065675),
          .packages=c("pomp","magrittr"),.errorhandling="remove") %dopar% {
            rawModel %>%
              mif2(start=unlist(guess),Nmif=50,Np=1000,transform=TRUE,
                   cooling.fraction.50=0.8,cooling.type="geometric",
                   rw.sd=rw.sd(beta_sd=0.02, Beta=0.02, mu=0.02, gamma=0.02, rho=0.02, N=0.02)) %>%
              mif2() -> mf
            ll <- logmeanexp(replicate(5,logLik(pfilter(mf))),se=TRUE)
            data.frame(loglik=ll[1],loglik.se=ll[2],as.list(coef(mf)))
          }
}) -> mles
```

A useful way to view the data is with a pairwise scatterplot matrix:

```
pairs(~loglik+beta_sd+Beta+mu+gamma+rho+N,data=mles)
```

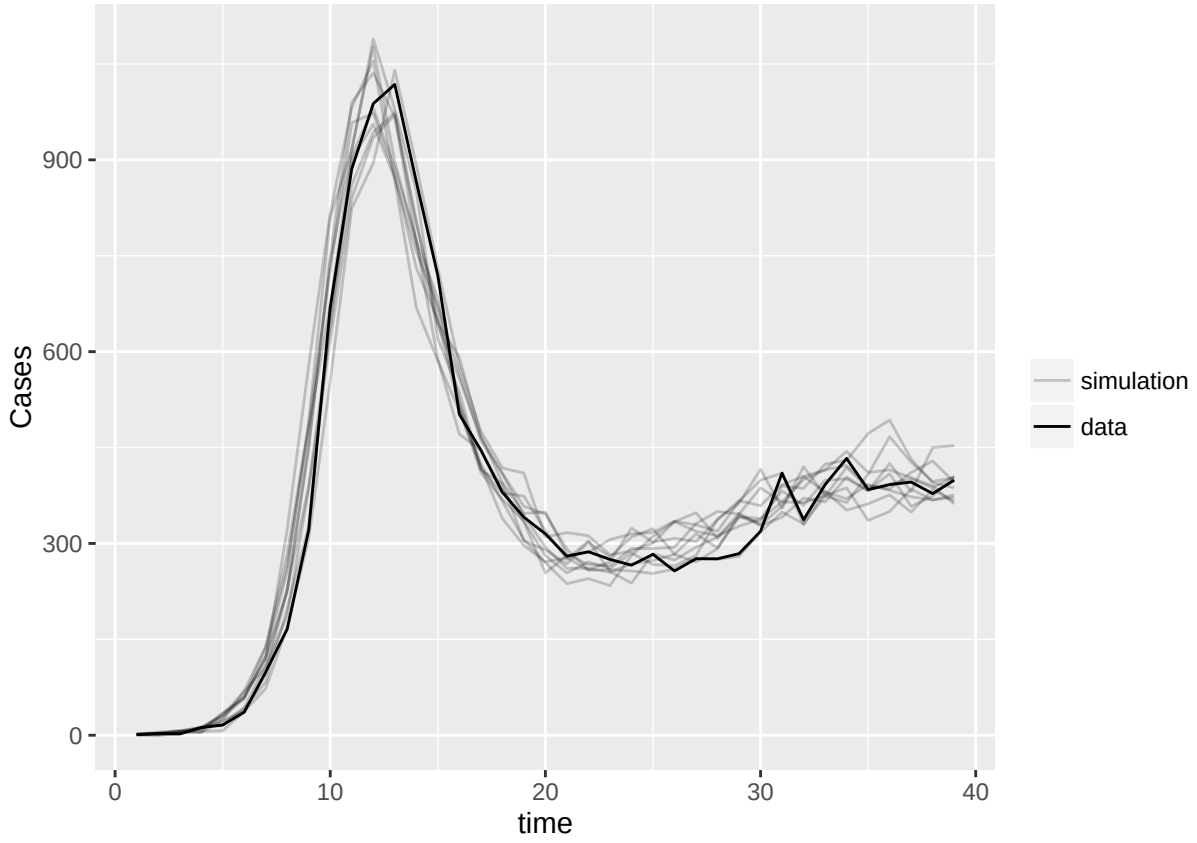


From looking at the matrix, there does not seem to be any linear correlations between the parameters in this model. Most of the paramters seem to gravitate near one location for the best log likelihood (except for beta_sd and N).

Iterated Filtering Fit to Data

The iterated filtering algorithm produces a maximum likelihood estimate for the parameter values. The fit to the data can be seen by doing the following:

```
mles %>%
  subset(loglik==max(loglik)) %>% unlist() -> mle
simulate(rawModel,params=mle,nsim=10,as.data.frame=TRUE,include.data=TRUE) %>%
  ggplot(mapping=aes(x=time,y=Cases,group=sim,alpha=(sim=="data")))+
  scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                     labels=c(`FALSE`="simulation",`TRUE`="data"))+
  geom_line()
```



Likelihood Profiles

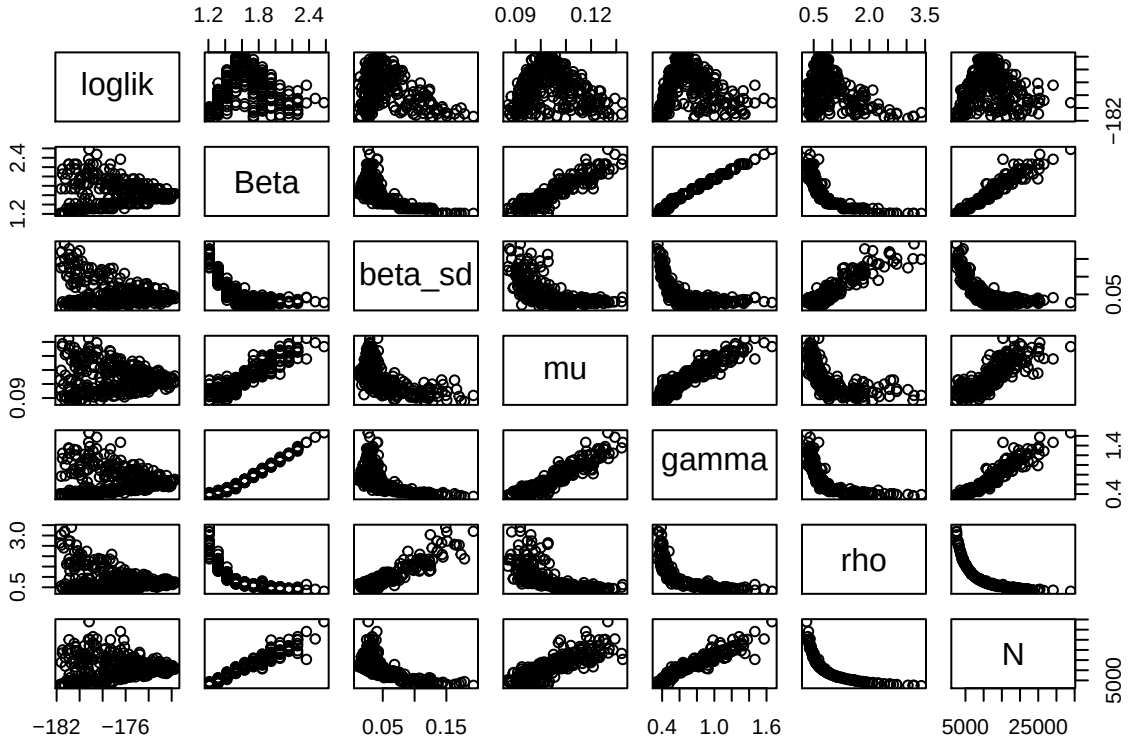
To get better estimates and obtain likelihood-ratio-test based confidence intervals, I've constructed profile likelihoods for two important parameters: beta (transmission term) and gamma (recovery rate). In a likelihood profile, one varies the focal parameter across some range, maximizing the likelihood at each value of the focal parameter over the remaining parameters.

Raw Beta Profile The beta profile is constructed below.

```
profileDesign(
  Beta=seq(from=1.0, to=3.0, length=20),
  lower=c(beta_sd=0.05, mu=0.01, gamma=0.1, rho=0.05, N=10000),
  upper=c(beta_sd=0.5, mu=1.0, gamma=0.99, rho=0.99, N=10000),
  nprof=50
) -> betaPD

bake("RawFakedata_Beta-profile.rds",{
  foreach (p=iter(betaPD,"row"),
           .combine=rbind,
           .errorhandling="remove",
           .packages=c("pomp","magrittr","reshape2","plyr"),
           .inorder=FALSE,
           .options.mpi=list(seed=1680158025)
  ) %dopar% {
    rawModel %>%
      mif2(start=unlist(p),Nmif=50,Np=1000,transform=TRUE,
           cooling.fraction.50=0.8,cooling.type="geometric",
           rw.sd=rw.sd(beta_sd=0.02, mu=0.02, gamma=0.02, rho=0.02, N=0.02)) %>%
      mif2() -> mf

    pf <- replicate(5,pfilter(mf,Np=1000))  ## independent particle filters
    ll <- sapply(pf,logLik)
    ll <- logmeanexp(ll,se=TRUE)
    nfail <- sapply(pf,getElement,"nfail")  ## number of filtering failures

    data.frame(as.list(coef(mf)),
               loglik = ll[1],
               loglik.se = ll[2],
               nfail.min = min(nfail),
               nfail.max = max(nfail))
  } %>% arrange(Beta,-loglik)
}) -> beta_prof
```

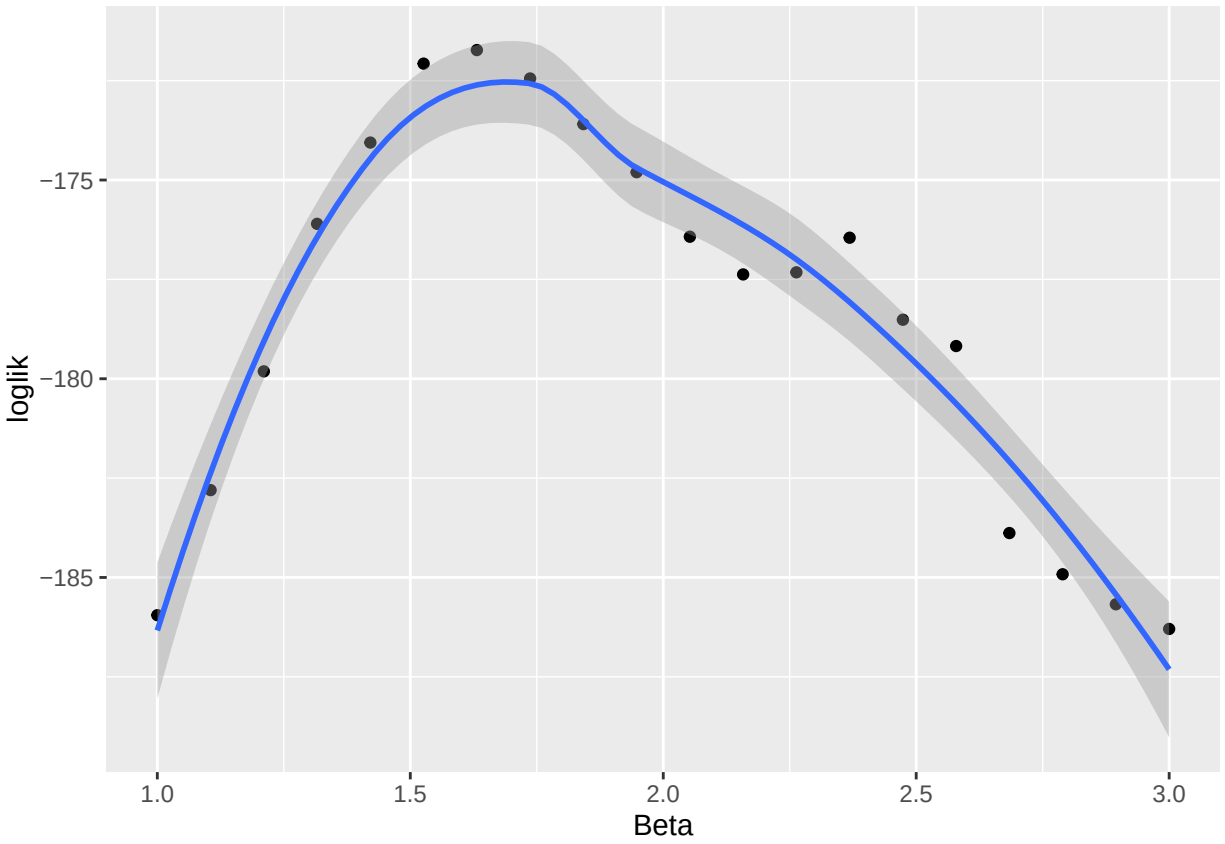
Pairwise scatterplot matrix for Beta:

```
pairs(~loglik+Beta+beta_sd+mu+gamma+rho+N,data=beta_prof,subset=loglik>max(loglik)-10)
```



There appears to be a positive linear correlation between Beta and gamma, as well as Beta and N and Beta and mu in this profile design.

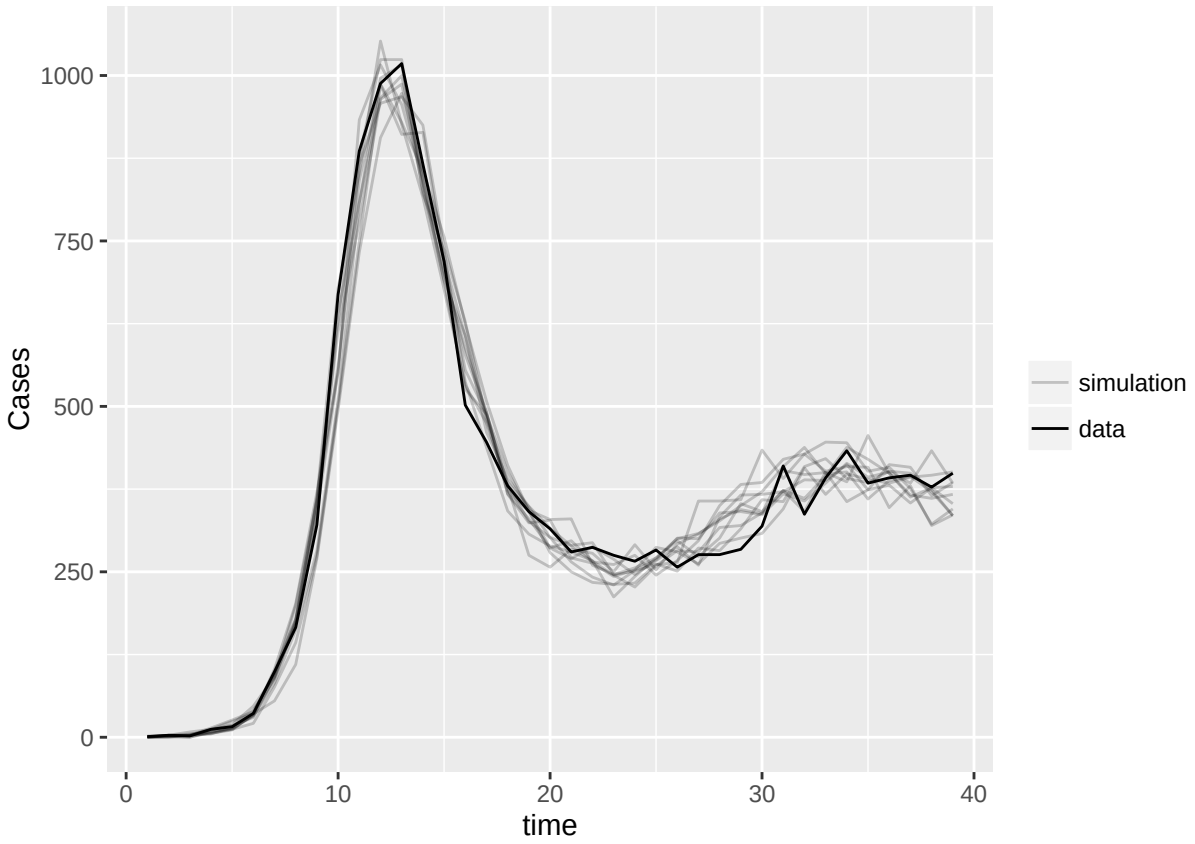
```
beta_prof %>%
  mutate(Beta=signif(Beta,8)) %>%
  ddply(~Beta,subset,loglik==max(loglik)) %>%
  ggplot(aes(x=Beta,y=loglik))+geom_point()+geom_smooth()
```



This graph shows Beta plotted as a function of log likeihood. The maximum log likelihood appears to be close to 1.5, which is the true parameter value. The gray ribbon surrounding the blue line represents the confidence interval estimate, and it is seen that 65% of the points actually fall within the 95% confidence interval.

Now let us plot the data and 10 simulated realizations of the model process on the same axes.

```
beta_prof %>%
  subset(loglik==max(loglik)) %>% unlist() -> mle
simulate(rawModel,params=mle,nsim=10,as.data.frame=TRUE,include.data=TRUE) %>%
  ggplot(mapping=aes(x=time,y=Cases,group=sim,alpha=(sim=="data")))+
  scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                     labels=c(`FALSE`="simulation",`TRUE`="data"))+
  geom_line()
```



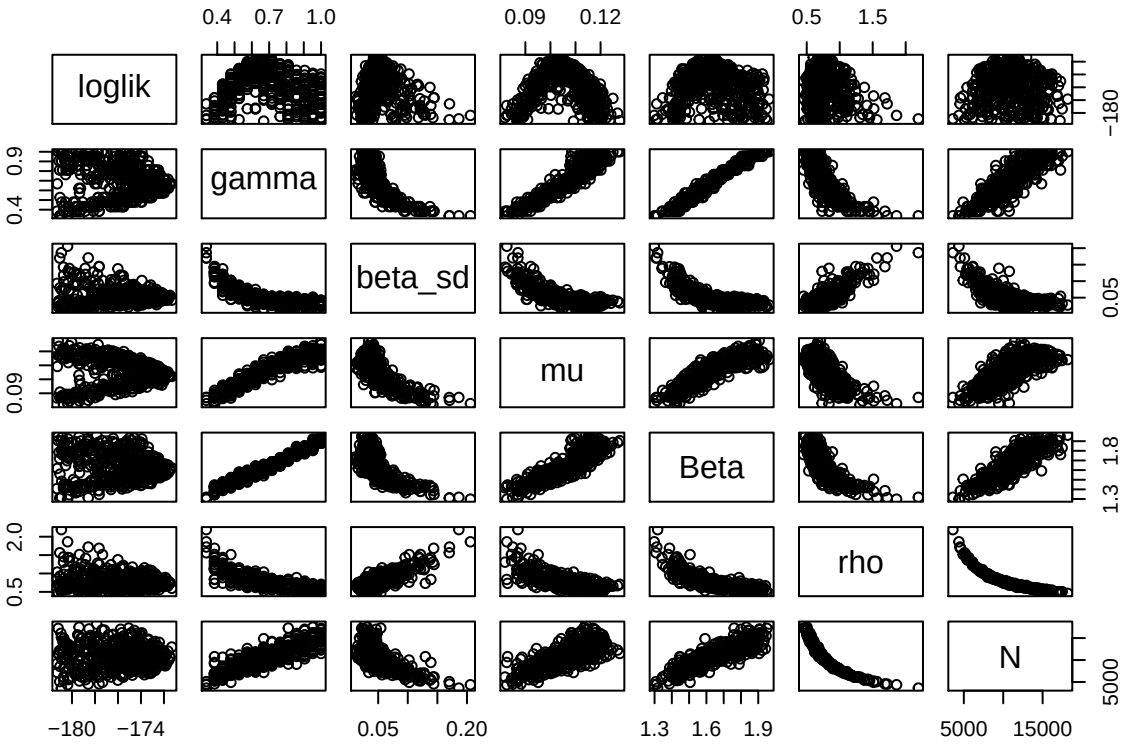
Raw Gamma Profile The gamma profile is constructed below.

```
profileDesign(
  gamma=seq(from=0.1, to=1.0, length=20),
  lower=c(beta_sd=0.05, mu=0.01, Beta=1.0, rho=0.05, N=10000),
  upper=c(beta_sd=0.5, mu=1.0, Beta=3.0, rho=0.99, N=10000),
  nprof=50
) -> gammaPD

bake("RawFakedata_gamma-profile.rds",{
  foreach (p=iter(gammaPD,"row"),
           .combine=rbind,
           .errorhandling="remove",
           .packages=c("pomp","magrittr","reshape2","plyr"),
           .inorder=FALSE,
           .options.mpi=list(seed=1680158025)
  ) %dopar% {
    rawModel %>%
      mif2(start=unlist(p),Nmif=50,Np=1000,transform=TRUE,
           cooling.fraction.50=0.8,cooling.type="geometric",
           rw.sd=rw.sd(beta_sd=0.02, mu=0.02, Beta=0.02, rho=0.02, N=0.02)) %>%
      mif2() -> mf

    pf <- replicate(5,pfilter(mf,Np=1000))  ## independent particle filters
    ll <- sapply(pf,logLik)
    ll <- logmeanexp(ll,se=TRUE)
```

```
    nfail <- sapply(pf,getElement,"nfail")  ## number of filtering failures

    data.frame(as.list(coef(mf)),
               loglik = ll[1],
               loglik.se = ll[2],
               nfail.min = min(nfail),
               nfail.max = max(nfail))
  } %>% arrange(gamma,-loglik)
}) -> gamma_prof
```

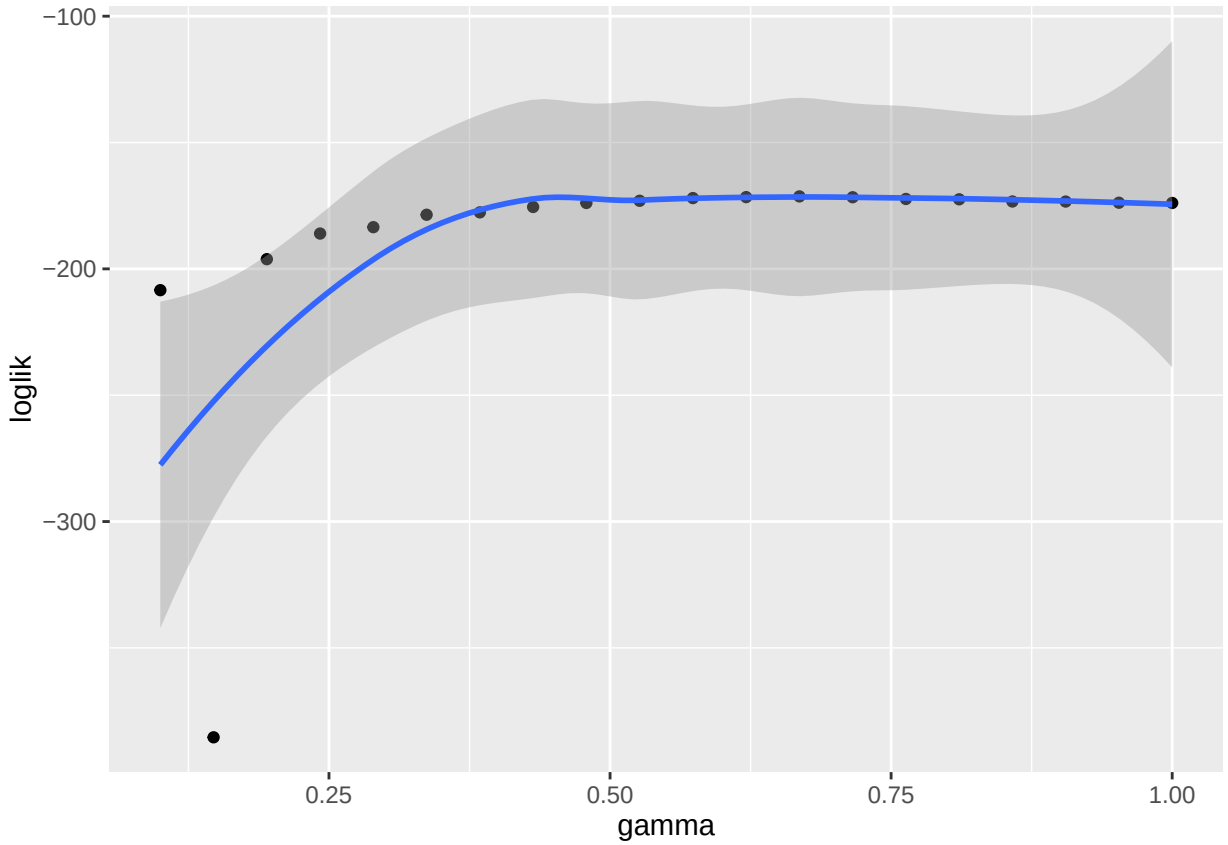
Pairwise scatterplot matrix for Gamam:

```
pairs(~loglik+gamma+beta_sd+mu+Beta+rho+N,data=gamma_prof,subset=loglik>max(loglik)-10)
```



Gamma also apperas to have a positive linear correlation with beta, mu, and N.

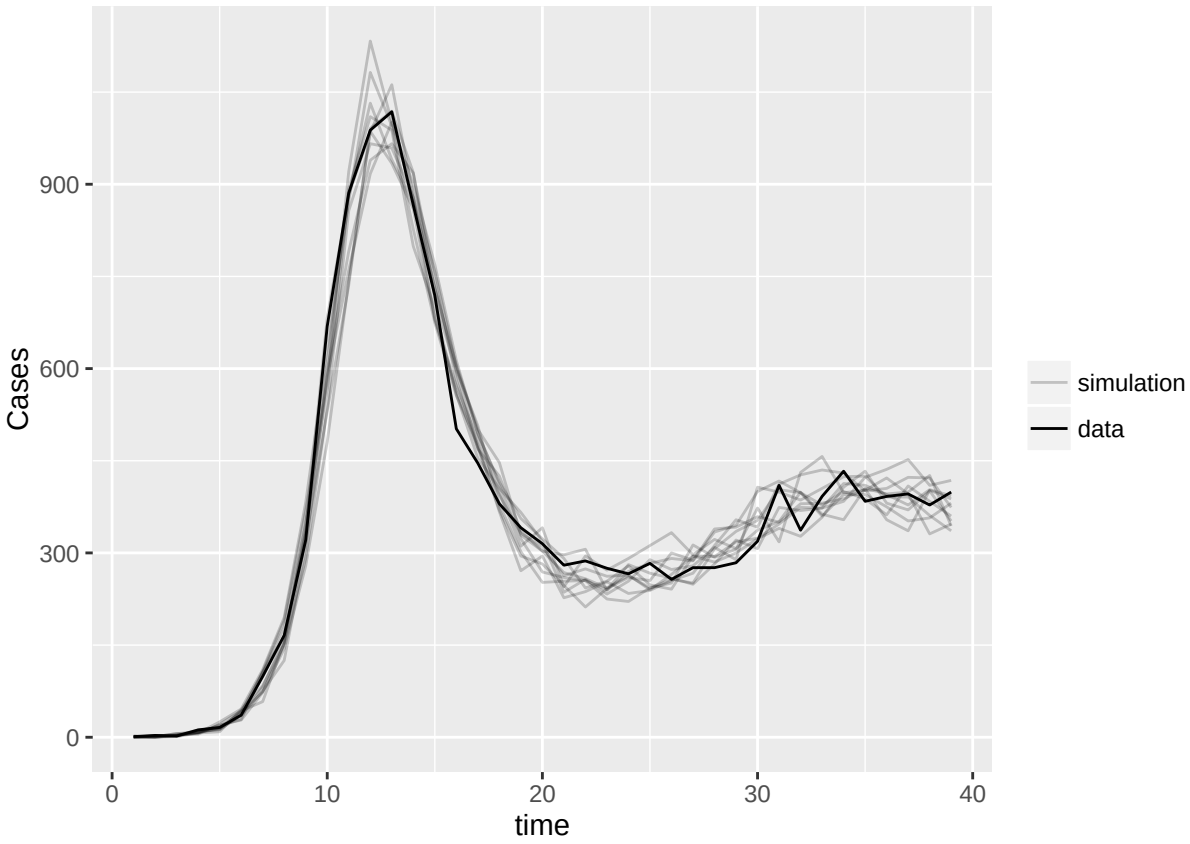
```
gamma_prof %>%
  mutate(gamma=signif(gamma,8)) %>%
  ddply(~gamma,subset,loglik==max(loglik)) %>%
  ggplot(aes(x=gamma,y=loglik))+geom_point()+geom_smooth()
```



This graph shows gamma plotted as a function of log likeihood. The maximum log likelihood appears to be between the range of 0.5-1.0, with 0.6 being its true parameter value. The gray ribbon surrounding the blue line represents the confidence interval estimate, and it is seen that 90% of the points actually fall within the 95% confidence interval. Likely higher coverage due to a wider CI.

Now let us plot the data and 10 simulated realizations of the model process on the same axes.

```
gamma_prof %>%
  subset(loglik==max(loglik)) %>% unlist() -> mle
simulate(rawModel,params=mle,nsim=10,as.data.frame=TRUE,include.data=TRUE) %>%
  ggplot(mapping=aes(x=time,y=Cases,group=sim,alpha=(sim=="data")))+
  scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                     labels=c(`FALSE`="simulation",`TRUE`="data"))+
  geom_line()
```



The raw iterated filtering and profile likelihood fits all closely resemble the data. The 65% achieved coverage of the 95% CI for the Beta parameter estimate is not great, but gamma has 90% coverage. All the simulated realizations using parameter estimates from this stochastic method appear to fit the data very well.

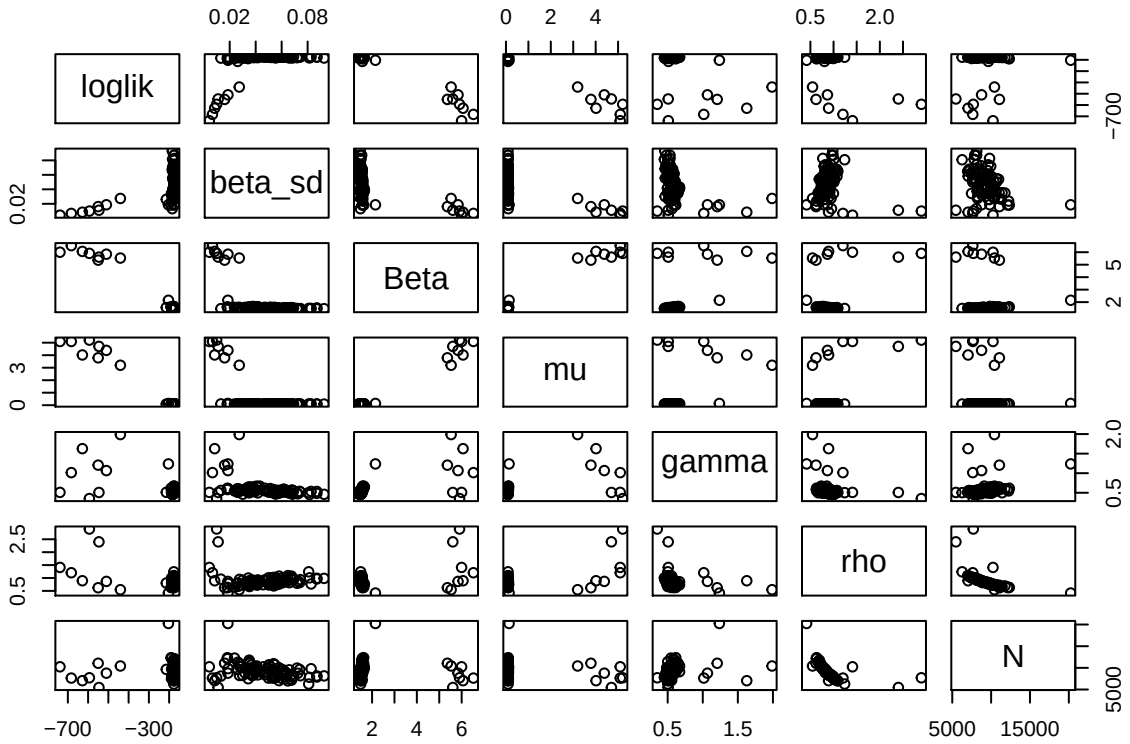
Cumulative Iterated Filtering

The cumulative iterated filtering algorithm is defined here:

```
registerDoParallel()
bake(file="CumFakeData-mif.rds", {
  guesses <- sobolDesign(lower=c(beta_sd=0.05, Beta=1.0, mu=0.01, gamma=0.1, rho=0.05, N=10000),
                         upper=c(beta_sd=0.5, Beta=3.0, mu=1.0, gamma=0.99, rho=0.99, N=10000),
                         nseq=100)
  foreach (guess=iter(guesses,"row"),.combine=rbind,
           .options.mpi=list(seed=334065675),
           .packages=c("pomp","magrittr"),.errorhandling="remove") %dopar% {
             cumModel %>%
               mif2(start=unlist(guess),Nmif=50,Np=1000,transform=TRUE,
                    cooling.fraction.50=0.8,cooling.type="geometric",
                    rw.sd=rw.sd(beta_sd=0.02, Beta=0.02, mu=0.02, gamma=0.02, rho=0.02, N=0.02)) %>%
               mif2() -> mf
             ll <- logmeanexp(replicate(5,logLik(pfilter(mf))),se=TRUE)
             data.frame(loglik=ll[1],loglik.se=ll[2],as.list(coef(mf)))
           }
}) -> mles_cum
```

A useful way to view the data is with a pairwise scatterplot matrix:

```
pairs(~loglik+beta_sd+Beta+mu+gamma+rho+N,data=mles_cum)
```

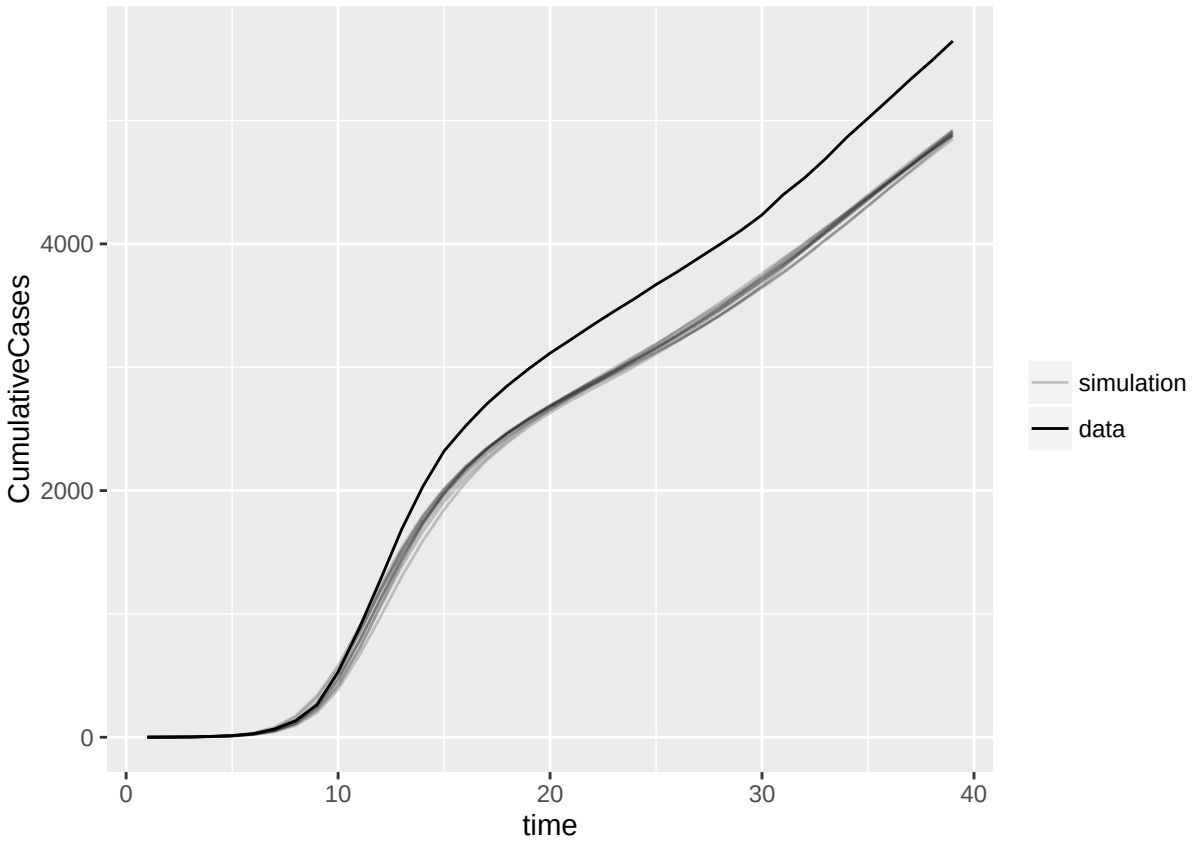


From looking at the matrix, there does not seem to be any linear correlations between the parameters in this model. Most of the paramters seem to gravitate near one location for the best log likelihood.

Iterated Filtering Fit to Data

The iterated filtering algorithm produces a maximum likelihood estimate for the parameter values. The fit to the data can be seen by doing the following:

```
mles_cum %>%
  subset(loglik==max(loglik)) %>% unlist() -> mle
simulate(cumModel,params=mle,nsim=10,as.data.frame=TRUE,include.data=TRUE) -> fits
fits$CumulativeCases[1:39] <- fits$actualCumulative[1:39]
  ggplot(data=fits, mapping=aes(x=time,y=CumulativeCases,group=sim,alpha=(sim=="data")))+
  scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                     labels=c(`FALSE`="simulation",`TRUE`="data"))+
  geom_line()
```



Likelihood Profiles

To get better estimates and obtain likelihood-ratio-test based confidence intervals, I've constructed profile likelihoods for two important parameters: beta (transmission term) and gamma (recovery rate). In a likelihood profile, one varies the focal parameter across some range, maximizing the likelihood at each value of the focal parameter over the remaining parameters.

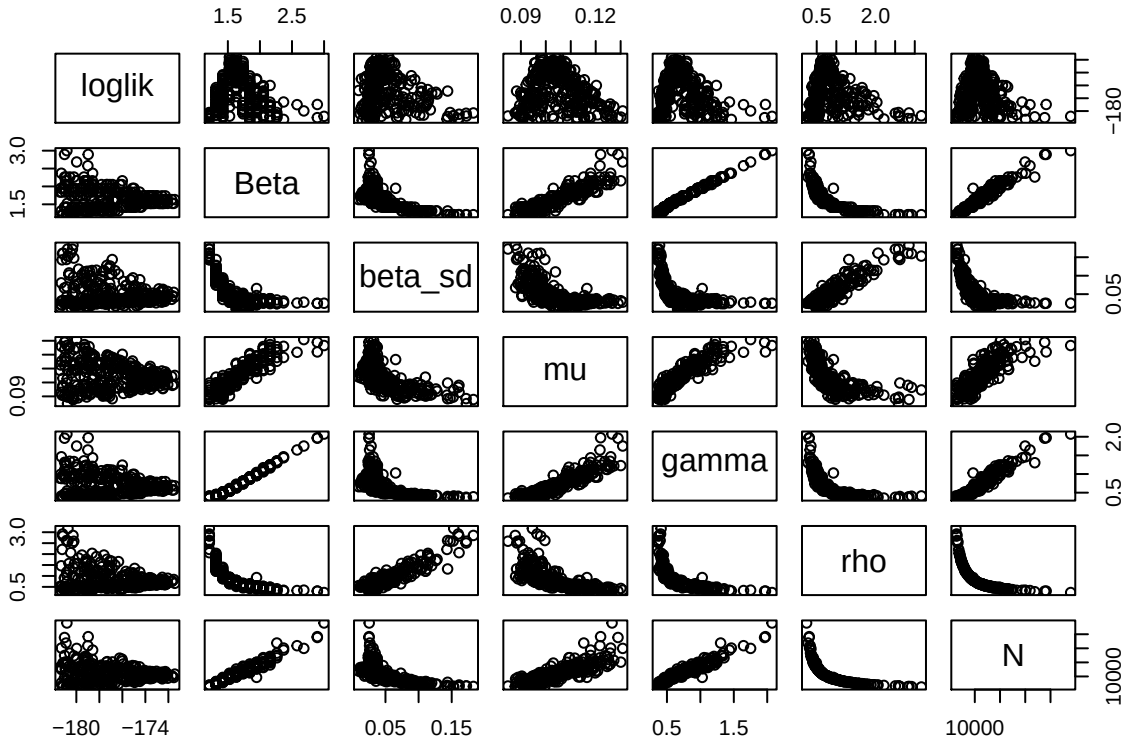
Cumulative Beta Profile The beta profile is constructed below.

```
profileDesign(
  Beta=seq(from=1.0, to=3.0, length=20),
  lower=c(beta_sd=0.05, mu=0.01, gamma=0.1, rho=0.05, N=10000),
  upper=c(beta_sd=0.5, mu=1.0, gamma=0.99, rho=0.99, N=10000),
  nprof=50
) -> cumBetaPD

bake("CumFakedata_Beta-profile.rds",{
  foreach (p=iter(cumBetaPD,"row"),
           .combine=rbind,
           .errorhandling="remove",
           .packages=c("pomp","magrittr","reshape2","plyr"),
           .inorder=FALSE,
           .options.mpi=list(seed=1680158025)
  ) %dopar% {
    cumModel %>%
```

```
      mif2(start=unlist(p),Nmif=50,Np=1000,transform=TRUE,
           cooling.fraction.50=0.8,cooling.type="geometric",
           rw.sd=rw.sd(beta_sd=0.02, mu=0.02, gamma=0.02, rho=0.02, N=0.02)) %>%
      mif2() -> mf

    pf <- replicate(5,pfilter(mf,Np=1000))  ## independent particle filters
    ll <- sapply(pf,logLik)
    ll <- logmeanexp(ll,se=TRUE)
    nfail <- sapply(pf,getElement,"nfail")  ## number of filtering failures

    data.frame(as.list(coef(mf)),
               loglik = ll[1],
               loglik.se = ll[2],
               nfail.min = min(nfail),
               nfail.max = max(nfail))
  } %>% arrange(Beta,-loglik)
}) -> beta_prof_cum
```

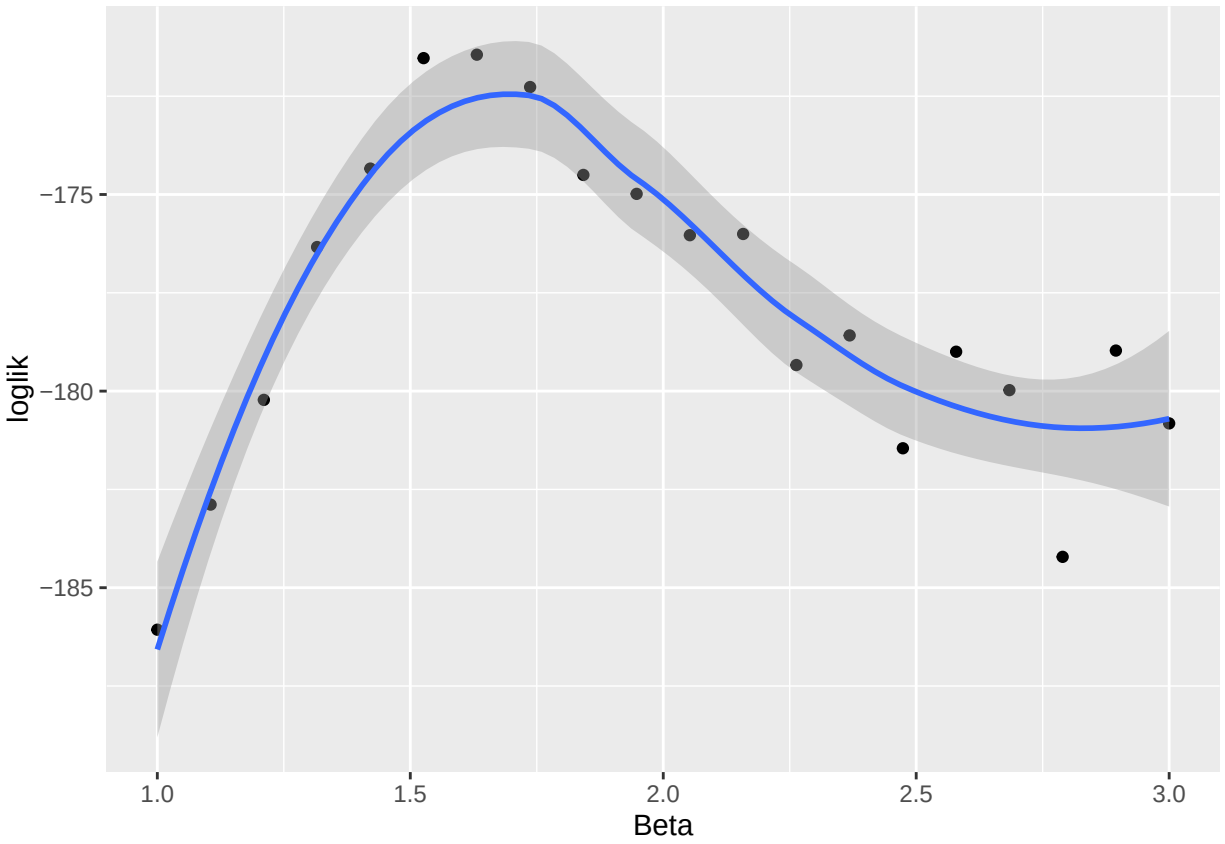
Pairwise scatterplot matrix for Beta:

```
pairs(~loglik+Beta+beta_sd+mu+gamma+rho+N,data=beta_prof_cum,subset=loglik>max(loglik)-10)
```



There appears to be a positive linear correlation between Beta and gamma, as well as Beta and N and Beta and mu in this profile design.

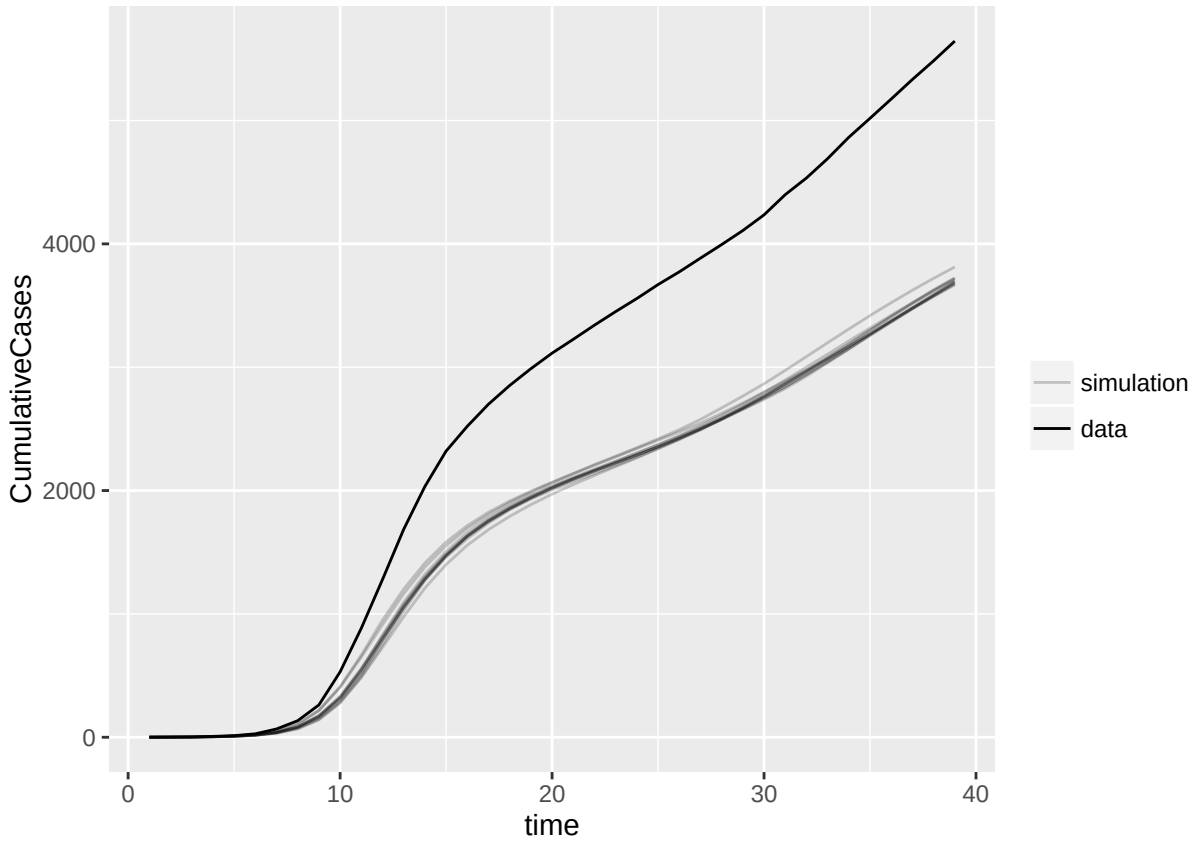
```
beta_prof_cum %>%
  mutate(Beta=signif(Beta,8)) %>%
  ddply(~Beta,subset,loglik==max(loglik)) %>%
  ggplot(aes(x=Beta,y=loglik))+geom_point()+geom_smooth()
```



This graph shows Beta plotted as a function of log likeihood. The maximum log likelihood appears to be close to 1.6, with the actual value being 1.5. The gray ribbon surrounding the blue line represents the confidence interval estimate, and it is seen that 70% of the points actually fall within the 95% confidence interval.

Now let us plot the data and 10 simulated realizations of the model process on the same axes.

```
beta_prof_cum %>%
  subset(loglik==max(loglik)) %>% unlist() -> mle
simulate(cumModel,params=mle,nsim=10,as.data.frame=TRUE,include.data=TRUE) -> betaSim
betaSim$CumulativeCases[1:39] <- betaSim$actualCumulative[1:39]
ggplot(data=betaSim, mapping=aes(x=time,y=CumulativeCases,group=sim,alpha=(sim=="data")))+
scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                   labels=c(`FALSE`="simulation",`TRUE`="data"))+
 geom_line()
```



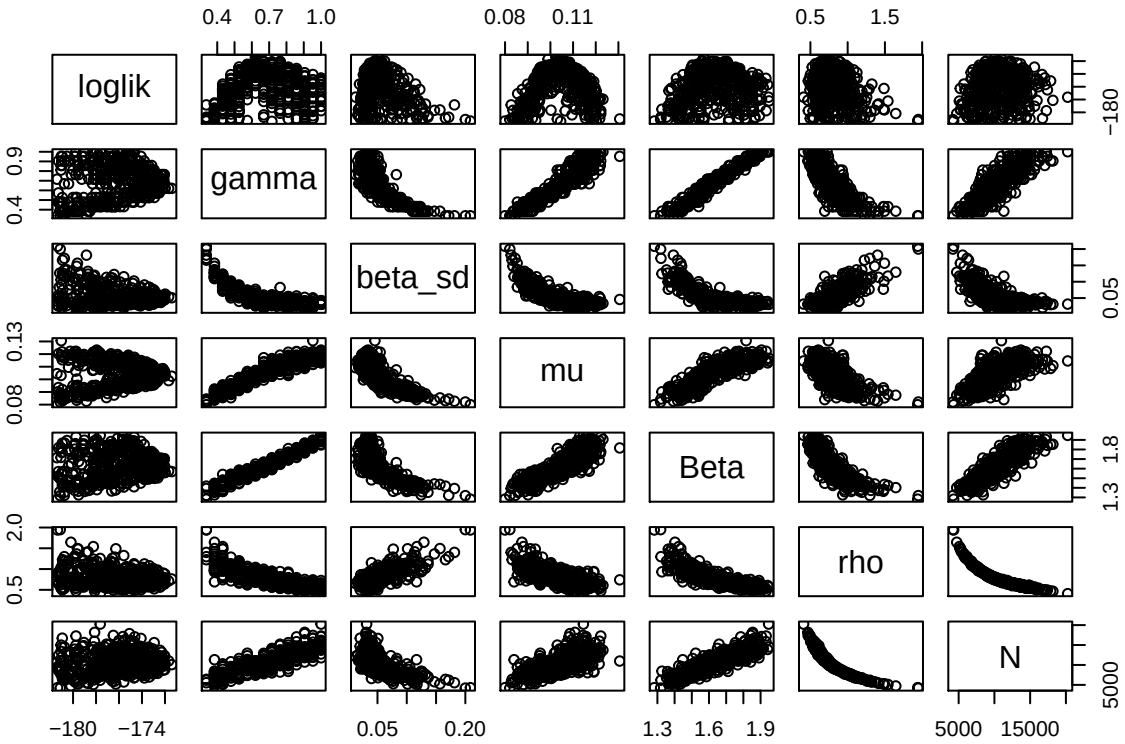
Cumulative Gamma Profile The gamma profile is constructed below.

```
profileDesign(
  gamma=seq(from=0.1, to=1.0, length=20),
  lower=c(beta_sd=0.05, mu=0.01, Beta=1.0, rho=0.05, N=10000),
  upper=c(beta_sd=0.5, mu=1.0, Beta=3.0, rho=0.99, N=10000),
  nprof=50
) -> cumGammaPD

bake("CumFakedata_gamma-profile.rds",{
  foreach (p=iter(cumGammaPD,"row"),
           .combine=rbind,
           .errorhandling="remove",
           .packages=c("pomp","magrittr","reshape2","plyr"),
           .inorder=FALSE,
           .options.mpi=list(seed=1680158025)
  ) %dopar% {
    cumModel %>%
      mif2(start=unlist(p),Nmif=50,Np=1000,transform=TRUE,
           cooling.fraction.50=0.8,cooling.type="geometric",
           rw.sd=rw.sd(beta_sd=0.02, mu=0.02, Beta=0.02, rho=0.02, N=0.02)) %>%
      mif2() -> mf

    pf <- replicate(5,pfilter(mf,Np=1000))  ## independent particle filters
    ll <- sapply(pf,logLik)
    ll <- logmeanexp(ll,se=TRUE)
```

```
    nfail <- sapply(pf,getElement,"nfail")  ## number of filtering failures

    data.frame(as.list(coef(mf)),
               loglik = ll[1],
               loglik.se = ll[2],
               nfail.min = min(nfail),
               nfail.max = max(nfail))
  } %>% arrange(gamma,-loglik)
}) -> gamma_prof_cum
```

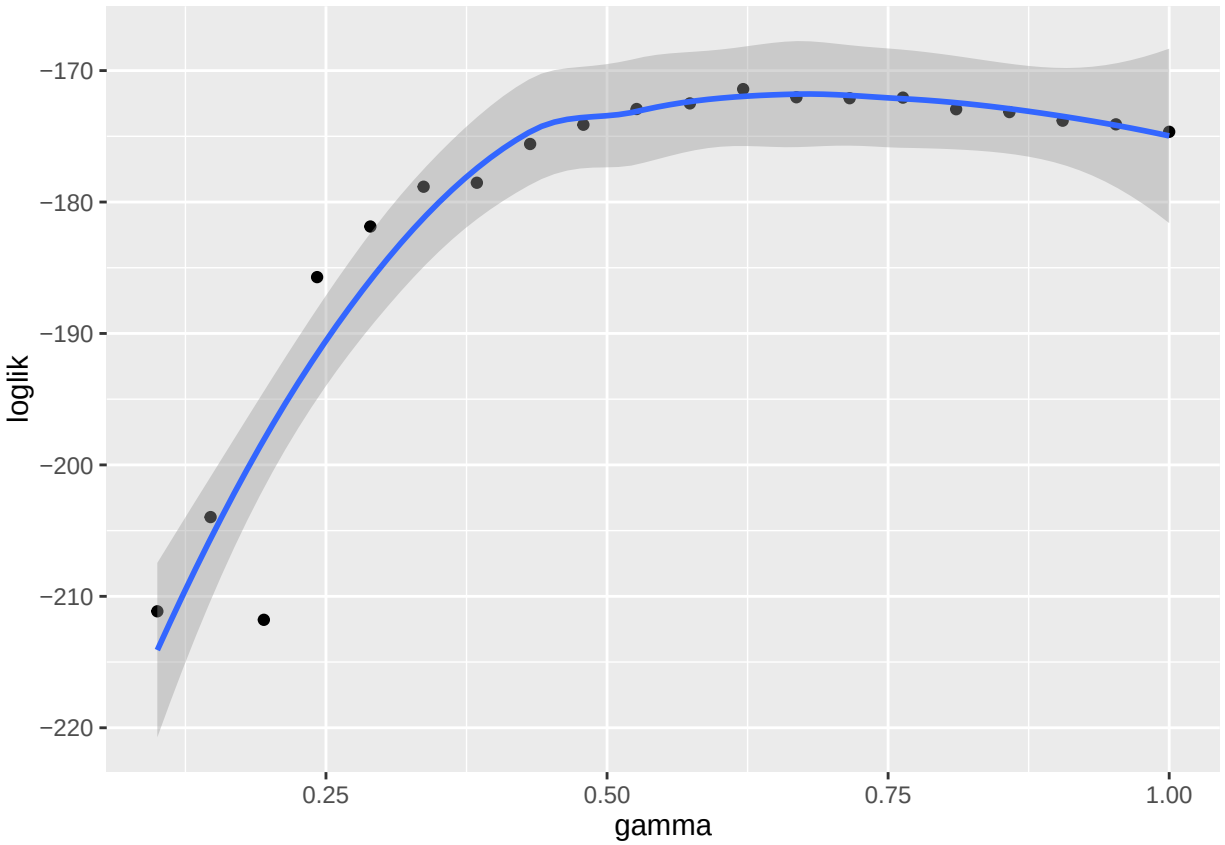
Pairwise scatterplot matrix for Gamma:

```
pairs(~loglik+gamma+beta_sd+mu+Beta+rho+N,data=gamma_prof_cum,subset=loglik>max(loglik)-10)
```



Gamma also appears to have a positive correlation with Beta, mu, and N.

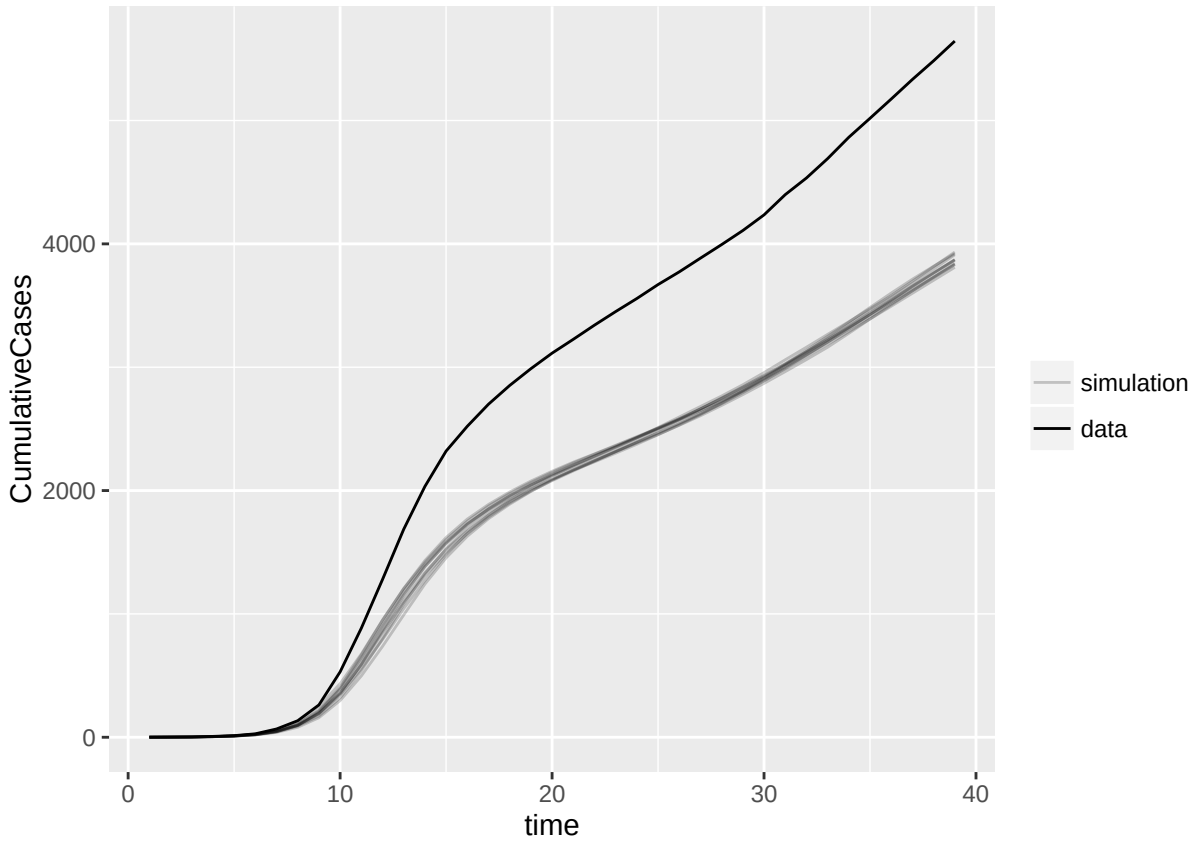
```
gamma_prof_cum %>%
  mutate(gamma=signif(gamma,8)) %>%
  ddply(~gamma,subset,loglik==max(loglik)) %>%
  ggplot(aes(x=gamma,y=loglik))+geom_point()+geom_smooth()
```



This graph shows gamma plotted as a function of log likeihood. The maximum log likelihood appears to be in a range between 0.5-0.75, with 0.6 being its true parameter value. The gray ribbon surrounding the blue line represents the confidence interval estimate, and it is seen that 80% of the points actually fall within the 95% confidence interval.

Now let us plot the data and 10 simulated realizations of the model process on the same axes.

```
gamma_prof_cum %>%
  subset(loglik==max(loglik)) %>% unlist() -> mle
simulate(cumModel,params=mle,nsim=10,as.data.frame=TRUE,include.data=TRUE) -> gammaSim
gammaSim$CumulativeCases[1:39] <- gammaSim$actualCumulative[1:39]
ggplot(data=gammaSim,mapping=aes(x=time,y=CumulativeCases,group=sim,alpha=(sim=="data")))+
scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                   labels=c(`FALSE`="simulation",`TRUE`="data"))+
  geom_line()
```



The cumulative iterated filtering and profile likelihood fits all seem to fail to takeoff as much as the data. The 70% achieved coverage of the 95% CI for the Beta parameter estimate and the 80% coverage for gamma do not do much better than the raw data. All the simulated realizations using parameter estimates from this stochastic method appear to slow down more rapidly at the point of inflection than the data does.

Sampling the posterior using particle Markov chain Monte Carlo

In order to do full-information Bayesian inference, we can use particle Markov chain Monte Carlo, implemented in pomp in the pmcmc function. First we must add a dprior component to our models.

```
dprior <- Csnippet("
    lik = dunif(beta_sd, 0.01, 1.0, 1)+dunif(Beta, 1.0, 3.0, 1)+dunif(mu, 0.01, 1.0, 1)+
          dunif(gamma, 0.1, 2.0, 1)+dunif(rho, 0.05, 0.99, 1)+dunif(N, 1000, 10000, 1);
    lik = (give_log) ? lik : exp(lik);
    ")

rawModel <- pomp(sampleData, time="time", t0=0,
                 rprocess=euler.sim(step.fun, delta.t=0.1), initializer = init,
                 statenames=c("S", "I", "RealCases"), paramnames=c("Beta", "gamma", "mu", "rho", "N","beta
                 obsnames = "Cases", zeronames = "RealCases", rmeasure = rmeas, dmeasure = dmeas,
                 skeleton = vectorfield(skel), dprior=dprior,
                 toEstimationScale=logtrans, fromEstimationScale = exptrans)

cumModel <- pomp(cumulativeSampleData, time="time", t0=0,
                rprocess=euler.sim(step.funCum, delta.t=0.1), initializer = initCum,
                statenames=c("S", "I", "RealCases", "CumulativeCases"), paramnames=c("Beta", "gamma", "mu
```

```
              "rho", "N","beta_sd"), obsnames = "Cases", zeronames = "RealCases",
              rmeasure = rmeas, dmeasure = dmeas, skeleton = vectorfield(skelCum),
              toEstimationScale=logtrans, fromEstimationScale = exptrans, dprior=dprior)
```

Raw PMCMC

Perform a random walk markov chain monte carlo algorithm.

```
#Particle Markov Chain Monte Carlo using estimates obtained from mif2
bake(file="RawData-pmcmc.rds",{
  beta_prof %>% ddply(~Beta,subset,loglik==max(loglik)) %>%
    subset(beta_sd > 0.01 & beta_sd < 1.0 & Beta > 1.0 & Beta < 3.0 & mu < 1.0 &
           gamma < 2.0 & rho > 0.05 & rho < 0.99,
           select=-c(loglik,loglik.se)) -> starts
  foreach (start=iter(starts,"row"),.combine=c,
           .options.mpi=list(seed=23781975),
           .packages=c("pomp","magrittr"),.errorhandling="remove") %dopar%
           {
             rawModel %>%
               pmcmc(Nmcmc=2000,Np=200,start=unlist(start),
                     proposal=mvn.rw.adaptive(rw.sd=c(Beta=0.3,beta_sd=0.05,mu=0.05,gamma=0.1,rho=0.15,
                                              scale.start=100,shape.start=100)) -> chain
             chain %>% pmcmc(Nmcmc=10000,proposal=mvn.rw(covmat(chain)))
           }
}) -> rawChains
```

The following code attempts to determine whether the chains have converged:

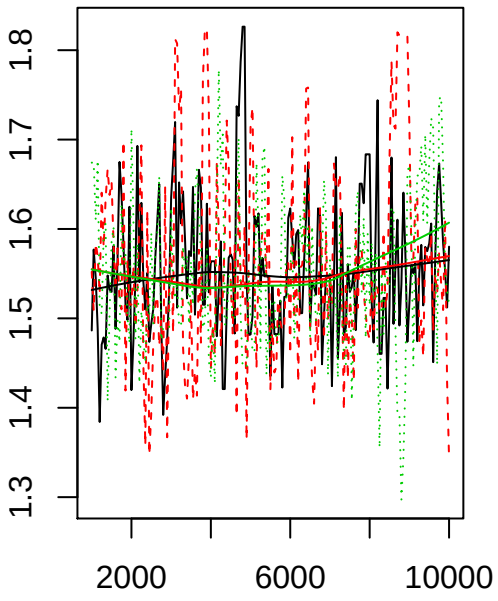
```
rawChains %>% conv.rec() -> rawTraces
rejectionRate(rawTraces[,c("Beta","beta_sd","mu", "gamma", "rho", "N")])
```

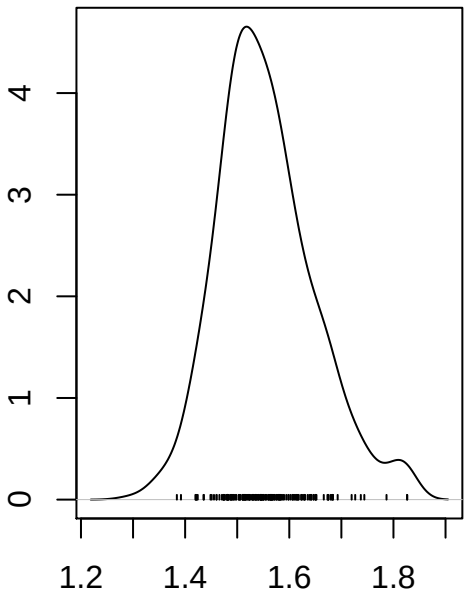
```
##      Beta   beta_sd        mu     gamma       rho         N
## 0.9069667 0.9069667 0.9069667 0.9069667 0.9069667 0.9069667
```

```
autocorr.diag(rawTraces[,c("Beta","beta_sd", "mu", "gamma", "rho", "N")])
```

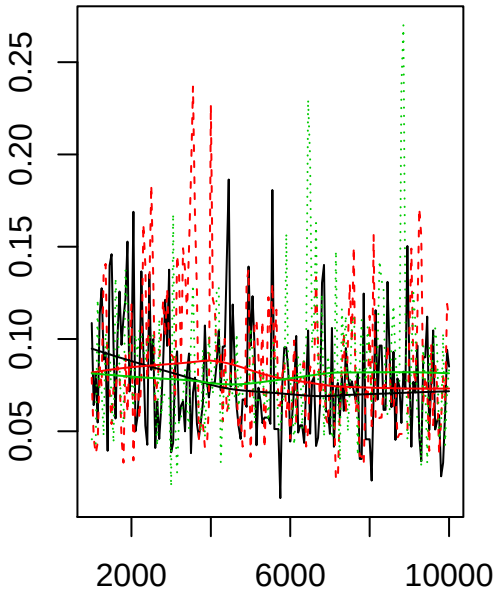
```
##             Beta   beta_sd        mu     gamma       rho         N
## Lag 0  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## Lag 1  0.9762821 0.9631136 0.9448919 0.9789624 0.9769558 0.9770110
## Lag 5  0.8947004 0.8402815 0.7683219 0.9035007 0.8957987 0.8945950
## Lag 10 0.8099162 0.7185709 0.6209534 0.8256748 0.8096633 0.8064720
## Lag 50 0.4227843 0.3033271 0.2636252 0.4766361 0.3641620 0.3662486
```

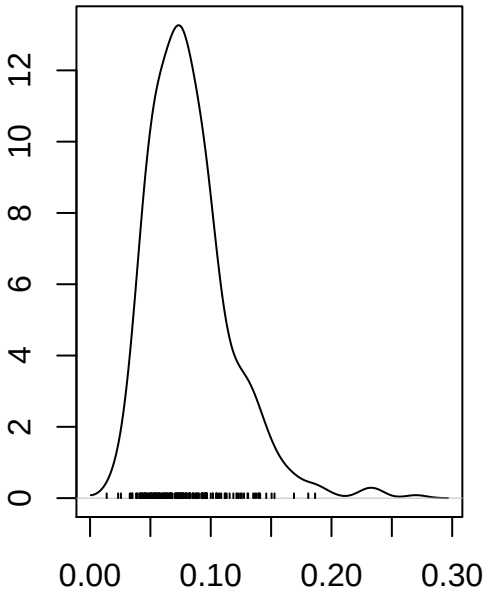
```
rawTraces <- window(rawTraces,thin=50,start=1000)
plot(rawTraces[,"Beta"])
```



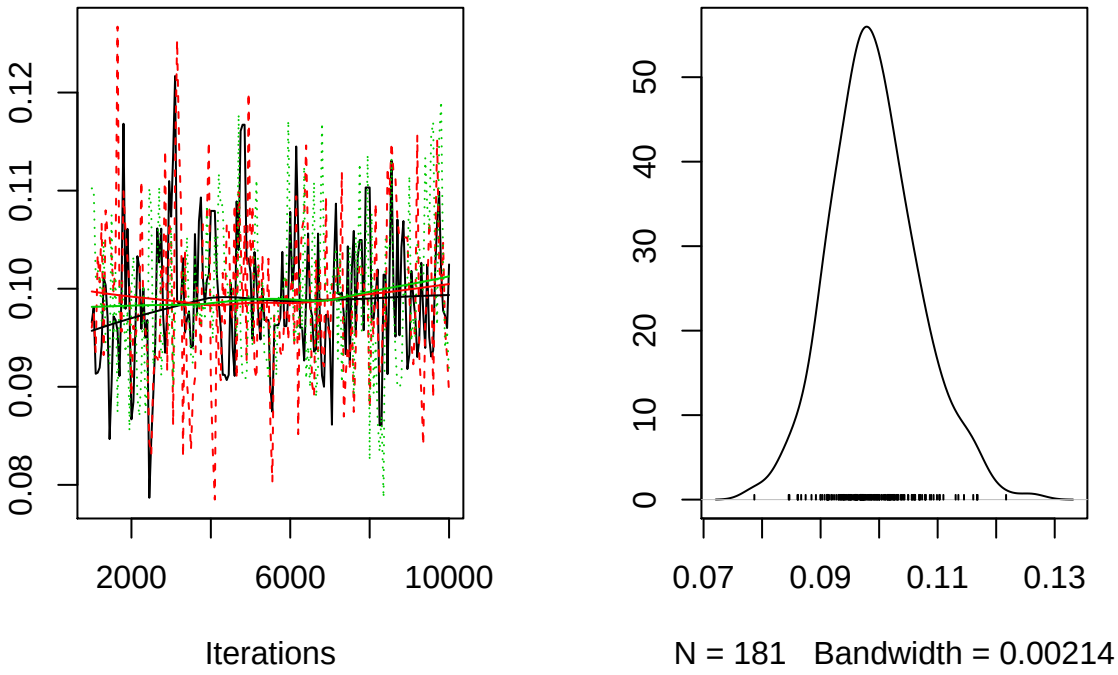



```
plot(rawTraces[,"beta_sd"])
```

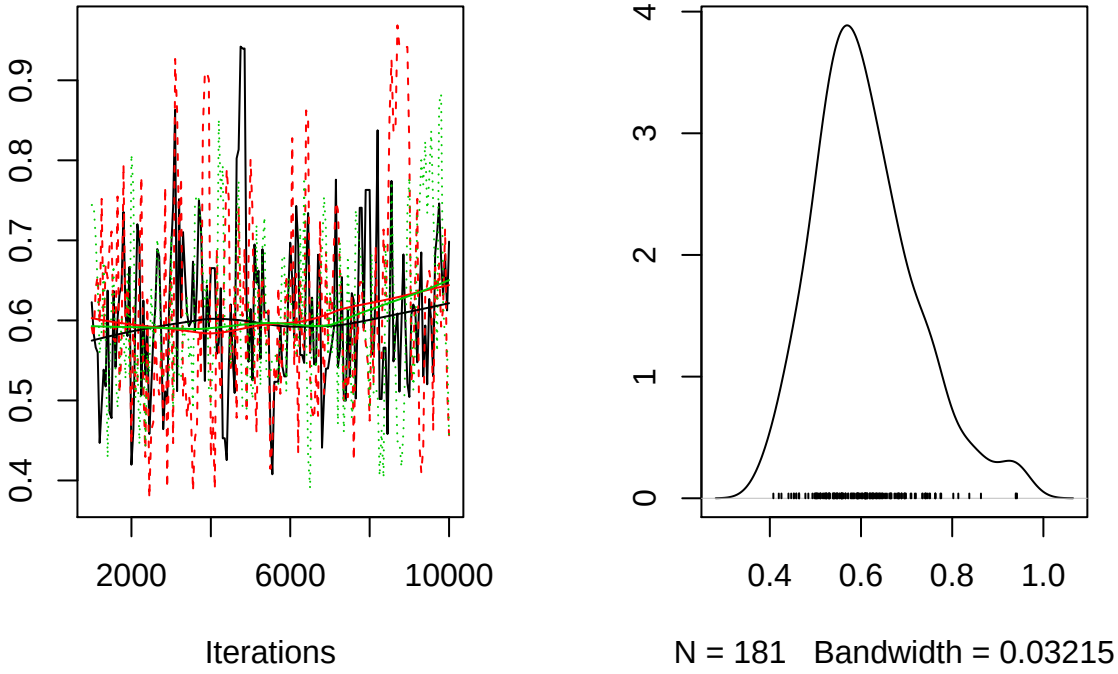




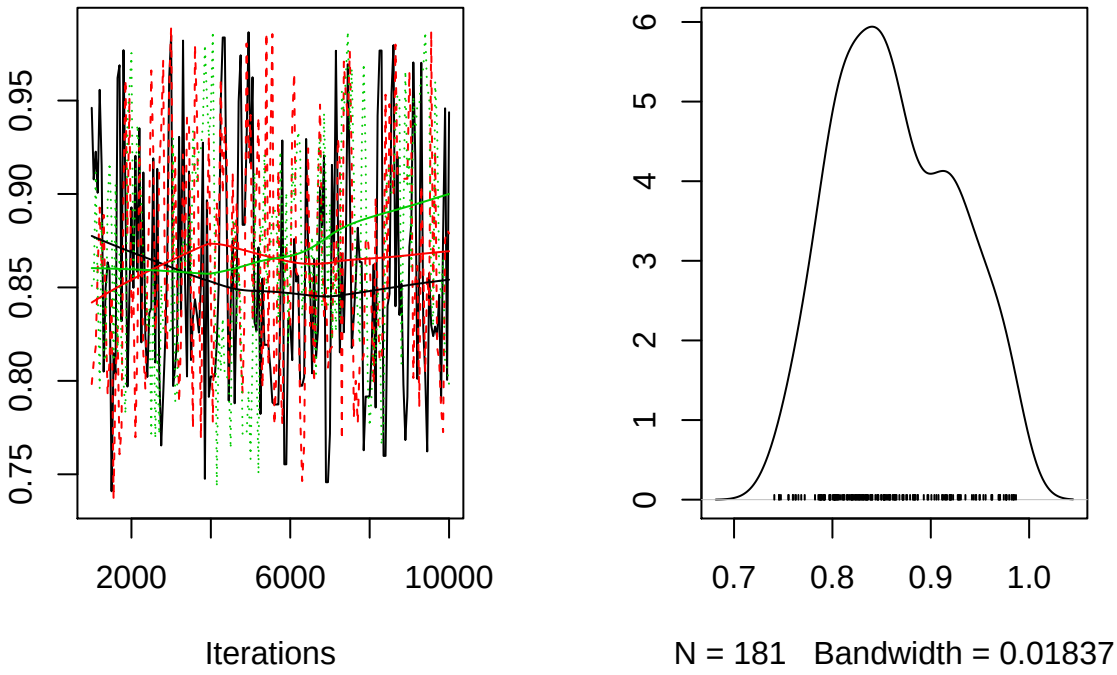

```
plot(rawTraces[,"mu"])
```



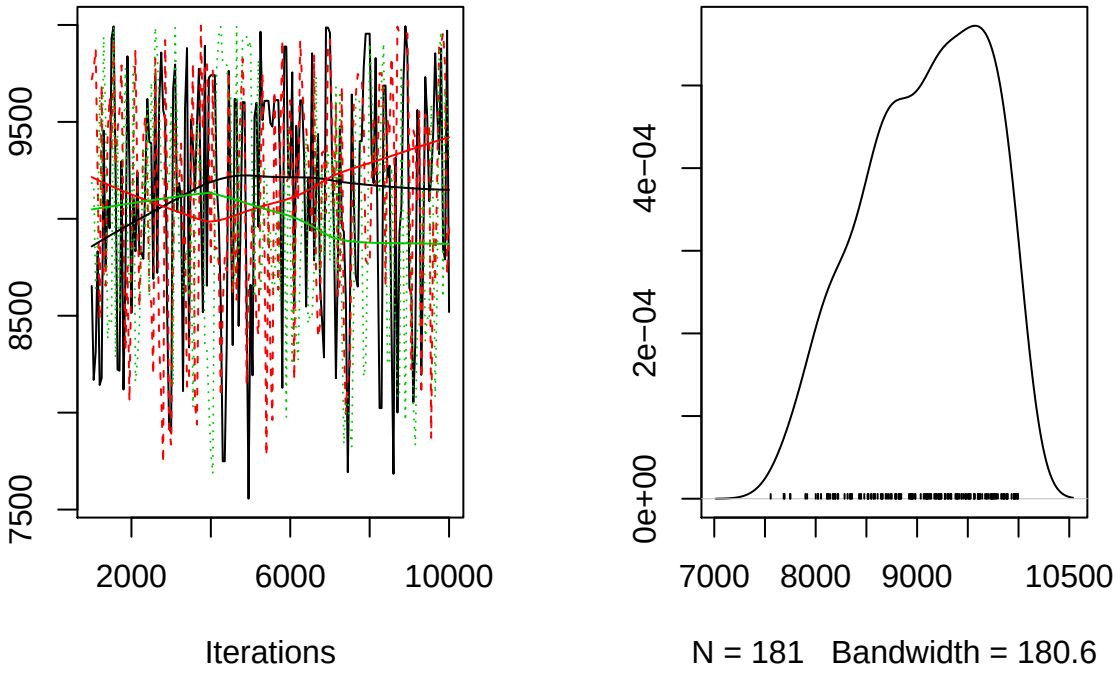

```
plot(rawTraces[,"gamma"])
```




```
plot(rawTraces[,"rho"])
```




```
plot(rawTraces[,"N"])
```




```
gelman.diag(rawTraces[,c("Beta","beta_sd","mu", "gamma", "rho", "N")])
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## Beta          1.03       1.04
## beta_sd       1.07       1.18
```

```
## mu             1.01       1.02
## gamma          1.03       1.06
## rho            1.06       1.20
## N              1.06       1.21
##
## Multivariate psrf
##
## 1.09
```

It appears that the MCMC iterations are close to converging to their stationary distributions. Let's simulate from the posterior median parameter values.

```
summary(rawTraces[,c("Beta","beta_sd","mu", "gamma", "rho", "N")])

theta <- summary(rawTraces)$quantiles[c("Beta","beta_sd","mu","gamma", "rho", "N"),'50%']
simulate(rawModel,params=theta,nsim=10,as.data.frame=TRUE,include.data=TRUE) %>%
  ggplot(mapping=aes(x=time,y=Cases,group=sim,alpha=(sim=="data")))+
  scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                     labels=c(`FALSE`="simulation",`TRUE`="data"))+
  geom_line()
```

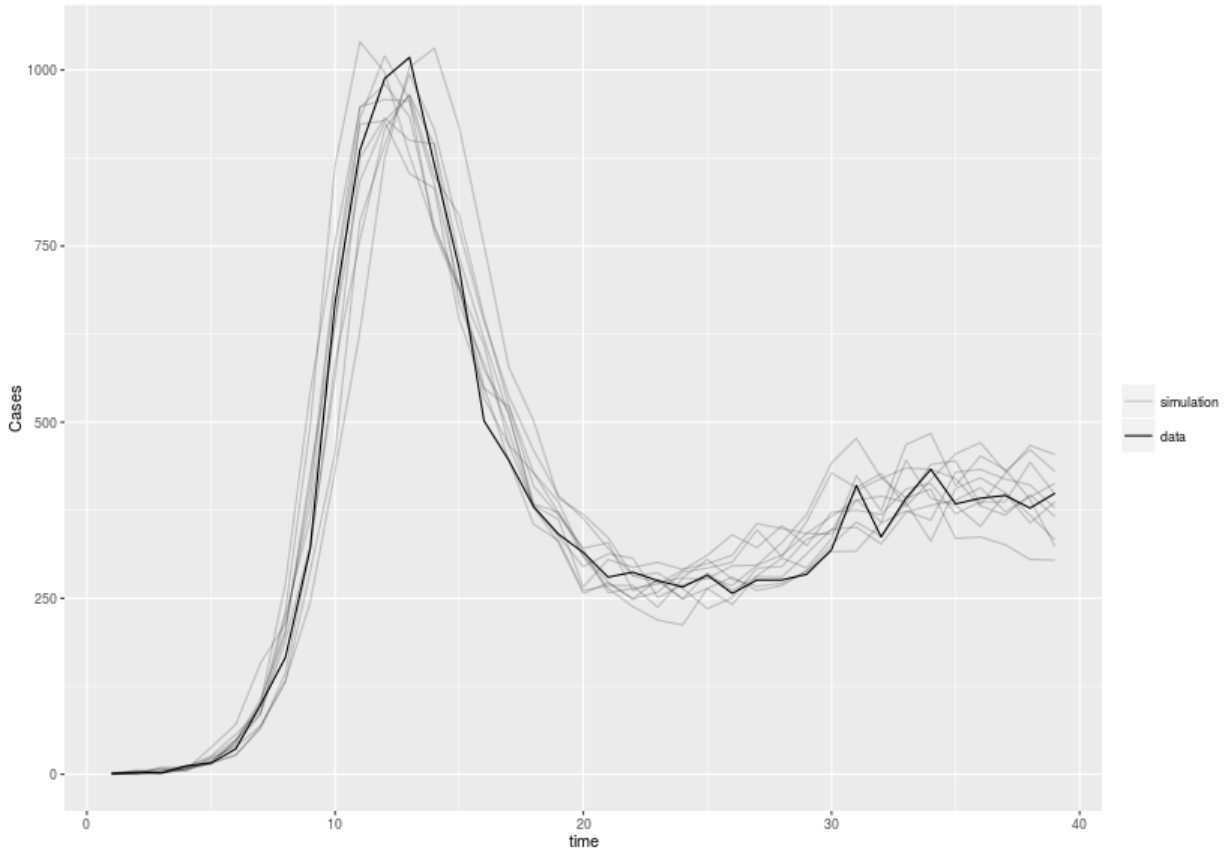
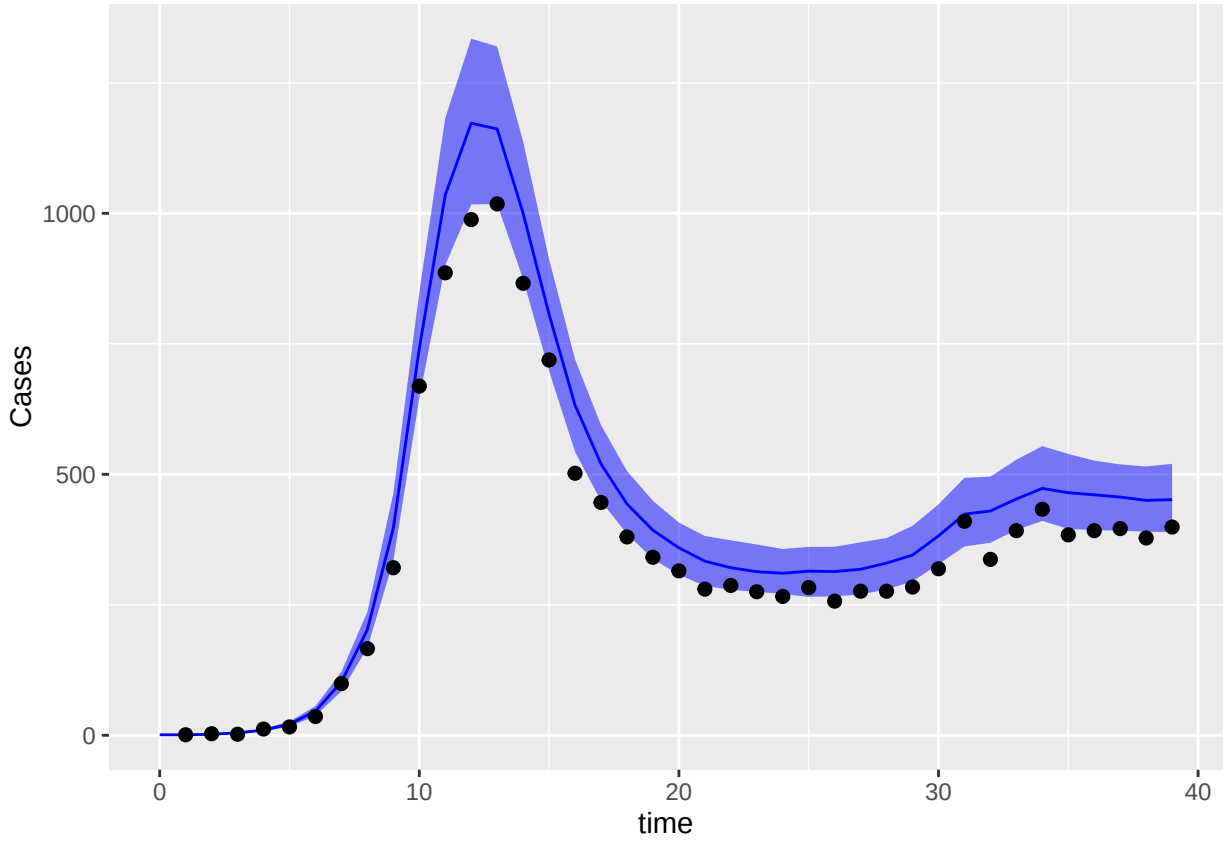


Figure 5:

If one wants to get an estimate of the smoothed state variables, the filter.traj command can be used to get samples from the distribution. The following code extracts a singel trajectory from each MCMC sample from the posterior distribution and plots the median and 95% interval at each point.

```
rawChains %>% filter.traj() %>% melt() %>%
  subset(rep > 1000 & rep %% 50 == 0) %>%
  dcast(L1+rep+time~variable) %>%
  ddply(~time,summarize,
        prob=c(0.025,0.5,0.975),
        q=quantile(RealCases,prob)) %>%
  mutate(qq=mapvalues(prob,from=c(0.025,0.5,0.975),to=c("lo","med","hi"))) %>%
  dcast(time~qq,value.var='q') %>%
  ggplot()+
  geom_ribbon(aes(x=time,ymin=lo,ymax=hi),alpha=0.5,fill='blue')+
  geom_line(aes(x=time,y=med),color='blue')+
  geom_point(data=sampleData,aes(x=time,y=Cases),color='black',size=2)+
  labs(y="Cases")
```



It is seen that the 95% confidence interval is very narrow for the raw incidence curve here, and thus suggests that it does not quanitfy uncertainity very well. Also many of the points fail to fit very well within the confidence interval, thus suggesting superficial accurate fits when the entire time series is considered.

Cumulative PMCMC

Perform a random walk markov chain monte carlo algorithm.

```
bake(file="CumData-pmcmc.rds",{
  beta_prof_cum %>% ddply(~Beta,subset,loglik==max(loglik)) %>%
    subset(beta_sd > 0.01 & beta_sd < 1.0 & Beta > 1.0 & Beta < 3.0 & mu < 1.0 &
             gamma < 2.0 & rho > 0.05 & rho < 0.99,
```

```
          select=-c(loglik,loglik.se)) -> starts
  foreach (start=iter(starts,"row"),.combine=c,
           .options.mpi=list(seed=23781975),
           .packages=c("pomp","magrittr"),.errorhandling="remove") %dopar%
           {
             cumModel %>%
               pmcmc(Nmcmc=2000,Np=200,start=unlist(start),
                     proposal=mvn.rw.adaptive(rw.sd=c(Beta=0.3,beta_sd=0.05,mu=0.05,gamma=0.1,rho=0.15,
                                              scale.start=100,shape.start=100)) -> chain
             chain %>% pmcmc(Nmcmc=10000,proposal=mvn.rw(covmat(chain)))
           }
}) -> cumChains
```

The following code attempts to determine whether the chains have converged:

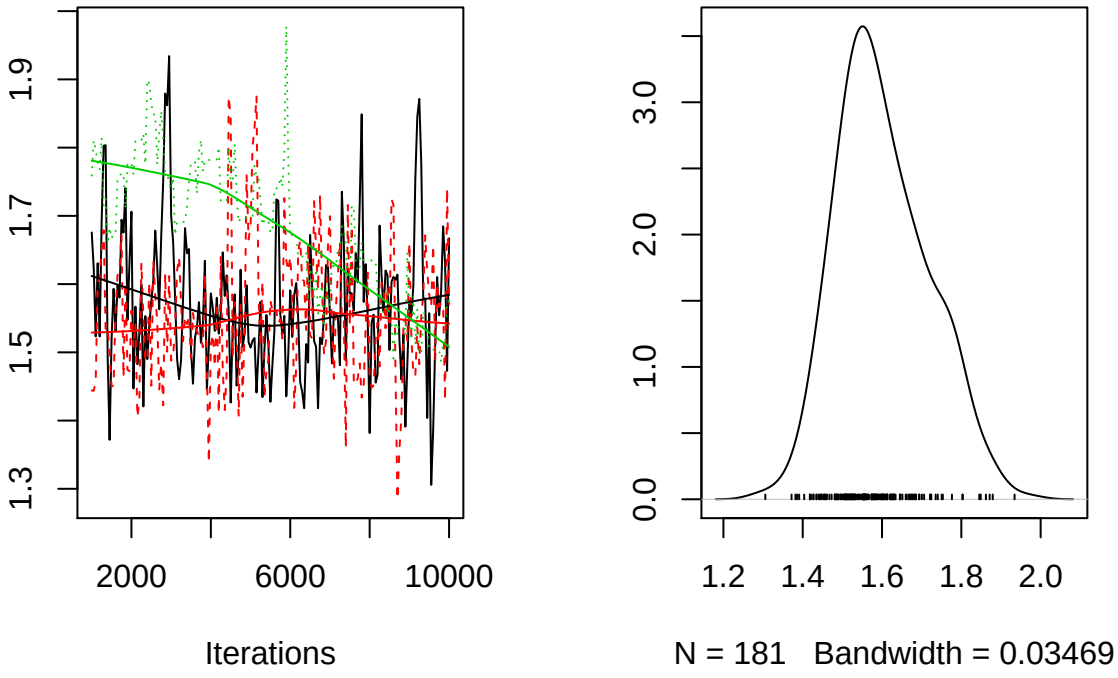
```
cumChains %>% conv.rec() -> traces
rejectionRate(traces[,c("Beta","beta_sd","mu", "gamma", "rho", "N")])
```

```
##      Beta   beta_sd        mu     gamma       rho         N
## 0.8110667 0.8110667 0.8110667 0.8110667 0.8110667 0.8110667
```

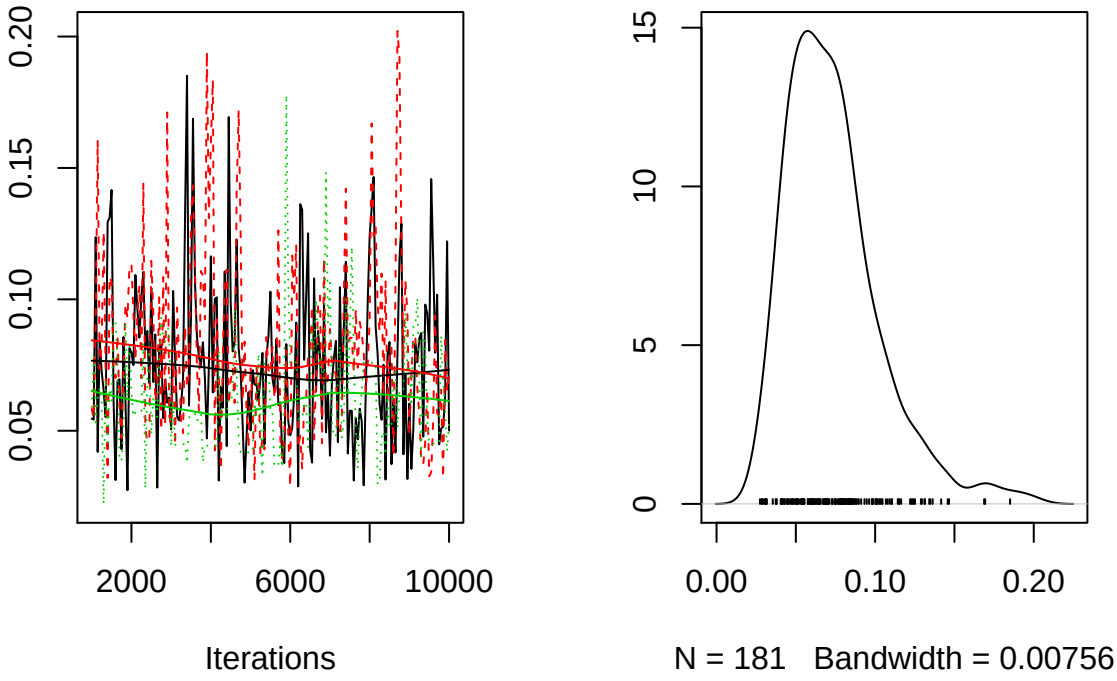
```
autocorr.diag(traces[,c("Beta","beta_sd", "mu", "gamma", "rho", "N")])
```

```
##             Beta   beta_sd        mu     gamma       rho         N
## Lag 0  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## Lag 1  0.9811259 0.9629695 0.9640327 0.9865015 0.9721973 0.9756589
## Lag 5  0.9158625 0.8407179 0.8488470 0.9379092 0.8734269 0.8878154
## Lag 10 0.8484034 0.7227384 0.7479669 0.8864169 0.7777230 0.7999368
## Lag 50 0.5716308 0.3102238 0.4425972 0.6396167 0.3776868 0.4067659
```

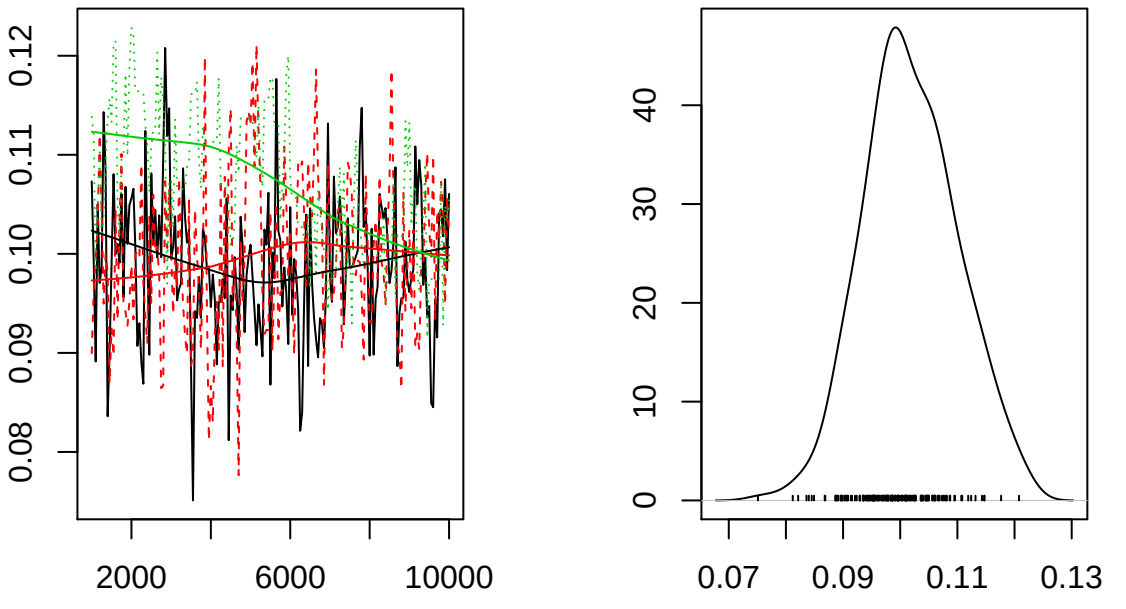
```
traces <- window(traces,thin=50,start=1000)
plot(traces[,"Beta"])
```



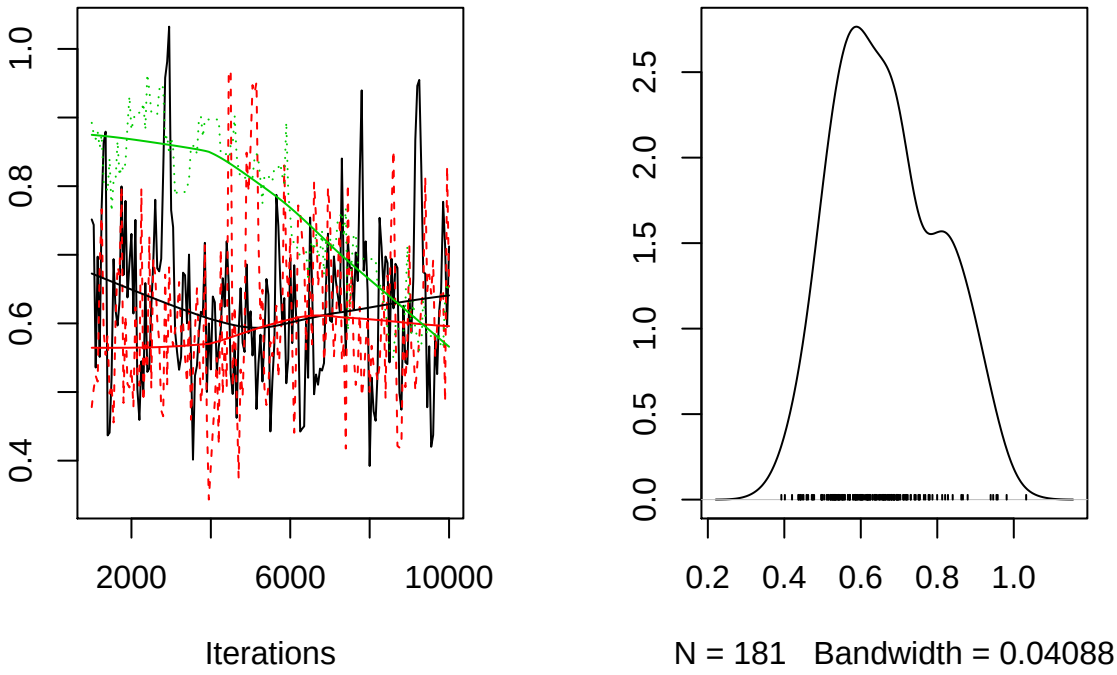
```
plot(traces[,"beta_sd"])
```



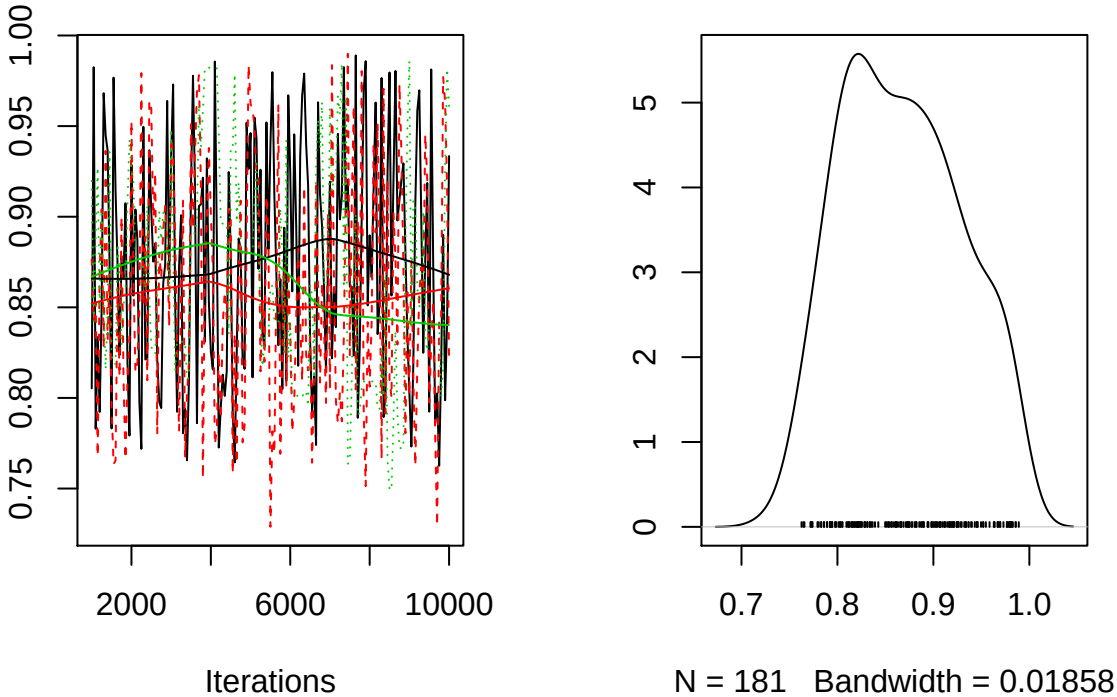

```
plot(traces[,"mu"])
```



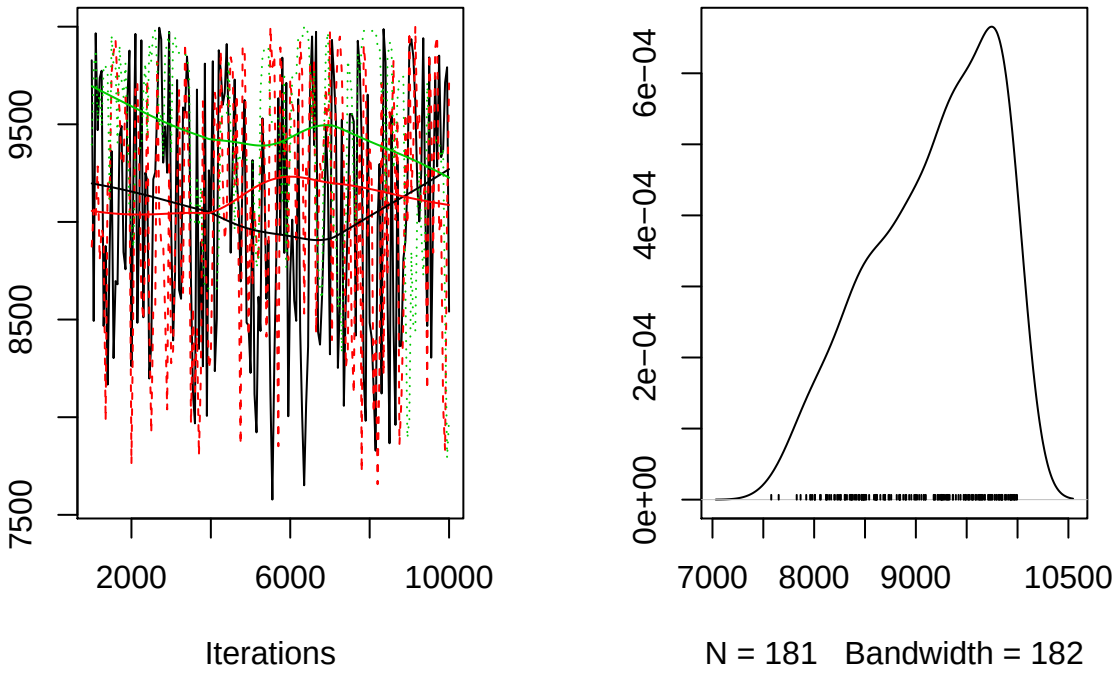

```
plot(traces[,"gamma"])
```




```
plot(traces[,"rho"])
```




```
plot(traces[,"N"])
```




```
gelman.diag(traces[,c("Beta","beta_sd","mu", "gamma", "rho", "N")])
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## Beta          1.08       1.26
## beta_sd       1.03       1.09
## mu            1.10       1.32
## gamma         1.10       1.32
## rho           1.05       1.19
## N             1.08       1.25
##
## Multivariate psrf
##
## 1.14
```

It appears that the MCMC iterations are close to converging to their stationary distributions. Let's simulate from the posterior median parameter values.

```
summary(traces[,c("Beta","beta_sd","mu", "gamma", "rho", "N")])

theta <- summary(traces)$quantiles[c("Beta","beta_sd","mu","gamma", "rho", "N"),'50%']
simulate(cumModel,params=theta,nsim=10,as.data.frame=TRUE,include.data=TRUE) %>%
  ggplot(mapping=aes(x=time,y=CumulativeCases,group=sim,alpha=(sim=="data")))+
  scale_alpha_manual(name="",values=c(`TRUE`=1,`FALSE`=0.2),
                     labels=c(`FALSE`="simulation",`TRUE`="data"))+
  geom_line()
```

If one wants to get an estimate of the smoothed state variables, the filter.traj command can be used to get samples from the distribution. The following code extracts a singel trajectory from each MCMC sample from the posterior distribution and plots the median and 95% interval at each point.

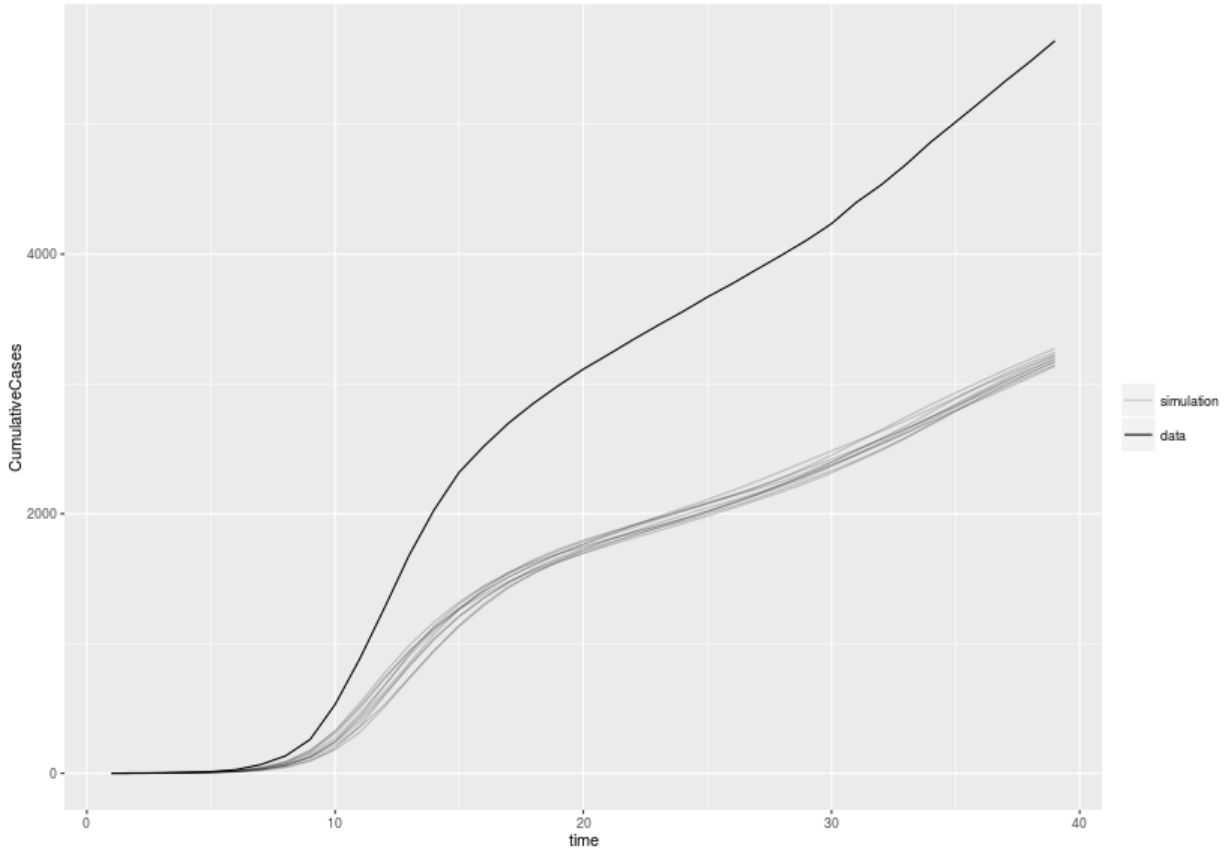
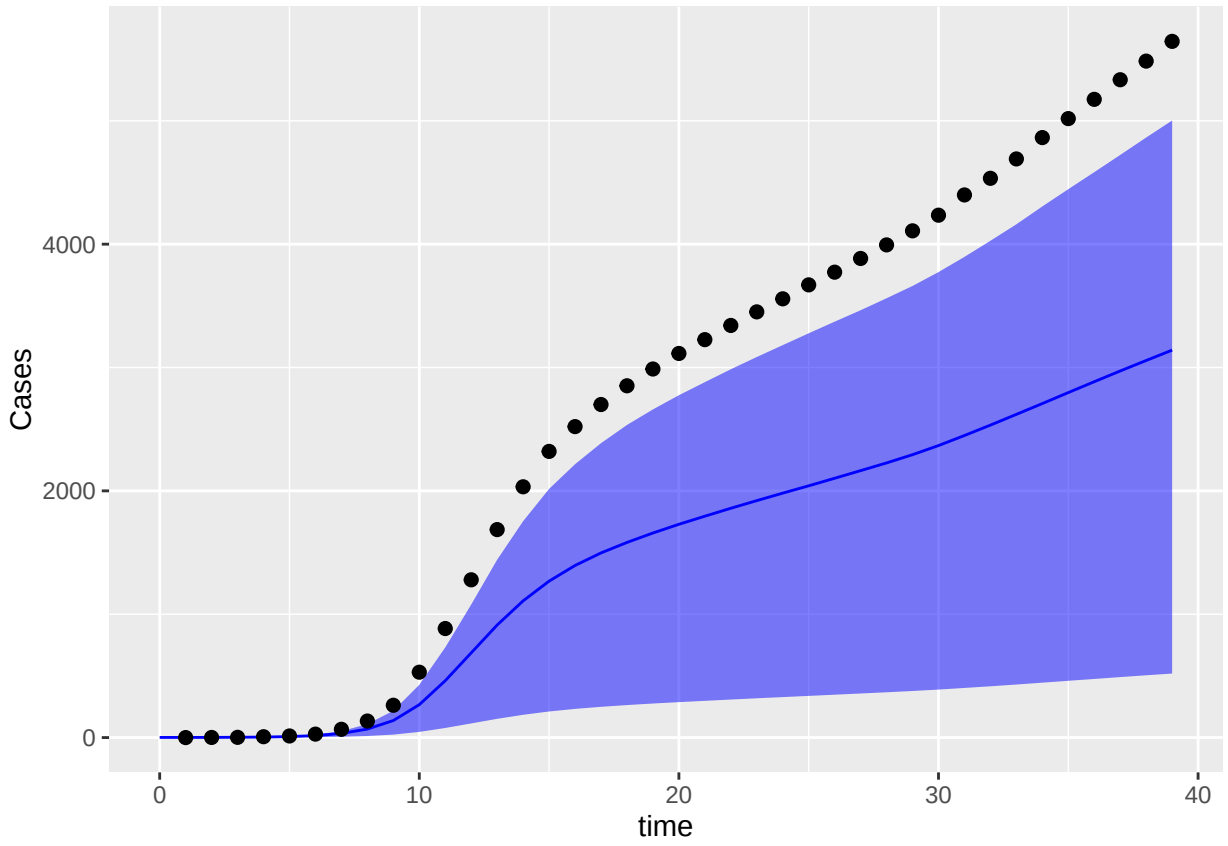

CumulativeCases
4000
2000
0
0
10
20
30
40
time
simulation
data

Figure 6:

```
cumChains %>% filter.traj() %>% melt() %>%
  subset(rep > 1000 & rep %% 50 == 0) %>%
  dcast(L1+rep+time~variable) %>%
  ddply(~time,summarize,
        prob=c(0.025,0.5,0.975),
        q=quantile(CumulativeCases,prob)) %>%
  mutate(qq=mapvalues(prob,from=c(0.025,0.5,0.975),to=c("lo","med","hi"))) %>%
  dcast(time~qq,value.var='q') %>%
  ggplot()+
  geom_ribbon(aes(x=time,ymin=lo,ymax=hi),alpha=0.5,fill='blue')+
  geom_line(aes(x=time,y=med),color='blue')+
  geom_point(data=cumulativeSampleData,aes(x=time,y=actualCumulative),color='black',size=2)+
  labs(y="Cases")
```



It is seen that the 95% confidence interval is very robust for the cumulative incidence curve here, and thus suggests that it quanitfies uncertainity very well. However, many of the points fail to fit very well within the confidence interval, and it is seen that the estimations under predict the true incidence.

Discussion

It appears that even when one includes the entire time-series of an epidemic, not just the takeoff, that fitting deterministic models to cumulative data will more accurately quantify uncertainity, but will often underpredict the true incidence. Raw models will fit very well to the data, but one risks the fact that its variance is not very high, and thus does not quanitfy uncertainity in the parameter values. It should be noted that in this case the raw model does actually predict the parameters fairly well due to the fact that we know the parameters used to simulate the epidemic data. When one considers fitting cumulative and raw

data with stochastic methods, you see that the cumulative incidence curves starts to lose its variance in the simulated fits to the data, but still produces wider confidence intervals for parameter estimates than raw incidence, again quantifying uncertainity more accurately. However, you again see that cumulative incidence fits tend to under predict the true incidence. Raw data fits very well with stochastic methods but appears to have lower achieved confience interval coverage. In conclusion, when one considers the entire time-series of an epidemic, cumulative incidence will quanitfy uncertainity in parameter values better than raw data, but you will get closer fits and closer counts to actual incidene with raw data.

References

1. King AA, Ionides EL, Bretó CM, Ellner SP, Ferrari MJ, Kendall BE, et al. pomp: Statistical inference for partially observed Markov processes [Internet]. 2016. Available: http://kingaa.github.io/pomp

2. King AA, de Celles MD, Magpantay FMG, Rohani P. Avoidable errors in the modelling of outbreaks of emerging pathogens, with special reference to ebola. Proceedings of the Royal Society B: Biological Sciences. The Royal Society; 2015;282: 20150347–20150347. doi:10.1098/rspb.2015.0347

3. Grad YH, Miller JC, Lipsitch M. Cholera modeling. Epidemiology. Ovid Technologies (Wolters Kluwer Health); 2012;23: 523–530. doi:10.1097/ede.0b013e3182572581

4. Ionides EL, Nguyen D, é YA, Stoev S, King AA. Inference for dynamic and latent variable models via iterated, perturbed bayes maps. Proceedings of the National Academy of Sciences. Proceedings of the National Academy of Sciences; 2015;112: 719–724. doi:10.1073/pnas.1410597112