STAT 656: HW1

Haoze Li

2023-09-03

Synthetic data

1

By definition, we have the conditional distributions

$$y_i|y_{i-1} \sim \mathcal{N}(\rho y_{i-1}, \sigma^2),\ \forall i = 1, \dots, n$$

Using Bayes' formula, we can decompose the joint distribution of $(y_0, y_1, \dots, y_n)$ as

$$p(y_0, y_1, \dots, y_n) = p(y_1|y_0)p(y_2|y_1, y_0)\dots p(y_n|y_0, \dots, y_{n-1})$$

Since y_i is independent of $y_0, y_1, \dots, y_{i-2}$, we can simplify the joint distribution as follows:

$$p(y_0, y_1, \dots, y_n) = \prod_{i=1}^{n} p(y_i|y_{i-1})$$

Therefore, the log-likelihood function is

$$L(\rho, \sigma^2|y_0, y_1, \dots, y_n) = \sum_{i=1}^{n} \log p(y_i|y_{i-1})$$

$$= \sum_{i=1}^{n} \left(-\frac{1}{2} \log 2\pi\sigma^2 - \frac{(y_i - \rho y_{i-1})^2}{2\sigma^2} \right)$$

$$= -\frac{n}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \rho y_{i-1})^2$$

2

First we import data:

```r
d1 <- read.csv('computation_data_hw_1.csv')
head(d1)
```

```
##   X           x
## 1 1  0.00000000
## 2 2  0.76224052
## 3 3  0.09081971
## 4 4  0.15237869
## 5 5  0.53305524
## 6 6 -0.43105303
```

The log-likelihood function we obtained previously can be rewritten as

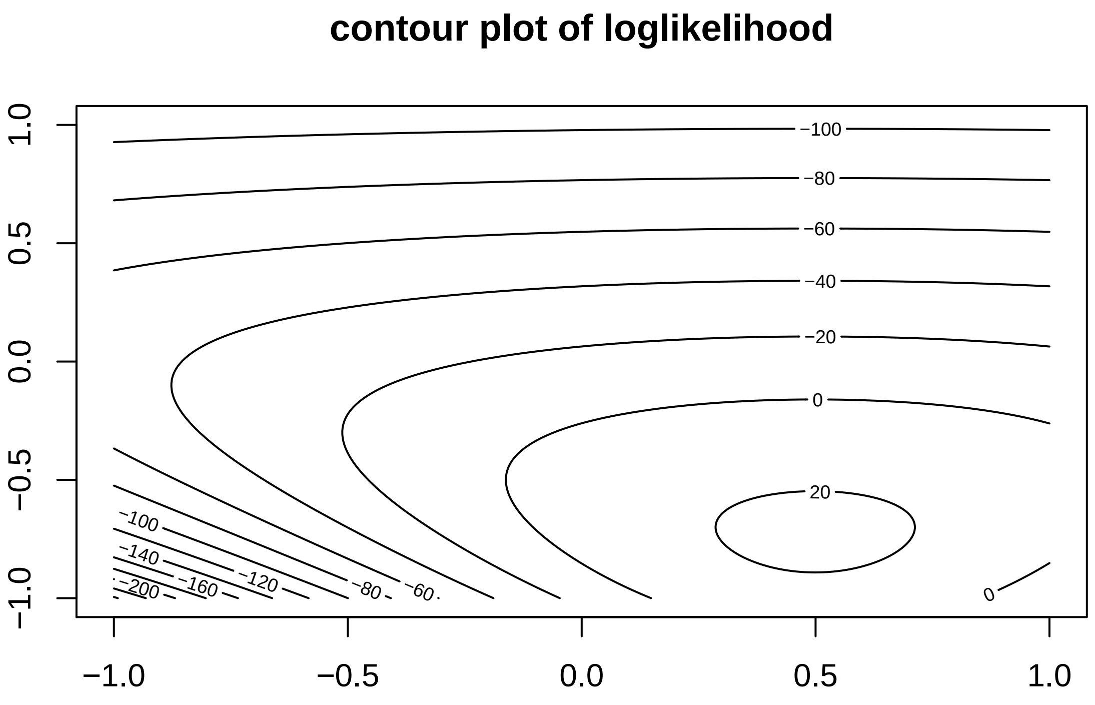
$$L(\rho, \log \sigma) = -n \log \sigma - \frac{1}{2e^{2 \log \sigma}} \sum_{i=1}^{n} (y_i - \rho y_{i-1})^2$$

Thus, we can implement the function for calculating $L(\rho, \log \sigma)$ as follows:

```r
ar_loglik <- function(rho, sig, d){
    n <- nrow(d)
    s <- 0
    for(i in 1:(n - 1)){
        s <- s + (d$x[i + 1] - rho * d$x[i])^2
    }
    -n * sig - 0.5 * exp(-2 * sig) * s
}
```

We can visualize it as a contour plot:

```r
rho <- seq(-1, 1, length.out = 500)
sig <- seq(-1, 1, length.out = 500)
loglik = outer(rho, sig, ar_loglik, d1)
contour(rho, sig, loglik, nlevels=10, main="contour plot of loglikelihood")
```



3

Note that the log of the posterior function is equal to the sum of log-likelihood function and log of prior (up to a constant), we can thus write the R function as follows (We omit any constant here). To show the work, we can derive the fomula as follows:

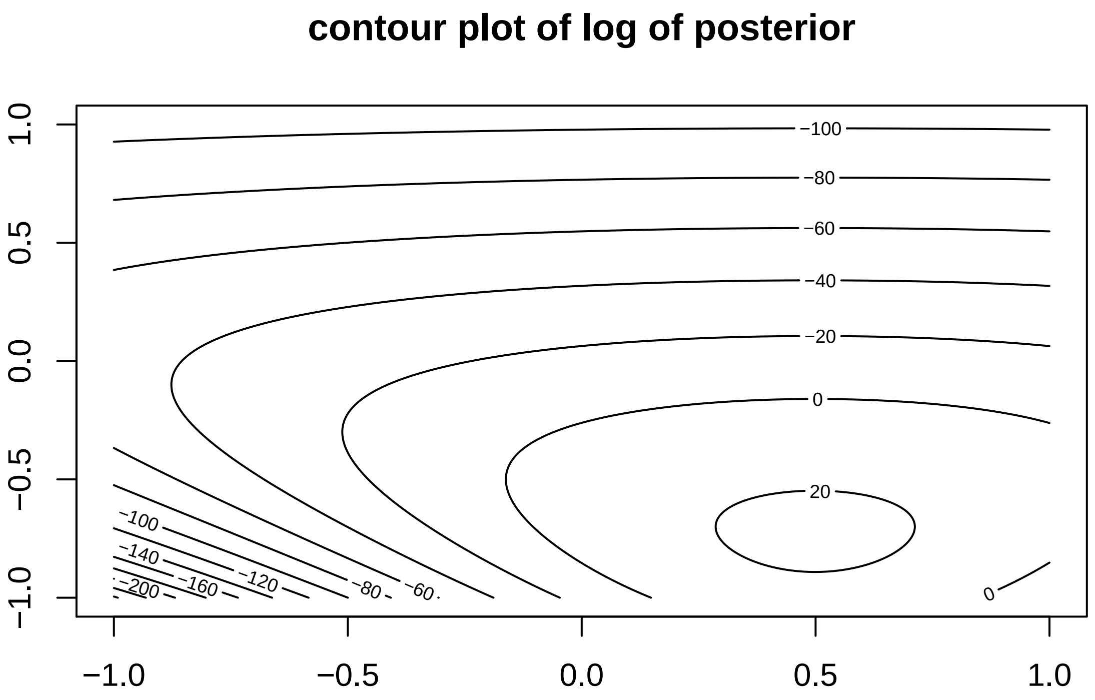
$$\log p(\rho, \log \sigma | y_0, y_1, \ldots, y_n) = \log p(\rho) + \log p(\log \sigma) + L(\rho, \log \sigma | y_0, y_1, \ldots, y_n)$$

$$= -\frac{(\log \sigma)^2}{200} + L(\rho, \log \sigma | y_0, y_1, \ldots, y_n) + \text{constant}$$

```
ar_logpost <- function(rho, sig, d){
    ar_loglik(rho, sig, d) - (sig^2 / 200)
}
```

Again, we can visualize it with contour plot:

```
logpost = outer(rho, sig, ar_logpost, d1)
contour(rho, sig, logpost, nlevels=10, main="contour plot of log of posterior")
```



The difference between the log of posterior and log-likelohood function is not obvious in this example. So we can say this prior is uninformative. In fact, the prior of ρ is uniform distribution on $(-1, 1)$, which just represents a random guess for all possible values of ρ (and thus is uninformative). The variance of the prior of $\log\sigma$ is 100, which is very large and thus is uninformative as well.

5

In previous question, we have got a vector of posterior density on the discrete grid obtained by split $(-1, 1) \times (-1, 1)$ into 25000 parts. Now, we can sample from this grid approximation to the posterior.

```
set.seed(1)
idx <- sample(seq(1, length(logpost)), size=1000, replace=TRUE,
                    prob=exp(c(logpost)) / sum(exp(c(logpost))))
d1_sample <- expand.grid(rho, sig)[idx,]
names(d1_sample) <- c('rho', 'logsigma')
head(d1_sample)
```

```
##             rho   logsigma
## 24388 0.5511022 -0.8076152
## 49425 0.6993988 -0.6072144
## 31389 0.5551102 -0.7515030
## 40403 0.6112224 -0.6793587
## 46897 0.5871743 -0.6272545
## 37350 0.3987976 -0.7034068
```

7

First we calculate the quantiles:

```
# quantiles of rho
(d1_quantiles_rho <- quantile(d1_sample$rho,
                              probs=c(0.025, 0.25, 0.5, 0.75, 0.975)))
```

```
##      2.5%       25%       50%       75%     97.5% 
## 0.3146293 0.4348697 0.4989980 0.5551102 0.6793587
```

```
# quantiles of logsigma
(d1_quantiles_logsigma <- quantile(d1_sample$logsigma,
                                   probs=c(0.025, 0.25, 0.5, 0.75, 0.975)))
```

```
##       2.5%        25%        50%        75%      97.5% 
## -0.8677355 -0.7715431 -0.7194389 -0.6713427 -0.5791583
```

Then we can get the mean, median and standard deviation:

```
# means
(d1_mean_rho <- mean(d1_sample$rho))
```

```
## [1] 0.4963808
```

```
(d1_mean_logsigma <- mean(d1_sample$logsigma))
```

```
## [1] -0.7194629
```

```
# medians
(d1_median_rho <- median(d1_sample$rho))
```

```
## [1] 0.498998
```

```
(d1_median_logsigma <- median(d1_sample$logsigma))
```

```
## [1] -0.7194389
```

```
# standard deviations
(d1_sd_rho <- sd(d1_sample$rho))
```

```
## [1] 0.09155298
```

```
(d1_sd_logsigma <- sd(d1_sample$logsigma))
```

```
## [1] 0.07316152
```

Last, we can get the skewness and kurtosis:

```
# install.packages("moments")
library(moments)

# skewnesses
(d1_skewness_rho <- skewness(d1_sample$rho))
```

```
## [1] 0.1190664
```

```
(d1_skewness_logsigma <- skewness(d1_sample$logsigma))
```

```
## [1] 0.01033475
```

```
# kurtosis
(d1_kurtosis_rho <- kurtosis(d1_sample$rho))
```

```
## [1] 3.526542
```

```
(d1_kurtosis_logsigma <- kurtosis(d1_sample$logsigma))
```

```
## [1] 2.840722
```

8

The sampler can be written as follows:

```
n <- nrow(d1) # time points of the AR process

ar_sampler <- function(size=1000, parsize=1000, rho, sig, post, length, d, init_value=0){
    logpost = outer(rho, sig, post, d)
    res <- data.frame(matrix(ncol = length, nrow = 0))
    for(i in 1:size){
        idx <- sample(seq(1, length(logpost)), size=parsize, replace=TRUE,
                      prob=exp(c(logpost)) / sum(exp(c(logpost))))
        d1_sample <- expand.grid(rho, sig)[idx,]
        names(d1_sample) <- c('rho', 'logsigma')
        mean_rho <- mean(d1_sample$rho)
        mean_sigma <- exp(mean(d1_sample$logsigma))
        newrow <- c(init_value)
        for(j in 1:length){
            eps <- rnorm(1, sd=mean_sigma)
            newrow <- append(newrow, newrow[j - 1] * mean_rho + eps)
        }
        res[i, ] <- newrow
    }
    res
}
```

then we can sample from with the sampler:

```
y_rep <- ar_sampler(rho=rho, sig=sig, post=ar_logpost, length=n, d=d1)
# summary(y_rep)
```

Here, we first sample 1000 $(\rho, \log \sigma)$ pairs from the posterior distribution and then get their means. Given these means as parameters of the AR(1) process, we sample y_i's sequentially by setting $y_0 = 0$. We drew 1000 y^{rep} by repeating this procedure.

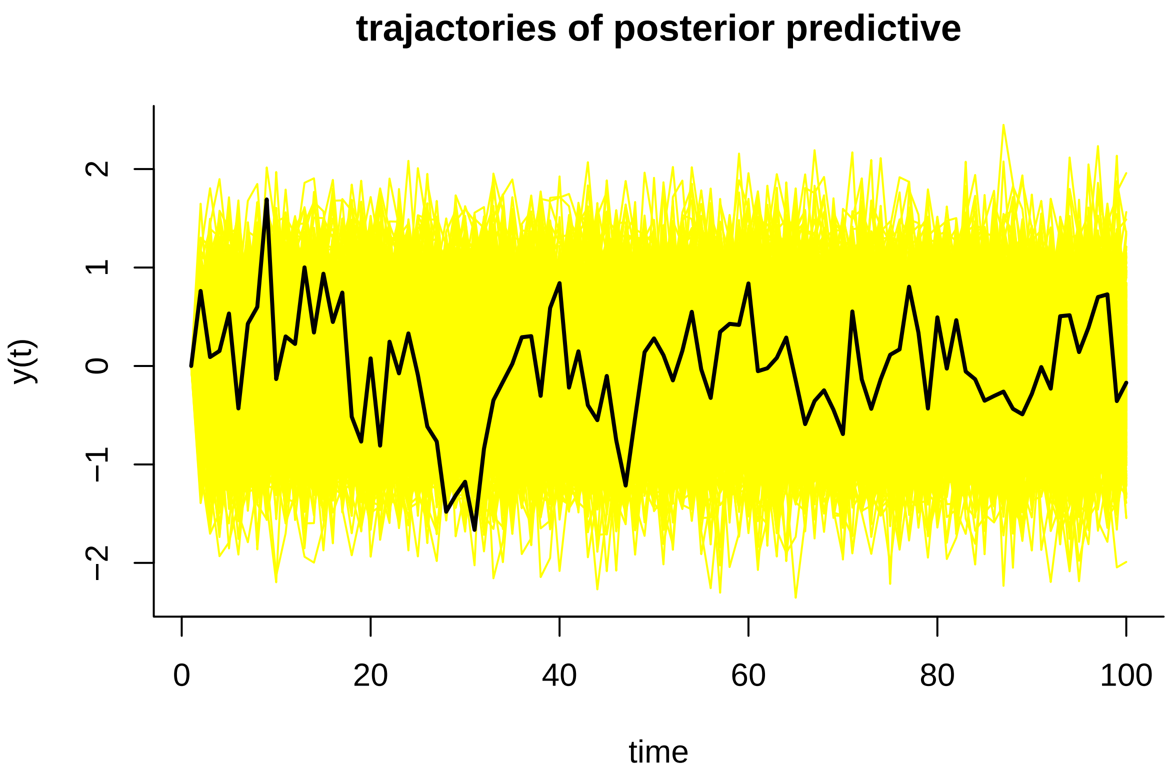
9

We plot the trajectories for each sampled AR(1) process:

```
plot(c(1, 100), c(min(y_rep), max(y_rep)), bty ='l', type="n",
     xlab='time', ylab='y(t)',
     main="trajectories of posterior predictive")
apply(y_rep[1:nrow(y_rep),], 1,
               function(a) {lines(a, col="yellow")})
```

```
## NULL
```

```
# original trajectory
lines(d1$x, col="black", lwd=2)
```



We can see from the plot that the model fit is overall satisfactory. And thus we concude that the model is approriate.

Real data

1

It is sufficient for analyzing the data because we have some basic information of the target variable. However, if I had the resources to collect more information, I would ask if they are vaccined.

2

Since the range of the data is too wide, I would first use a log tranformation on every non zero data point and then fit an AR(1) model. The model can be written as:

$$\log y_i = \rho \log y_{i-1} + \epsilon_i$$

for $i = 1, \ldots, n$ and $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ independently.

3

From the dataset, we observe that y_i is always larger than y_{i-1}. Thus, ρ is probably positive. Also, since we don't have any knowledge for σ^2, we can use a Gaussian prior as we had in synthetic data part before. Concretely, the priors we decide to use here is

$$\rho \sim \text{Uniform}(0, 1); \quad \log \sigma \sim \mathcal{N}(0, 10^2)$$

4

Now we fit the model using the functions we got in sythetic data part:

```r
d2 <- read.csv("covid_us.txt")

# transform data
d2$cases <- log(d2$cases)
d2$deaths[d2$deaths > 0] <- log(d2$deaths[d2$deaths > 0])
d2_val <- d2[which(d2$date == "2020-07-01"):nrow(d2),]
d2 <- d2[1:which(d2$date == "2020-06-30"),]
head(d2)
```

```
##         date     cases deaths
## 1 2020-01-21 0.0000000      0
## 2 2020-01-22 0.0000000      0
## 3 2020-01-23 0.0000000      0
## 4 2020-01-24 0.6931472      0
## 5 2020-01-25 1.0986123      0
## 6 2020-01-26 1.6094379      0
```

Note that the log-likelihood function is exactly the same as the one in sythetic data case, only with the transformed data. And the prior distribution is also proportional to the prior distribution before. Hence, we can get the posterior distribution for this example up to a constant, which is

$$\log p(\rho, \log\sigma | y_0, y_1, \ldots, y_n) = \log p(\rho) + \log p(\log\sigma) + L(\rho, \log\sigma | y_0, y_1, \ldots, y_n)$$

$$= -\frac{(\log\sigma)^2}{200} + L(\rho, \log\sigma | \log y_0, \log y_1, \ldots, \log y_n) + \text{constant}$$

where

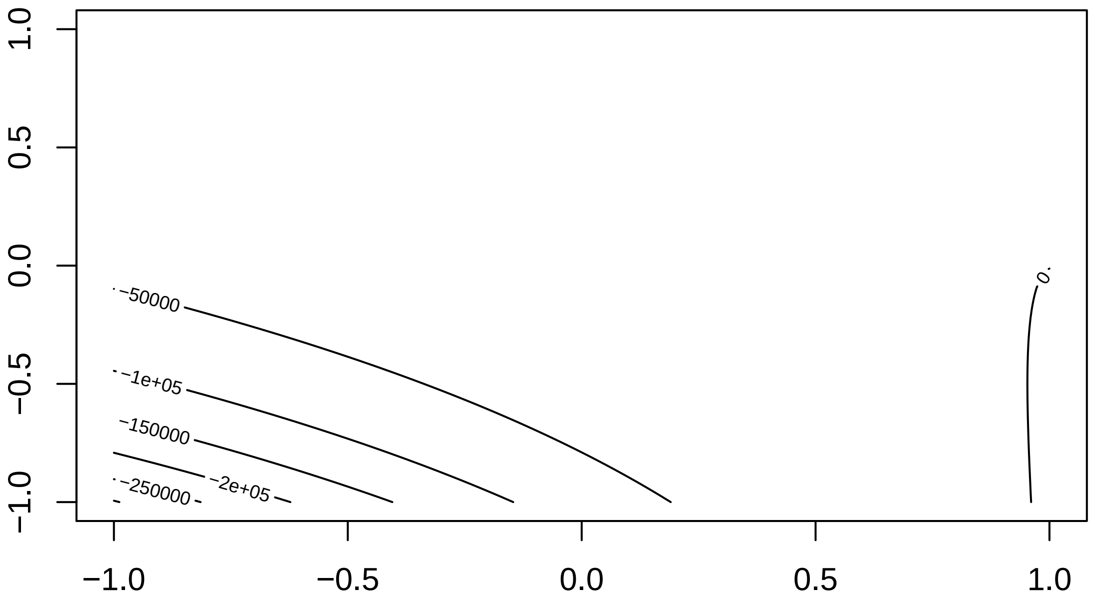
$$L(\rho, \log\sigma) = -n\log\sigma - \frac{1}{2e^{2\log\sigma}}\sum_{i=1}^{n}(\log y_i - \rho\log y_{i-1})^2$$

We can implement the log of posterior as follows:

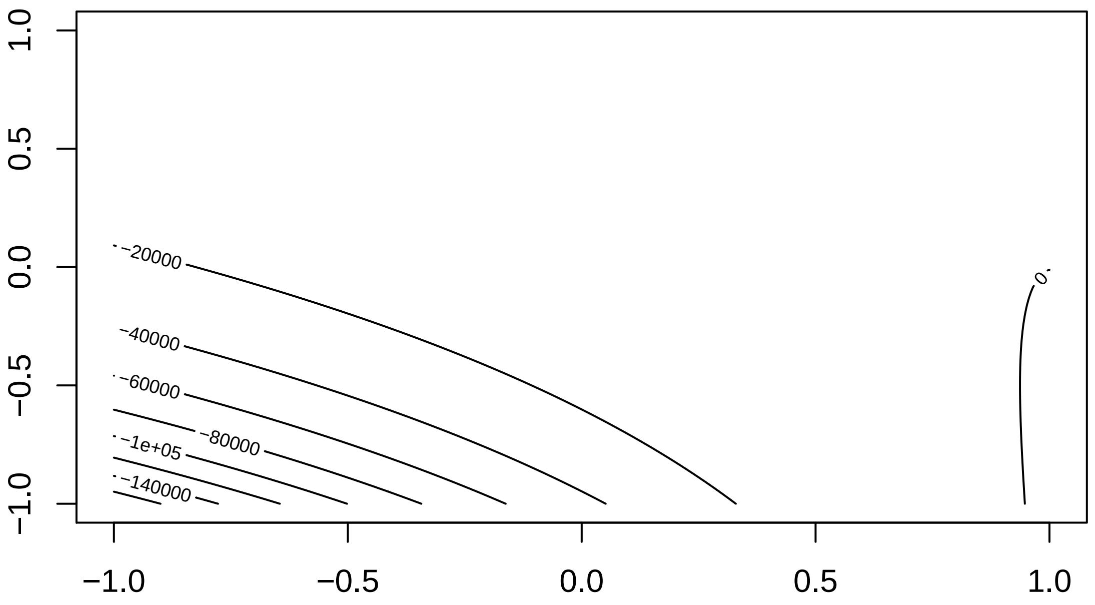
```r
ar_logpost_cases <- function(rho, sig, d){
    n <- nrow(d)
    s <- 0
    for(i in 1:(n - 1)){
        s <- s + (d$cases[i + 1] - rho * d$cases[i])^2
    }
    -n * sig - 0.5 * exp(-2 * sig) * s - (sig^2 / 200)
}

ar_logpost_deaths <- function(rho, sig, d){
    n <- nrow(d)
    s <- 0
    for(i in 1:(n - 1)){
        s <- s + (d$deaths[i + 1] - rho * d$deaths[i])^2
    }
    -n * sig - 0.5 * exp(-2 * sig) * s - (sig^2 / 200)
}
```

We can visualize both of them as follows:

```r
rho <- seq(-1, 1, length.out = 500)
sig <- seq(-1, 1, length.out = 500)
logpost_cases = outer(rho, sig, ar_logpost_cases, d2)
contour(rho, sig, logpost_cases, nlevels=10)
```



```r
logpost_deaths <- outer(rho, sig, ar_logpost_deaths, d2)
contour(rho, sig, logpost_deaths, nlevels=10)
```



Now we can sample from the posterior distribution:

```r
set.seed(1)
idx_cases <- sample(seq(1, length(logpost_cases)), size=1000, replace=TRUE,
                    prob=exp(c(logpost_cases)) / sum(exp(c(logpost_cases))))
cases_sample <- expand.grid(rho, sig)[idx_cases,]
names(cases_sample) <- c('rho', 'logsigma')
head(cases_sample)
```

```
##        rho   logsigma
## 500      1 -1.0000000
## 500.1    1 -1.0000000
## 1000     1 -0.9959920
## 2500     1 -0.9839679
## 500.2    1 -1.0000000
## 2500.1   1 -0.9839679
```

```
idx_deaths <- sample(seq(1, length(logpost_deaths)), size=1000, replace=TRUE,
                    prob=exp(c(logpost_deaths)) / sum(exp(c(logpost_deaths))))
deaths_sample <- expand.grid(rho, sig)[idx_deaths,]
names(deaths_sample) <- c('rho', 'logsigma')
head(deaths_sample)
```

```
##        rho   logsigma
## 1000     1 -0.9959920
## 1500     1 -0.9919840
## 500      1 -1.0000000
## 3500     1 -0.9759519
## 500.1    1 -1.0000000
## 500.2    1 -1.0000000
```

First we calculate the quantiles:

```
# quantiles of rho
(cases_quantiles_rho <- quantile(cases_sample$rho,
                                probs=c(0.025, 0.25, 0.5, 0.75, 0.975)))
```

```
##  2.5%   25%   50%   75% 97.5%
##     1     1     1     1     1
```

```
# quantiles of logsigma
(cases_quantiles_logsigma <- quantile(cases_sample$logsigma,
                                     probs=c(0.025, 0.25, 0.5, 0.75, 0.975)))
```

```
##       2.5%        25%        50%        75%      97.5%
## -1.0000000 -1.0000000 -0.9959920 -0.9919840 -0.9758517
```

```
(deaths_quantiles_rho <- quantile(deaths_sample$rho,
                                 probs=c(0.025, 0.25, 0.5, 0.75, 0.975)))
```

```
##     2.5%      25%      50%      75%    97.5%
## 0.995992 1.000000 1.000000 1.000000 1.000000
```

```
# quantiles of logsigma
(deaths_quantiles_logsigma <- quantile(deaths_sample$logsigma,
                                      probs=c(0.025, 0.25, 0.5, 0.75, 0.975)))
```

```
##       2.5%        25%        50%        75%      97.5%
## -1.0000000 -1.0000000 -0.9959920 -0.9919840 -0.9759519
```

Then we can get the mean, median and standard deviation:

```
# cases
# means
(cases_mean_rho <- mean(cases_sample$rho))
```

```
## [1] 0.9999519
```

```
(cases_mean_logsigma <- mean(cases_sample$logsigma))
```

```
## [1] -0.9942285
```

```
# medians
(cases_median_rho <- median(cases_sample$rho))
```

```
## [1] 1
```

```r
(cases_median_logsigma <- median(cases_sample$logsigma))
```

```
## [1] -0.995992
```

```r
# standard deviations
(cases_sd_rho <- sd(cases_sample$rho))
```

```
## [1] 0.0004366322
```

```r
(cases_sd_logsigma <- sd(cases_sample$logsigma))
```

```
## [1] 0.00780297
```

```r
# deaths
# means
(deaths_mean_rho <- mean(deaths_sample$rho))
```

```
## [1] 0.9996914
```

```r
(deaths_mean_logsigma <- mean(deaths_sample$logsigma))
```

```
## [1] -0.9946493
```

```r
# medians
(deaths_median_rho <- median(deaths_sample$rho))
```

```
## [1] 1
```

```r
(deaths_median_logsigma <- median(deaths_sample$logsigma))
```

```
## [1] -0.995992
```

```r
# standard deviations
(deaths_sd_rho <- sd(deaths_sample$rho))
```

```
## [1] 0.001083975
```

```r
(deaths_sd_logsigma <- sd(deaths_sample$logsigma))
```

```
## [1] 0.007015561
```

Last, we can get the skewness and kurtosis:

```r
# cases
# skewnesses
(cases_skewness_rho <- skewness(cases_sample$rho))
```

```
## [1] -8.963564
```

```r
(cases_skewness_logsigma <- skewness(cases_sample$logsigma))
```

```
## [1] 2.408139
```

```r
# kurtosis
(cases_kurtosis_rho <- kurtosis(cases_sample$rho))
```

```
## [1] 81.34548
```

```r
(cases_kurtosis_logsigma <- kurtosis(cases_sample$logsigma))
```

```
## [1] 12.88046
```

```r
# deaths
# skewnesses
(deaths_skewness_rho <- skewness(deaths_sample$rho))
```

```
## [1] -3.324376
```

```r
(deaths_skewness_logsigma <- skewness(deaths_sample$logsigma))
```

```
## [1] 1.948389
```

```r
# kurtosis
(deaths_kurtosis_rho <- kurtosis(deaths_sample$rho))
```

```
## [1] 12.76197
```

```r
(deaths_kurtosis_logsigma <- kurtosis(deaths_sample$logsigma))
```

```
## [1] 8.659671
```

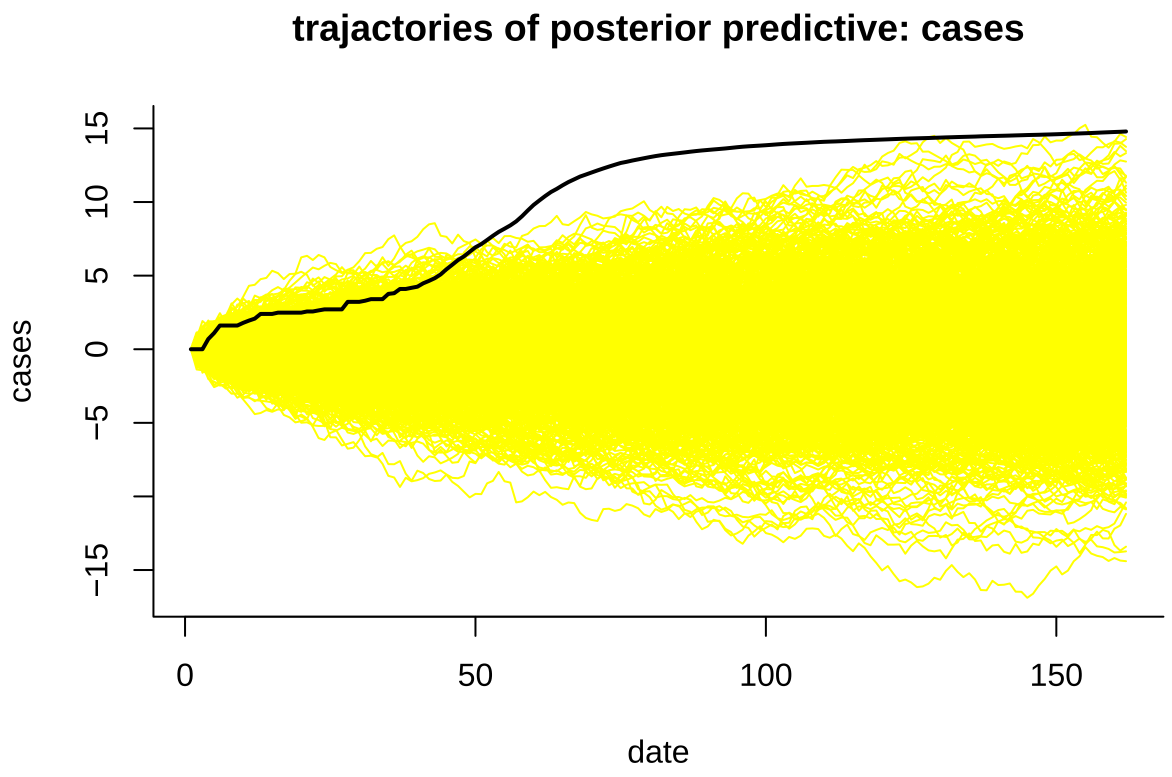
5

Let's conduct posterior predictive check now:

```r
cases_rep <- ar_sampler(rho=rho, sig=sig, post=ar_logpost_cases, length=nrow(d2), d=d2)
deaths_rep <- ar_sampler(rho=rho, sig=sig,post=ar_logpost_deaths, length=nrow(d2), d=d2)

plot(c(1, nrow(d2)), c(min(cases_rep), max(cases_rep)), bty ='l', type="n",
     xlab='date', ylab='cases',
     main="trajactories of posterior predictive: cases")
apply(cases_rep[1:nrow(cases_rep),], 1,
              function(a) {lines(a, col="yellow")})
```

```
## NULL
```

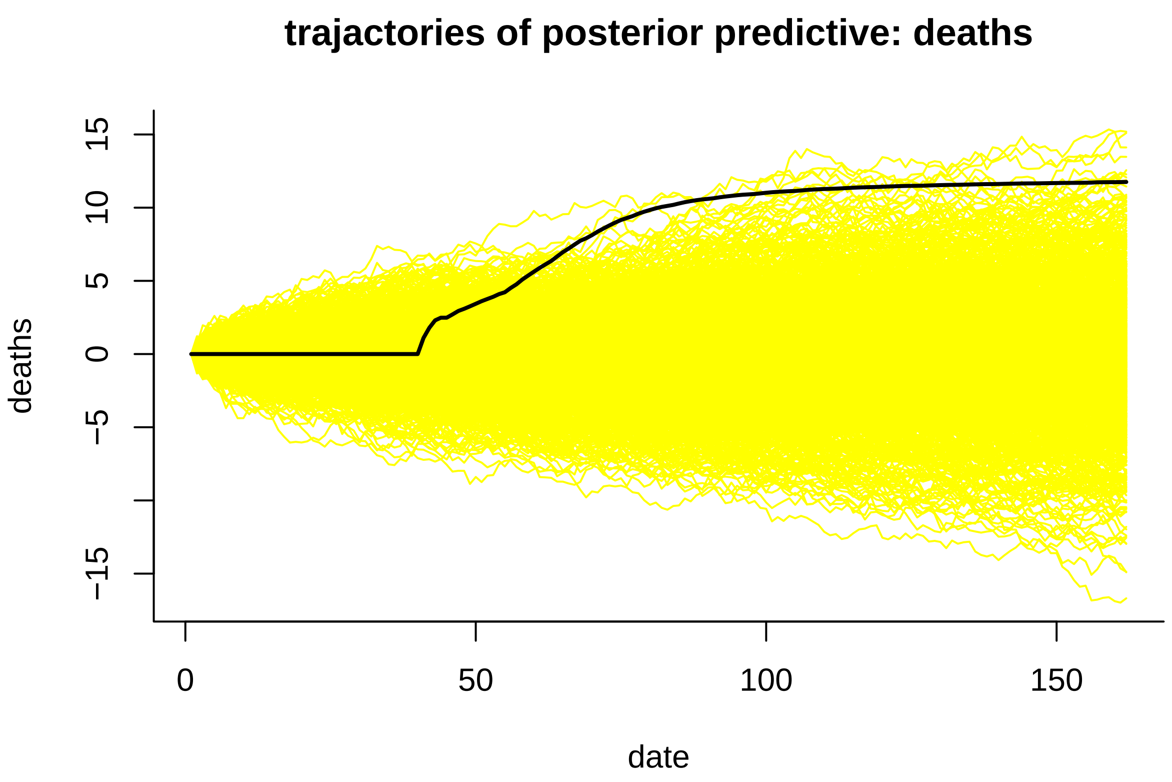
```r
# original trajectory
lines(d2$cases, col="black", lwd=2)
```



```r
plot(c(1, nrow(d2)), c(min(deaths_rep), max(deaths_rep)), bty ='l', type="n",
     xlab='date', ylab='deaths',
     main="trajactories of posterior predictive: deaths")
apply(deaths_rep[1:nrow(cases_rep),], 1,
              function(a) {lines(a, col="yellow")})
```

```
## NULL
```

```r
# original trajectory
lines(d2$deaths, col="black", lwd=2)
```



From the two plots, we can see that the model does not provide good predictions.One of the reasons is that the AR(1) process cannot capture the increasing trend successfully. One might need to add a bias term into the model.

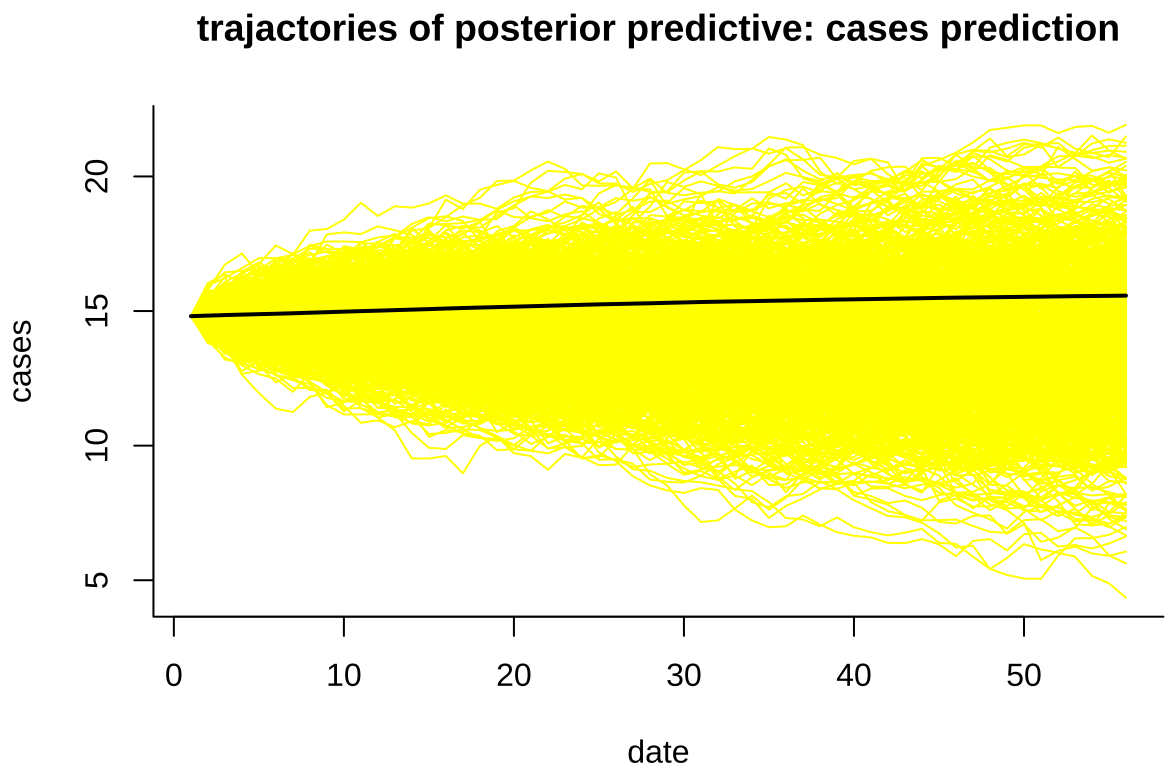
6

We can make posterior predictions with the dataset `d2_val`.

```r
cases_val <- ar_sampler(rho=rho, sig=sig,
                        post=ar_logpost_cases, length=nrow(d2_val),
                        d=d2_val, init_value=d2_val$cases[1])
deaths_val <- ar_sampler(rho=rho, sig=sig,
                         post=ar_logpost_deaths, length=nrow(d2_val),
                         d=d2_val, init_value=d2_val$deaths[1])

plot(c(1, nrow(d2_val)), c(min(cases_val), max(cases_val)), bty ='l', type="n",
     xlab='date', ylab='cases',
     main="trajectories of posterior predictive: cases prediction")
apply(cases_val[1:nrow(cases_val),], 1,
              function(a) {lines(a, col="yellow")})
```

```
## NULL
```

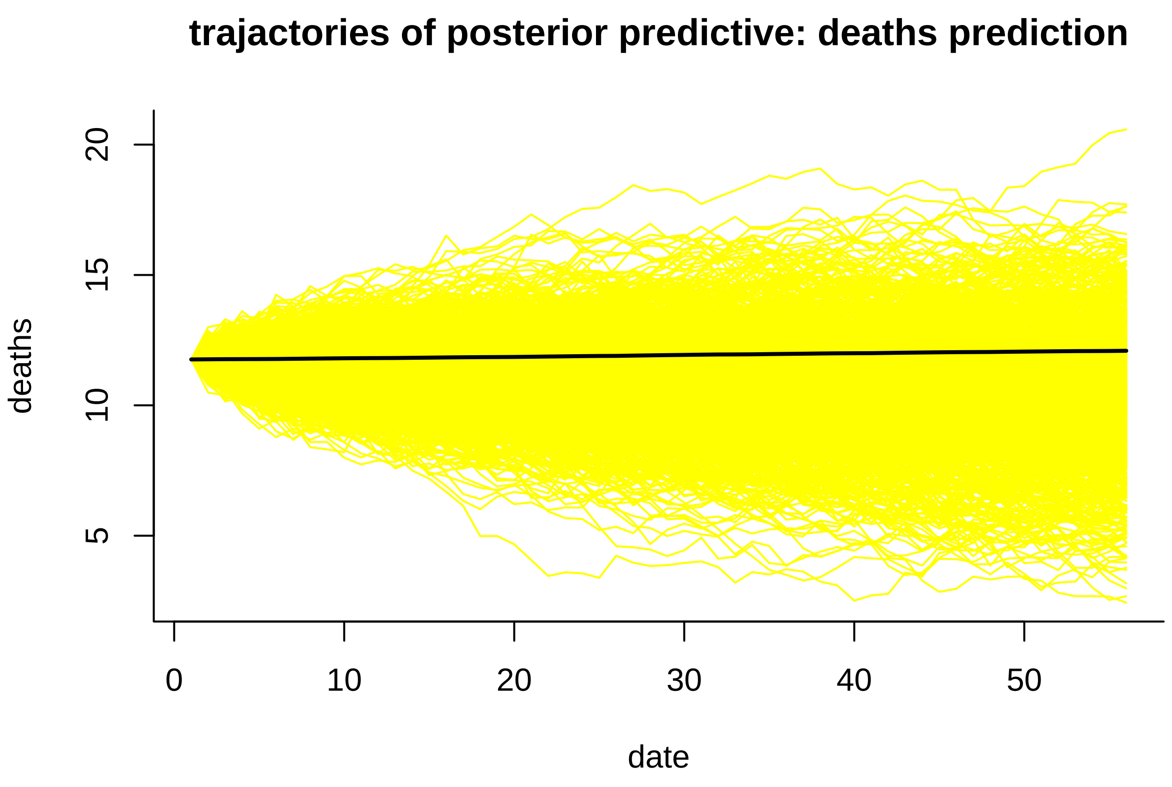
```r
# original trajectory
lines(d2_val$cases, col="black", lwd=2)
```



```r
plot(c(1, nrow(d2_val)), c(min(deaths_val), max(deaths_val)), bty ='l', type="n",
     xlab='date', ylab='deaths',
     main="trajactories of posterior predictive: deaths prediction")
apply(deaths_val[1:nrow(deaths_val),], 1,
              function(a) {lines(a, col="yellow")})
```

```
## NULL
```

```r
# original trajectory
lines(d2_val$deaths, col="black", lwd=2)
```



As we can see, the prediction for the future is not satisfactory. Part of the reason is the regression coefficient ρ changes (decreases) over time.

7

Import dataset:

```
d3 <- read.csv("covid_us-states.txt")
unique(d3$state)
```

```
## [1] "California" "Texas"      "Indiana"
```

```
d3 <- d3[1:which(d3$date == "2020-06-30"),]
```

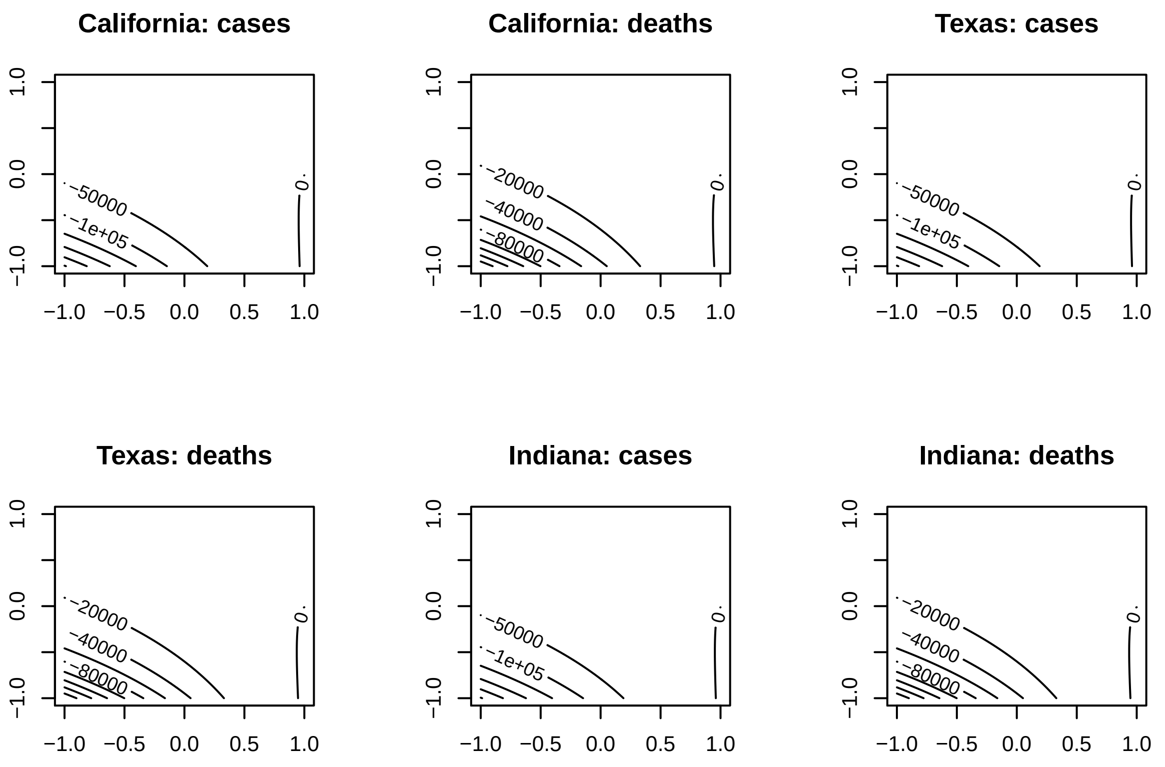
```
## Warning in 1:which(d3$date == "2020-06-30"): numerical expression has 3
## elements: only the first used
```

```
d3$cases[d3$cases > 0] <- log(d3$cases[d3$cases > 0])
d3$deaths[d3$deaths > 0] <- log(d3$deaths[d3$deaths > 0])
d3_cali <- d3[d3$state == "California",]
d3_tex <- d3[d3$state == "Texas",]
d3_indi <- d3[d3$state == "Indiana",]
```

We can repeat step 4 for each dataset:

```
par(mfrow=c(2, 3))
rho <- seq(-1, 1, length.out = 500)
sig <- seq(-1, 1, length.out = 500)
logpost_cases_cali = outer(rho, sig, ar_logpost_cases, d3_cali)
contour(rho, sig, logpost_cases, nlevels=10,
        main="California: cases")

logpost_deaths_cali <- outer(rho, sig, ar_logpost_deaths, d3_cali)
```

```r
contour(rho, sig, logpost_deaths, nlevels=10,
        main="California: deaths")

logpost_cases_tex = outer(rho, sig, ar_logpost_cases, d3_tex)
contour(rho, sig, logpost_cases, nlevels=10,
        main="Texas: cases")

logpost_deaths_tex <- outer(rho, sig, ar_logpost_deaths, d3_tex)
contour(rho, sig, logpost_deaths, nlevels=10,
        main="Texas: deaths")

logpost_cases_indi = outer(rho, sig, ar_logpost_cases, d3_indi)
contour(rho, sig, logpost_cases, nlevels=10,
        main="Indiana: cases")

logpost_deaths_indi <- outer(rho, sig, ar_logpost_deaths, d3_indi)
contour(rho, sig, logpost_deaths, nlevels=10,
        main="Indiana: deaths")
```



Sample from posterior distribution:

```r
set.seed(1)
idx_cases_cali <- sample(seq(1, length(logpost_cases_cali)), size=1000, replace=TRUE,
                    prob=exp(c(logpost_cases_cali)) / sum(exp(c(logpost_cases_cali))))
cases_sample_cali <- expand.grid(rho, sig)[idx_cases_cali,]
names(cases_sample_cali) <- c('rho', 'logsigma')


idx_deaths_cali <- sample(seq(1, length(logpost_deaths_cali)), size=1000, replace=TRUE,
                    prob=exp(c(logpost_deaths_cali)) / sum(exp(c(logpost_deaths_cali))))
```

```
deaths_sample_cali <- expand.grid(rho, sig)[idx_deaths_cali,]
names(deaths_sample_cali) <- c('rho', 'logsigma')

idx_cases_tex <- sample(seq(1, length(logpost_cases_tex)), size=1000, replace=TRUE,
                        prob=exp(c(logpost_cases_tex)) / sum(exp(c(logpost_cases_tex))))
cases_sample_tex <- expand.grid(rho, sig)[idx_cases_tex,]
names(cases_sample_tex) <- c('rho', 'logsigma')


idx_deaths_tex <- sample(seq(1, length(logpost_deaths_tex)), size=1000, replace=TRUE,
                        prob=exp(c(logpost_deaths_tex)) / sum(exp(c(logpost_deaths_tex))))
deaths_sample_tex <- expand.grid(rho, sig)[idx_deaths_tex,]
names(deaths_sample_tex) <- c('rho', 'logsigma')

idx_cases_indi <- sample(seq(1, length(logpost_cases_indi)), size=1000, replace=TRUE,
                        prob=exp(c(logpost_cases_indi)) / sum(exp(c(logpost_cases_indi))))
cases_sample_indi <- expand.grid(rho, sig)[idx_cases_indi,]
names(cases_sample_indi) <- c('rho', 'logsigma')


idx_deaths_indi <- sample(seq(1, length(logpost_deaths_indi)), size=1000, replace=TRUE,
                        prob=exp(c(logpost_deaths_indi)) / sum(exp(c(logpost_deaths_indi))))
deaths_sample_indi <- expand.grid(rho, sig)[idx_deaths_indi,]
names(deaths_sample_indi) <- c('rho', 'logsigma')
```

Now we can get the summary statistics of each sample:

```
for(i in c("cali$rho", "tex$rho", "indi$rho",
           "cali$logsigma", "tex$logsigma", "indi$logsigma")){
    cat(i, "cases quantiles: ")
    cat(quantile(eval(parse(text=paste("cases_sample", i, sep="_"))),
                                  probs=c(0.025, 0.25, 0.5, 0.75, 0.975)))
    cat("\n")
    cat(i, "deaths quantiles: ")
    cat(quantile(eval(parse(text=paste("deaths_sample", i, sep="_"))),
                                  probs=c(0.025, 0.25, 0.5, 0.75, 0.975)))
    cat("\n")

    # means
    cat(i, "cases mean: ")
    cat(mean(eval(parse(text=paste("cases_sample", i, sep="_")))))
    cat("\n")
    cat(i, "deaths mean: ")
    cat(mean(eval(parse(text=paste("deaths_sample", i, sep="_")))))
    cat("\n")

    # medians
    cat(i, "cases median: ")
    cat(median(eval(parse(text=paste("cases_sample", i, sep="_")))))
    cat("\n")
    cat(i, "deaths median: ")
    cat(median(eval(parse(text=paste("deaths_sample", i, sep="_")))))
    cat("\n")
```

```r
    # standard deviations
    cat(i, "cases sd: ")
    cat(sd(eval(parse(text=paste("cases_sample", i, sep="_")))))
    cat("\n")
    cat(i, "deaths sd: ")
    cat(sd(eval(parse(text=paste("deaths_sample", i, sep="_")))))
    cat("\n")

    # skewnesses
    cat(i, "cases skewness: ")
    cat(skewness(eval(parse(text=paste("cases_sample", i, sep="_")))))
    cat("\n")
    cat(i, "deaths skewness: ")
    cat(skewness(eval(parse(text=paste("deaths_sample", i, sep="_")))))
    cat("\n")

    # kurtosis
    cat(i, "cases kurtosis: ")
    cat(kurtosis(eval(parse(text=paste("cases_sample", i, sep="_")))))
    cat("\n")
    cat(i, "deaths kurtosis: ")
    cat(kurtosis(eval(parse(text=paste("deaths_sample", i, sep="_")))))
    cat("\n")
}
```

```
## cali$rho cases quantiles: 0.995992 1 1 1 1
## cali$rho deaths quantiles: 0.995992 1 1 1 1
## cali$rho cases mean: 0.9997635
## cali$rho deaths mean: 0.9991102
## cali$rho cases median: 1
## cali$rho deaths median: 1
## cali$rho cases sd: 0.0009783056
## cali$rho deaths sd: 0.001823198
## cali$rho cases skewness: -4.14996
## cali$rho deaths skewness: -1.929936
## cali$rho cases kurtosis: 20.19005
## cali$rho deaths kurtosis: 6.350665
## tex$rho cases quantiles: 0.995992 1 1 1 1
## tex$rho deaths quantiles: 0.991984 0.995992 1 1 1
## tex$rho cases mean: 0.9997435
## tex$rho deaths mean: 0.9985531
## tex$rho cases median: 1
## tex$rho deaths median: 1
## tex$rho cases sd: 0.0009814645
## tex$rho deaths sd: 0.002434843
## tex$rho cases skewness: -3.562776
## tex$rho deaths skewness: -1.57544
## tex$rho cases kurtosis: 13.69338
## tex$rho deaths kurtosis: 4.829709
## indi$rho cases quantiles: 0.995992 1 1 1 1
## indi$rho deaths quantiles: 0.991984 0.995992 1 1 1
## indi$rho cases mean: 0.9995471
## indi$rho deaths mean: 0.9985251
## indi$rho cases median: 1
## indi$rho deaths median: 1
## indi$rho cases sd: 0.00130699
## indi$rho deaths sd: 0.002407937
## indi$rho cases skewness: -2.701735
## indi$rho deaths skewness: -1.598422
## indi$rho cases kurtosis: 9.252609
## indi$rho deaths kurtosis: 5.347341
## cali$logsigma cases quantiles: -1 -1 -0.995992 -0.991984 -0.9759519
## cali$logsigma deaths quantiles: -1 -1 -0.995992 -0.991984 -0.9719439
## cali$logsigma cases mean: -0.9944369
## cali$logsigma deaths mean: -0.9946733
## cali$logsigma cases median: -0.995992
## cali$logsigma deaths median: -0.995992
## cali$logsigma cases sd: 0.007150281
## cali$logsigma deaths sd: 0.007280988
## cali$logsigma cases skewness: 1.769655
## cali$logsigma deaths skewness: 1.941137
## cali$logsigma cases kurtosis: 6.520238
## cali$logsigma deaths kurtosis: 7.22971
## tex$logsigma cases quantiles: -1 -1 -0.995992 -0.991984 -0.9679359
## tex$logsigma deaths quantiles: -1 -1 -0.995992 -0.991984 -0.9718437
## tex$logsigma cases mean: -0.9931222
## tex$logsigma deaths mean: -0.993515
## tex$logsigma cases median: -0.995992
## tex$logsigma deaths median: -0.995992
## tex$logsigma cases sd: 0.008913931
## tex$logsigma deaths sd: 0.00853492
## tex$logsigma cases skewness: 2.102563
## tex$logsigma deaths skewness: 2.088324
## tex$logsigma cases kurtosis: 9.111731
## tex$logsigma deaths kurtosis: 8.904721
## indi$logsigma cases quantiles: -1 -1 -0.995992 -0.987976 -0.9599198
## indi$logsigma deaths quantiles: -1 -1 -0.995992 -0.987976 -0.9639279
## indi$logsigma cases mean: -0.9906613
## indi$logsigma deaths mean: -0.9916433
## indi$logsigma cases median: -0.995992
## indi$logsigma deaths median: -0.995992
## indi$logsigma cases sd: 0.01109487
## indi$logsigma deaths sd: 0.009776376
## indi$logsigma cases skewness: 1.788361
## indi$logsigma deaths skewness: 1.942104
## indi$logsigma cases kurtosis: 6.755089
## indi$logsigma deaths kurtosis: 8.154806
```

We can see from the output that they are more or less agree with each other.

8

For the Covid-19 dataset, we fitted an AR(1) model. First, for each nonzero cases/deaths, we transform it to log scale. Then the AR(1) process simply models the cumulative cases/deaths as a linear function of cases/deaths in previous day. However, the model did not fit well. Part of the reason is the nature of this virus: cases and deaths increases exponentially, i.e., cases and deaths typically would increase faster as time goes. AR(1) cannot capture this property and thus fails.