STAT 656: Bayesian Data Analysis
Fall 2024
Homework 1

Juanwu Lu[*]

Synthetic Data

The *autoregressive model* is frequently used to analyze time series data. The simplest autoregressive model has order 1, and is abbreviated as AR(1). This model assumes that an observation y_i at time point i ($i = 1, \dots, n$) is generated according to

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

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ independently, and *rho* and σ are unknown parameters. For simplicity, we shall assume that y_0 is a fixed constant. We will also assume $|\rho| < 1$.

1. (5 points) **Solution:**

   Given the formulation above, the log-likelihood function is calculated as follows:

$$\begin{aligned}\log L(\rho, \sigma^2 | y_0, y_1, \dots, y_n) &= \log \prod_{i=1}^{n} (2\pi\sigma^2)^{-\frac{1}{2}} \cdot \exp\left\{-\frac{(y_i - \rho y_{i-1})^2}{2\sigma^2}\right\} \\ &= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \rho y_{i-1})^2 \\ &= -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \rho y_{i-1})^2.\end{aligned}$$

2. (10 points) **Solution:**

   The code implementation is as follows:

```
# Read data
if (file.exists("data/computation_data_hw_1.csv")) {
  data <- read.csv("data/computation_data_hw_1.csv")
  y <- data[["x"]]
} else {
  stop("Cannot find the file 'data/computation_data_hw_1.csv' at ", getwd())
}

# Define the log-likelihood function
ar_loglik <- function(rho, log_sig) {
  y <- .GlobalEnv$y
  n <- length(y)
  sig <- exp(log_sig)
```

[*]College of Engineering, Purdue University, West Lafayette, IN, USA

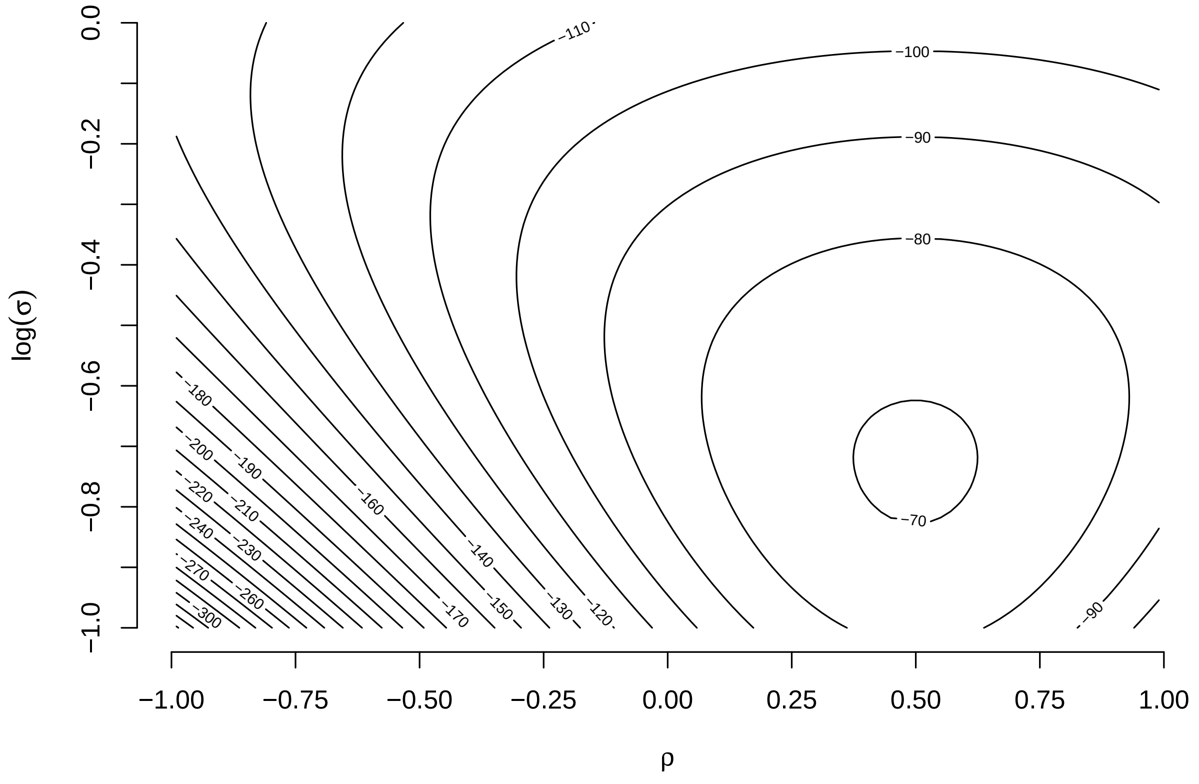
```r
  rho <- rep(as.numeric(rho), times = n - 1)
  denom <- n * log(2 * pi) + 2 * n * log_sig
  log_lik <- -0.5 * (
    denom + (y[1]^2 + sum((y[2:n] - rho * y[1:(n - 1)])^2)) / sig^2
  )
  return(log_lik)
}

# Visualization
rho <- seq(-0.99, 0.99, length = 100)
log_sig <- seq(-1.0, 0.0, length = 100)
loglik <- outer(rho, log_sig, Vectorize(ar_loglik))
contour(
  x = rho,
  y = log_sig,
  z = loglik,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  title = "Log-Likelihood Function",
  nlevels = 20,
  axes = FALSE,
)
```

```
## Warning in plot.window(xlim, ylim, ...): "title" is not a graphical parameter
```

```
## Warning in title(...): "title" is not a graphical parameter
```

```r
axis(side = 1, at = seq(-1.0, 1.0, by = 0.25))
axis(side = 2, at = seq(-1.0, 0.0, by = 0.10))
```



3. (10 points) **Solution:**

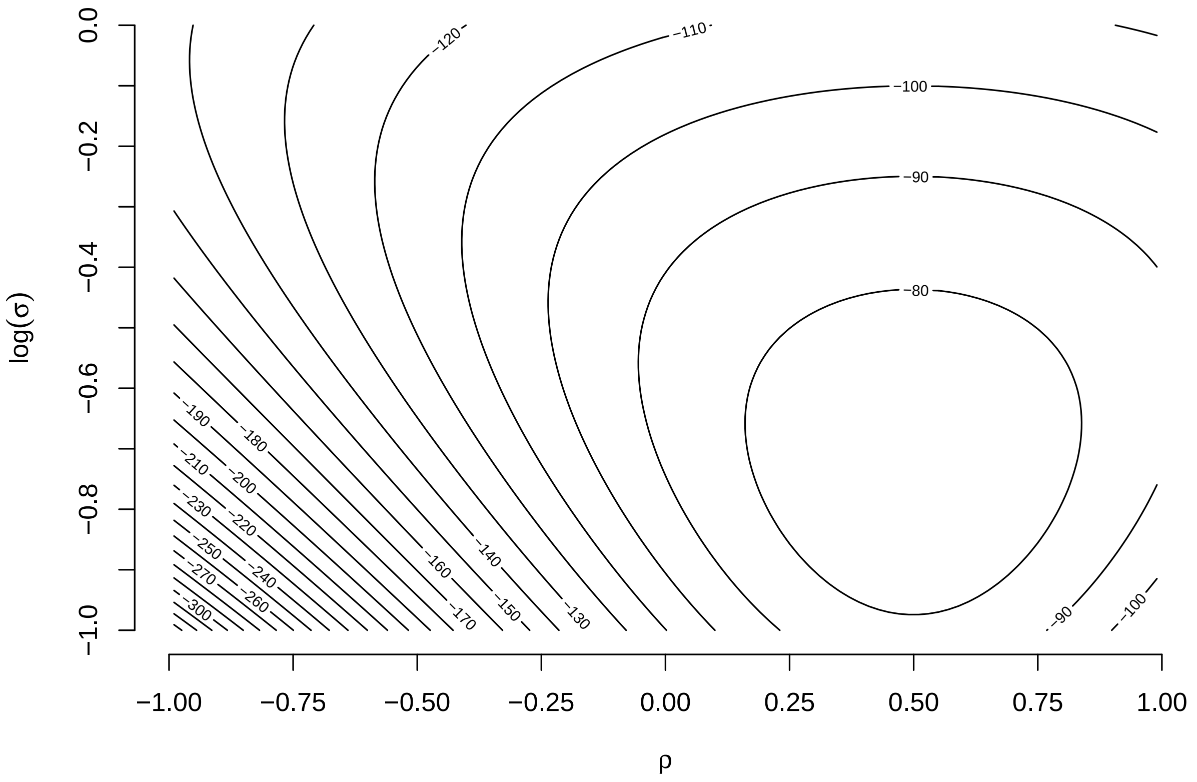
Since the prior is independent, we have $p(\rho, \log(\sigma)) = p(\rho)p(\log(\sigma)) = \frac{1}{2} \cdot \frac{1}{10\sqrt{2\pi}} \exp\left\{-\frac{\log(\sigma)^2}{200}\right\}$. Therefore, the posterior density is proportional to the product of the likelihood and the prior, and then the log of the posterior density is calculated up to a constant as follows:

$$\begin{aligned}\log p(\rho, \log(\sigma)|y_0, y_1, \ldots, y_n) &= \log L(\rho, \log(\sigma)|y_0, y_1, \ldots, y_n) + \log p(\rho, \log(\sigma)) + C \\ &= -n\log(\sigma) - \frac{1}{2\exp(2 * \log(\sigma))}\sum_{i=1}^{n}(y_i - \rho y_{i-1})^2 - \frac{\log(\sigma)^2}{200} + C'\end{aligned}$$

where C is the constant log-normalizer and C' is a constant irrelevant to ρ and $\log(\sigma)$. The visualization of the log of the posterior density is as follows:

```r
ar_logpost <- function(rho, log_sig) {
  log_prior <- -log(2) - 0.5 * (log(2 * pi) + log(100) + log_sig^2 / 100)
  log_post <- ar_loglik(rho, log_sig) + log_prior
  return(log_post)
}

logpost <- outer(rho, log_sig, Vectorize(ar_logpost))
contour(
  x = rho,
  y = log_sig,
  z = logpost,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  nlevels = 20,
  axes = FALSE,
)
axis(side = 1, at = seq(-1.0, 1.0, by = 0.25))
axis(side = 2, at = seq(-1.0, 0.0, by = 0.10))
```



Compared to the log-likelihood function, the log posterior density is more concentrated around the maximum likelihood estimate of $(\rho, \log(\sigma))^\intercal$. The prior is not overly informative since the shape of the posterior density is still similar to the likelihood function, indicating that the posterior is mainly determined by the likelihood function.

4. (10 points) **Solution:**

From the visualization in 3., we can see that the posterior density is concentrated around $0.25 \leq \rho \leq 0.75$ and $-0.9 \leq \log(\sigma) \leq -0.5$. Therefore, we can choose a grid of ρ and $\log(\sigma)$ as follows:

```r
set.seed(42)
rho_grid <- seq(0.25, 0.75, length = 100)
log_sig_grid <- seq(-0.9, -0.5, length = 100)
grid_log_post <- outer(rho_grid, log_sig_grid, Vectorize(ar_logpost))
# Calculate the normalized probability density
probs <- exp(grid_log_post - max(grid_log_post))
probs <- probs / sum(probs)
# Randomly sample from the grid
indices <- sample(
  x = seq_len(length(as.vector(probs))),
  size = 1000,
  replace = TRUE,
  prob = probs
)
rho_sample <- rho_grid[((indices - 1) %% nrow(probs)) + 1]
log_sig_sample <- log_sig_grid[((indices - 1) %/% nrow(probs)) + 1]
```

5. (5 points) **Solution:**

The code implementation is as follows:

```r
library(moments)

# Calculate summaries for rho
print(quantile(rho_sample, probs = c(0.025, 0.25, 0.5, 0.75, 0.975)))
```

```
##      2.5%       25%       50%       75%     97.5% 
## 0.3357323 0.4419192 0.4974747 0.5542929 0.6691919
```

```r
sprintf("Mean: %.4f", mean(rho_sample))
```

```
## [1] "Mean: 0.5002"
```

```r
sprintf("Standard Deviation: %.4f", sd(rho_sample))
```

```
## [1] "Standard Deviation: 0.0847"
```

```r
sprintf("Skewnewss: %.4f", skewness(rho_sample))
```

```
## [1] "Skewnewss: 0.0490"
```

```r
sprintf("Kurtosis: %.4f", kurtosis(rho_sample))
```

```
## [1] "Kurtosis: 3.0670"
```

```r
# Calculate summaries for log(sigma)
print(quantile(log_sig_sample, probs = c(0.025, 0.25, 0.5, 0.75, 0.975)))
```

```
##       2.5%        25%        50%        75%      97.5% 
## -0.8516162 -0.7707071 -0.7222222 -0.6696970 -0.5646465
```

```r
sprintf("Mean: %.4f", mean(log_sig_sample))
```

```
## [1] "Mean: -0.7199"
```

```r
sprintf("Standard Deviation: %.4f", sd(log_sig_sample))
```

```
## [1] "Standard Deviation: 0.0710"
```

```r
sprintf("Skewness: %.4f", skewness(log_sig_sample))
```

```
## [1] "Skewness: 0.1683"
```

```r
sprintf("Kurtosis: %.4f", kurtosis(log_sig_sample))
```

```
## [1] "Kurtosis: 2.7730"
```

6. (10 points) **Solution:**

The code implementation is as follows:

```r
ar_post_predictive <- function(rho, log_sig) {
  # Sample from the posterior predictive distribution
  y <- .GlobalEnv$y
  n <- length(y)
  new_y <- rep(0.0, times = n)
  sig <- exp(log_sig)
  for (i in 2:n) {
    new_y[i] <- rnorm(n = 1, mean = rho * new_y[i - 1], sd = sig)
  }
  return(new_y)
}
```

```
params <- data.frame(rho = rho_sample, log_sig = log_sig_sample)
samples <- matrix(NA, nrow = nrow(params), ncol = length(y))
for (i in seq_len(nrow(params))) {
  samples[i, ] <- ar_post_predictive(params[i, "rho"], params[i, "log_sig"])
}
```

The above code makes use of the grid samples drawn in problem 4. For each pair of $(\rho, \log(\sigma))$, we use the AR(1) model to generate a new sample of sequences. The summary statistics of the posterior predictive distribution are as follows:

```
print("Sequence means:")
```

```
## [1] "Sequence means:"
```

```
print(colMeans(samples))
```

```
##   [1]  0.0000000000 -0.0047681988 -0.0019792654 -0.0018532894 -0.0349612745
##   [6] -0.0534930980 -0.0504204039 -0.0250051576  0.0087108463 -0.0069861131
##  [11]  0.0151475799  0.0119437223 -0.0124981784 -0.0147181086 -0.0095207567
##  [16] -0.0264066932 -0.0181890113 -0.0090347402  0.0119296348 -0.0101964259
##  [21] -0.0171148698 -0.0048811360 -0.0079157382  0.0231390506  0.0194940894
##  [26]  0.0125344169  0.0055016256  0.0021447649  0.0126945975 -0.0040268132
##  [31] -0.0186832362 -0.0045891829  0.0033932494  0.0210361994  0.0271254869
##  [36]  0.0207458603  0.0261321589 -0.0076168507 -0.0157112720 -0.0058691160
##  [41] -0.0242675255 -0.0056608656 -0.0231461736 -0.0067463403 -0.0008398088
##  [46] -0.0029217855 -0.0168447601  0.0023636063  0.0106066591 -0.0074663974
##  [51]  0.0183917648  0.0098816763  0.0033097755 -0.0213814711 -0.0128319243
##  [56]  0.0146220978 -0.0009408223  0.0073294289 -0.0133172729 -0.0389761421
##  [61] -0.0291574610 -0.0194010667 -0.0096349505 -0.0264908224  0.0059469389
##  [66]  0.0058553509 -0.0240765049 -0.0060451499 -0.0139014324  0.0017078806
##  [71] -0.0156887923 -0.0337133590 -0.0141836931 -0.0106669384 -0.0111976680
##  [76] -0.0238671950 -0.0078890984  0.0310415176  0.0228528593  0.0205011147
##  [81]  0.0118425975 -0.0199361761  0.0137116263  0.0059716095  0.0150293149
##  [86] -0.0002212806 -0.0068151597 -0.0227551295  0.0174926333  0.0011145912
##  [91] -0.0301007269 -0.0032666686  0.0017014288 -0.0081048702 -0.0100689443
##  [96] -0.0071409824  0.0219371072  0.0248663111  0.0009138676 -0.0086203099
```

```
print("Sequence standard deviations:")
```

```
## [1] "Sequence standard deviations:"
```

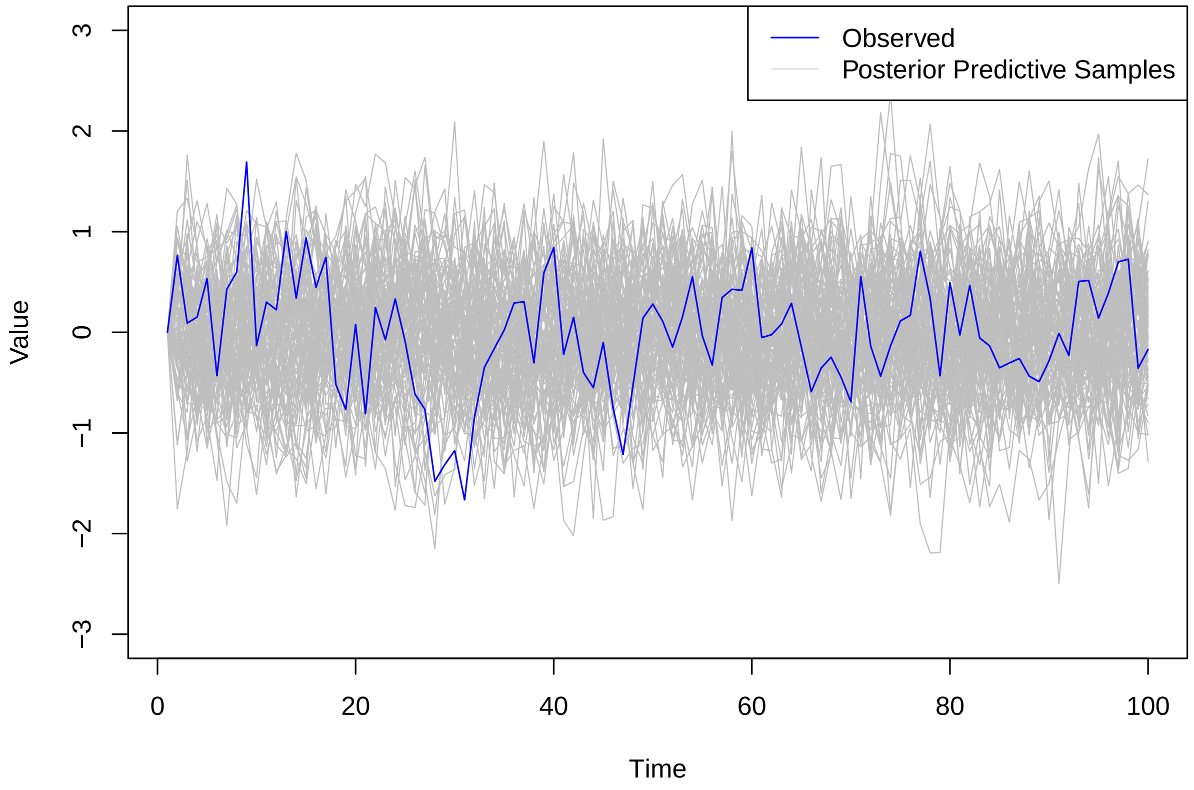
```
print(apply(samples, 2, sd))
```

```
##   [1] 0.0000000 0.4875799 0.5509644 0.5491307 0.5646703 0.5777193 0.5872822
##   [8] 0.5823673 0.5608815 0.5692878 0.5816537 0.5517229 0.5654348 0.5837800
##  [15] 0.5620184 0.5582978 0.5517426 0.5445177 0.5719960 0.5662387 0.5742013
##  [22] 0.5476983 0.5913086 0.5771088 0.5762680 0.5785766 0.6054432 0.5748847
##  [29] 0.5843245 0.5768187 0.5828568 0.5667542 0.5788029 0.5793277 0.5589459
##  [36] 0.5576004 0.5605176 0.5567362 0.5583023 0.5755939 0.5762962 0.5889772
##  [43] 0.5768393 0.5641803 0.5891536 0.5754784 0.5540364 0.6022881 0.5939347
##  [50] 0.5882779 0.5739056 0.5880374 0.5918485 0.5718155 0.5718703 0.5689348
##  [57] 0.5537982 0.5624192 0.5729930 0.5599766 0.5657569 0.5668950 0.5504416
##  [64] 0.5612458 0.5663921 0.5732462 0.5806640 0.5830129 0.5701818 0.5704153
##  [71] 0.5618780 0.5681715 0.5969084 0.5880126 0.5880703 0.5666169 0.5749689
##  [78] 0.5714826 0.5794300 0.5787407 0.5578098 0.5531926 0.5732864 0.5613480
##  [85] 0.5699433 0.5690359 0.5541614 0.5828233 0.5935380 0.5810655 0.5792932
##  [92] 0.5680596 0.5715865 0.5691965 0.5761821 0.5759740 0.5769301 0.5691927
##  [99] 0.5556785 0.5510210
```

7. (10 points) **Solution:**

   The visualization of the posterior predictive samples against the observed data is as follows:

```r
plot(
  x = seq_len(length(y)),
  y = samples[1, ],
  col = "gray",
  type = "l",
  lwd = 0.75,
  xlab = "Time",
  ylab = "Value",
  ylim = c(-3.0, 3.0)
)
for (i in 2:100) {
  lines(x = seq_len(length(y)), y = samples[i, ], col = "gray", lwd = 0.75)
}
lines(x = seq_len(length(y)), y = y, col = "blue", lwd = 1.0)
legend(
  "topright",
  legend = c("Observed", "Posterior Predictive Samples"),
  col = c("blue", "gray"),
  lwd = c(1, 0.5),
  bg = "white",
)
```



   From the visualization, we see that the posterior predictive samples have a wider range than the observed data, indicating that the model has a higher uncertainty in prediction and may not be able to capture the true data distribution well. Therefore, the model is **not a good fit** for the observed data.

My expectation for a good model is that the posterior predictive samples are close to the observed data with a similar tendency and range.

Real Data

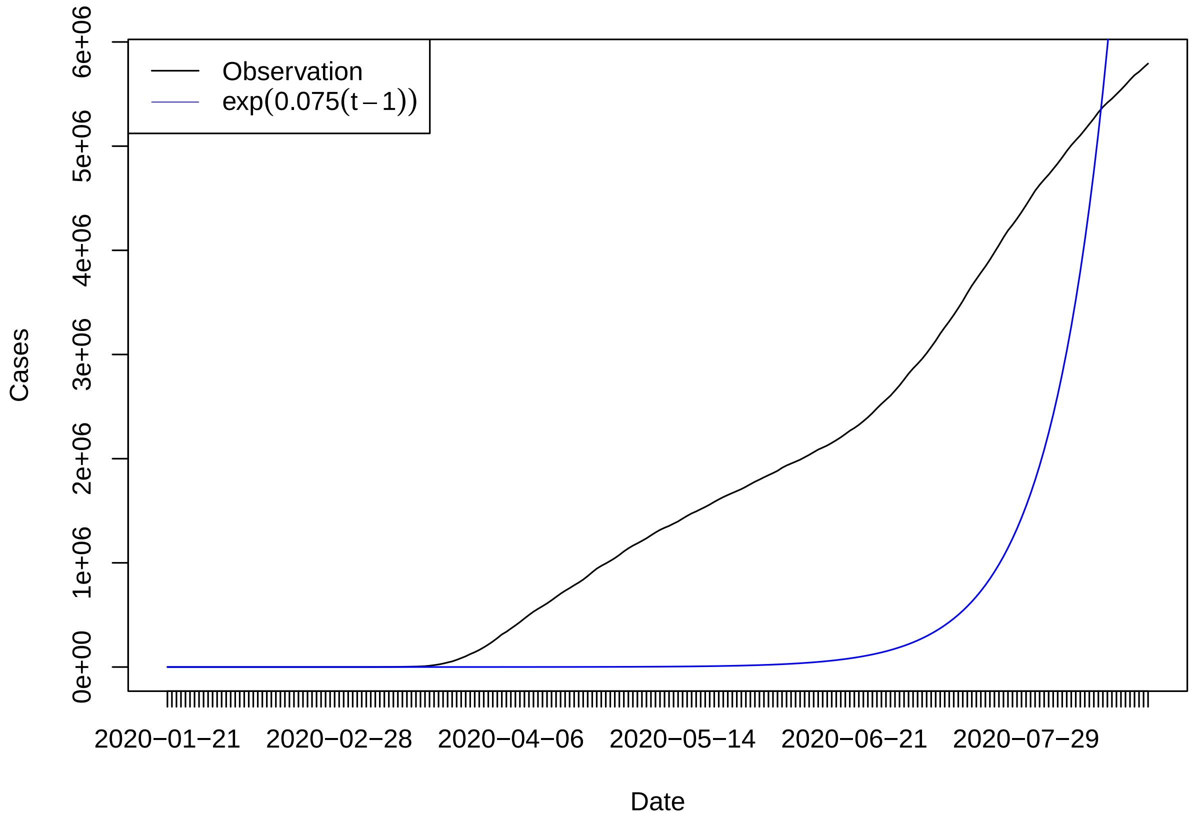
1. (5 points) **Solution:**

   No, I do not believe that the information given is sufficient for analyzing the data. The GitHub page has only provided the dataset file, source of data, and definitions of different sorts of cases. All these pieces of information are related to the observation, i.e., COVID-19 cases, but none of them reflect any information to support the prior distribution of the parameters in the model. Other information such as demographic statistics, geofencing data, activity data, etc., can be useful for data analysis.

2. (5 points) **Solution:**

   Before establishing the model, the raw data is visualized as follows:

```r
# Read data
if (file.exists("data/covid_us.txt")) {
  data <- read.table("data/covid_us.txt", header = TRUE, sep = ",")
  y <- data[["cases"]]
} else {
  stop("File not found: 'computation_data_hw_1.csv' at ", getwd())
}
plot(
  x = seq_len(length(y)),
  y = y,
  type = "l",
  xlab = "Date",
  ylab = "Cases",
  xaxt = "n"
)
lines(
  x = seq_len(length(y)),
  y = exp(0.075 * (seq_len(length(y)) - 1)),
  col = "blue"
)
axis(side = 1, labels = data[["date"]], at = seq_len(length(y)))
legend(
  "topleft",
  legend = c("Observation", expression(y=exp(0.075 * (t - 1)))),
  col = c("black", "blue"),
  lwd = c(1, 0.5),
  bg = "white",
)
```

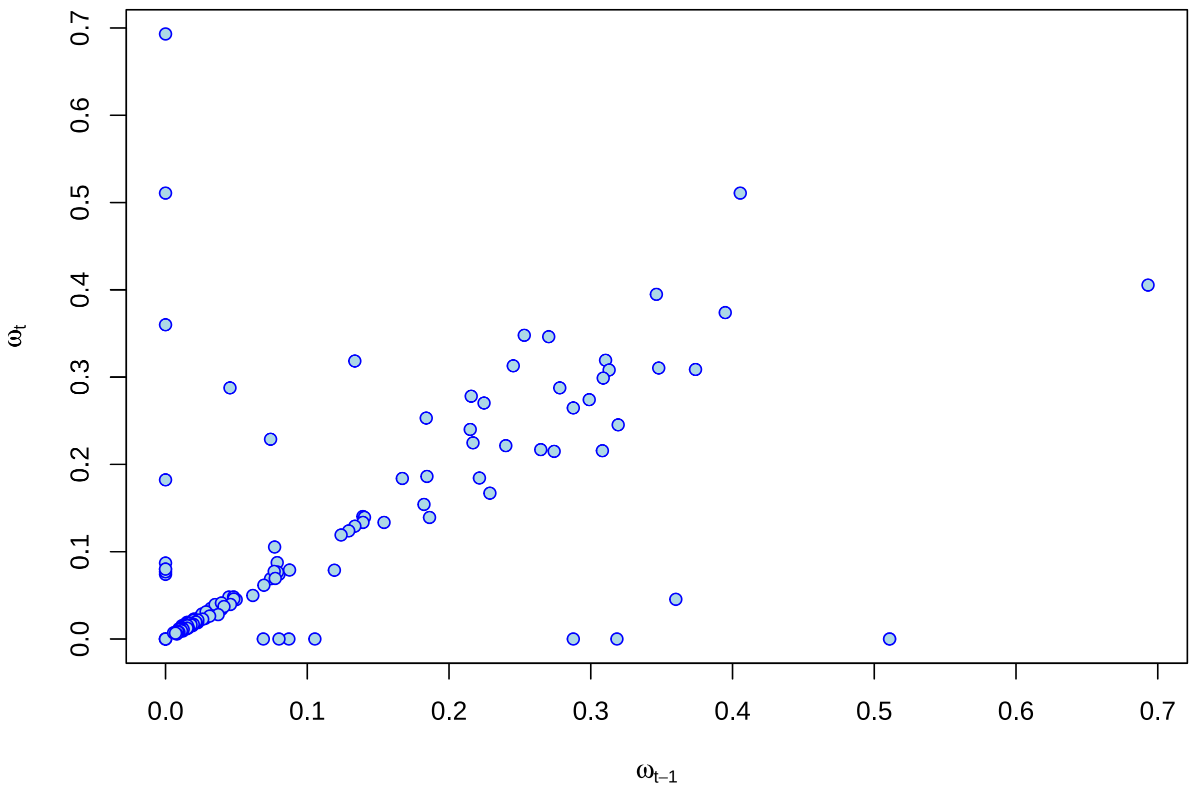


From the visualization, it is clear that the raw data has an exponential growth trend (blue line), which can not be directly captured by the AR(1) model. Therefore, the raw data needs to be transformed before being used for fitting an AR(1) model. The transformation consists of two steps: first, taking the logarithm of the raw data, and then, differentiating the transformed with `lag = 1` to filter out the non-stationary trend that is not captured by the AR(1) model. The transformed data is given as

$$\omega_t = \log(y_t) - \log(y_{t-1}).$$

For consistency, $\omega_1 = 0$. The visualization of the transformed data is as follows:

```r
omg <- c(0, diff(log(y), lag = 1))
plot(
  x = omg[seq(1, length(omg) - 1)],
  y = omg[seq(2, length(omg))],
  type = "p",
  pch = 21,
  col = "blue",
  bg = "lightblue",
  xlab = expression(omega[t - 1]),
  ylab = expression(omega[t]),
)
```



As shown in the visualization, the transformed data has a more linear relationship between consecutive observations. Finally, the AR(1) model is constructed as:

$$\omega_i = \rho\omega_{i-1} + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2).$$

3. (5 points) **Solution:**

   Based on the visualization above, a majority of the point cloud is growing linearly with a slope around 1.0 and slopes are non-negative. Therefore, the prior distribution of ρ can be set as a Gamma distribution $\rho \sim \text{Gamma}(1, 1)$ with a mean of 1.0. For the log-variance $\log(\sigma)$, without losing generality, the prior can be set a normal distribution $\log(\sigma) \sim \mathcal{N}(0, 1^2)$.

4. (10 points) **Solution:**

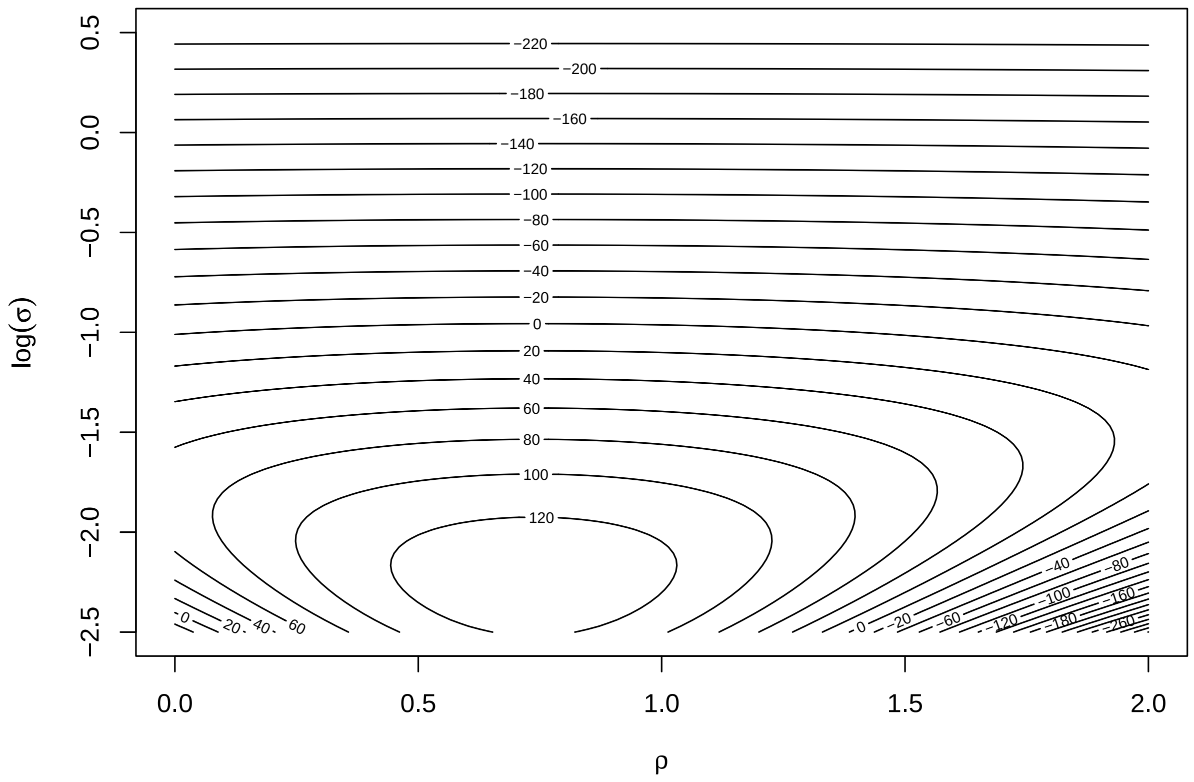
   With the above specified prior distributions, the log-posterior density is calculated as follows:

$$\begin{aligned}\log p(\rho, \log(\sigma)|\omega_1, \ldots, \omega_n) &= \log L(\rho, \log(\sigma)|\omega_1, \ldots, \omega_n) + \log p(\rho) + \log p(\log(\sigma)) + C \\ &= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\omega_i - \rho\omega_{i-1})^2 + \log\left(\frac{1}{\Gamma(1)}\exp(-\rho)\right) - \frac{\log(\sigma)^2}{2} + C' \\ &= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\omega_i - \rho\omega_{i-1})^2 - \rho - \frac{\log(\sigma)^2}{2} + C',\end{aligned}$$

   Therefore, the model fitted using data from the first time point until June 30 is as follows:

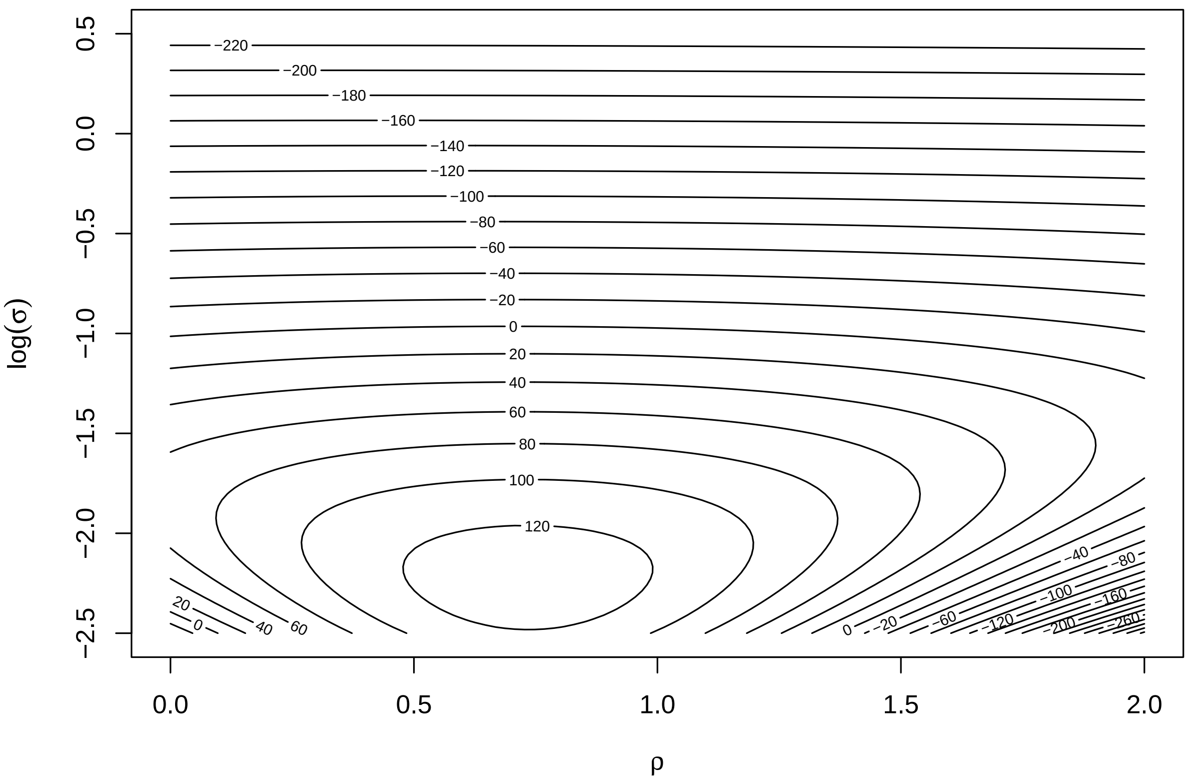
```r
y <- c(
  0,
  diff(log(y[as.Date(data$date) < as.Date("2020-06-30")]), lag = 1)
)
rho <- seq(0.0, 2.0, length = 100)
log_sig <- seq(-2.5, 0.5, length = 100)
```

```
log_lik <- outer(rho, log_sig, Vectorize(ar_loglik))
contour(
  x = rho,
  y = log_sig,
  z = log_lik,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  nlevels = 20,
)
```



The visualization of the logarithm of unnormalized posterior density is shown below:

```
ar_logpost <- function(rho, log_sig) {
  log_prior <- -rho - 0.5 * (log_sig^2)
  log_post <- ar_loglik(rho, log_sig) + log_prior
  return(log_post)
}
log_post <- outer(rho, log_sig, Vectorize(ar_logpost))
contour(
  x = rho,
  y = log_sig,
  z = log_post,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  nlevels = 20,
)
```



log(σ)
ρ
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
0.0
0.5
1.0
1.5
2.0
−220
−200
−180
−160
−140
−120
−100
−80
−60
−40
−20
0
20
40
60
80
100
120

5.