STAT 656 HW 1

Satoshi Ido (ID: 34788706)

10 September 2023

```
library("ggplot2")
library("MASS")
library("lmtest")
library("fs")
library("moments")
```

Synthetic data

Question 1

AR(1) model:

$$y_i = p \cdot y_{i-1} + e_i$$

Assume that the error terms (e_i) are independently and identically distributed (i.i.d.) with a gaussian distribution $N(0, \sigma^2)$.

The pdf of a gaussian distribution is given by:

$$f(e_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{e_i^2}{2\sigma^2}\right)$$

The likelihood function, $L(p, \sigma^2 | y_0, y_1, y_2, ..., y_n)$, is the joint pdf of the observed data $(y_0, y_1, y_2, ..., y_n)$ given the parameters (p, σ^2). Since the observations are assumed to be independent, the joint probability can be expressed as the product of individual probabilities:

$$L(p, \sigma^2 | y_0, y_1, y_2, ..., y_n) = \prod f(y_i | p, \sigma^2) \cdot f(y_0)$$

Hence, the likelihood function of the AR(1) model would be:

$$L(p, \sigma^2 | y_0, y_1, y_2, ..., y_n) = \prod f(y_i | y_{i-1}, p, \sigma^2) \cdot f(y_0) \quad \text{where } f(y_0) \text{ is constant} = 1$$

Using the pdf of a gaussian distribution, we can express the conditional probability $f(y_i | y_{i-1}, p, \sigma^2)$ as:

$$f(y_i | y_{i-1}, p, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(y_i - p \cdot y_{i-1})^2}{2\sigma^2}\right)$$

Taking the log of the likelihood function to obtain the log-likelihood function:

$$\begin{aligned}\log L(p, \sigma^2 | y_0, y_1, y_2, ..., y_n) &= \log\left(\prod f(y_i | y_{i-1}, p, \sigma^2)\right) \\ &= \sum \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(y_i - p \cdot y_{i-1})^2}{2\sigma^2}\right)\right) \\ &= -\frac{n}{2} \cdot \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \cdot \sum (y_i - p \cdot y_{i-1})^2\end{aligned}$$

The first term represents a constant that does not depend on the parameters and can be ignored during optimization.
The second term quantifies the sum of squared residuals, which measures the discrepancy between the observed values and the predictions made by the AR(1) model.

Question 2

Do stuff with data loaded from computation_data_hw_1.csv # You can make the data also an input to the function, or treat it as a global variable

```
# # Create the input_dir (input directory)
# current_note_path <- getwd()
# INPUT_DIR <- file.path(current_note_path, "656/hw/hw1/data")

# # If INPUT_DIR has not been created yet, create it
# if (!dir.exists(INPUT_DIR)) {
#   dir.create(INPUT_DIR)
# }

# # Create the output_dir (output directory)
# OUTPUT_DIR <- file.path(current_note_path, "656/hw/hw1/outputs")

# # If OUTPUT_DIR has not been created yet, create it
# if (!dir.exists(OUTPUT_DIR)) {
#   dir.create(OUTPUT_DIR)
# }

# # Read CSV files using a function to specify the directory automatically
# read_csv <- function(name, ...) {
#   path <- file.path(INPUT_DIR, paste0(name, ".csv"))
#   print(paste("Load:", path))
#   return(read.csv(path, ...))
# }
```

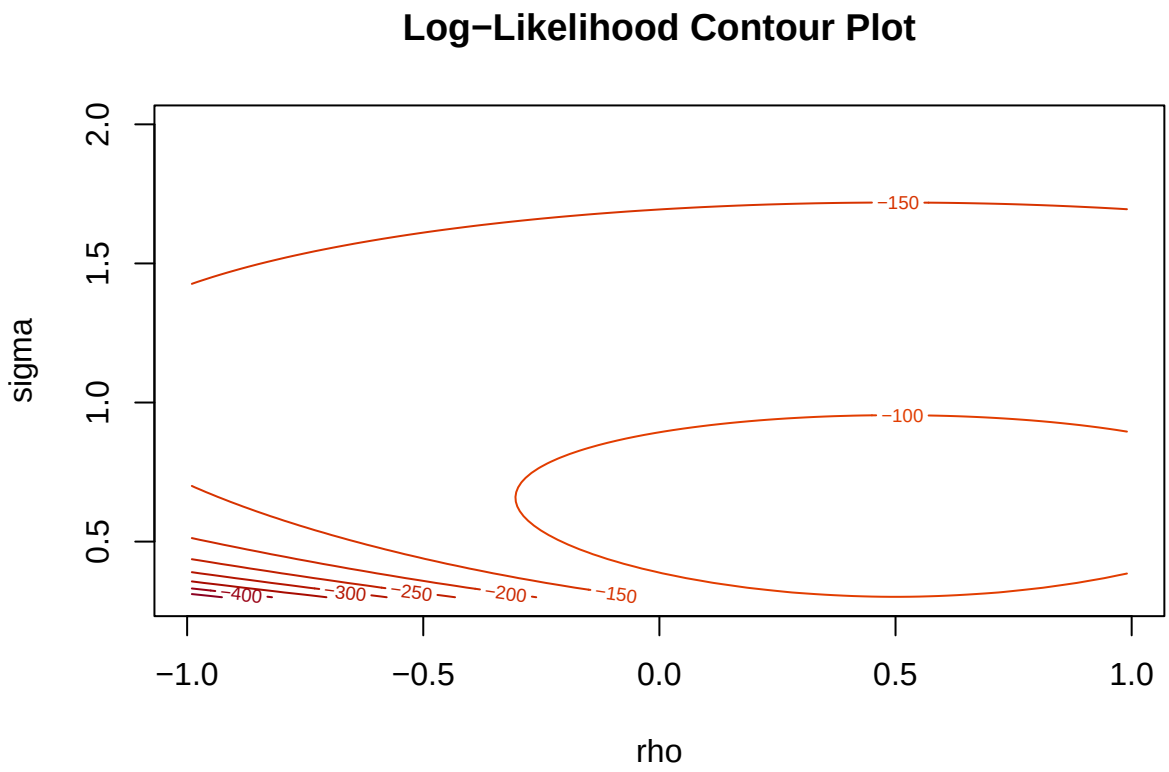
Move the file to INPUT_DIR if it is not already there. comment these out if you already have done this.

```
# # Define the source directory and destination directory
# current_note_path <- getwd()
# source_dir <- file.path(current_note_path, "656/hw/hw1")
# destination_dir <- INPUT_DIR

# # Get a list of CSV files in the source directory
# csv_files <- fs::dir_ls(source_dir, regexp = "\\.csv$", recurse = TRUE)
```

```r
# # Move each CSV file to the destination directory
# for (file in csv_files) {
#   fs::file_move(file, destination_dir)
#   cat("Moved file:", file, "\n")
# }
```

```r
# Create the input_dir (input directory)
current_note_path <- getwd()
INPUT_DIR <- file.path(current_note_path, "656/hw/hw1/data")
# Create the output_dir (output directory)
OUTPUT_DIR <- file.path(current_note_path, "656/hw/hw1/outputs")
# Create the function to read CSV easily
read_csv <- function(name, ...) {
  path <- file.path(INPUT_DIR, paste0(name, ".csv"))
  print(paste("Load:", path))
  return(read.csv(path, ...))
}

# Import the data
# data <- read_csv("computation_data_hw_1")[,2]
data <- read.csv("/Users/satoshiido/Documents/statistical-analysis/656/hw/hw1/data/computation_data_hw_

# The function which calculates a log-likelihood given rho and sigma
ar_loglik <- function(rho, sigma) {
  n <- length(data)
  resid <- 0
  for (i in 2:n) {
      resid <- resid + (data[i] - rho * data[i-1])^2
    }
  # Slightly adjust the variance from sigma^2 to log(sigma)
  loglik <- (-(n/2) * log(2 * pi * sigma^2) - 1/(2 * exp(2 * log(sigma))) * resid)
  return(loglik)
}
```

```r
# Compute log-likelihood for different values of rho and log(sigma)
rho <- seq(-0.99, 0.99, length.out = 100)
sigma <- seq(0.3, 2, length.out = 100)
# Compute log-likelihood values and store them in the matrix
loglik <- outer(rho, sigma, ar_loglik)
# Create a contour plot
## Set the color
cols <- hcl.colors(30, "YlOrRd")
par(mfrow = c(1, 1))
contour(x = rho, y = sigma, z = loglik, xlab = "rho", ylab = "sigma", main = "Log-Likelihood Contour Pl
```



Question 3

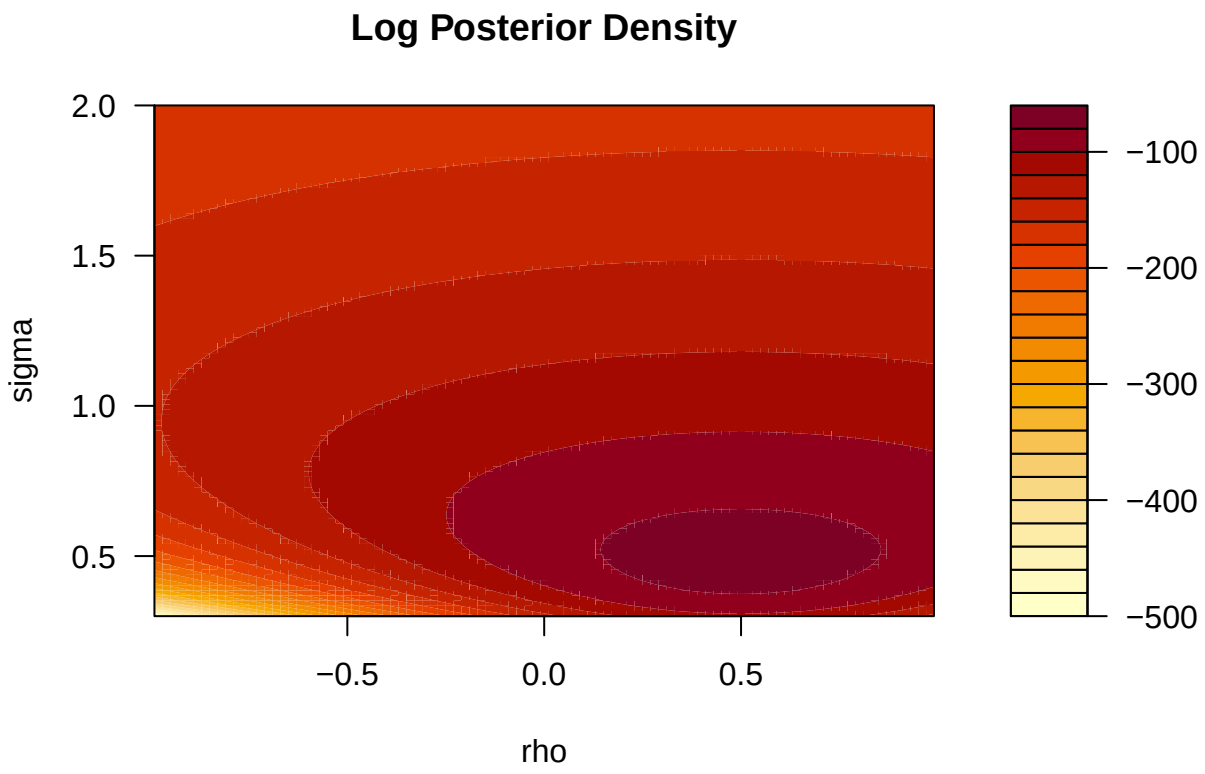
The Log posterior function corresponds to the log-likelihood function in a manner of its summation. Whether the chosen priors are suitable or too informative depends on the situation. The uniform prior on rho means you regard all values between -1 and 1 are equally possible for rho.
That could be reasonable if we have no prior knowledge about rho beforehand, but it rules out anything less than -1 or more than 1, which could be too limiting if rho's actual value is outside that range.
The normal prior on log(sigma) lets log(sigma) vary a lot, but it sets a highest probability at log(sigma)=0, meaning sigma=1. Unless we expect sigma near 1, this prior may unncessary influential on outcomes.

```
# Log prior
## p ~ Uniform(-1, 1), log(sigma) ~ N(0, 10^2)
logprior <- function(rho, sigma){
  # Set the prior probabilities with log transformation in order for the future use of log posterior
  # Uniform(-1, 1)
  log_prior_rho <- ifelse(rho >= -1 && rho <= 1, 0, -Inf)
  # N(0, 10^2)
  log_prior_log_sigma <- dnorm(log(sigma), mean = 0, sd = 10, log = TRUE)

  # Return the value
  return(log_prior_rho + log_prior_log_sigma)
}

# Log likelihood
loglikelihood <- function(rho, sigma){
```

```
  # Likelihood for an AR(1) model
  n <- length(data)

  resid <- 0
  for (i in 2:n) {
      resid <- resid + (data[i] - rho * data[i-1])^2
    }

  likelihood <- (-(n/2) * log(2 * pi * sigma^2) - 1/(2 * exp(2 * log(sigma))) * resid)
  # Return log loglikelihood
  return(likelihood)
}

# Log posterior
logposterior <- function(rho, sigma){
  # Return log posterior (up to a constant) for (p,log(sigma))T
  return(logprior(rho, sigma) + loglikelihood(rho, sigma))
}
```

Visualization of the log posterior (up to a constant) for (p,log(sigma))T

```
# Compute log-likelihood for different values of rho and sigma
rho2 <- seq(-0.99, 0.99, length.out = 100)
sigma2 <- seq(0.3, 2, length.out = 100)

# Compute the log posterior for each pair of values and store them in a matrix
log_posterior_values <- outer(rho2, sigma2, FUN = Vectorize(logposterior))

# Plot the log posterior (upto a constant) for (p,log(sigma))T
filled.contour(
      rho2
      , sigma2
      , log_posterior_values
      , xlab = "rho"
      , ylab = "sigma"
      , main = "Log Posterior Density"
      )
```



Question 5

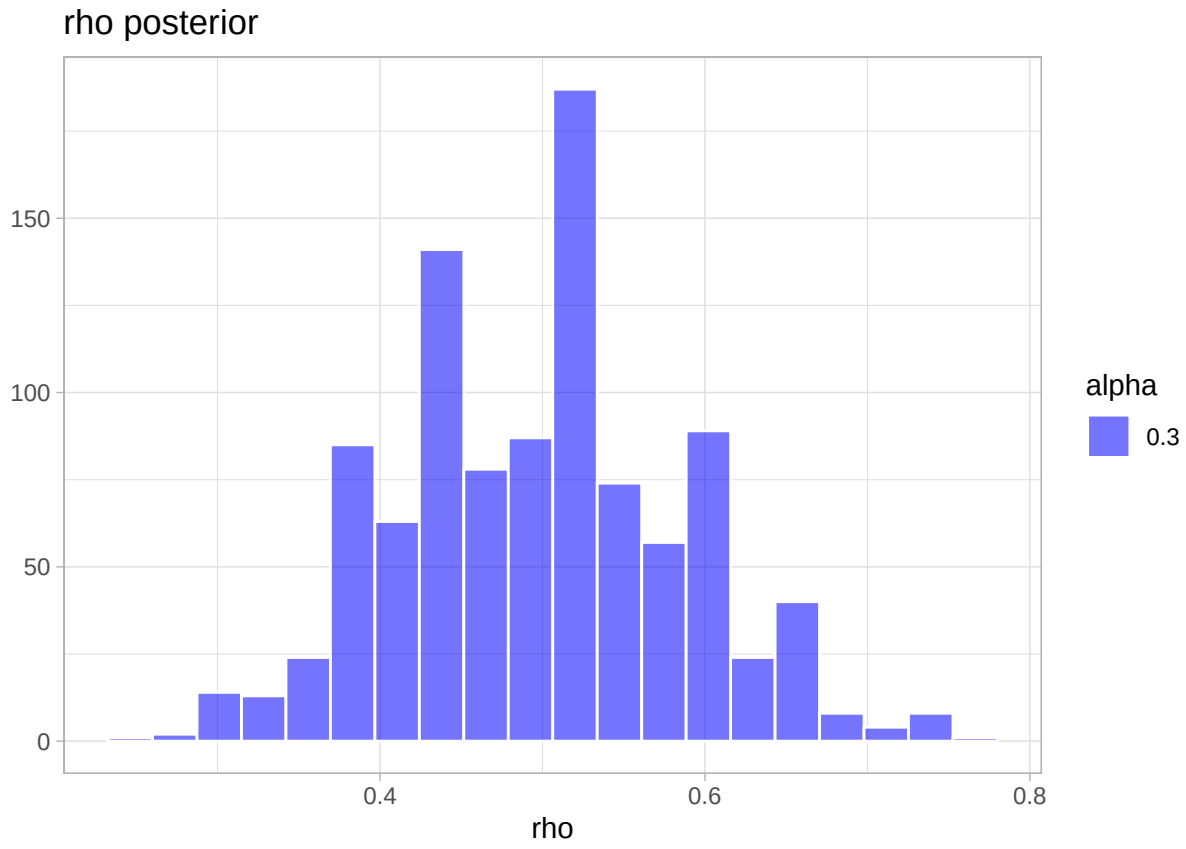
I selected sigma values from 0 to 2, and rho from -0.99 to 0.99 based on the contour plot in problem 3. I used sample which are in the area of highest posterior density and include some areas of lower density for more robust approximation of the posterior distribution.

```
# Compute log-likelihood for different values of rho and log(sigma)
rho3 <- seq(-0.99, 0.99, length.out = 100)
# log_sigma2 <- seq(-1, 0, length.out = 100)
sigma3 <- seq(0, 2, length.out = 100)

# Get unnormalized posterior values
posterior_values <- exp(log_posterior_values)

# Normalize posterior values so their sum becomes equal to 1
posterior_prop_values <- posterior_values / sum(posterior_values)

# Create a data frame of all possible parameter combinations
param_grid <- expand.grid(rho = rho3, sigma = sigma3)

# Convert posterior values to a vector
posterior_prop_values_vec <- as.vector(posterior_prop_values)

# Draw 1000 values from the discrete approximation to the posterior
sample_indices <- sample(length(posterior_prop_values_vec), size = 1000, prob = posterior_prop_values_v
```

```r
sampled_params <- param_grid[sample_indices, ]

# Check the first few rows of sampled_params to see the sampled parameter values
head(sampled_params)
```

```
##        rho     sigma
## 975   0.49 0.1818182
## 975.1 0.49 0.1818182
## 1374  0.47 0.2626263
## 980   0.59 0.1818182
## 978   0.55 0.1818182
## 1175  0.49 0.2222222
```

```r
# Plot the sampled parameters
qplot(rho, data = sampled_params, geom = "histogram",
      color = I("white"),
      fill = I("blue"),
      bins = 20, alpha = 0.3) + theme_light() + labs(title = "rho posterior")
```



Question 7

```r
# rho
quantile(sampled_params$rho, c(0.025, 0.25, 0.5, 0.75, 0.975))
```

```
##  2.5%   25%   50%   75% 97.5%
##  0.33  0.43  0.49  0.55  0.67
```

```r
# sigma
quantile(sampled_params$sigma, c(0.025, 0.25, 0.5, 0.75, 0.975))
```

```
##      2.5%       25%       50%       75%     97.5%
## 0.1414141 0.2020202 0.2222222 0.2424242 0.3232323
```

```r
# Summary of rho including skewness and kurtosis
summary(sampled_params$rho);skewness(sampled_params$rho);kurtosis(sampled_params$rho)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2500  0.4300  0.4900  0.4978  0.5500  0.7700
```

```
## [1] 0.1174842
```

```
## [1] 2.821087
```

```r
# Summary of rho including skewness and kurtosis
summary(sampled_params$sigma);skewness(sampled_params$sigma);kurtosis(sampled_params$sigma)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1010  0.2020  0.2222  0.2223  0.2424  0.4040
```

```
## [1] 0.4706782
```

```
## [1] 3.418469
```

Question 8

```r
simulate_ar1 <- function(rho, sigma, n) {
  # Initialize the series
  y_rep <- numeric(n)

  # Add initial value
  y_rep[1] <- rnorm(1, 0, sigma)

  # Generate the remaining values
  for (i in 2:n) {
    y_rep[i] <- rho * y_rep[i - 1] + rnorm(1, 0, sigma)
  }

  return(y_rep)
}
```

Generate 1000 draws from the posterior predictive distribution

```
set.seed(123)  # for reproducibility
# initialize a list to store the samples
# posterior_predictive_samples <- list()
posterior_predictive_samples <- matrix(nrow = 1000, ncol = 100)

# Generate 1000 samples
for (i in 1:1000) {
  # Draw parameters from the posterior
  rho <- sampled_params[i, 1]
  sigma <- sampled_params[i, 2]

  # Simulate a new dataset using these parameters
  y_rep <- simulate_ar1(rho, sigma, 100)

  # Store the simulated dataset
  posterior_predictive_samples[i,] <- simulate_ar1(rho, sigma, 100)
}
```

Summarize the posterior predictive distribution

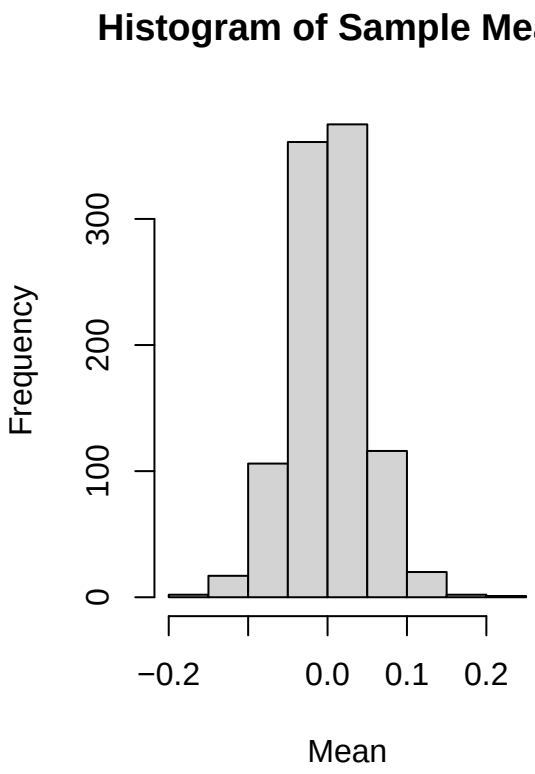
```
# Calculate the mean and variance of each simulated dataset
sample_means <- apply(posterior_predictive_samples, 1, mean)
sample_vars <- apply(posterior_predictive_samples, 1, var)

# Display the summary of the means and variances
summary_means <- summary(sample_means)
summary_vars <- summary(sample_vars)

print(summary_means)
```

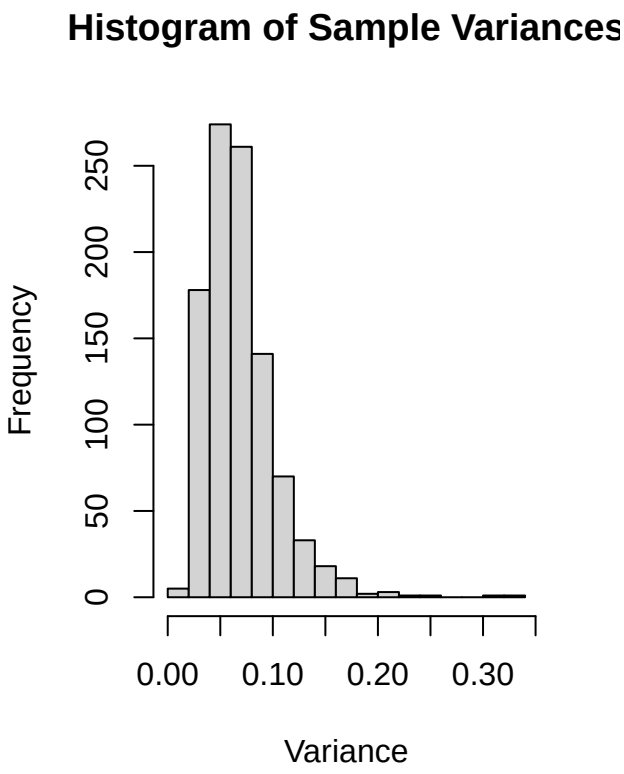
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.176328 -0.028968  0.002267  0.001639  0.032215  0.236233
```

```
print(summary_vars)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01396 0.04554 0.06251 0.06878 0.08251 0.32053
```

```
# Plot histograms of the means and variances
## Create a 1x2 plot layout
par(mfrow = c(1, 2))
hist(sample_means, main = "Histogram of Sample Means", xlab = "Mean")
hist(sample_vars, main = "Histogram of Sample Variances", xlab = "Variance")
```





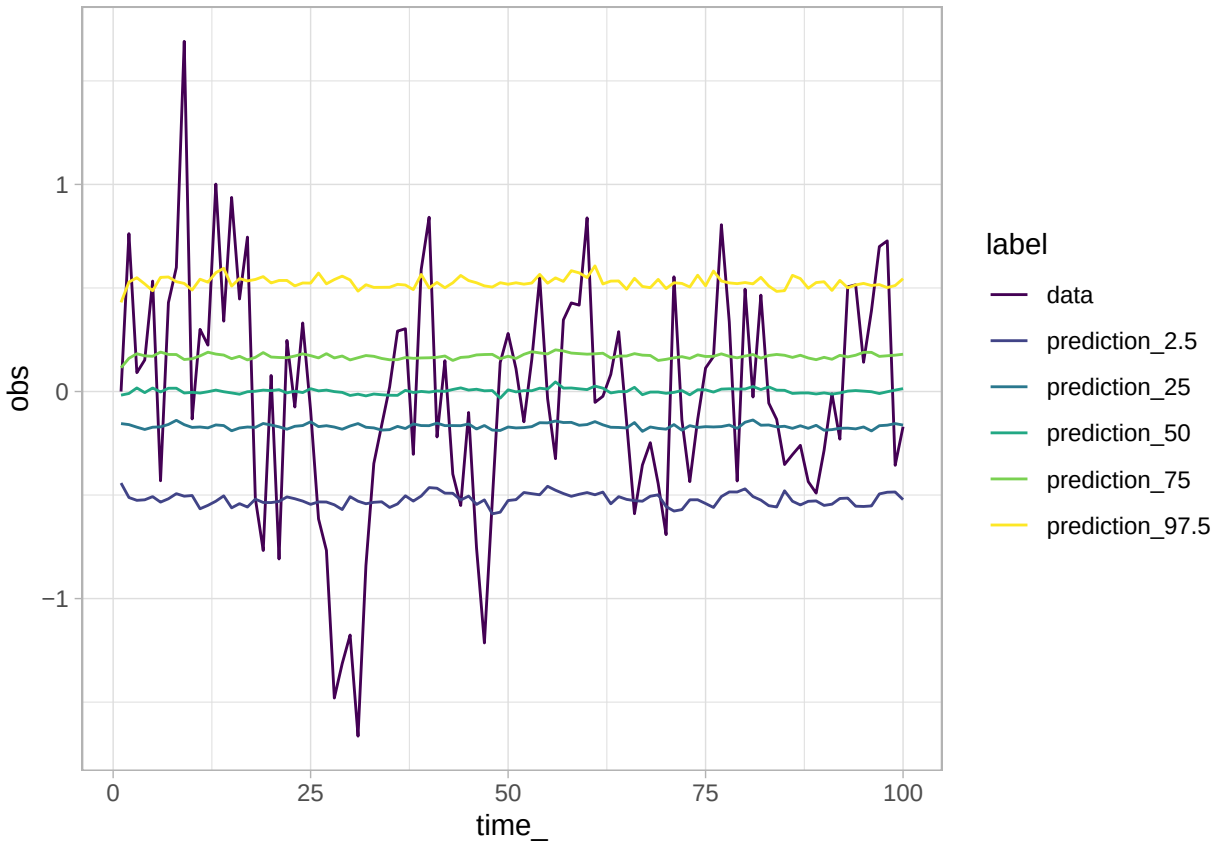

```
quant <- apply(posterior_predictive_samples, 2, quantile, c(0.025, 0.5, 0.975))
quantile(posterior_predictive_samples[[1]], c(0.025, 0.5, 0.975))
```

```
##       2.5%        50%      97.5%
## -0.1291648 -0.1291648 -0.1291648
```

Question 9

The observed data fall within the range of the posterior predictive samples, and the summary statistics is roughly similar. Hence, the model fits the data.

```
# create a matrix to store the 5 percentiles
quant <- apply(posterior_predictive_samples, 2, quantile, c(0.025, 0.25, 0.5, 0.75, 0.975))

plot1 <- data.frame(
  time_ = rep(1:100, 6),
  obs = c(data, quant[1, ], quant[2, ], quant[3, ], quant[4, ], quant[5, ]),
  label = c(rep("data", 100), rep("prediction_2.5", 100), rep("prediction_25", 100), rep("prediction_50
)

ggplot(plot1, aes(x = time_, y = obs, color = label)) +
    geom_line() +
    theme_light() +
    scale_color_viridis_d()
```



Real data

Question 1

I assume the information in the Github is sufficient enough to analyze cases numbers and deaths due to COVID-19.
It sufficiently describes the methodology of collection for each variable. Yet, if we could pull the additional data from other resources, we could have more accurate analysis. For example, the ratio of vaccinated patients by year.

Question 2

If I use AR(1) model, I would try transforming the data to make it stationary such as take a log of data because the data itself changed dramatically at some points. The number itself rose rapidly.
Hence, the AR(1) model can be remodeled as:

$$\log y_i = p \cdot \log y_{i-1} + e_i$$

```
# data import
us <- read.csv("/Users/satoshiido/Documents/statistical-analysis/656/hw/hw1/data/covid_us.txt")
states <- read.csv("/Users/satoshiido/Documents/statistical-analysis/656/hw/hw1/data/covid_us-states.tx
head(us);head(states)
```

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

```
##   X       date      state fips cases deaths
## 1 1 2020-01-25 California    6     1      0
## 2 2 2020-01-26 California    6     2      0
## 3 3 2020-01-27 California    6     2      0
## 4 4 2020-01-28 California    6     2      0
## 5 5 2020-01-29 California    6     2      0
## 6 6 2020-01-30 California    6     2      0
```

```
tail(us);tail(states)
```

```
##           date   cases deaths
## 213 2020-08-20 5587401 174137
## 214 2020-08-21 5636428 175298
## 215 2020-08-22 5681459 176248
## 216 2020-08-23 5713799 176694
## 217 2020-08-24 5754178 177198
## 218 2020-08-25 5792911 178411
```

```
##       X       date      state fips  cases deaths
## 578 578 2020-08-24 California    6 676236  12250
## 579 579 2020-08-24    Indiana   18  89602   3225
## 580 580 2020-08-24      Texas   48 607621  11824
## 581 581 2020-08-25 California    6 682320  12409
## 582 582 2020-08-25    Indiana   18  90460   3241
## 583 583 2020-08-25      Texas   48 613326  12012
```

Question 3

As we can see from the data, the number of cases and deaths were increasing over time and never decreased. From this information, I would assume ρ is positive. Therefore, I would make ρ postive

$$\rho \sim uniform(0,1); \quad \log \sigma \sim N(0, 10^2)$$

Question 4

```
# take the log of the data
us$cases <- log(us$cases)

# take the log of the data which only has the positive values
us$deaths[us$deaths > 0] <- log(us$deaths[us$deaths > 0])
```

```r
# display the data where the number of cases is positive
us_val <- us[which(us$date == "2020-07-01") : nrow(us),]
us <- us[1:which(us$date == "2020-06-30"),]
head(us)
```

```
##         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
```