STAT 656: HW1

Rohan Dekate

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 $i(i = 1, \cdots, n)$ is generated according to $y_i = \rho y_{i-1} + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ independently, and ρ and σ are unknown parameters. For simplicity, we shall assume that y_0 is a fixed constant. We will also assume $|\rho| < 1$.

1. Write the log-likelihood function $\log L(\rho, \sigma^2 | y_0, y_1, \cdots, y_n)$ for $(\rho, \sigma^2)^T$ for the AR(1) model.

The model is $y_i = \rho y_{i-1} + \epsilon_i$, where $\epsilon_i \sim N(0, \sigma^2)$.

This implies that the conditional distribution of y_i given y_{i-1} is:

$$y_i | y_{i-1} \sim N(\rho y_{i-1}, \sigma^2)$$

The PDF for a single observation y_i conditional on y_{i-1} from the definition of a Normal distribution is:

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

The likelihood function is the product of the individual observations from $i = 1$ to n:

$$L(\rho, \sigma^2 | y_0, \ldots, y_n) = \prod_{i=1}^{n} f(y_i | y_{i-1}; \rho, \sigma^2)$$

Substituting the PDF from the previous step:

$$L(\rho, \sigma^2 | y_0, \ldots, y_n) = \prod_{i=1}^{n} \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \rho y_{i-1})^2}{2\sigma^2}\right)\right]$$

Simplifying the product gives:

$$L(\rho, \sigma^2 | y_0, \ldots, y_n) = (2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \rho y_{i-1})^2\right)$$

Finally, the log-likelihood function is:

$$\log L(\rho, \sigma^2 | y_0, \ldots, y_n) = \log\left[(2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \rho y_{i-1})^2\right)\right]$$

Using the properties of logarithms, we separate the terms:

$$\log L(\rho, \sigma^2 | \ldots) = \log((2\pi\sigma^2)^{-n/2}) + \log\left(\exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \rho y_{i-1})^2\right)\right)$$

This simplifies to:

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

Expanding the logarithm term gives the final expression:

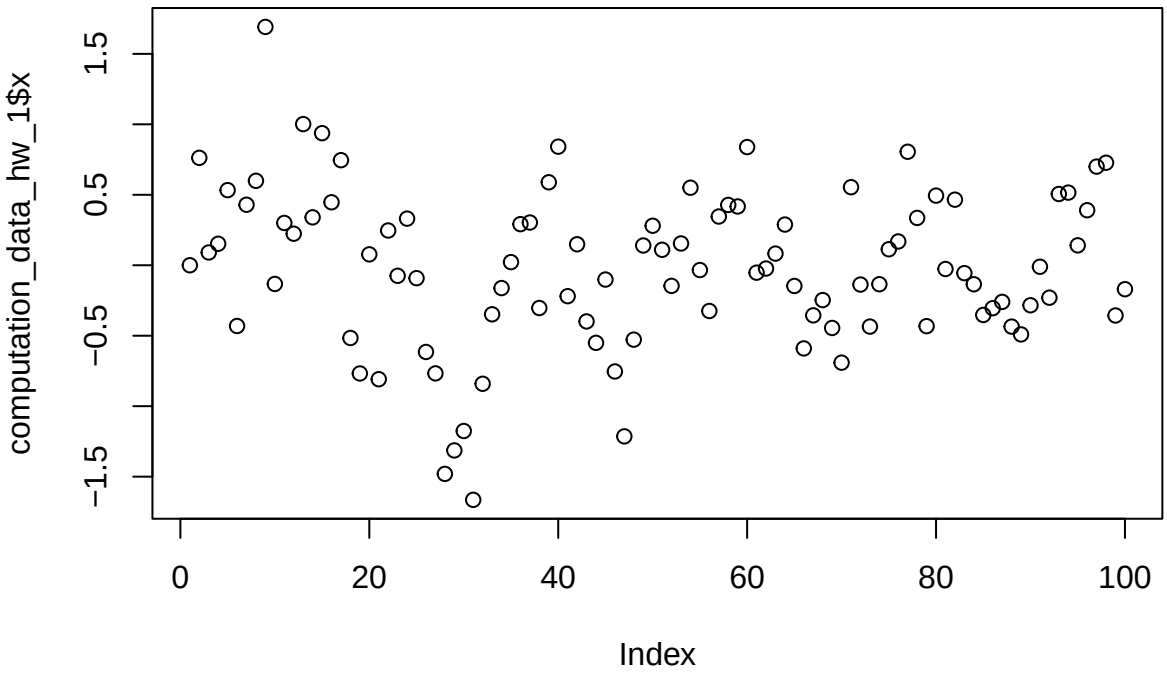
$$\log L(\rho, \sigma^2 | y_0, \dots, y_n) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \rho y_{i-1})^2$$

For rest of this problem, we shall take ρ, $\log\sigma$ to be our estimands of interest, and consider data in the file `computation_data_hw_1.csv`, generated from this type of process with $y_0 = 0$.

```
computation_data_hw_1 <- read.csv("./computation_data_hw_1.csv")
head(computation_data_hw_1)
```

```
##   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
```

```
plot(computation_data_hw_1$x)
```



2. Write an R function that computes the log of the likelihood functions for $(\rho, \log(\sigma))^T$ for this data.

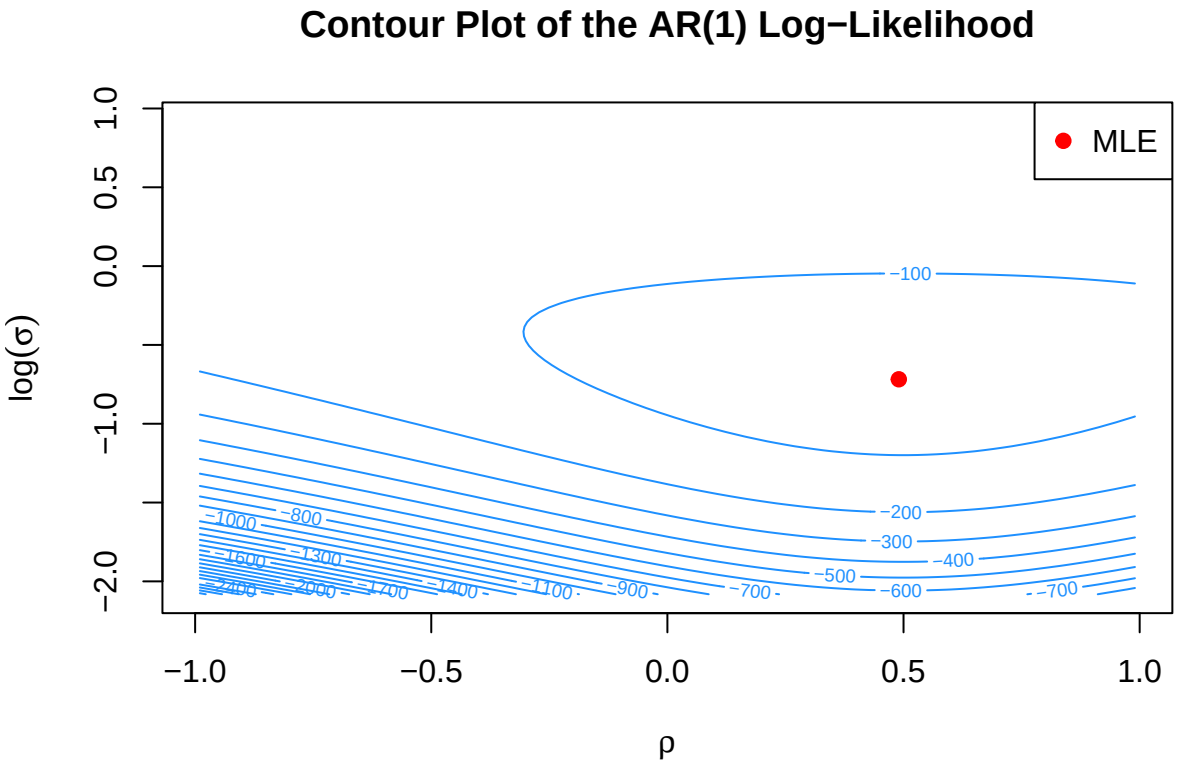
```
ar_loglik <- function(rho, log_sig)
{
  n = length(computation_data_hw_1$x) # Number of rows/samples
```

```
  # sigma = exp(log_sig)
  # sigma^2 = (exp(log_sig))^2 = exp(2 * log_sig)
  sigma_sq = exp(2 * log_sig)

  y0 = 0
  y_current = computation_data_hw_1$x
  y_obs = c(y0, y_current[-n])
  sum_sq_err = sum((y_current - rho*y_obs)^2)

  log_lik = -(n/2)*log(2*pi) - (n/2)*log(sigma_sq) - (1/(2*sigma_sq)) * sum_sq_err
  return(log_lik)
}
```

Provide a visualization of this log-likelihood as a contour plot. Hint: The outer and contour function in R can be useful for creating the visualization, see also the code of lectures 2 and 3.

```
rho_grid <- seq(-0.99, 0.99, length.out = 100)
log_sig_grid <- seq(log(sd(computation_data_hw_1$x)) - 1.5, log(sd(computation_data_hw_1$x)) + 1.5, leng
#log_sig_grid = seq(-1, 1, length.out = 100)
log_lik_surface = outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_loglik))
```

```
contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_lik_surface,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  main = "Contour Plot of the AR(1) Log-Likelihood",
  nlevels = 20,
  col = "dodgerblue"
)

# Add a point for the maximum likelihood estimate for reference
max_lik_index <- which(log_lik_surface == max(log_lik_surface), arr.ind = TRUE)
points(rho_grid[max_lik_index[1]], log_sig_grid[max_lik_index[2]], pch = 19, col = "red")
legend("topright", "MLE", pch = 19, col = "red", bg = "white")
```

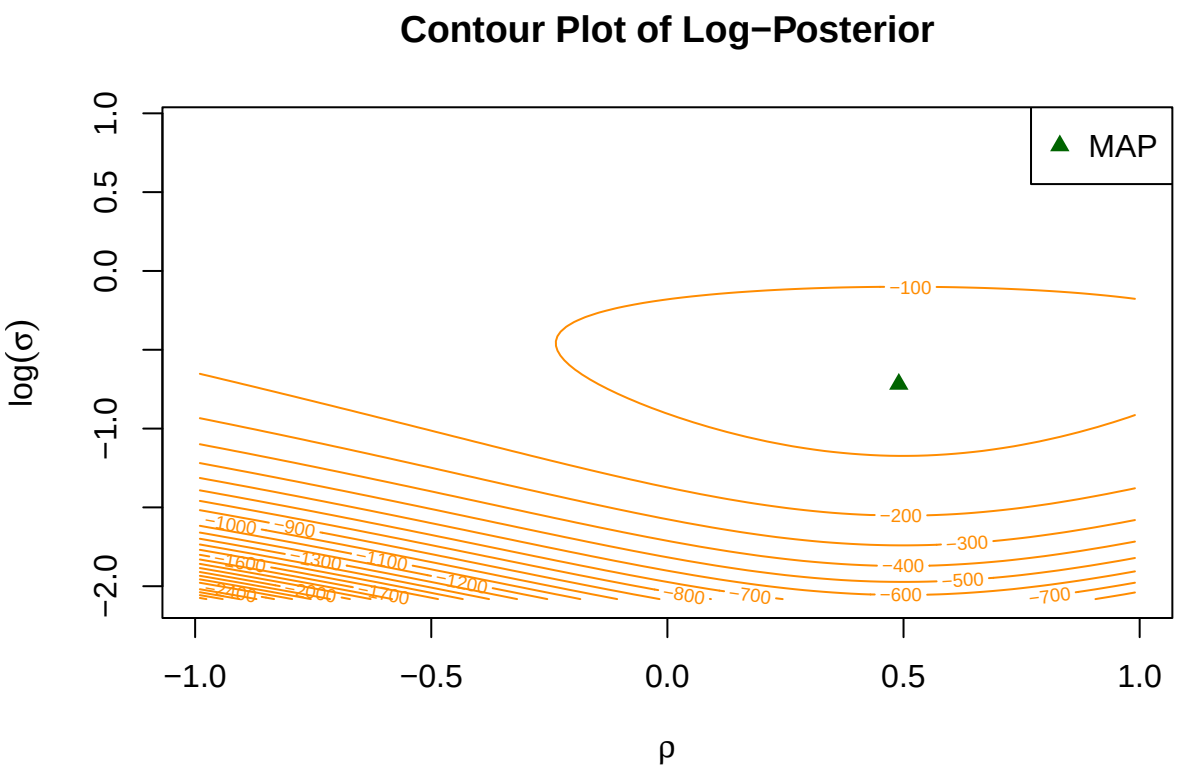


3. For the purposes of this problem, suppose we specify $\rho \sim \text{Uniform}(-1, 1)$, $\log(\sigma) \sim \mathcal{N}(0, 10^2)$ independently *a priori* (note that this may not be an appropriate prior for the parameters of an AR(1) model in general). Write an R function that computes the log of the posterior density (upto a constant) for $(\rho, \log(\sigma))^T$ under this prior. Provide a visualization of this function as above. How does this function compare to the log-likelihood function? Would you say that this prior specification is overly informative? Why or why not?

```r
ar_logposterior <- function(rho, log_sig) {

  log_lik = ar_loglik(rho, log_sig)

  # Log prior for rho ~ Uniform(-1, 1)
  # The density is 1/(1 - (-1)) = 1/2 for -1 < rho < 1
  # The log density is log(1/2) = -log(2) for -1 < rho < 1
  # and -Inf otherwise
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)

  # Log prior for log_sig ~ Normal(0, 10^2)
  log_prior_log_sig = -0.5 * log(2 * pi * 100) - (log_sig - 0)^2 / (2 * 100)

  log_posterior = log_lik + log_prior_rho + log_prior_log_sig

  return(log_posterior)
}
```

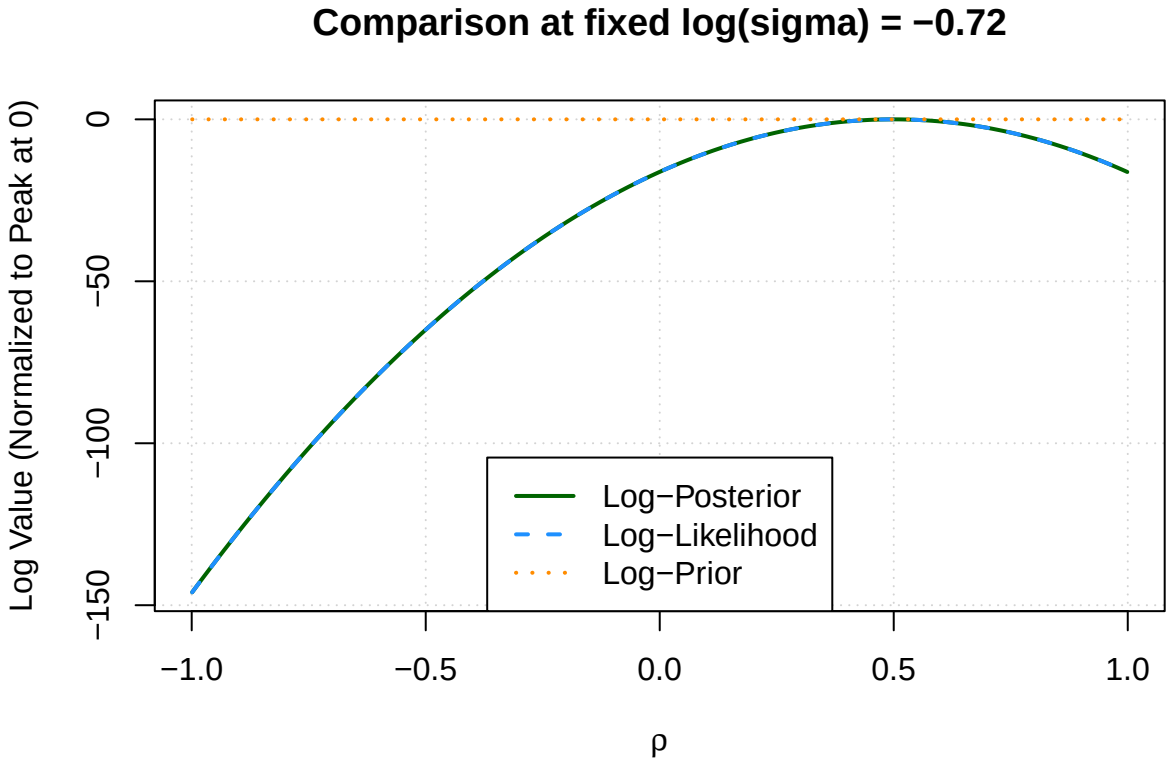
```r
log_post_grid <- outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_logposterior))
```

```r
contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_post_grid,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  main = "Contour Plot of Log-Posterior",
  nlevels = 20,
  col = "darkorange"
)

# Add a point for the maximum a posteriori (MAP) estimate for reference
map_index <- which(log_post_grid == max(log_post_grid, na.rm=TRUE), arr.ind = TRUE)
points(rho_grid[map_index[1]], log_sig_grid[map_index[2]], pch = 17, col = "darkgreen")
legend("topright", "MAP", pch = 17, col = "darkgreen", bg = "white")
```




```r
fixed_log_sig <- log_sig_grid[map_index[2]]

# Create a finer rho vector for a smoother plot
rho_vec <- seq(-0.999, 0.999, length.out = 500)

# Calculate log-likelihood and log-posterior along this vector
log_lik_vals <- outer(rho_vec, fixed_log_sig, FUN = Vectorize(ar_loglik))

# Define and calculate log-prior for the line plot
ar_logprior <- function(rho, log_sig) {
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)
  log_prior_log_sig = dnorm(log_sig, mean = 0, sd = 10, log = TRUE)
  return(log_prior_rho + log_prior_log_sig)
```

```
}
log_prior_vals <- ar_logprior(rho_vec, fixed_log_sig)

log_post_vals <- log_lik_vals + log_prior_vals

# Plot 3: Combined line plot
# We normalize the values to peak at 0 for easier comparison of their shapes
plot(
  rho_vec,
  log_post_vals - max(log_post_vals),
  type = 'l',
  col = "darkgreen",
  lwd = 2,
  xlab = expression(rho),
  ylab = "Log Value (Normalized to Peak at 0)",
  main = paste("Comparison at fixed log(sigma) =", round(fixed_log_sig, 2)),
  #ylim = c(-50, 5), # Adjust ylim if the plot is clipped
  panel.first = grid()
)

# Add lines for the log-likelihood and log-prior
lines(rho_vec, log_lik_vals - max(log_lik_vals), col = "dodgerblue", lwd = 2, lty = 2)
lines(rho_vec, log_prior_vals - max(log_prior_vals), col = "darkorange", lwd = 2, lty = 3)

# Add legend
legend(
  "bottom",
  legend = c("Log-Posterior", "Log-Likelihood", "Log-Prior"),
  col = c("darkgreen", "dodgerblue", "darkorange"),
  lwd = 2,
  lty = c(1, 2, 3),
  bg = "white"
)
```



The log-posterior and log-likelihood plots are essentially the same. This is natural as $Posterior \propto Likelihood \times Prior$ from the Bayes' rule. A constant prior implies that the two functions will have the same shape.

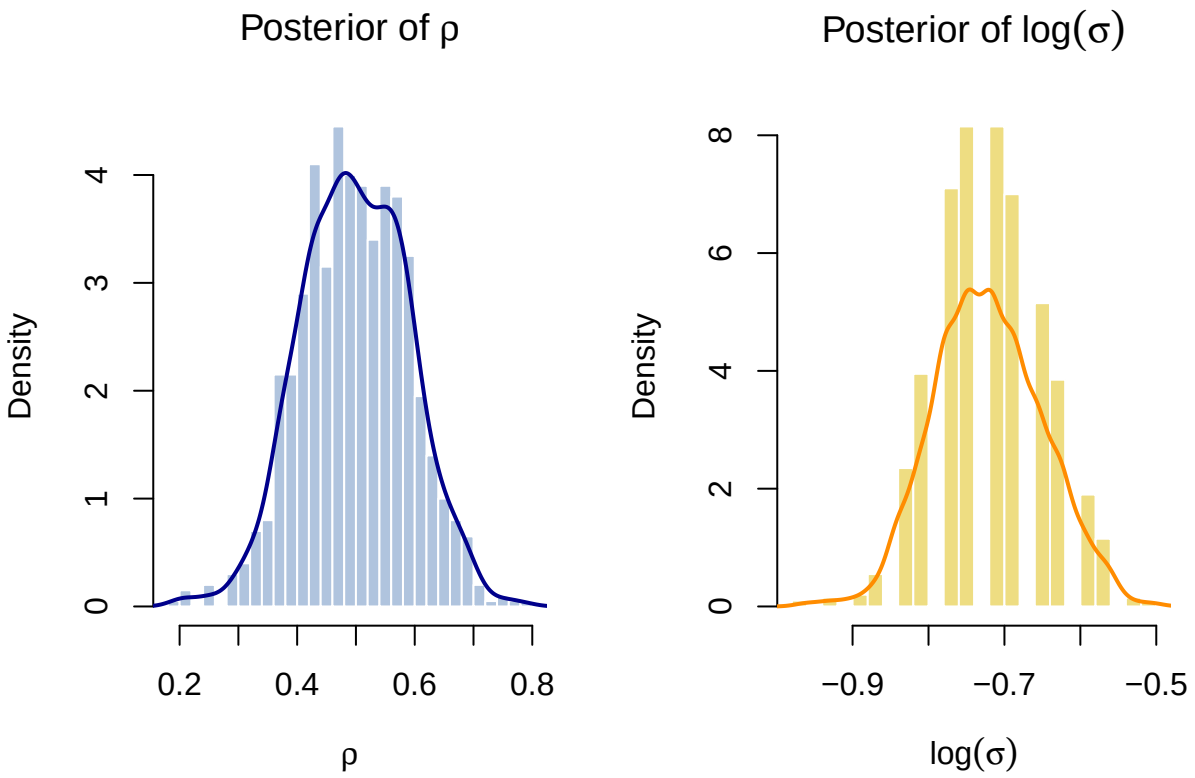
The prior specification is not informative as it is flat and assigns equal probabilities to all possible values of the parameter within the specified range.

4. Draw 1000 values of $(\rho, \log(\sigma))^T$ from a discrete grid approximation to the posterior. Be sure to describe your choice of discrete grid. Hint: The previous step can be helpful in this regard, together with the R function sample. Again, look at the code associated with the lectures.

I use the same grid as the one generated to create the contour plots. As we can see from the posterior contour plot, it effectively covers the region of the parameter space where the posterior probability is concentrated.

```
# For numerical stability
max_log_post <- max(log_post_grid, na.rm = TRUE)
post_grid <- exp(log_post_grid - max_log_post)

# Normalize the grid values to create a probability distribution that sums to 1.
prob_grid <- post_grid / sum(post_grid, na.rm = TRUE)

# Draw 1000 samples from posterior
n_points <- length(rho_grid) * length(log_sig_grid)
grid_indices <- 1:n_points

n_samples <- 1000

sampled_indices <- sample(
    grid_indices,
    size = n_samples,
```

```r
    replace = TRUE,
    prob = as.vector(prob_grid)
)

# Convert to 2D grid
sampled_row_col <- arrayInd(sampled_indices, dim(log_post_grid))

# Extract rho and log(sigma)
sampled_rho <- rho_grid[sampled_row_col[, 1]]
sampled_log_sig <- log_sig_grid[sampled_row_col[, 2]]

# Combine the sampled parameter values into a data frame.
posterior_samples <- data.frame(
    rho = sampled_rho,
    log_sig = sampled_log_sig
)

# Set up a 1x2 plotting area
par(mfrow = c(1, 2), mar = c(5, 4, 4, 2))

# Histogram for rho
hist(posterior_samples$rho,
     main = expression(paste("Posterior of ", rho)),
     xlab = expression(rho),
     freq = FALSE, # Plot density instead of frequency
     col = "lightsteelblue",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples$rho), col = "darkblue", lwd = 2)

# Histogram for log(sigma)
hist(posterior_samples$log_sig,
     main = expression(paste("Posterior of ", log(sigma))),
     xlab = expression(log(sigma)),
     freq = FALSE, # Plot density instead of frequency
     col = "lightgoldenrod",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples$log_sig), col = "darkorange", lwd = 2)
```




```
# Reset plotting parameters
par(mfrow = c(1, 1))
```

5. Use these draws to calculate the following summaries for each of ρ and $\log(\sigma)$: 0.025, 0.25, 0.5, 0.75, 0.975 quantiles, mean, standard deviation, skewness, and kurtosis. You can use the library moments if you want.

```
library(moments)
# Define the quantiles we want to compute
probs <- c(0.025, 0.25, 0.5, 0.75, 0.975)

# Summarize rho
rho_quantiles <- quantile(posterior_samples$rho, probs = probs)
rho_mean <- mean(posterior_samples$rho)
rho_sd <- sd(posterior_samples$rho)
rho_skewness <- skewness(posterior_samples$rho)
rho_kurtosis <- kurtosis(posterior_samples$rho)

# Print rho summaries
cat("\nParameter: rho\n")
```

```
##
## Parameter: rho
```

```
print(data.frame(
    Statistic = c(paste0(probs*100, "% Quantile"), "Mean", "Std. Dev.", "Skewness", "Kurtosis"),
    Value = c(rho_quantiles, rho_mean, rho_sd, rho_skewness, rho_kurtosis)
), row.names = FALSE)
```

```
##       Statistic       Value
```

```
##   2.5% Quantile  0.33000000
##    25% Quantile  0.43000000
##    50% Quantile  0.49000000
##    75% Quantile  0.57000000
##  97.5% Quantile  0.67000000
##            Mean  0.49850000
##       Std. Dev.  0.09103646
##        Skewness -0.04717296
##        Kurtosis  2.97612185
```

```r
# Summarize log(sigma)
log_sig_quantiles <- quantile(posterior_samples$log_sig, probs = probs)
log_sig_mean <- mean(posterior_samples$log_sig)
log_sig_sd <- sd(posterior_samples$log_sig)
log_sig_skewness <- skewness(posterior_samples$log_sig)
log_sig_kurtosis <- kurtosis(posterior_samples$log_sig)

# Print log(sigma) summaries
cat("\nParameter: log(sigma)\n")
```

```
##
## Parameter: log(sigma)
```

```r
print(data.frame(
    Statistic = c(paste0(probs*100, "% Quantile"), "Mean", "Std. Dev.", "Skewness", "Kurtosis"),
    Value = c(log_sig_quantiles, log_sig_mean, log_sig_sd, log_sig_skewness, log_sig_kurtosis)
), row.names = FALSE)
```

```
##       Statistic       Value
##   2.5% Quantile -0.83924308
##    25% Quantile -0.77863702
##    50% Quantile -0.71803096
##    75% Quantile -0.68772793
##  97.5% Quantile -0.56651581
##            Mean -0.72142490
##       Std. Dev.  0.07116356
##        Skewness  0.06221675
##        Kurtosis  2.93481328
```

6. Write an R function that takes parameters $(\rho, \log(\sigma))^T$ and simulates a new dataset y^{rep} according to the AR process. Recall that we can simulate from the posterior predictive distribution of new datasets y^{rep} given todays dataset as follows: first simulate a parameter set from the posterior distribution, and use this to simulate a new dataset. Use your two earlier R functions to generate 1000 such posterior predictive samples. Summarize your draws.

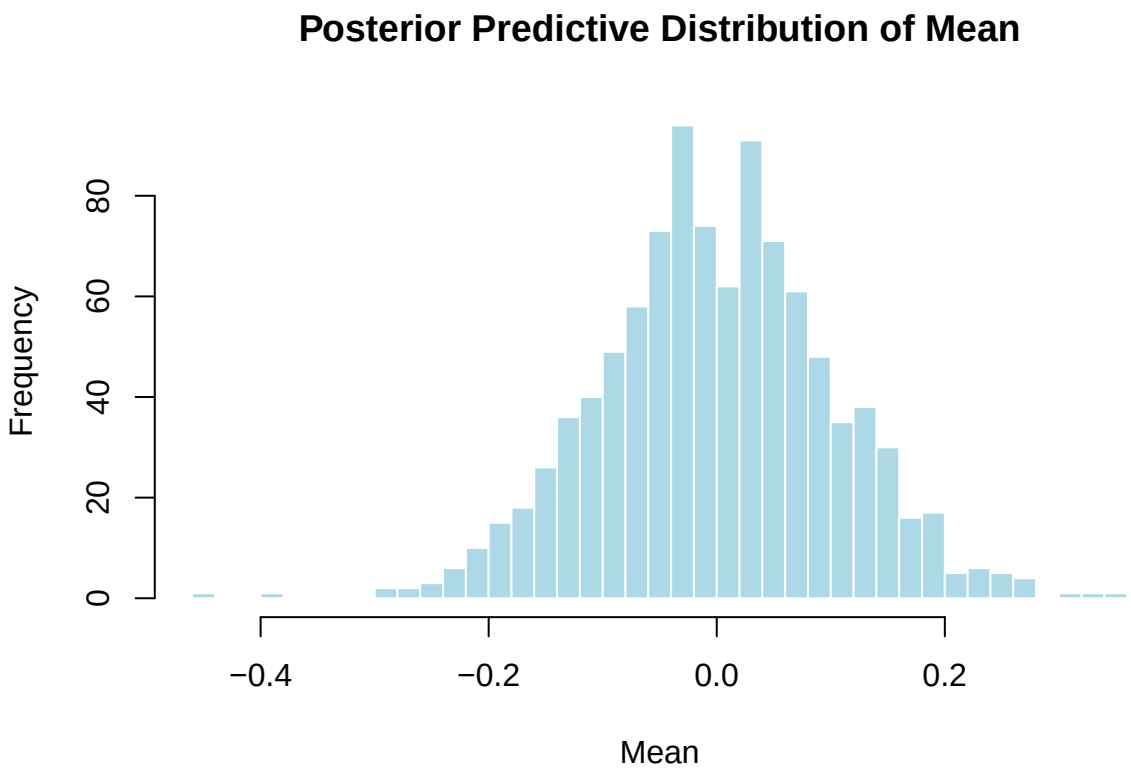
```r
simulate_ar_process <- function(rho, log_sig, n) {

  sigma <- exp(log_sig)
  y_rep <- numeric(n)

  # The posterior for rho is within (-1, 1), ensuring stationarity.
  # We start the process by drawing from the stationary distribution.
  #y_rep[1] <- rnorm(1, mean = 0, sd = sigma / sqrt(1 - rho^2))
```

```
  y_rep[1] <- 0 # Fixed constant

  # Simulate the rest of the process
  for (i in 2:n) {
    y_rep[i] <- rho * y_rep[i-1] + rnorm(1, mean = 0, sd = sigma)
  }

  return(y_rep)
}
```

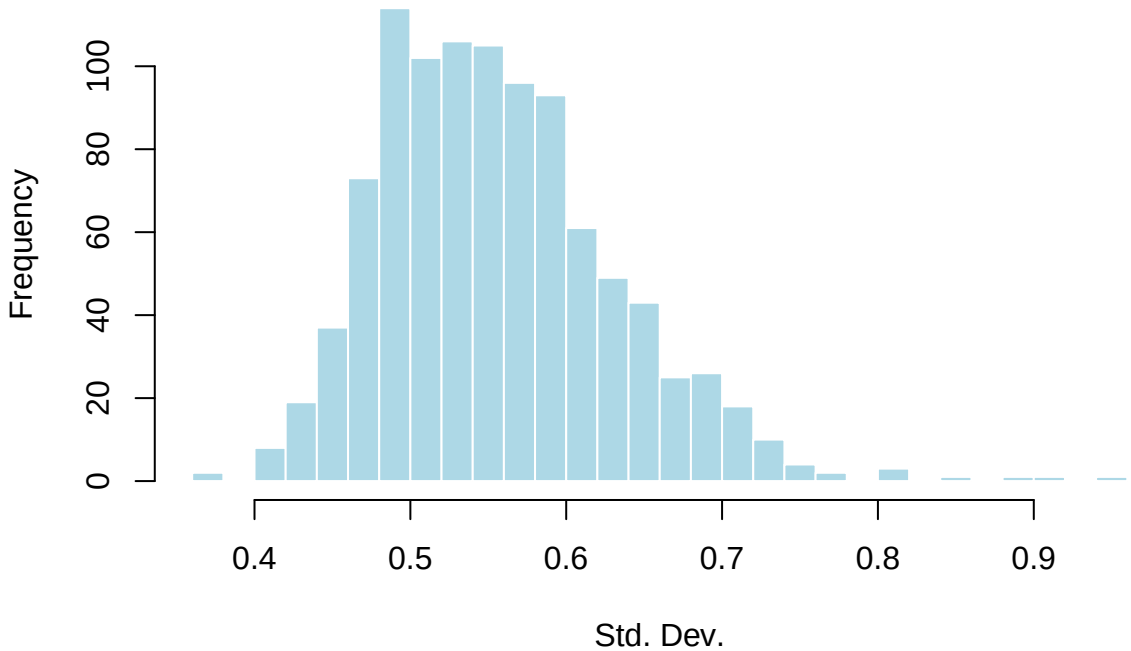
```
n_reps <- 1000
n_data <- nrow(computation_data_hw_1)
posterior_predictive_samples <- vector("list", n_reps)

for (i in 1:n_reps) {
  current_rho <- posterior_samples$rho[i]
  current_log_sig <- posterior_samples$log_sig[i]

  # Simulate a new dataset using these parameters
  posterior_predictive_samples[[i]] <- simulate_ar_process(
    rho = current_rho,
    log_sig = current_log_sig,
    n = n_data
  )
}
```

```
rep_means <- sapply(posterior_predictive_samples, mean)
rep_sds <- sapply(posterior_predictive_samples, sd)
rep_skewness <- sapply(posterior_predictive_samples, skewness)
rep_kurtosis <- sapply(posterior_predictive_samples, kurtosis)
```

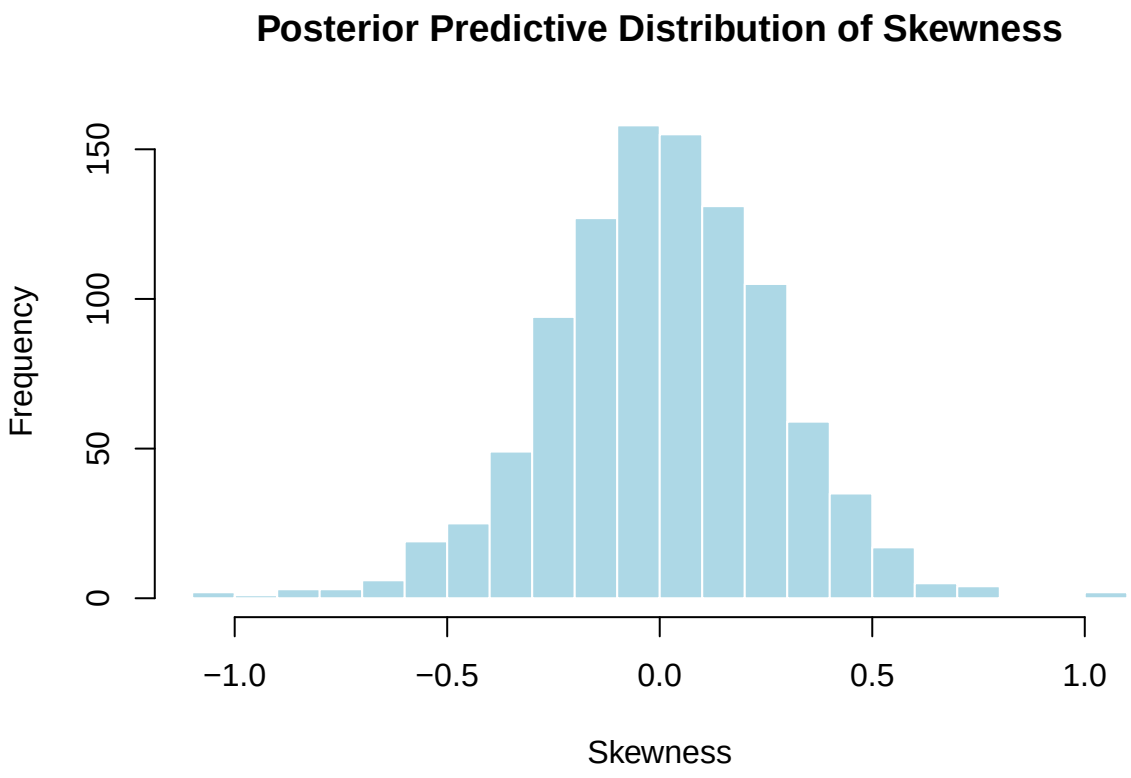
```
hist(rep_means, main = "Posterior Predictive Distribution of Mean", xlab = "Mean", col="lightblue", bor
```



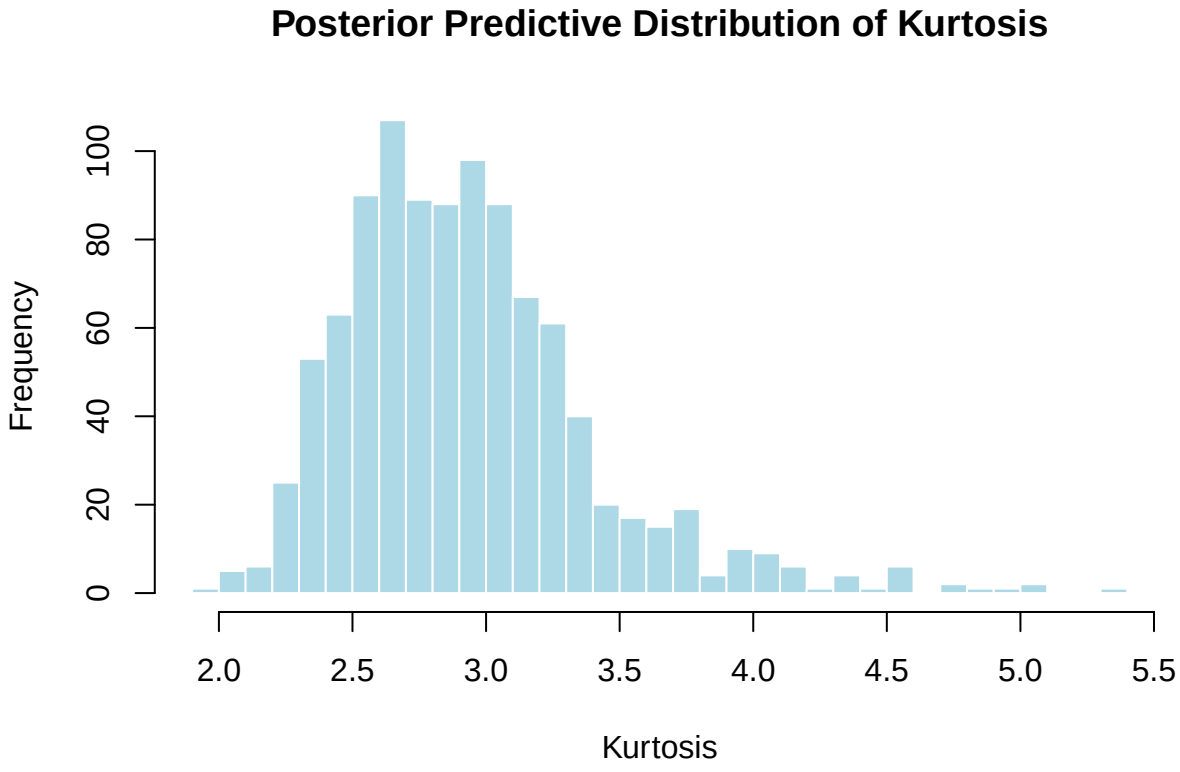

```
hist(rep_sds, main = "Posterior Predictive Distribution of Std. Dev.", xlab = "Std. Dev.", col="lightblu
```




```
hist(rep_skewness, main = "Posterior Predictive Distribution of Skewness", xlab = "Skewness", col="ligh
```




```
hist(rep_kurtosis, main = "Posterior Predictive Distribution of Kurtosis", xlab = "Kurtosis", col="ligh
```

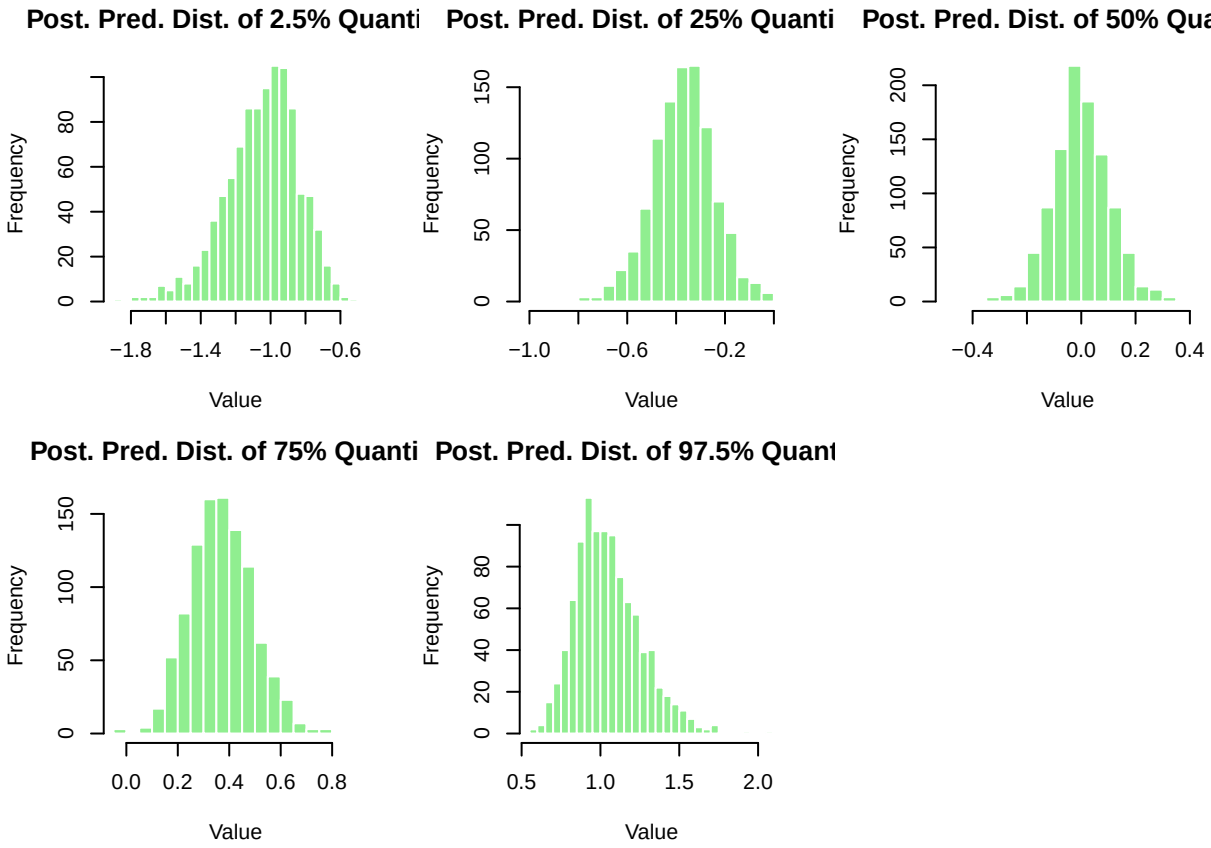



```
quantile_probs <- c(0.025, 0.25, 0.5, 0.75, 0.975)

rep_quantiles <- t(sapply(posterior_predictive_samples, quantile, probs = quantile_probs))
colnames(rep_quantiles) <- paste0(quantile_probs * 100, "%")
```

```r
# Print a summary table
quantile_summary_df <- data.frame(
  Quantile = colnames(rep_quantiles),
  Posterior_Pred_Mean = colMeans(rep_quantiles),
  Posterior_Pred_95_CI_Lower = apply(rep_quantiles, 2, quantile, 0.025),
  Posterior_Pred_95_CI_Upper = apply(rep_quantiles, 2, quantile, 0.975)
)

print(quantile_summary_df, row.names=FALSE)
```

```
##  Quantile Posterior_Pred_Mean Posterior_Pred_95_CI_Lower
##      2.5%       -1.0475522227                 -1.5267708
##       25%       -0.3709546467                 -0.6335391
##       50%        0.0004005226                 -0.2041520
##       75%        0.3712688727                  0.1541219
##     97.5%        1.0504971156                  0.7086058
##  Posterior_Pred_95_CI_Upper
##                  -0.6946707
##                  -0.1324806
##                   0.2178627
##                   0.6170834
##                   1.5252860
```

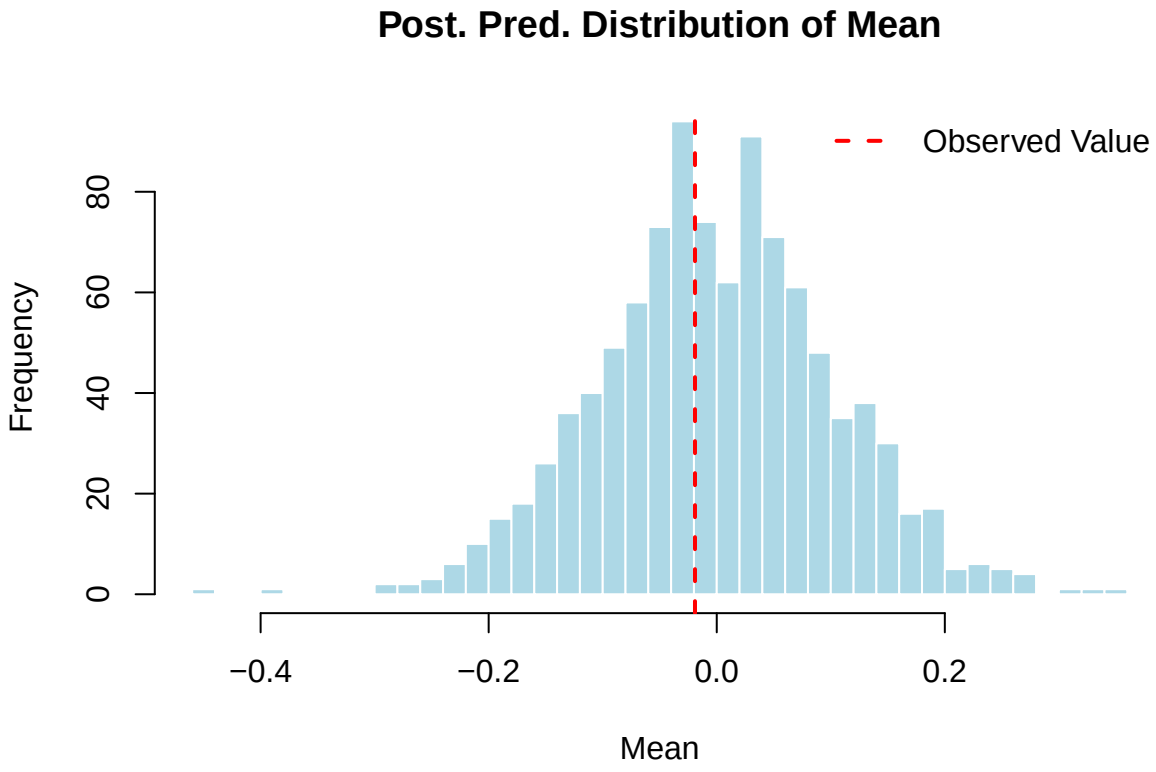
```r
par(mfrow = c(2, 3), mar = c(4, 4, 3, 2))

for (i in 1:length(quantile_probs)) {
  hist(rep_quantiles[, i],
       main = paste("Post. Pred. Dist. of", colnames(rep_quantiles)[i], "Quantile"),
       xlab = "Value",
       col = "lightgreen",
       border = "white",
       breaks = 30)
}

# Reset plotting parameters
par(mfrow = c(1, 1))
```



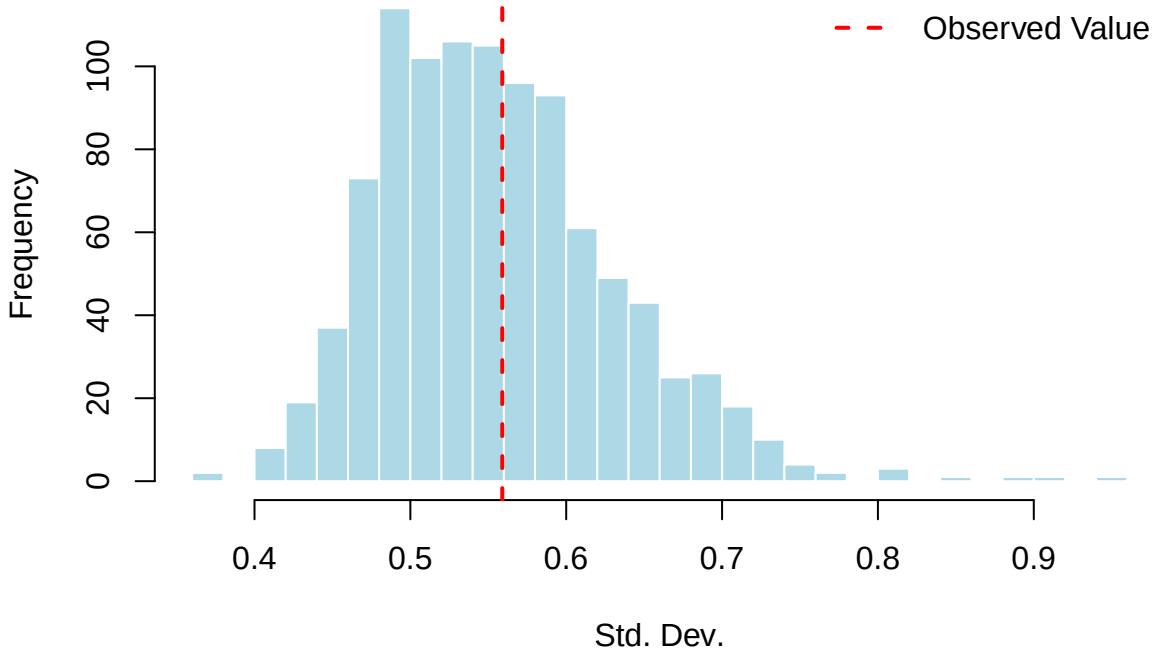
7. Compare the observed data to these posterior predictive summaries. Also create a plot where the observed data is superposed on these posterior predictive trajectories. What can you say about the model fit? Does the model appear appropriate?

```
obs_data <- computation_data_hw_1$x
obs_stats <- c(
    mean(obs_data),
    sd(obs_data),
    skewness(obs_data),
    kurtosis(obs_data)
)
```

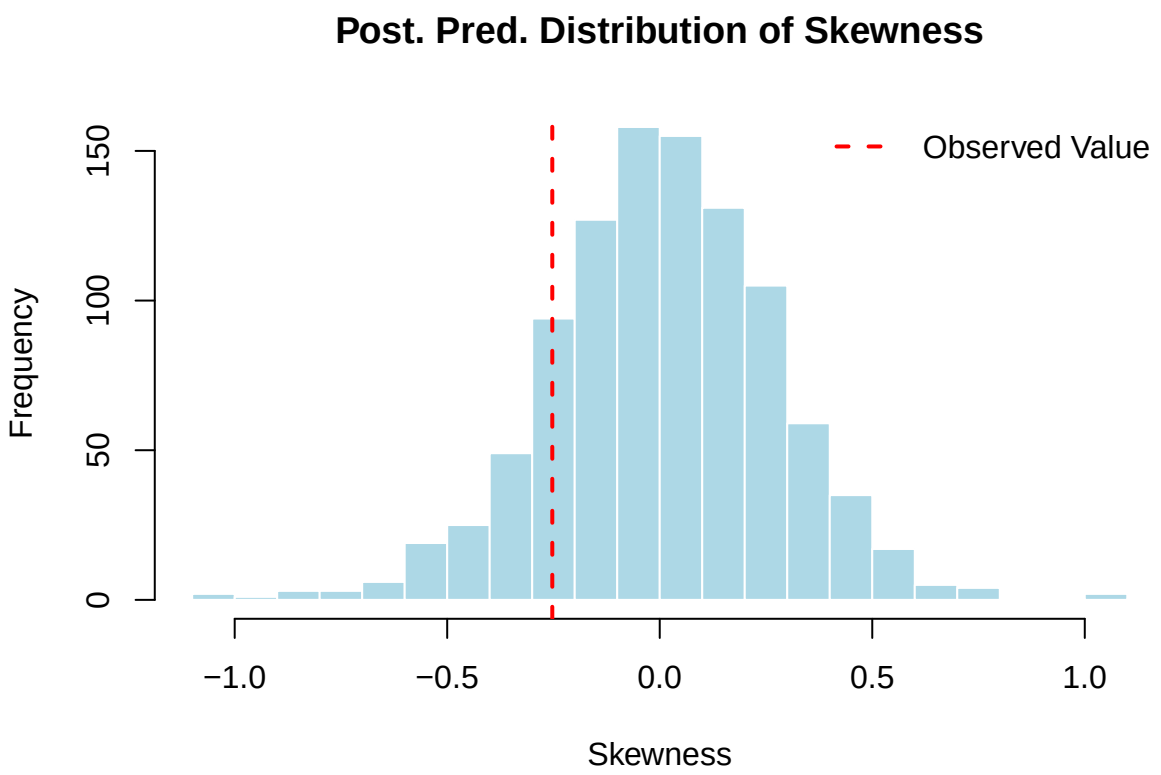
```
# Histogram for the Mean
hist(rep_means, main = "Post. Pred. Distribution of Mean", xlab = "Mean", col="lightblue", border="white
abline(v = obs_stats[1], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```



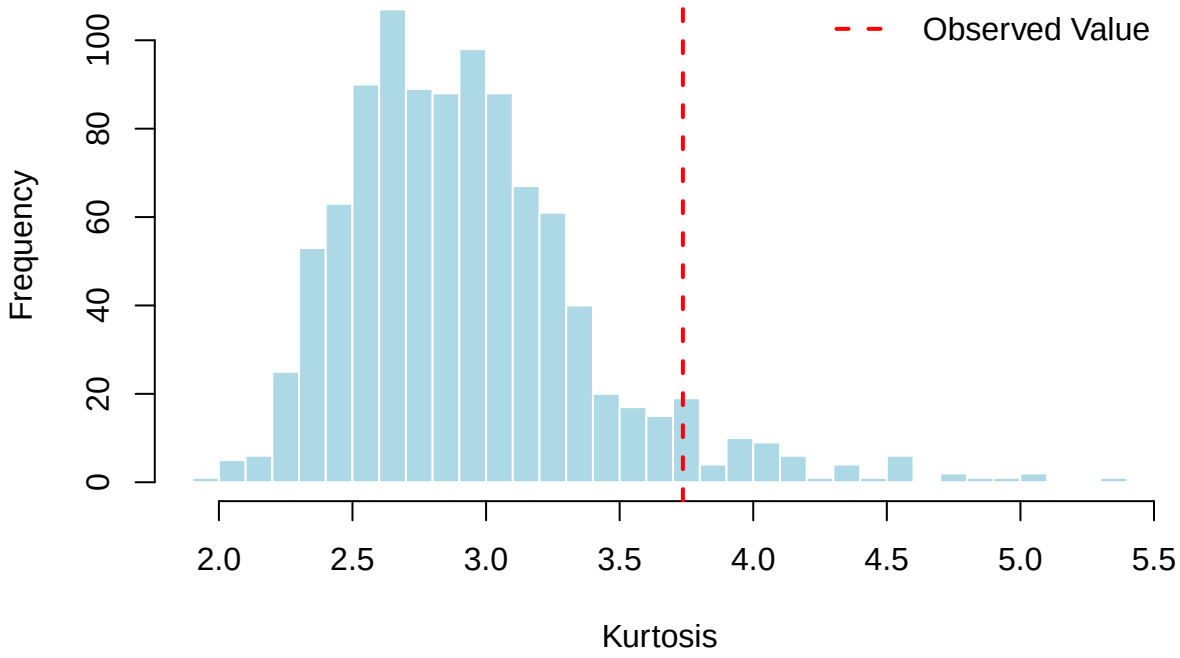

```r
# Histogram for the Standard Deviation
hist(rep_sds, main = "Post. Pred. Distribution of Std. Dev.", xlab = "Std. Dev.", col="lightblue", bord
abline(v = obs_stats[2], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```



```r
# Histogram for Skewness
hist(rep_skewness, main = "Post. Pred. Distribution of Skewness", xlab = "Skewness", col="lightblue", b
abline(v = obs_stats[3], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```



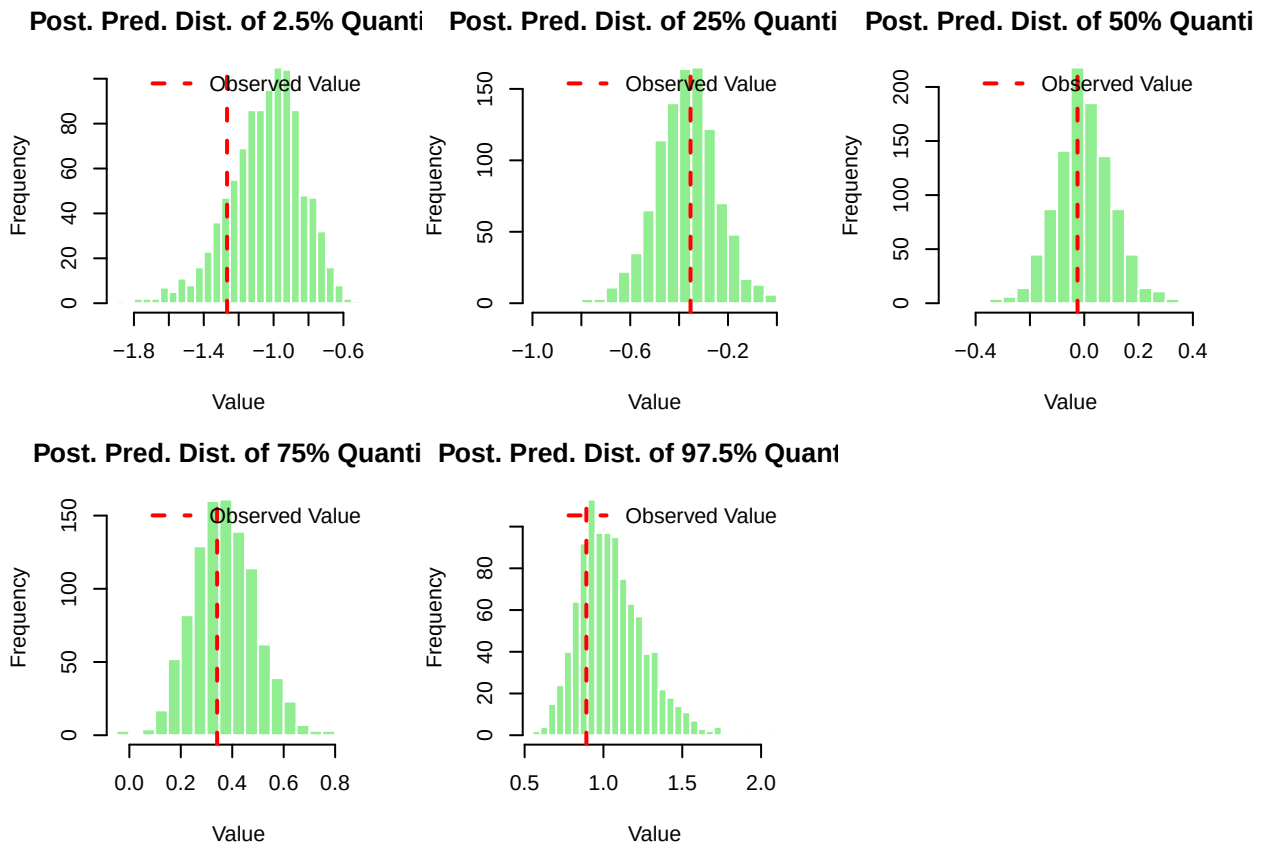

```r
# Histogram for Kurtosis
hist(rep_kurtosis, main = "Post. Pred. Distribution of Kurtosis", xlab = "Kurtosis", col="lightblue", b
abline(v = obs_stats[4], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```




```r
quantile_probs <- c(0.025, 0.25, 0.5, 0.75, 0.975)

rep_quantiles <- t(sapply(posterior_predictive_samples, quantile, probs = quantile_probs))
colnames(rep_quantiles) <- paste0(quantile_probs * 100, "%")

# Calculate the same quantiles for the observed data
obs_quantiles <- quantile(obs_data, probs = quantile_probs)

# Print a summary table
quantile_summary_df <- data.frame(
  Quantile = colnames(rep_quantiles),
  Observed_Value = obs_quantiles,
  Posterior_Pred_Mean = colMeans(rep_quantiles),
  Posterior_Pred_95_CI_Lower = apply(rep_quantiles, 2, quantile, 0.025),
  Posterior_Pred_95_CI_Upper = apply(rep_quantiles, 2, quantile, 0.975)
)
print(quantile_summary_df, row.names=FALSE)
```

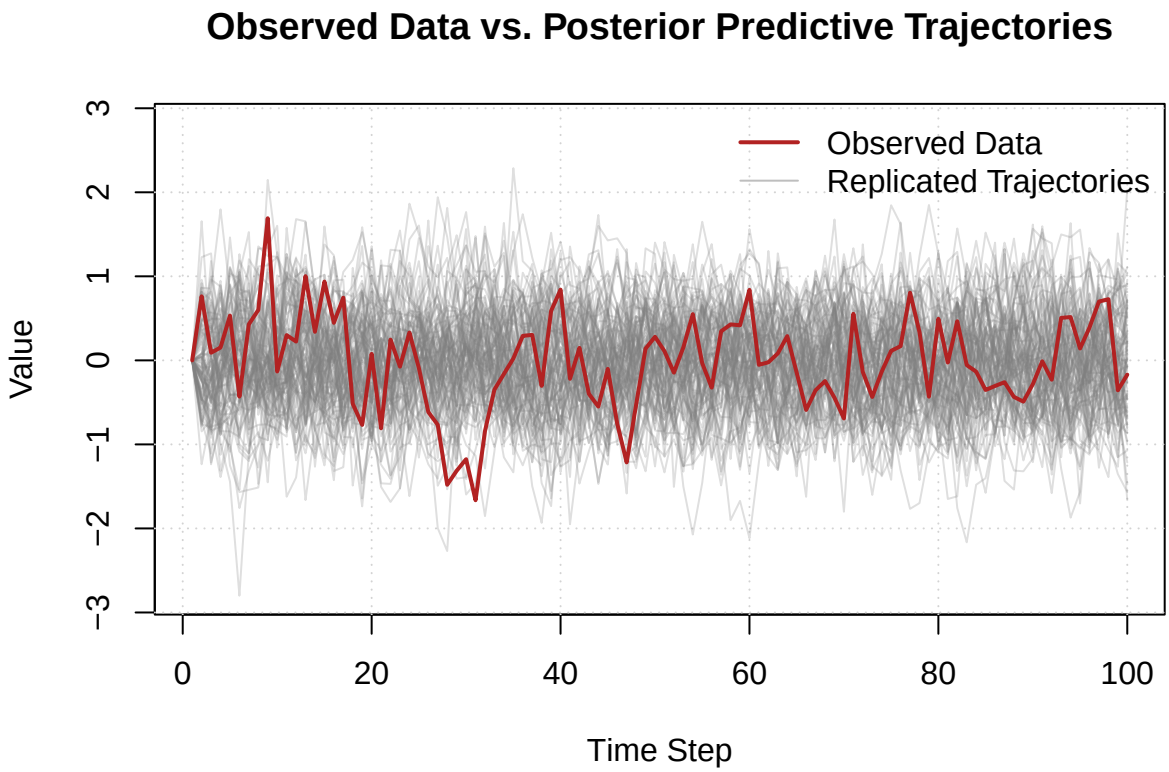
```
##  Quantile Observed_Value Posterior_Pred_Mean Posterior_Pred_95_CI_Lower
##      2.5%    -1.26647784       -1.0475522227                -1.5267708
##       25%    -0.35332512       -0.3709546467                -0.6335391
##       50%    -0.02484622        0.0004005226                -0.2041520
##       75%     0.34131050        0.3712688727                 0.1541219
##     97.5%     0.89171576        1.0504971156                 0.7086058
##  Posterior_Pred_95_CI_Upper
##                  -0.6946707
##                  -0.1324806
##                   0.2178627
##                   0.6170834
##                   1.5252860
```

```r
par(mfrow = c(2, 3), mar = c(4, 4, 3, 2))

for (i in 1:length(quantile_probs)) {
  hist(rep_quantiles[, i],
       main = paste("Post. Pred. Dist. of", colnames(rep_quantiles)[i], "Quantile"),
       xlab = "Value",
       col = "lightgreen",
       border = "white",
       breaks = 30)
  abline(v = obs_quantiles[i], col = "red", lwd = 2, lty = 2)
  legend("topright", "Observed Value", col = "red", lty = 2, lwd = 2, bty = "n")
}

# Reset plotting parameters
par(mfrow = c(1, 1))
```




```r
y_range <- range(c(obs_data, unlist(posterior_predictive_samples)))

# Plot the first 100 replicated trajectories as semi-transparent lines
plot(NULL,
     xlim = c(1, n_data),
     ylim = y_range,
     main = "Observed Data vs. Posterior Predictive Trajectories",
     xlab = "Time Step",
     ylab = "Value")
grid()
```

```r
# Add replicated data lines
for (i in 1:100) {
  lines(posterior_predictive_samples[[i]], col = rgb(0.5, 0.5, 0.5, 0.25))
}

# Superimpose the observed data as a bold, colored line
lines(obs_data, col = "firebrick", lwd = 2)

# Add a legend
legend("topright",
       legend = c("Observed Data", "Replicated Trajectories"),
       col = c("firebrick", "gray"),
       lwd = c(2, 1),
       bty = "n")
```



From the summary plots, we see that the posterior captures the mean and standard deviation of the observed data well but does not capture the kurtosis and skewness as well as the others. We also see this trend in the quantile plots where the quantiles at the extremes (2.5%, 97.5%) do not capture the observed value but the other quantiles capture the observed value very well.

From the final plot we see a lot of replicated trajectories that capture the general trend of the observed data. The model does not accurately fit the observed data everywhere but does a pretty good job. So I'd say that the model is appropriate given the high uncertainty in the observed data.

Real Data

Here we will use your functions from the previous question to determine the utility of the AR(1) model for predicting COVID-19 cases and deaths. Specifically, you will analyze two sets of data that contain the cases and deaths from COVID-19 in the United States overall, and for individual states, respectively. The data

on cases and deaths in the entire United States are in covid us.txt, and the data on cases and deaths for three individual states are in covid us-states.txt. Both datasets were downloaded from the New York Times' Coronavirus (COVID-19) Data in the United States Github page, which contains further information and details regarding these data.

```
covid_us <- read.table("./covid_us.txt", header = TRUE, sep = ",")
head(covid_us)
```

```
##         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
```

```
covid_us_states <- read.table("./covid_us-states.txt", header = TRUE, sep = ",")
head(covid_us_states)
```

```
##   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
```

1. Do you believe that the information given on the Github page is sufficient for analyzing the data? Why or why not? If not, what other information would you have requested that the New York Times provide, or what other information would you have collected if you had the resources to do so?

The GItHub page states that the New York Times compiled cumulative counts of coronavirus cases in the US at the national and state level (we omit the county level in this analysis). Since these numbers were recorded after testing it is possible that the cases are under estimated than the true values. There could be people who haven't undergone testing but carry the virus. Also, since the testing kits were limited that poses another source of observation error.

The `covid_us.txt` file contains date, cases and deaths column. So we can consider this as a time series data with number of cases and deaths for each date. There is no other information available. In a way this is similar to the Synthetic data `computation_data_hw_1.csv` but we have 2 columns instead of 1 as the observed data.

So it is sufficient to analyze using AR(1) process but we don't have knowledge about how the cases and deaths were arrived at. There is no context to determine an informative prior.

The `covid_us-states.txt` file simply adds a state column and a geographical identifier in the form of `fips`. But this is not sufficient to build an epidemic model.

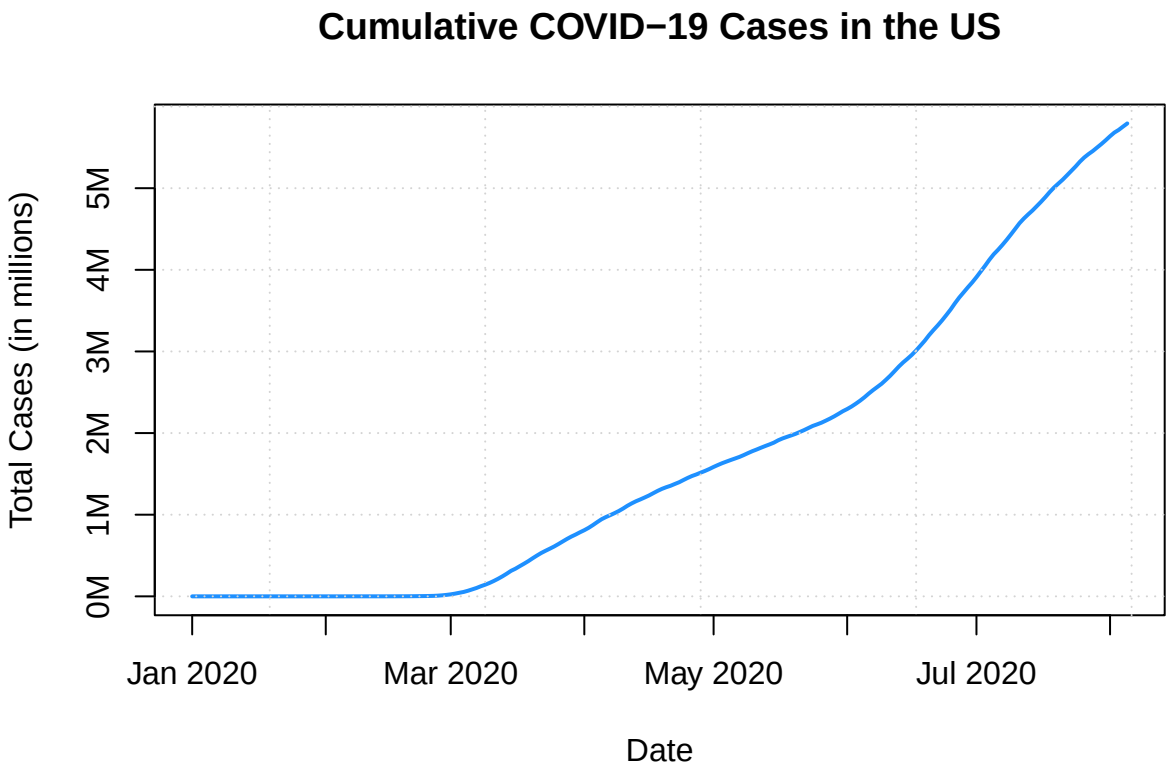
The files have sufficient information to build a blackbox autoregressive model.

I'd like to add information relvant to interventions like lockdowns, mask mandates, hospitalizations, demographic data (age, sex, occupation) etc. to add context to the observed variables.

2. Specify an AR(1) model for the data on the entire United States (not any of the individual states). In particular, would you model the raw data as an AR(1) process, our would you transform it first in some way? Explain.

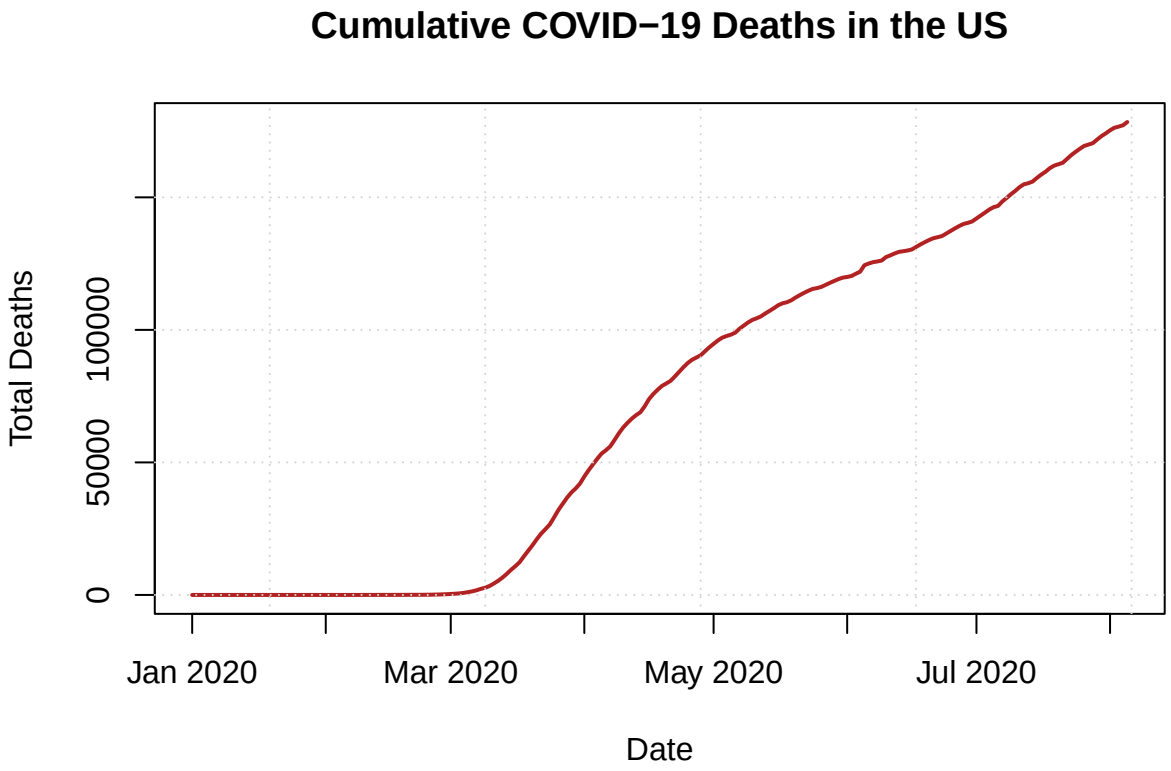
Since we know that the data is cumulatively reported it makes sense to model the change in the count rather than the cumulative value itself.

```
covid_us$date <- as.Date(covid_us$date)

plot(covid_us$date, covid_us$cases,
     type = 'l',
     col = 'dodgerblue',
     main = 'Cumulative COVID-19 Cases in the US',
     xlab = 'Date',
     ylab = 'Total Cases (in millions)',
     lwd = 2,
     xaxt = 'n',
     yaxt = 'n'
)
grid()
axis(2, at = seq(0, max(covid_us$cases), by = 1000000),
     labels = paste0(seq(0, max(covid_us$cases), by = 1000000) / 1000000, "M"))
axis.Date(1, at = seq(min(covid_us$date), max(covid_us$date), by = "month"),
          format = "%b %Y")
```




```
plot(covid_us$date, covid_us$deaths,
     type = 'l',
     col = 'firebrick',
     main = 'Cumulative COVID-19 Deaths in the US',
     xlab = 'Date',
     ylab = 'Total Deaths',
```

```
     lwd = 2,
     xaxt = 'n'
)
grid()
axis.Date(1, at = seq(min(covid_us$date), max(covid_us$date), by = "month"),
          format = "%b %Y")
```



We also see a gradual increase in the number of cases indicating that the data does not have a constant mean and variance. Thus it is non-stationary. The model used for synthetic data would be a poor choice for the covid data because it assumes stationary process.

I will transform the raw data to model the change in the counts, rather than the cumulative. As the covid dataset's variance is not constant I'd also log-transform the observations.

If C_t is the observed cumulative value of cases/death, I'd get the change in the count y_t as follows: $y_t = C_t - C_{t-1}$.

Applying the log transform will yield: $z_t = \log(y_t + 1)$ where 1 is added to ensure that the log value is defined for $y_t = 0$.

The process would look similar to the AR(1) process: $z_t = c + \rho \cdot z_{t-1} + \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ and t is the time index, c is the intercept. An intercept term is required because the process is non-stationary. However, we can get around this by making the process zero mean by subtracting the sample average from the log transformed data. Let $\bar{z}$ be the sample mean. We can make the observations zero mean by $z\prime_t = z_t - \bar{z}$. We will recover the same AR(1) model that we used for the synthetic data: $z\prime_t = \rho \cdot z\prime_{t-1} + \epsilon_t$.

3. Specify a prior for all of the model parameters. You can choose any prior you like, but you must justify your choice.

The parameters of my AR(1) model are the autoregressive coefficient ρ, and the standard deviation of the error term σ. As we had done in the case of synthetic data, we will work with $\log(\sigma)$.

We choose the following priors as before:

1. $\rho \sim \text{Uniform}(-1, 1)$: For AR(1) to be stationary, $|\rho| < 1$. This enforces zero probability for any value outside the range $(-1, 1)$. Although this choice (as before) is weakly informative, before observing the data we consider any value of ρ to be equally likely. Although we see a rising trend in the plots above for the covid cases and deaths, let's see if this weakly informative prior can model the process or not. We choose it owing to its flexibility and also because I have setup my code for synthetic data with this prior. We can change this prior if the model fit is not good.
2. $\log(\sigma) \sim \mathcal{N}(0, 1^2)$: $\sigma > 0$ strictly, thus $\log(\sigma) \in \mathbb{R}$. I chose this prior rather than $\sigma^2 = 10^2$ because the posterior variance was very wide. So I narrowed it down after checking the plots (which are not shown in this analysis). It is again a weakly informative prior allowing the data to determine the range of fluctuation of daily changes.

I have also modifed the log-likelihood function as the raw data is non-stationary and thereby not independent. (Changed n to $n - 1$ in log-likelihood)

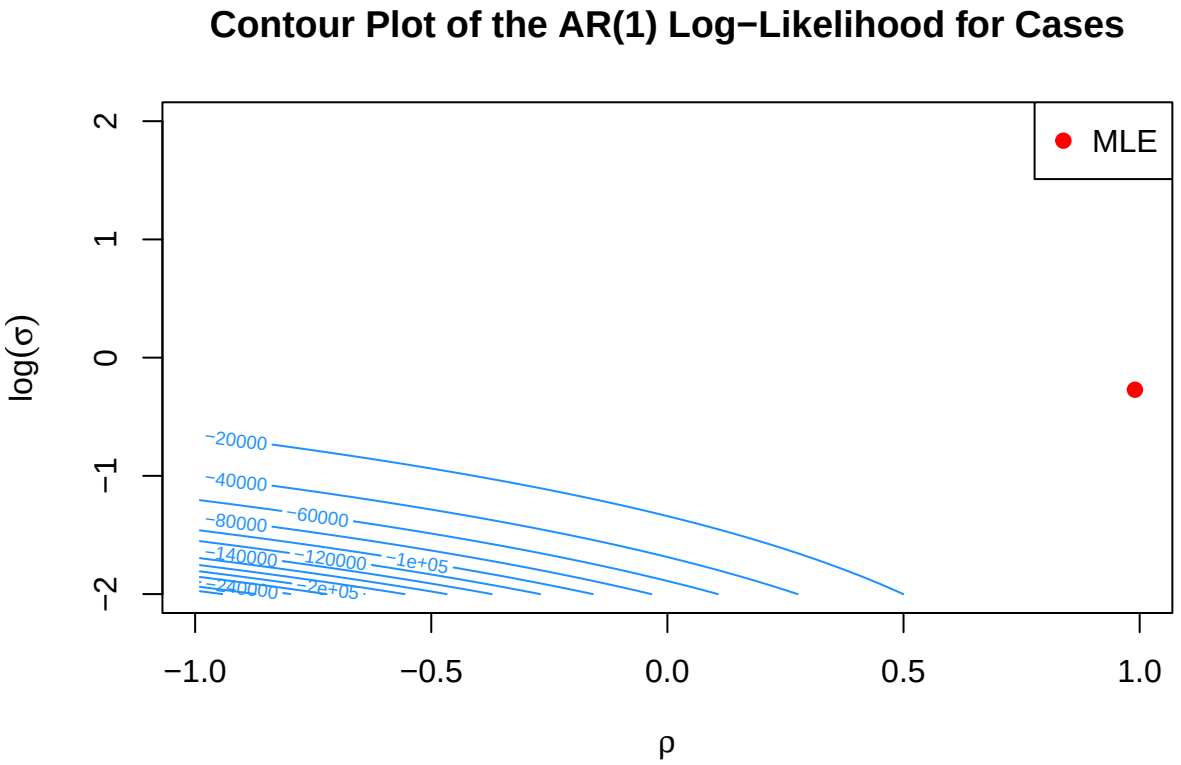
4. Fit your AR(1) model to data from the first time point until June 30. Calculate and provide a visualization of the posterior distribution of the estimand. Summarize the posterior distribution using moments as before.

```
# Process the data as described in (2)
# Filter data for the period up to and including June 30, 2020
training_data <- covid_us[covid_us$date <= as.Date("2020-06-30"), ]
daily_cases <- diff(training_data$cases) # Calculate change
daily_cases <- c(training_data$cases[1], daily_cases) # First value as initial data value
log_daily_cases <- log(daily_cases + 1) # log-transform
mean_log_daily_cases <- mean(log_daily_cases) # calculate mean
processed_data_cases <- (log_daily_cases - mean_log_daily_cases)
computation_data_covid <- data.frame(x = processed_data_cases)
```

```
ar_loglik_cases <- function(rho, log_sig)
{
  n = length(computation_data_covid$x) # Number of rows/samples

  # sigma = exp(log_sig)
  # sigma^2 = (exp(log_sig))^2 = exp(2 * log_sig)
  sigma_sq = exp(2 * log_sig)

  y_current = computation_data_covid$x
  y_prev <- c(0, y_current[-n])
  sum_sq_err = sum((y_current - rho*y_prev)^2)

  #log_lik = -(n/2)*log(2*pi) - (n/2)*log(sigma_sq) - (1/(2*sigma_sq)) * sum_sq_err
  log_lik <- -(n-1)/2 * log(2*pi) -
             (n-1)/2 * log(sigma_sq) -
             (1/(2*sigma_sq)) * sum_sq_err
  return(log_lik)
}
```

```
rho_grid <- seq(-0.99, 0.99, length.out = 200)
log_sig_grid <- seq(-2, 2, length.out = 200)
log_lik_surface = outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_loglik_cases))
```

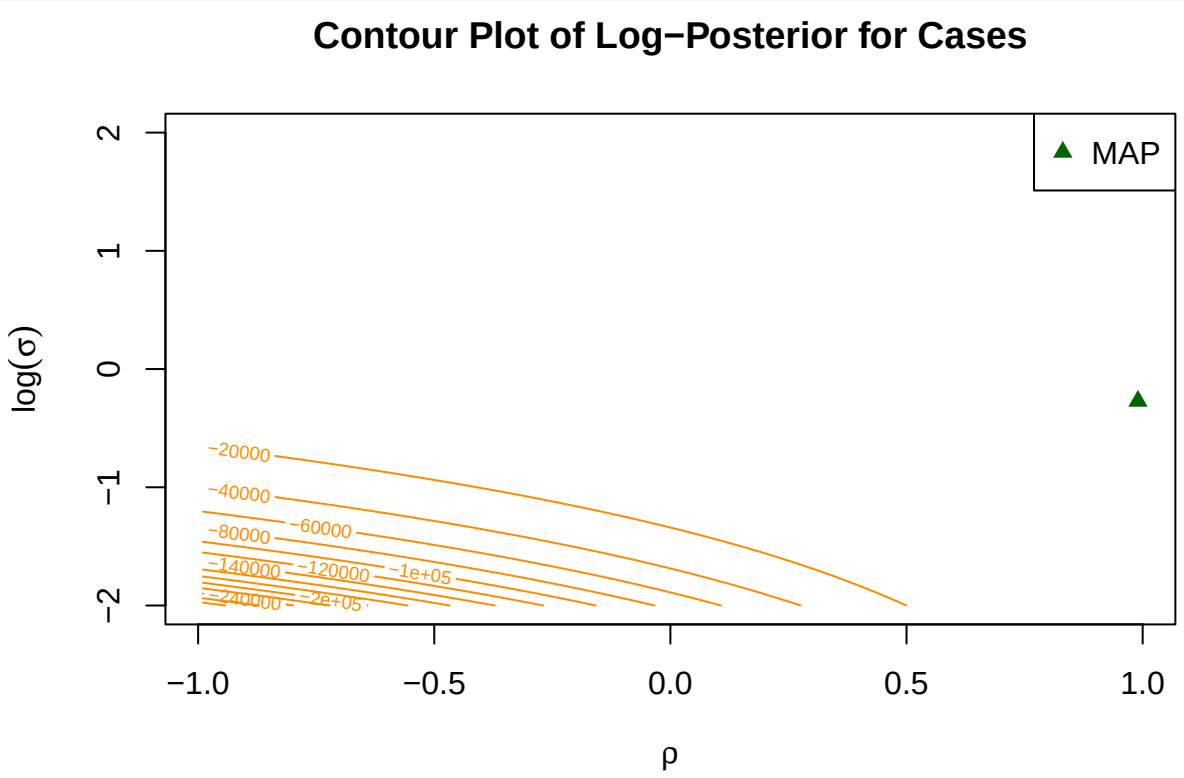
```r
contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_lik_surface,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  main = "Contour Plot of the AR(1) Log-Likelihood for Cases",
  nlevels = 20,
  col = "dodgerblue"
)

# Add a point for the maximum likelihood estimate for reference
max_lik_index <- which(log_lik_surface == max(log_lik_surface), arr.ind = TRUE)
points(rho_grid[max_lik_index[1]], log_sig_grid[max_lik_index[2]], pch = 19, col = "red")
legend("topright", "MLE", pch = 19, col = "red", bg = "white")
```




```r
ar_logposterior_cases <- function(rho, log_sig) {

  log_lik = ar_loglik_cases(rho, log_sig)

  # Log prior for rho ~ Uniform(-1, 1)
  # The density is 1/(1 - (-1)) = 1/2 for -1 < rho < 1
  # The log density is log(1/2) = -log(2) for -1 < rho < 1
  # and -Inf otherwise
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)

  # Log prior for log_sig ~ Normal(0, 10^2)
  log_prior_log_sig = -0.5 * log(2 * pi * 1) - (log_sig - 0)^2 / (2 * 1)
```

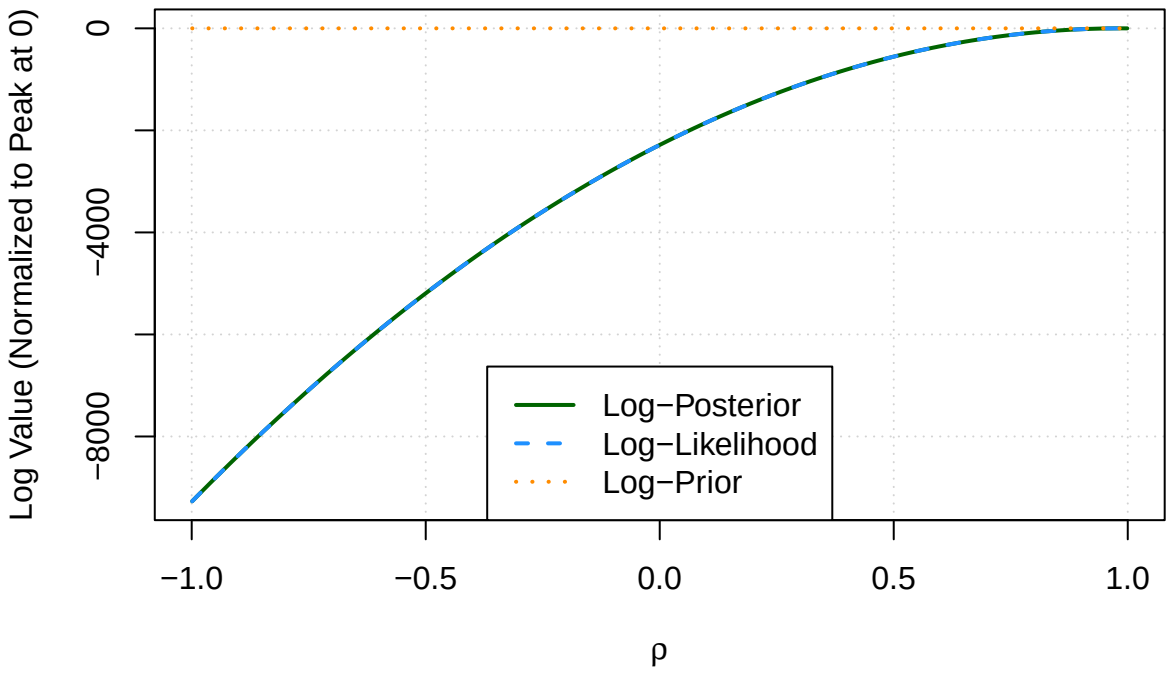
```
  log_posterior = log_lik + log_prior_rho + log_prior_log_sig

  return(log_posterior)
}
```

```
log_post_grid <- outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_logposterior_cases))

contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_post_grid,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  main = "Contour Plot of Log-Posterior for Cases",
  nlevels = 20,
  col = "darkorange"
)

# Add a point for the maximum a posteriori (MAP) estimate for reference
map_index <- which(log_post_grid == max(log_post_grid, na.rm=TRUE), arr.ind = TRUE)
points(rho_grid[map_index[1]], log_sig_grid[map_index[2]], pch = 17, col = "darkgreen")
legend("topright", "MAP", pch = 17, col = "darkgreen", bg = "white")
```



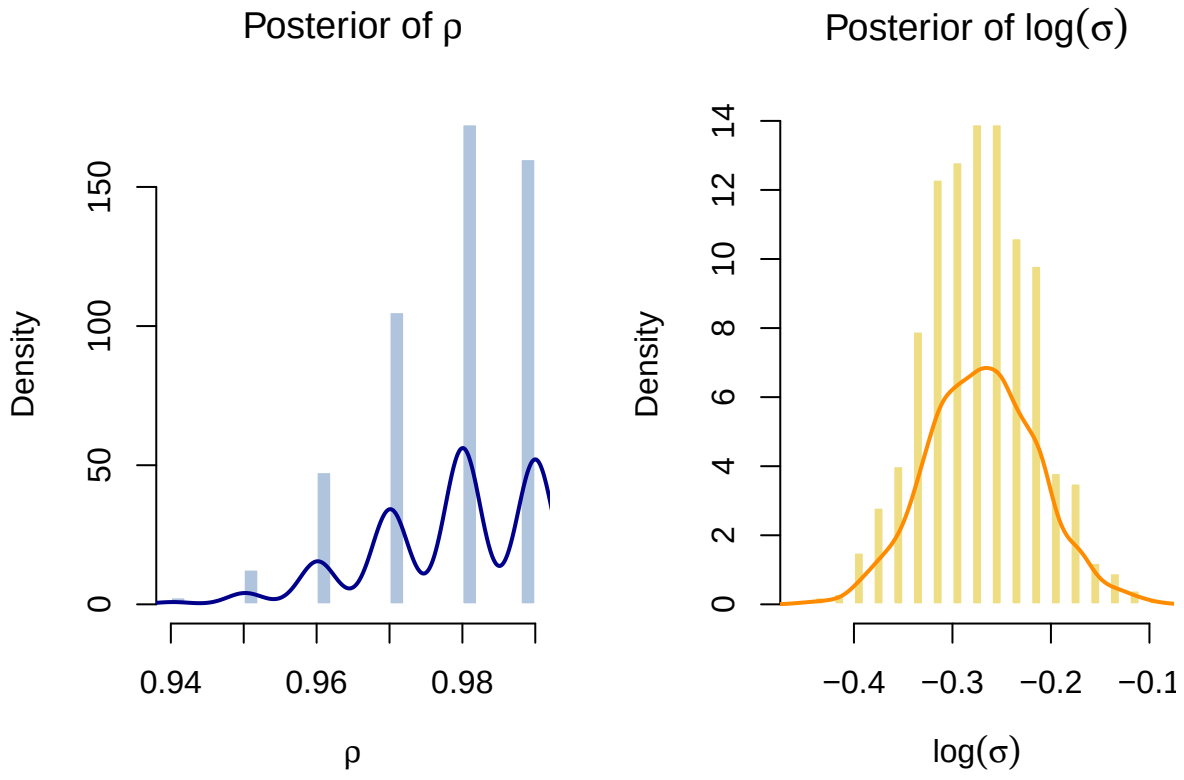

```
fixed_log_sig <- log_sig_grid[map_index[2]]

# Create a finer rho vector for a smoother plot
rho_vec <- seq(-0.999, 0.999, length.out = 500)

# Calculate log-likelihood and log-posterior along this vector
log_lik_vals <- outer(rho_vec, fixed_log_sig, FUN = Vectorize(ar_loglik_cases))
```

```r
# Define and calculate log-prior for the line plot
ar_logprior <- function(rho, log_sig) {
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)
  log_prior_log_sig = dnorm(log_sig, mean = 0, sd = 10, log = TRUE)
  return(log_prior_rho + log_prior_log_sig)
}
log_prior_vals <- ar_logprior(rho_vec, fixed_log_sig)

log_post_vals <- log_lik_vals + log_prior_vals

# Plot 3: Combined line plot
# We normalize the values to peak at 0 for easier comparison of their shapes
plot(
  rho_vec,
  log_post_vals - max(log_post_vals),
  type = 'l',
  col = "darkgreen",
  lwd = 2,
  xlab = expression(rho),
  ylab = "Log Value (Normalized to Peak at 0)",
  main = paste("Comparison at fixed log(sigma) =", round(fixed_log_sig, 2)),
  #ylim = c(-50, 5), # Adjust ylim if the plot is clipped
  panel.first = grid()
)

# Add lines for the log-likelihood and log-prior
lines(rho_vec, log_lik_vals - max(log_lik_vals), col = "dodgerblue", lwd = 2, lty = 2)
lines(rho_vec, log_prior_vals - max(log_prior_vals), col = "darkorange", lwd = 2, lty = 3)

# Add legend
legend(
  "bottom",
  legend = c("Log-Posterior", "Log-Likelihood", "Log-Prior"),
  col = c("darkgreen", "dodgerblue", "darkorange"),
  lwd = 2,
  lty = c(1, 2, 3),
  bg = "white"
)
```




```r
# For numerical stability
max_log_post <- max(log_post_grid, na.rm = TRUE)
post_grid <- exp(log_post_grid - max_log_post)

# Normalize the grid values to create a probability distribution that sums to 1.
prob_grid <- post_grid / sum(post_grid, na.rm = TRUE)

# Draw 1000 samples from posterior
n_points <- length(rho_grid) * length(log_sig_grid)
grid_indices <- 1:n_points

n_samples <- 1000

sampled_indices <- sample(
    grid_indices,
    size = n_samples,
    replace = TRUE,
    prob = as.vector(prob_grid)
)

# Convert to 2D grid
sampled_row_col <- arrayInd(sampled_indices, dim(log_post_grid))

# Extract rho and log(sigma)
sampled_rho <- rho_grid[sampled_row_col[, 1]]
sampled_log_sig <- log_sig_grid[sampled_row_col[, 2]]

# Combine the sampled parameter values into a data frame.
posterior_samples_cases <- data.frame(
    rho = sampled_rho,
```

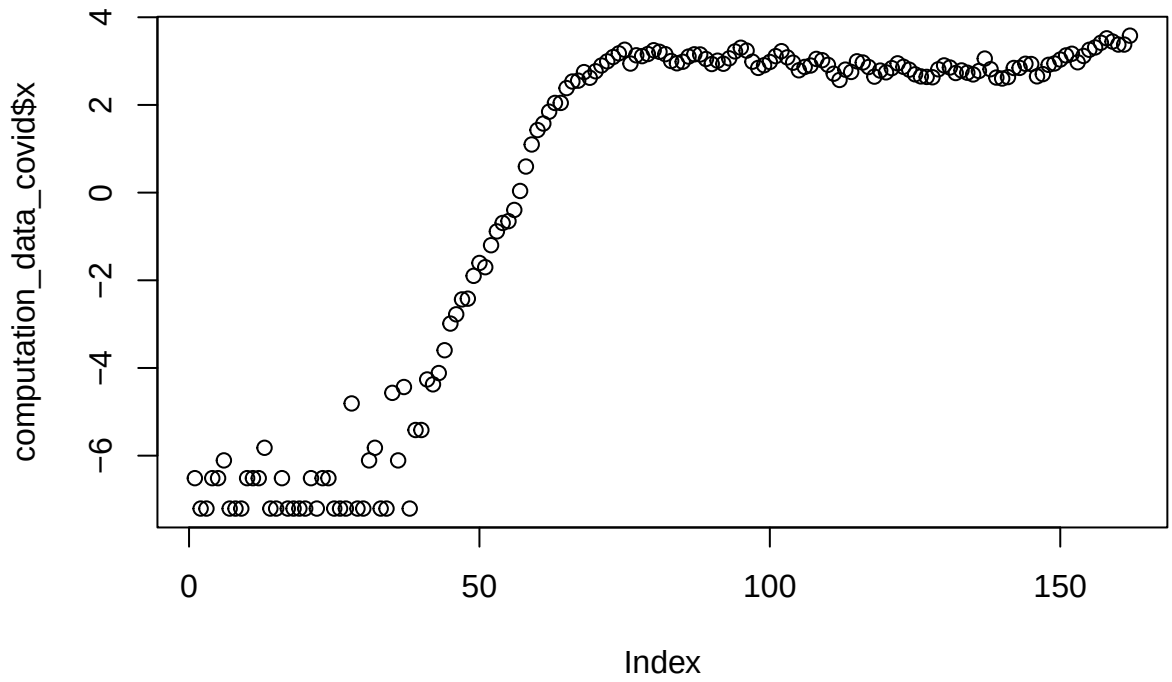
```
    log_sig = sampled_log_sig
)

# Set up a 1x2 plotting area
par(mfrow = c(1, 2), mar = c(5, 4, 4, 2))

# Histogram for rho
hist(posterior_samples_cases$rho,
     main = expression(paste("Posterior of ", rho)),
     xlab = expression(rho),
     freq = FALSE, # Plot density instead of frequency
     col = "lightsteelblue",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples_cases$rho), col = "darkblue", lwd = 2)

# Histogram for log(sigma)
hist(posterior_samples_cases$log_sig,
     main = expression(paste("Posterior of ", log(sigma))),
     xlab = expression(log(sigma)),
     freq = FALSE, # Plot density instead of frequency
     col = "lightgoldenrod",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples_cases$log_sig), col = "darkorange", lwd = 2)
```



```r
# Reset plotting parameters
par(mfrow = c(1, 1))
```

The posterior of ρ is wavy suggesting multiple possible modes of the data. This is not ideal. Let me plot the data which the model is fitting.

```r
plot(computation_data_covid$x)
```



A single ρ cannot capture multiple means. The data even after transformation clearly shows 2 different means.

```r
library(moments)
# Define the quantiles we want to compute
probs <- c(0.025, 0.25, 0.5, 0.75, 0.975)

summary_rho_cases <- c(
  mean = mean(posterior_samples_cases$rho),
  sd = sd(posterior_samples_cases$rho),
  skewness = skewness(posterior_samples_cases$rho),
  kurtosis = kurtosis(posterior_samples_cases$rho),
  quantiles = quantile(posterior_samples_cases$rho, probs = quantile_probs)
)
cat("\n--- Posterior Summary for rho ---\n")
```

```
##
## --- Posterior Summary for rho ---
```

```r
print(round(summary_rho_cases, 4))
```

```
##            mean              sd        skewness        kurtosis  quantiles.2.5%
##          0.9783          0.0108         -0.7897          3.1538          0.9502
##   quantiles.25%   quantiles.50%   quantiles.75% quantiles.97.5%
##          0.9701          0.9801          0.9900          0.9900
```

```r
summary_log_sig_cases <- c(
  mean = mean(posterior_samples_cases$log_sig),
  sd = sd(posterior_samples_cases$log_sig),
  skewness = skewness(posterior_samples_cases$log_sig),
  kurtosis = kurtosis(posterior_samples_cases$log_sig),
  quantiles = quantile(posterior_samples_cases$log_sig, probs = quantile_probs)
)
cat("\n--- Posterior Summary for log(sigma) ---\n")
```

```
##
## --- Posterior Summary for log(sigma) ---
```

```r
print(round(summary_log_sig_cases, 4))
```

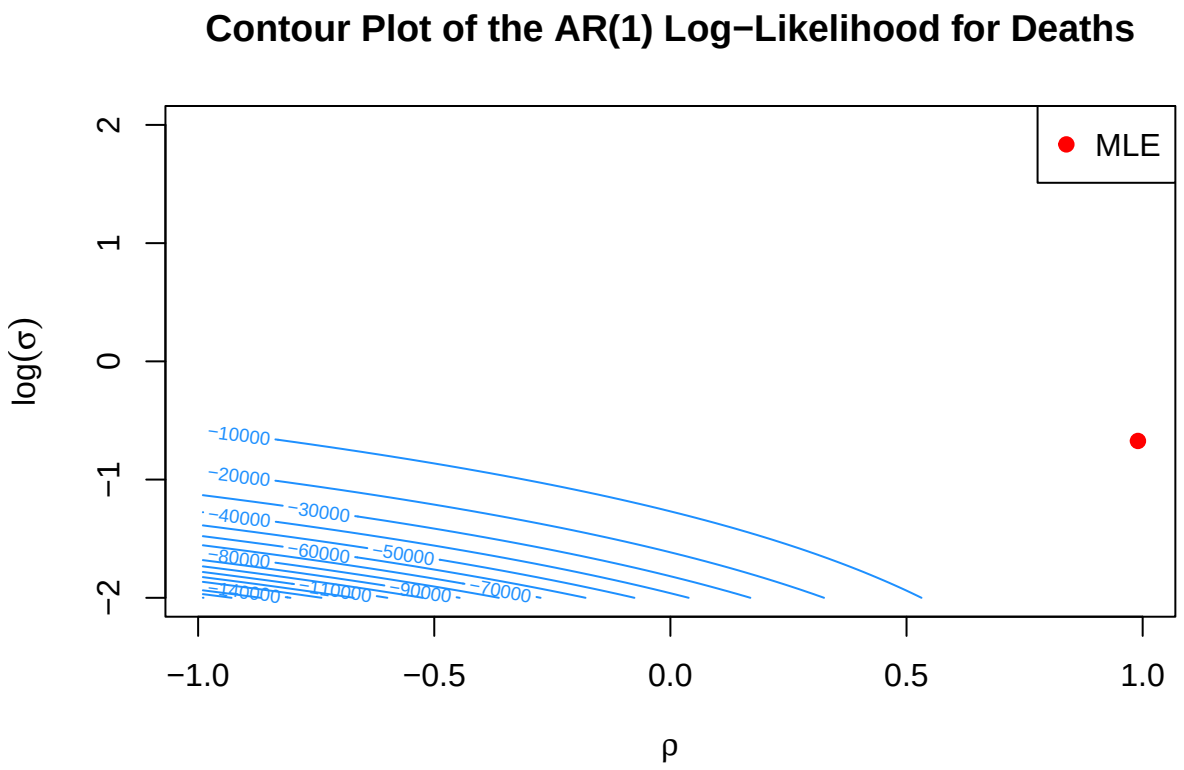
```
##             mean               sd         skewness         kurtosis  quantiles.2.5%
##          -0.2693           0.0558           0.0579           3.0087          -0.3719
##   quantiles.25%   quantiles.50%   quantiles.75% quantiles.97.5%
##          -0.3116         -0.2714         -0.2312         -0.1508
```

```r
daily_deaths <- diff(training_data$deaths)
daily_deaths <- c(training_data$deaths[1], daily_deaths)
log_daily_deaths <- log(daily_deaths + 1)
mean_log_daily_deaths <- mean(log_daily_deaths)
processed_data_deaths <- (log_daily_deaths - mean_log_daily_deaths)
computation_data_deaths <- data.frame(x = processed_data_deaths)
```

```r
ar_loglik_deaths <- function(rho, log_sig)
{
  n = length(computation_data_deaths$x) # Number of rows/samples

  # sigma = exp(log_sig)
  # sigma^2 = (exp(log_sig))^2 = exp(2 * log_sig)
  sigma_sq = exp(2 * log_sig)

  y_current = computation_data_deaths$x
  y_prev <- c(0, y_current[-n])
  sum_sq_err = sum((y_current - rho*y_prev)^2)

  #log_lik = -(n/2)*log(2*pi) - (n/2)*log(sigma_sq) - (1/(2*sigma_sq)) * sum_sq_err
  log_lik <- -(n-1)/2 * log(2*pi) -
             (n-1)/2 * log(sigma_sq) -
             (1/(2*sigma_sq)) * sum_sq_err

  return(log_lik)
}
```

```r
log_lik_surface = outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_loglik_deaths))

contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_lik_surface,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
```

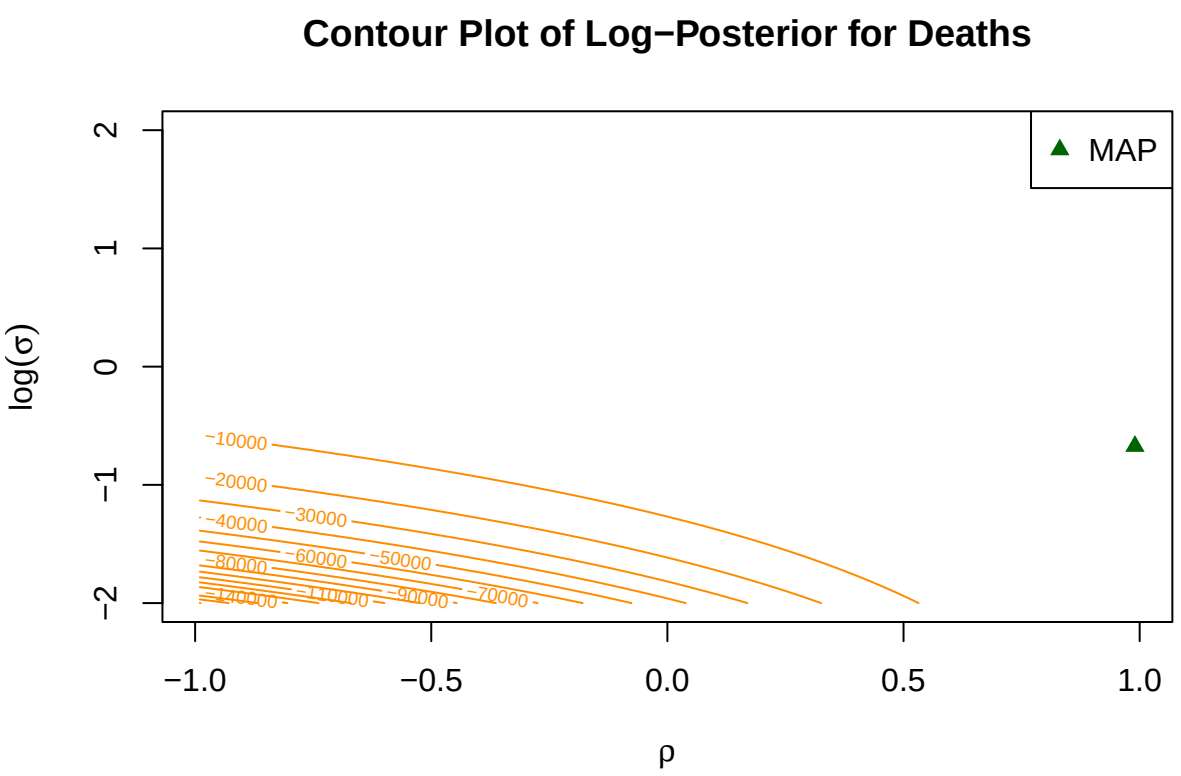
```
  main = "Contour Plot of the AR(1) Log-Likelihood for Deaths",
  nlevels = 20,
  col = "dodgerblue"
)

# Add a point for the maximum likelihood estimate for reference
max_lik_index <- which(log_lik_surface == max(log_lik_surface), arr.ind = TRUE)
points(rho_grid[max_lik_index[1]], log_sig_grid[max_lik_index[2]], pch = 19, col = "red")
legend("topright", "MLE", pch = 19, col = "red", bg = "white")
```




```
ar_logposterior_deaths <- function(rho, log_sig) {

  log_lik = ar_loglik_deaths(rho, log_sig)

  # Log prior for rho ~ Uniform(-1, 1)
  # The density is 1/(1 - (-1)) = 1/2 for -1 < rho < 1
  # The log density is log(1/2) = -log(2) for -1 < rho < 1
  # and -Inf otherwise
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)

  # Log prior for log_sig ~ Normal(0, 10^2)
  log_prior_log_sig = -0.5 * log(2 * pi * 1) - (log_sig - 0)^2 / (2 * 1)

  log_posterior = log_lik + log_prior_rho + log_prior_log_sig

  return(log_posterior)
}
```

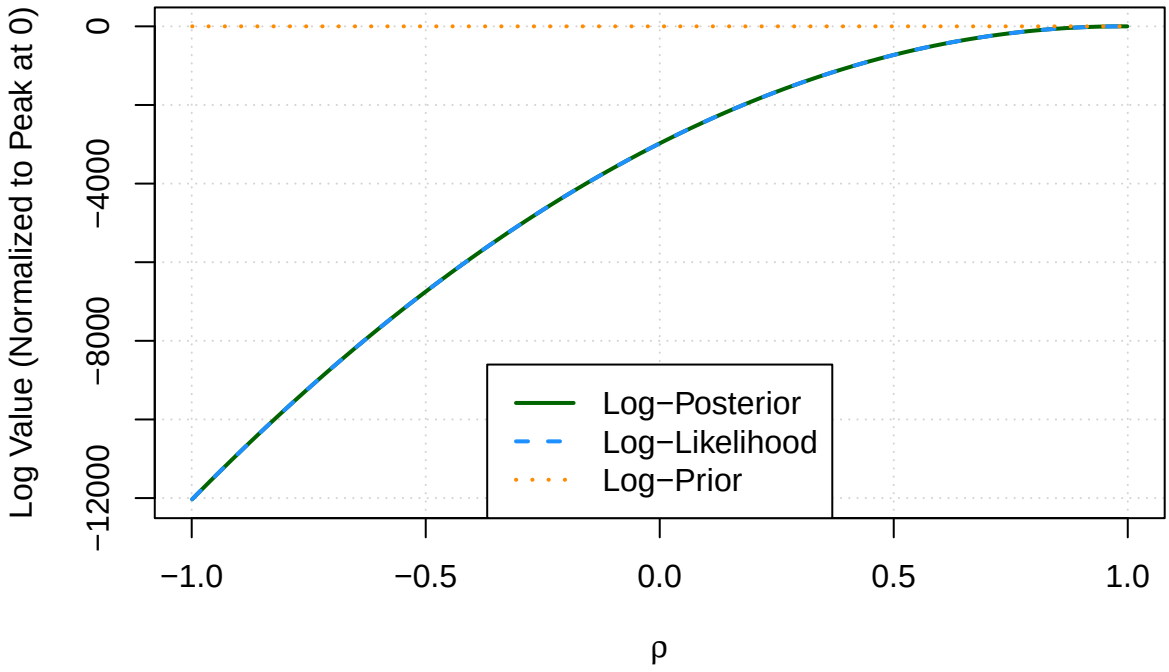
```
log_post_grid <- outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_logposterior_deaths))
```

```r
contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_post_grid,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  main = "Contour Plot of Log-Posterior for Deaths",
  nlevels = 20,
  col = "darkorange"
)

# Add a point for the maximum a posteriori (MAP) estimate for reference
map_index <- which(log_post_grid == max(log_post_grid, na.rm=TRUE), arr.ind = TRUE)
points(rho_grid[map_index[1]], log_sig_grid[map_index[2]], pch = 17, col = "darkgreen")
legend("topright", "MAP", pch = 17, col = "darkgreen", bg = "white")
```



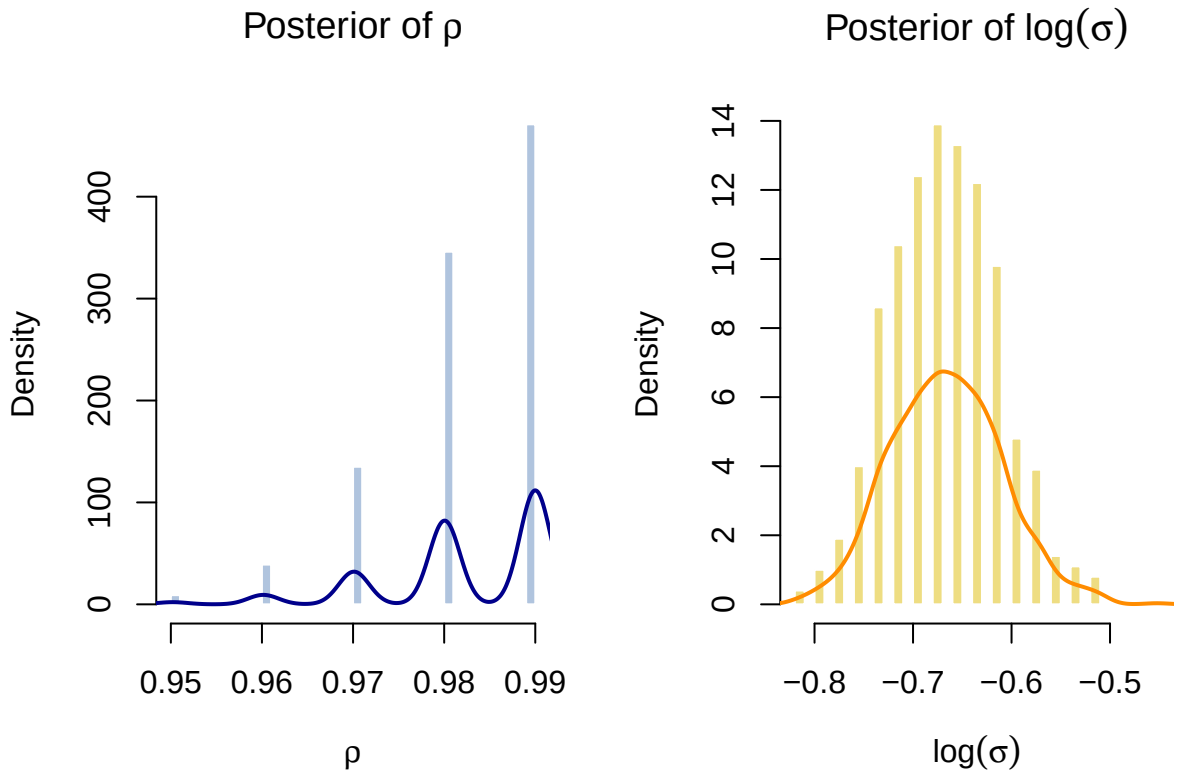

```r
fixed_log_sig <- log_sig_grid[map_index[2]]

# Create a finer rho vector for a smoother plot
rho_vec <- seq(-0.999, 0.999, length.out = 500)

# Calculate log-likelihood and log-posterior along this vector
log_lik_vals <- outer(rho_vec, fixed_log_sig, FUN = Vectorize(ar_loglik_deaths))

# Define and calculate log-prior for the line plot
ar_logprior <- function(rho, log_sig) {
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)
  log_prior_log_sig = dnorm(log_sig, mean = 0, sd = 10, log = TRUE)
  return(log_prior_rho + log_prior_log_sig)
```

```
}
log_prior_vals <- ar_logprior(rho_vec, fixed_log_sig)

log_post_vals <- log_lik_vals + log_prior_vals

# Plot 3: Combined line plot
# We normalize the values to peak at 0 for easier comparison of their shapes
plot(
  rho_vec,
  log_post_vals - max(log_post_vals),
  type = 'l',
  col = "darkgreen",
  lwd = 2,
  xlab = expression(rho),
  ylab = "Log Value (Normalized to Peak at 0)",
  main = paste("Comparison at fixed log(sigma) =", round(fixed_log_sig, 2)),
  #ylim = c(-50, 5), # Adjust ylim if the plot is clipped
  panel.first = grid()
)

# Add lines for the log-likelihood and log-prior
lines(rho_vec, log_lik_vals - max(log_lik_vals), col = "dodgerblue", lwd = 2, lty = 2)
lines(rho_vec, log_prior_vals - max(log_prior_vals), col = "darkorange", lwd = 2, lty = 3)

# Add legend
legend(
  "bottom",
  legend = c("Log-Posterior", "Log-Likelihood", "Log-Prior"),
  col = c("darkgreen", "dodgerblue", "darkorange"),
  lwd = 2,
  lty = c(1, 2, 3),
  bg = "white"
)
```




```r
# For numerical stability
max_log_post <- max(log_post_grid, na.rm = TRUE)
post_grid <- exp(log_post_grid - max_log_post)

# Normalize the grid values to create a probability distribution that sums to 1.
prob_grid <- post_grid / sum(post_grid, na.rm = TRUE)

# Draw 1000 samples from posterior
n_points <- length(rho_grid) * length(log_sig_grid)
grid_indices <- 1:n_points

n_samples <- 1000

sampled_indices <- sample(
    grid_indices,
    size = n_samples,
    replace = TRUE,
    prob = as.vector(prob_grid)
)

# Convert to 2D grid
sampled_row_col <- arrayInd(sampled_indices, dim(log_post_grid))

# Extract rho and log(sigma)
sampled_rho <- rho_grid[sampled_row_col[, 1]]
sampled_log_sig <- log_sig_grid[sampled_row_col[, 2]]

# Combine the sampled parameter values into a data frame.
posterior_samples_deaths <- data.frame(
    rho = sampled_rho,
```

```r
    log_sig = sampled_log_sig
)

# Set up a 1x2 plotting area
par(mfrow = c(1, 2), mar = c(5, 4, 4, 2))

# Histogram for rho
hist(posterior_samples_deaths$rho,
     main = expression(paste("Posterior of ", rho)),
     xlab = expression(rho),
     freq = FALSE, # Plot density instead of frequency
     col = "lightsteelblue",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples_deaths$rho), col = "darkblue", lwd = 2)

# Histogram for log(sigma)
hist(posterior_samples_deaths$log_sig,
     main = expression(paste("Posterior of ", log(sigma))),
     xlab = expression(log(sigma)),
     freq = FALSE, # Plot density instead of frequency
     col = "lightgoldenrod",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples_deaths$log_sig), col = "darkorange", lwd = 2)
```



```r
# Reset plotting parameters
par(mfrow = c(1, 1))
```

```r
summary_rho_deaths <- c(
  mean = mean(posterior_samples_deaths$rho),
  sd = sd(posterior_samples_deaths$rho),
  skewness = skewness(posterior_samples_deaths$rho),
  kurtosis = kurtosis(posterior_samples_deaths$rho),
  quantiles = quantile(posterior_samples_deaths$rho, probs = quantile_probs)
)
cat("\n--- Posterior Summary for rho ---\n")
```

```
##
## --- Posterior Summary for rho ---
```

```r
print(round(summary_rho_deaths, 4))
```

```
##            mean              sd        skewness        kurtosis  quantiles.2.5%
##          0.9823          0.0088         -1.1095          3.8855          0.9602
##   quantiles.25%   quantiles.50%   quantiles.75% quantiles.97.5%
##          0.9801          0.9801          0.9900          0.9900
```

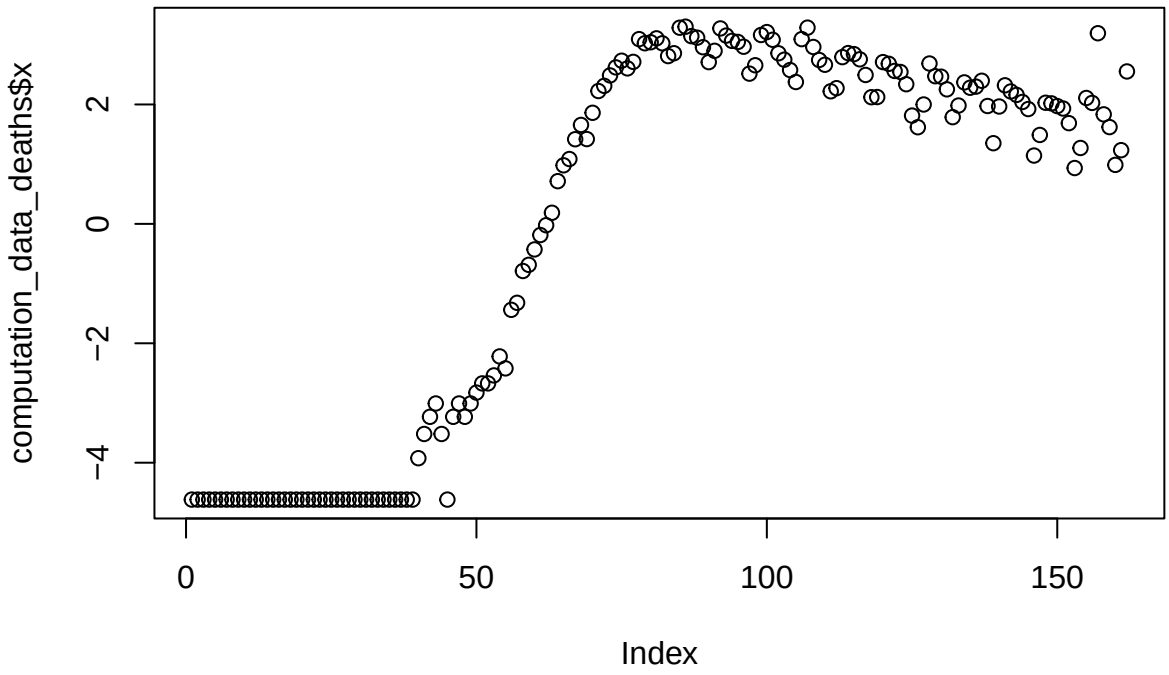
```r
summary_log_sig_deaths <- c(
  mean = mean(posterior_samples_deaths$log_sig),
  sd = sd(posterior_samples_deaths$log_sig),
  skewness = skewness(posterior_samples_deaths$log_sig),
  kurtosis = kurtosis(posterior_samples_deaths$log_sig),
  quantiles = quantile(posterior_samples_deaths$log_sig, probs = quantile_probs)
)
cat("\n--- Posterior Summary for log(sigma) ---\n")
```

```
##
## --- Posterior Summary for log(sigma) ---
```

```r
print(round(summary_log_sig_deaths, 4))
```

```
##            mean              sd        skewness        kurtosis  quantiles.2.5%
##         -0.6661          0.0561          0.1712          2.9867         -0.7739
##   quantiles.25%   quantiles.50%   quantiles.75% quantiles.97.5%
##         -0.7136         -0.6734         -0.6332         -0.5528
```

```r
plot(computation_data_deaths$x)
```



We see the same issue with the posterior values for the deaths dataset - multiple modes for ρ.

5. Conduct a posterior predictive check to diagnose the fit of your model for the data until June 30. Does your model provide good predictions? Why or why not? If not, describe how you would consider modifying it to address the observed inadequacies (you don't have to actually do this).

```
simulate_ar_process <- function(rho, log_sig, n) {

  sigma <- exp(log_sig)
  y_rep <- numeric(n)

  # The posterior for rho is within (-1, 1), ensuring stationarity.
  # We start the process by drawing from the stationary distribution.
  y_rep[1] <- rnorm(1, mean = 0, sd = sigma / sqrt(1 - rho^2))
  #y_rep[1] <- 0 # Fixed constant

  # Simulate the rest of the process
  for (i in 2:n) {
    y_rep[i] <- rho * y_rep[i-1] + rnorm(1, mean = 0, sd = sigma)
  }

  return(y_rep)
}
```

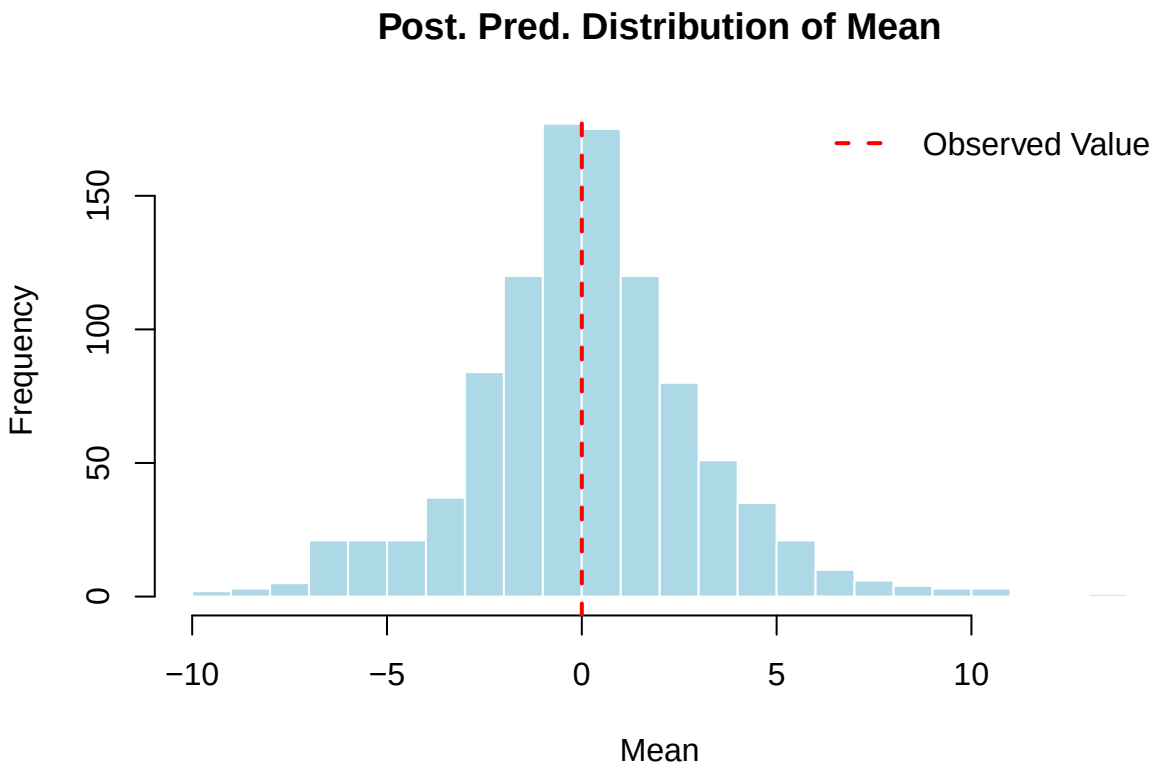
```
n_reps <- 1000
n_data <- nrow(computation_data_covid)
posterior_predictive_samples_cases <- vector("list", n_reps)

for (i in 1:n_reps) {
  current_rho <- posterior_samples_cases$rho[i]
  current_log_sig <- posterior_samples_cases$log_sig[i]
```

```
  # Simulate a new dataset using these parameters
  posterior_predictive_samples_cases[[i]] <- simulate_ar_process(
    rho = current_rho,
    log_sig = current_log_sig,
    n = n_data
  )
}
```

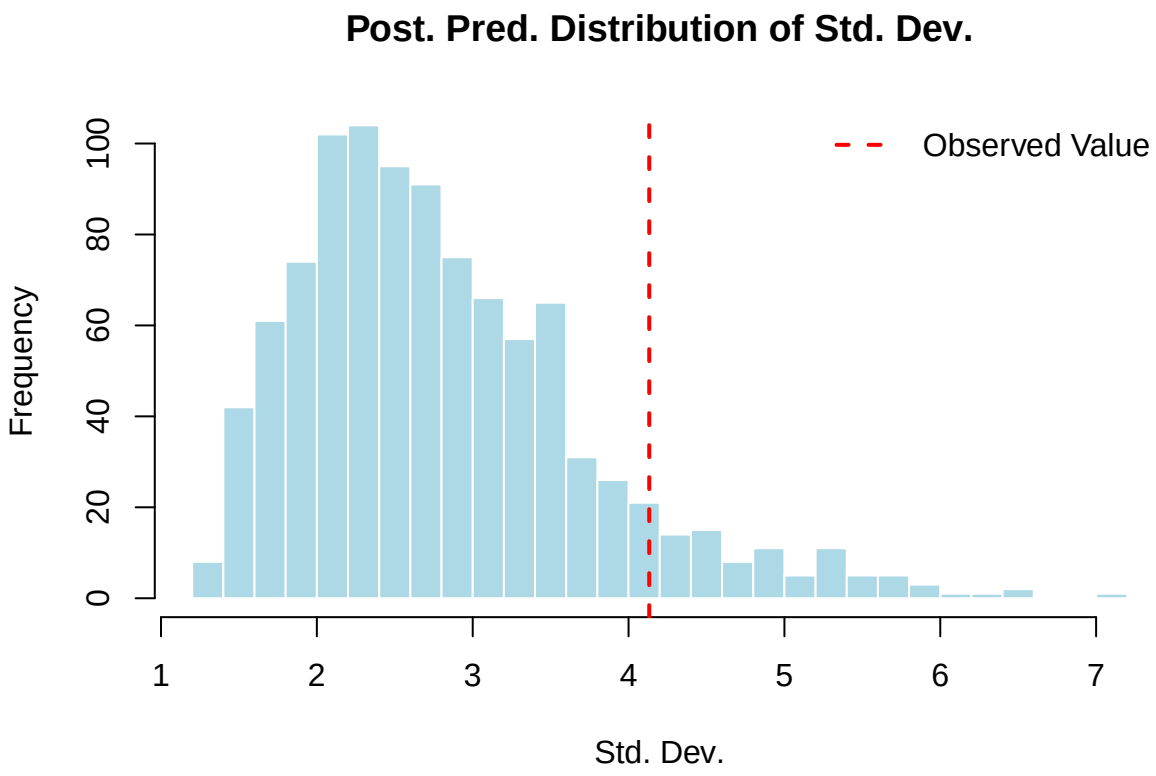
```
rep_means <- sapply(posterior_predictive_samples_cases, mean)
rep_sds <- sapply(posterior_predictive_samples_cases, sd)
rep_skewness <- sapply(posterior_predictive_samples_cases, skewness)
rep_kurtosis <- sapply(posterior_predictive_samples_cases, kurtosis)
```

```
obs_data <- computation_data_covid$x
obs_stats <- c(
    mean(obs_data),
    sd(obs_data),
    skewness(obs_data),
    kurtosis(obs_data)
)
```

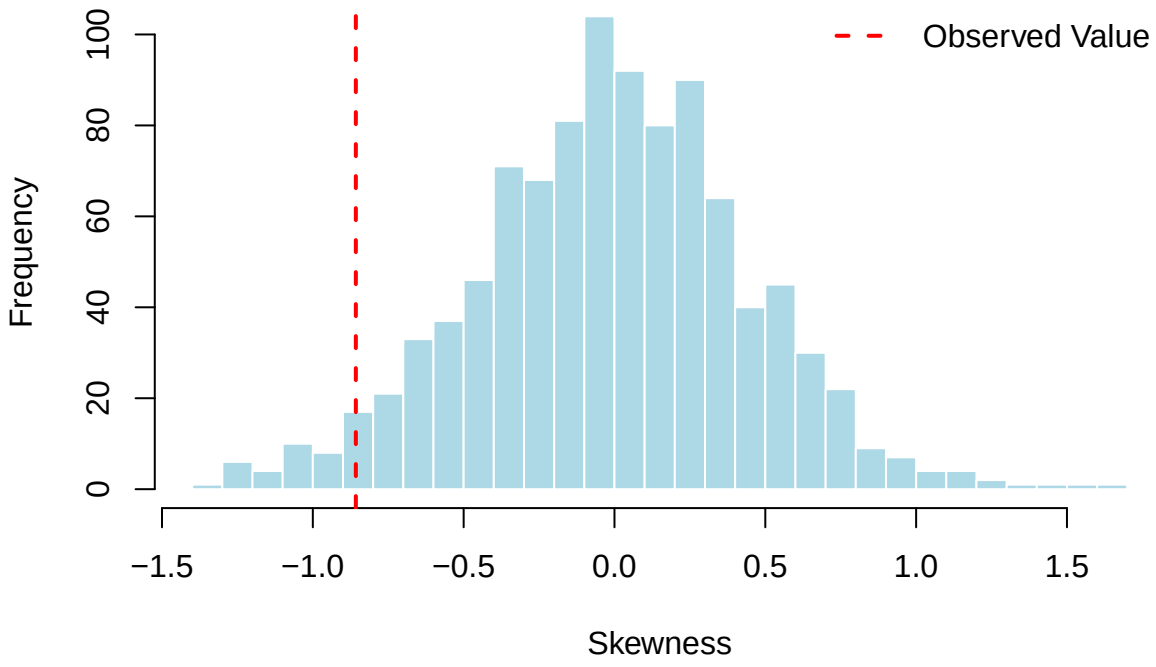
```
# Histogram for the Mean
hist(rep_means, main = "Post. Pred. Distribution of Mean", xlab = "Mean", col="lightblue", border="white
abline(v = obs_stats[1], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```



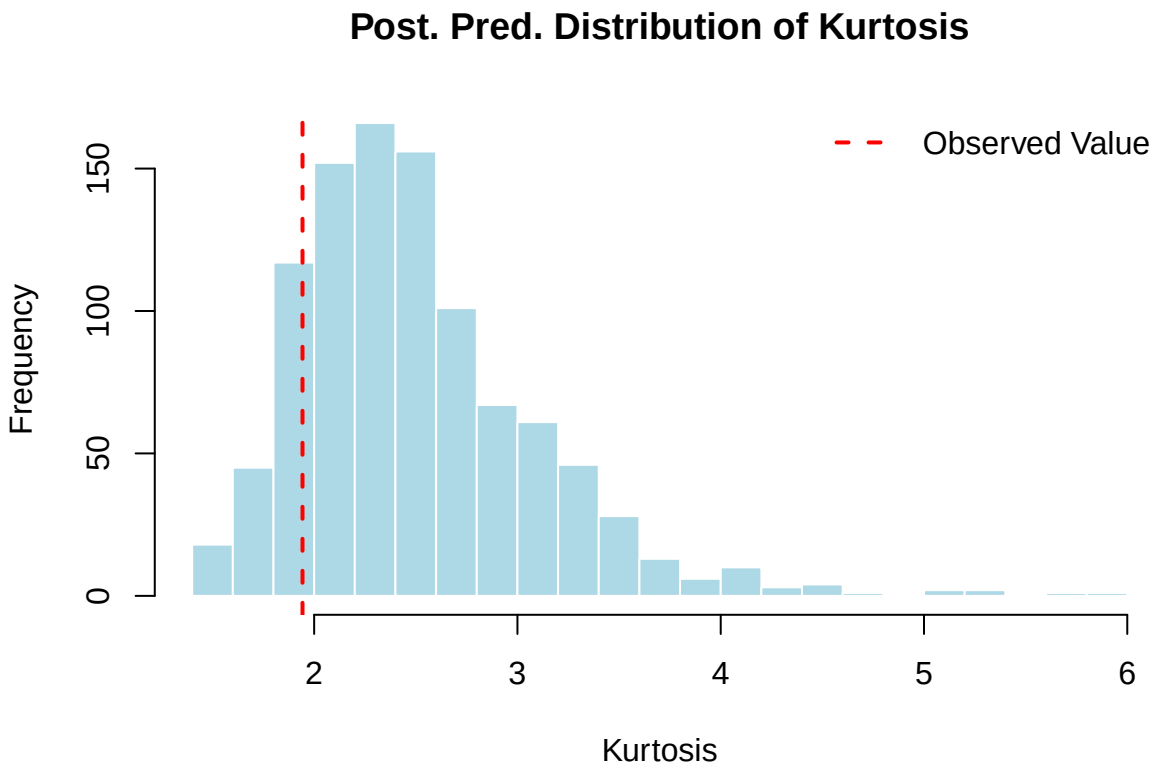

```
# Histogram for the Standard Deviation
hist(rep_sds, main = "Post. Pred. Distribution of Std. Dev.", xlab = "Std. Dev.", col="lightblue", borde
abline(v = obs_stats[2], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```




```r
# Histogram for Skewness
hist(rep_skewness, main = "Post. Pred. Distribution of Skewness", xlab = "Skewness", col="lightblue", b
abline(v = obs_stats[3], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```



```r
# Histogram for Kurtosis
hist(rep_kurtosis, main = "Post. Pred. Distribution of Kurtosis", xlab = "Kurtosis", col="lightblue", b
abline(v = obs_stats[4], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```

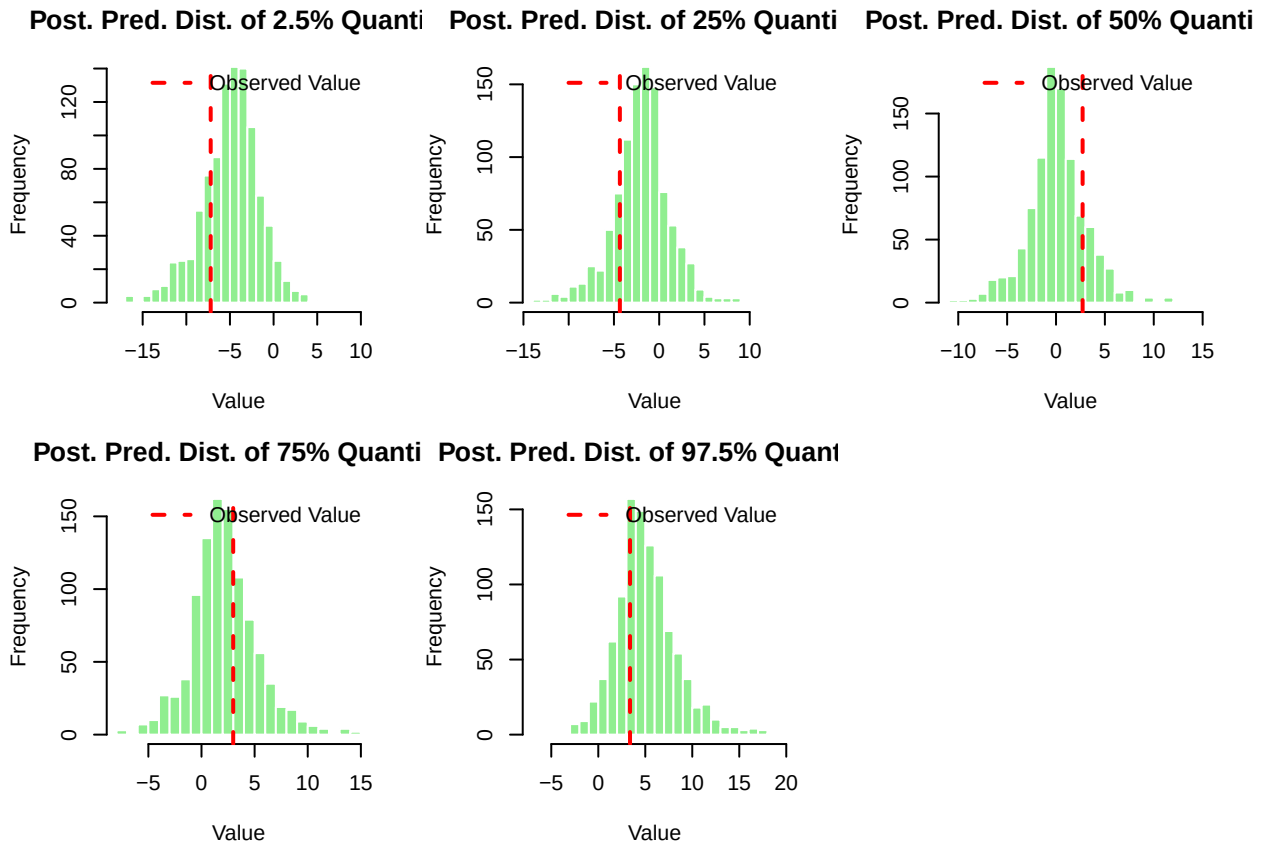



```r
quantile_probs <- c(0.025, 0.25, 0.5, 0.75, 0.975)

rep_quantiles <- t(sapply(posterior_predictive_samples_cases, quantile, probs = quantile_probs))
colnames(rep_quantiles) <- paste0(quantile_probs * 100, "%")

# Calculate the same quantiles for the observed data
obs_quantiles <- quantile(obs_data, probs = quantile_probs)

# Print a summary table
quantile_summary_df <- data.frame(
  Quantile = colnames(rep_quantiles),
  Observed_Value = obs_quantiles,
  Posterior_Pred_Mean = colMeans(rep_quantiles),
  Posterior_Pred_95_CI_Lower = apply(rep_quantiles, 2, quantile, 0.025),
  Posterior_Pred_95_CI_Upper = apply(rep_quantiles, 2, quantile, 0.975)
)
print(quantile_summary_df, row.names=FALSE)
```

```
##  Quantile Observed_Value Posterior_Pred_Mean Posterior_Pred_95_CI_Lower
##      2.5%      -7.206686          -4.9235871                -12.1136109
##       25%      -4.345666          -1.9489794                 -8.9900044
##       50%       2.720317           0.1258429                 -6.4192564
##       75%       2.984147           2.1654761                 -3.7779342
##     97.5%       3.377540           5.0917519                 -0.7764605
##  Posterior_Pred_95_CI_Upper
```

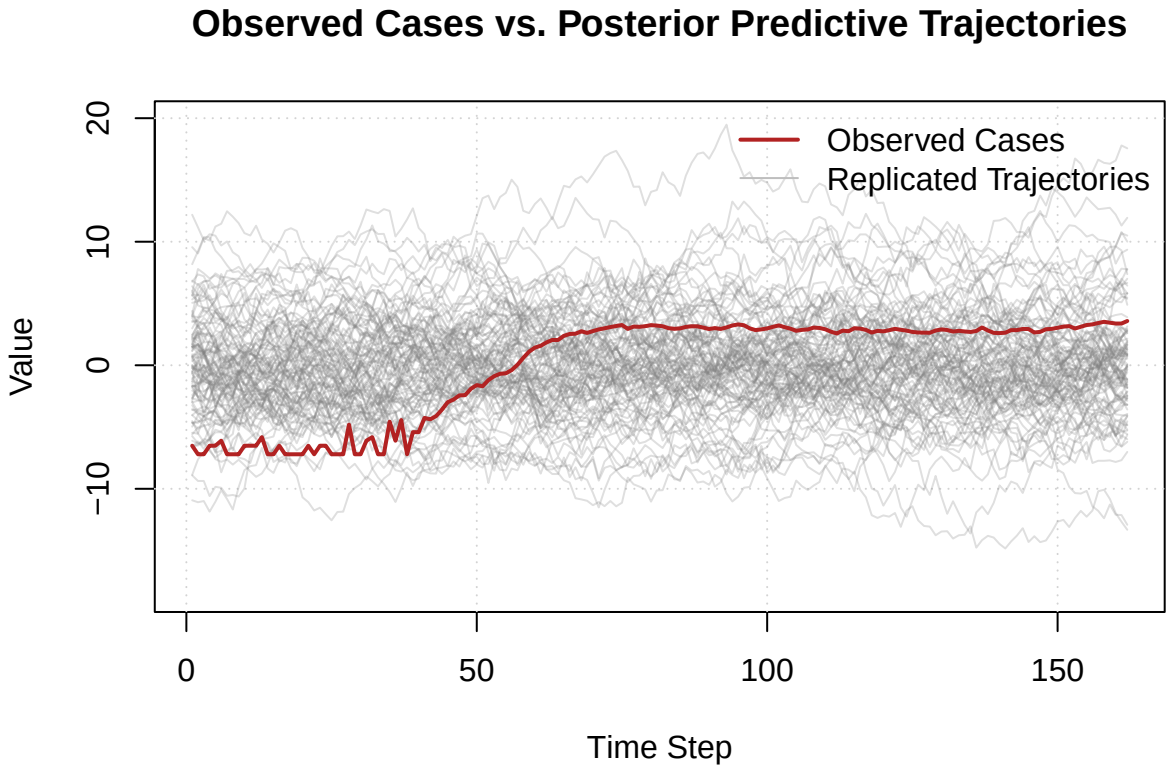
```
##                    1.108806
##                    3.862248
##                    6.436621
##                    9.166842
##                   12.495105
```

```r
par(mfrow = c(2, 3), mar = c(4, 4, 3, 2))

for (i in 1:length(quantile_probs)) {
  hist(rep_quantiles[, i],
       main = paste("Post. Pred. Dist. of", colnames(rep_quantiles)[i], "Quantile"),
       xlab = "Value",
       col = "lightgreen",
       border = "white",
       breaks = 30)
  abline(v = obs_quantiles[i], col = "red", lwd = 2, lty = 2)
  legend("topright", "Observed Value", col = "red", lty = 2, lwd = 2, bty = "n")
}

# Reset plotting parameters
par(mfrow = c(1, 1))
```




```r
y_range <- range(c(obs_data, unlist(posterior_predictive_samples_cases)))

# Plot the first 100 replicated trajectories as semi-transparent lines
plot(NULL,
     xlim = c(1, n_data),
     ylim = y_range,
```

```
     main = "Observed Cases vs. Posterior Predictive Trajectories",
     xlab = "Time Step",
     ylab = "Value")
grid()

# Add replicated data lines
for (i in 1:100) {
  lines(posterior_predictive_samples_cases[[i]], col = rgb(0.5, 0.5, 0.5, 0.25))
}

# Superimpose the observed data as a bold, colored line
lines(obs_data, col = "firebrick", lwd = 2)

# Add a legend
legend("topright",
       legend = c("Observed Cases", "Replicated Trajectories"),
       col = c("firebrick", "gray"),
       lwd = c(2, 1),
       bty = "n")
```



The model does not provide good predictions for the number cases. The chosen AR(1) model is too simple to model the abrupt change in the number of cases. We need more flexible ρ to model the gradual increase in the number of cases.

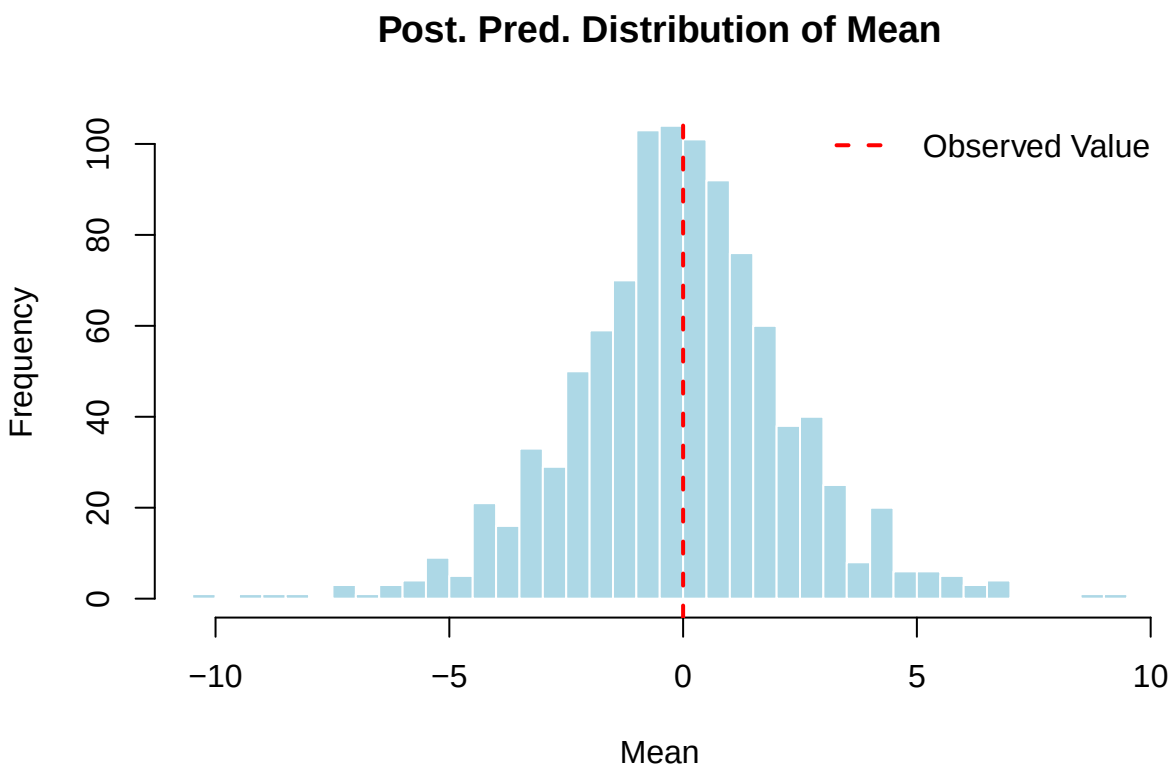
```
n_reps <- 1000
n_data <- nrow(computation_data_deaths)
posterior_predictive_samples_deaths <- vector("list", n_reps)
```

```r
for (i in 1:n_reps) {
  current_rho <- posterior_samples_deaths$rho[i]
  current_log_sig <- posterior_samples_deaths$log_sig[i]

  # Simulate a new dataset using these parameters
  posterior_predictive_samples_deaths[[i]] <- simulate_ar_process(
    rho = current_rho,
    log_sig = current_log_sig,
    n = n_data
  )
}
```

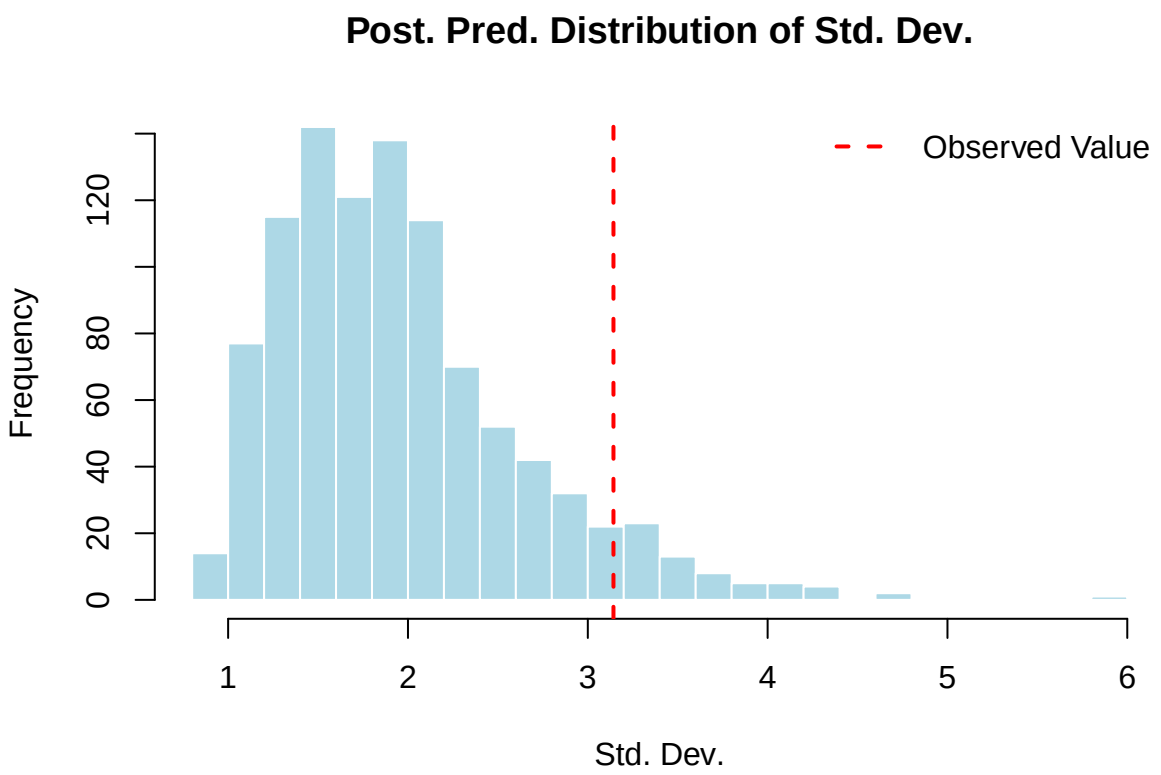
```r
rep_means <- sapply(posterior_predictive_samples_deaths, mean)
rep_sds <- sapply(posterior_predictive_samples_deaths, sd)
rep_skewness <- sapply(posterior_predictive_samples_deaths, skewness)
rep_kurtosis <- sapply(posterior_predictive_samples_deaths, kurtosis)
```

```r
obs_data <- computation_data_deaths$x
obs_stats <- c(
    mean(obs_data),
    sd(obs_data),
    skewness(obs_data),
    kurtosis(obs_data)
)
```

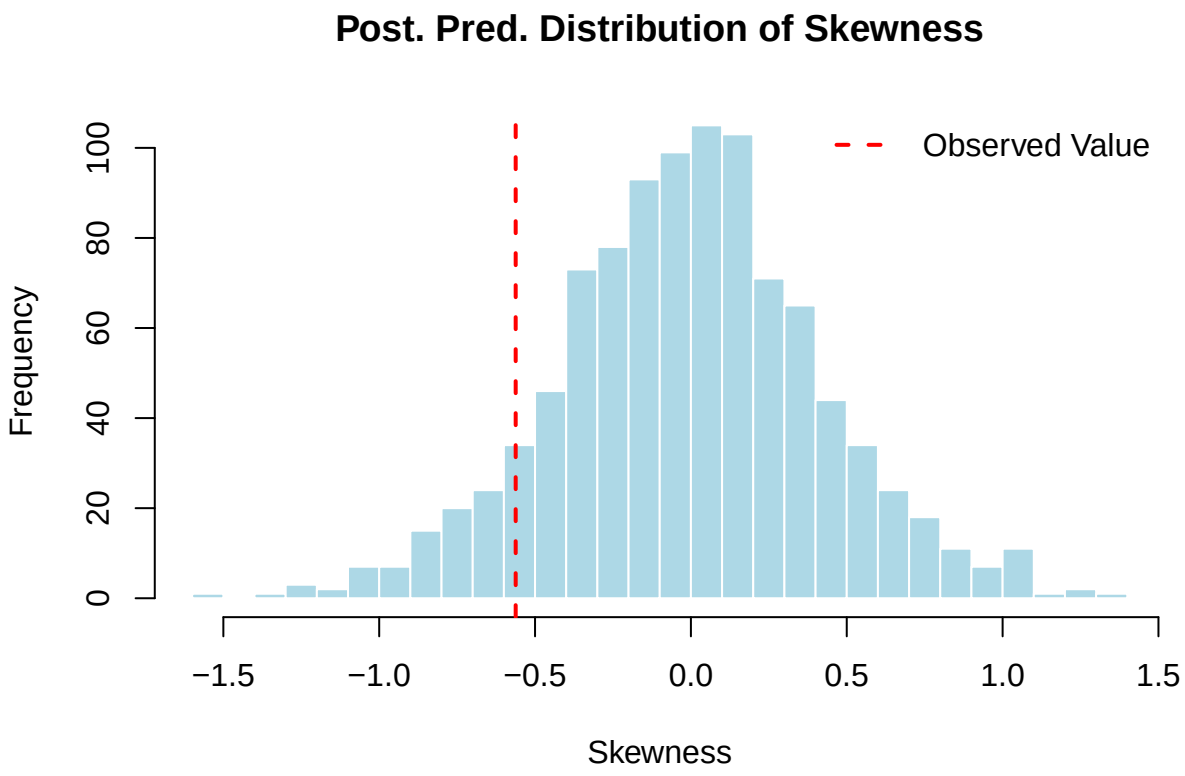
```r
# Histogram for the Mean
hist(rep_means, main = "Post. Pred. Distribution of Mean", xlab = "Mean", col="lightblue", border="white
abline(v = obs_stats[1], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```



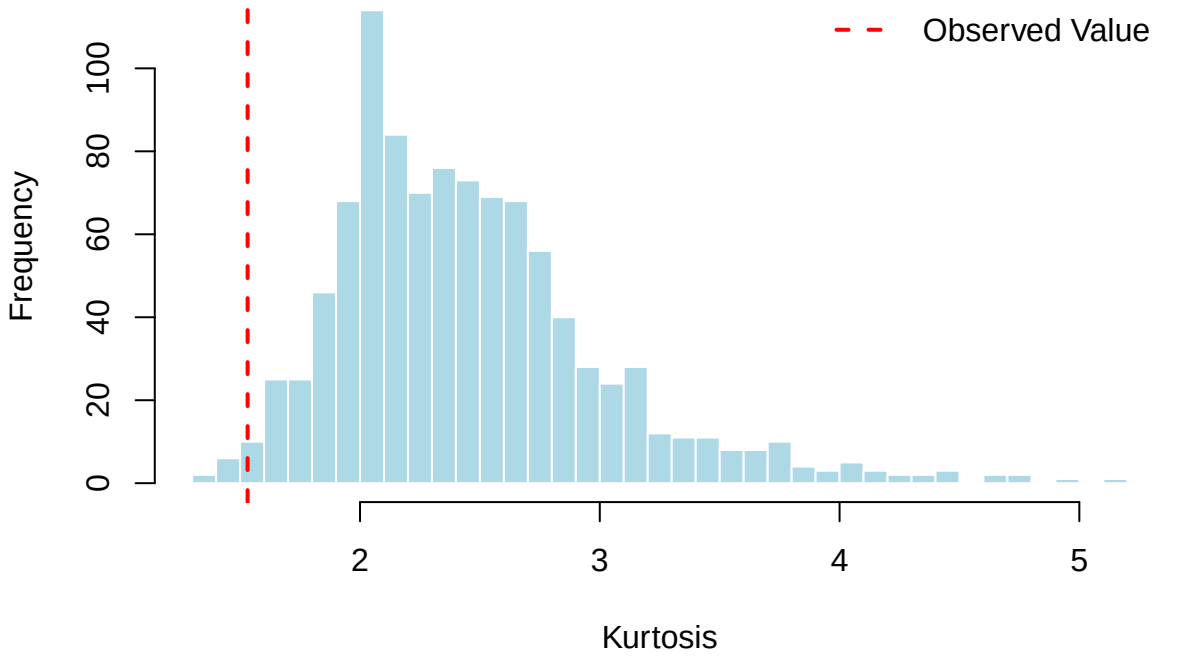
```r
# Histogram for the Standard Deviation
hist(rep_sds, main = "Post. Pred. Distribution of Std. Dev.", xlab = "Std. Dev.", col="lightblue", borde
abline(v = obs_stats[2], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```



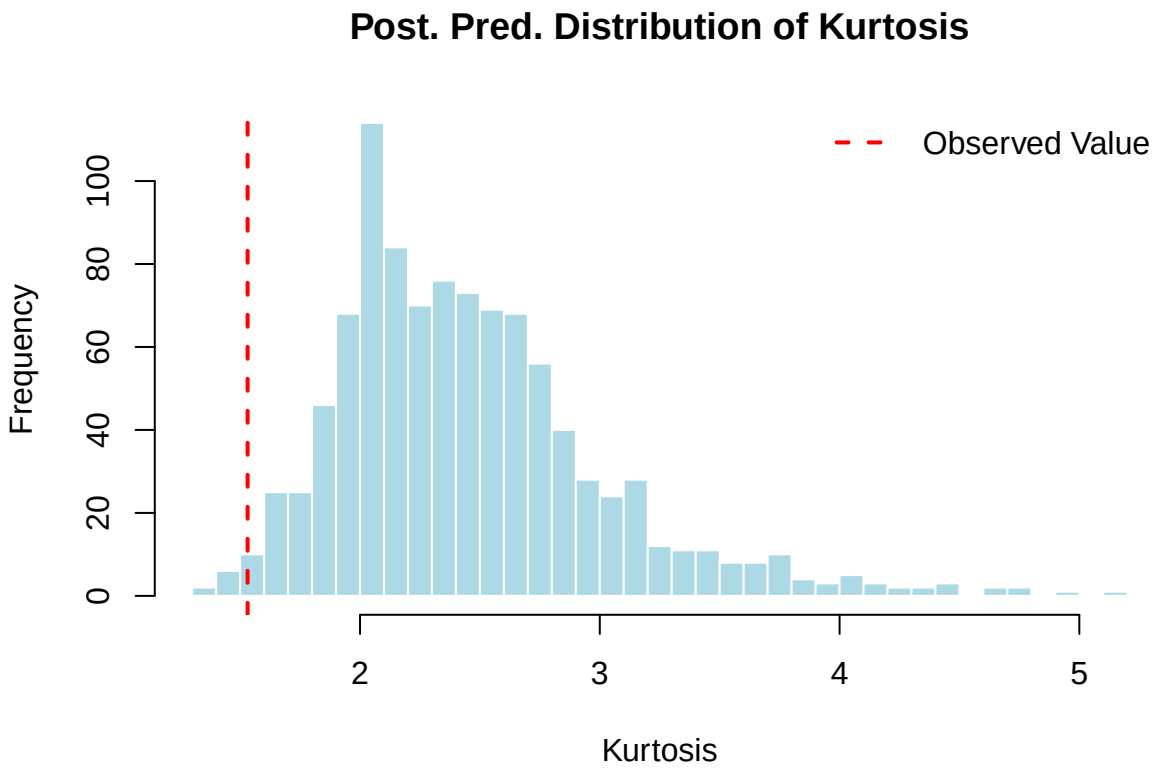

```r
# Histogram for Skewness
hist(rep_skewness, main = "Post. Pred. Distribution of Skewness", xlab = "Skewness", col="lightblue", bo
abline(v = obs_stats[3], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```




```r
# Histogram for Kurtosis
hist(rep_kurtosis, main = "Post. Pred. Distribution of Kurtosis", xlab = "Kurtosis", col="lightblue", b
abline(v = obs_stats[4], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```



```r
# Histogram for Kurtosis
hist(rep_kurtosis, main = "Post. Pred. Distribution of Kurtosis", xlab = "Kurtosis", col="lightblue", b
abline(v = obs_stats[4], col = "red", lwd = 2, lty=2)
legend("topright", "Observed Value", col="red", lty=2, lwd=2, bty="n")
```

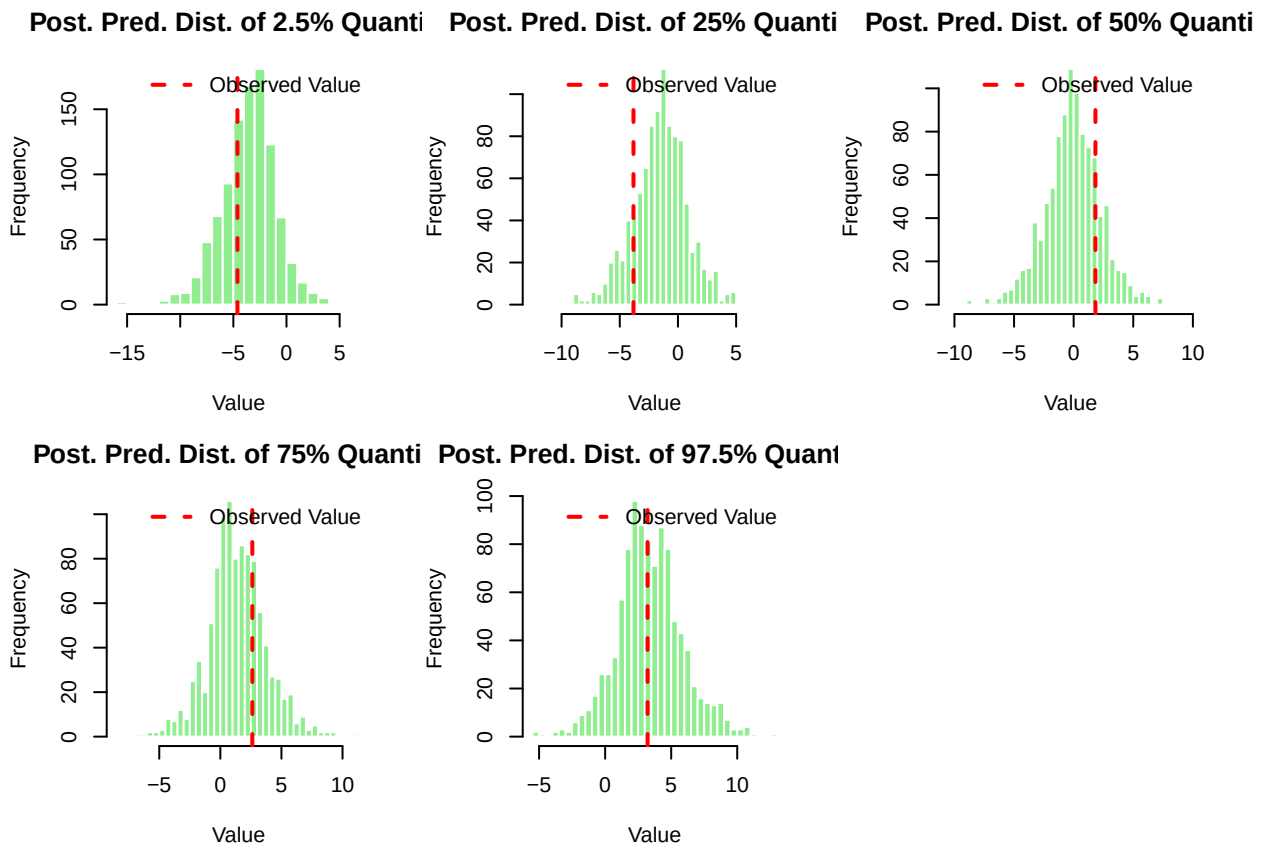



```r
quantile_probs <- c(0.025, 0.25, 0.5, 0.75, 0.975)

rep_quantiles <- t(sapply(posterior_predictive_samples_deaths, quantile, probs = quantile_probs))
colnames(rep_quantiles) <- paste0(quantile_probs * 100, "%")

# Calculate the same quantiles for the observed data
obs_quantiles <- quantile(obs_data, probs = quantile_probs)

# Print a summary table
quantile_summary_df <- data.frame(
  Quantile = colnames(rep_quantiles),
  Observed_Value = obs_quantiles,
  Posterior_Pred_Mean = colMeans(rep_quantiles),
  Posterior_Pred_95_CI_Lower = apply(rep_quantiles, 2, quantile, 0.025),
  Posterior_Pred_95_CI_Upper = apply(rep_quantiles, 2, quantile, 0.975)
)
print(quantile_summary_df, row.names=FALSE)
```

```
##  Quantile Observed_Value Posterior_Pred_Mean Posterior_Pred_95_CI_Lower
##      2.5%      -4.617547         -3.56049869                  -8.820325
##       25%      -3.823034         -1.52321490                  -6.481415
##       50%       1.824143         -0.07328685                  -4.936437
##       75%       2.616784          1.36681152                  -3.529079
##     97.5%       3.211999          3.38393046                  -1.449429
##  Posterior_Pred_95_CI_Upper
```

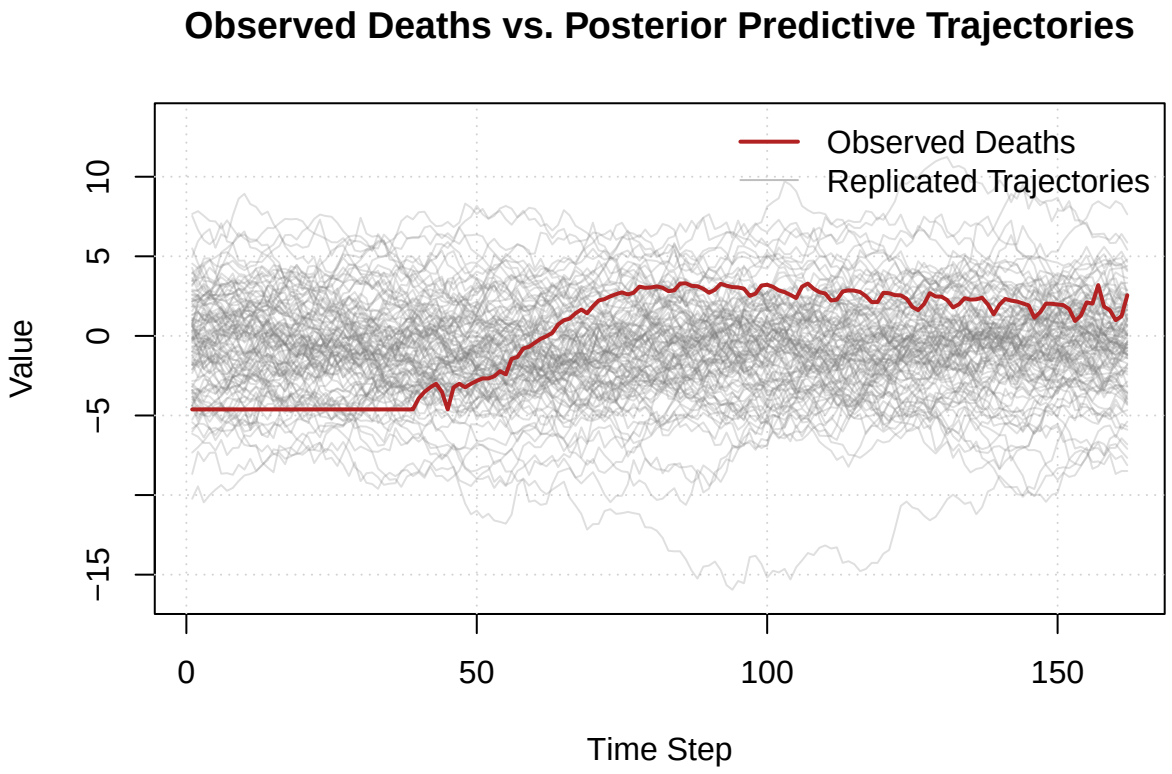
```
##                     1.282335
##                     3.242044
##                     4.652003
##                     6.462172
##                     8.767068
```

```r
par(mfrow = c(2, 3), mar = c(4, 4, 3, 2))

for (i in 1:length(quantile_probs)) {
  hist(rep_quantiles[, i],
       main = paste("Post. Pred. Dist. of", colnames(rep_quantiles)[i], "Quantile"),
       xlab = "Value",
       col = "lightgreen",
       border = "white",
       breaks = 30)
  abline(v = obs_quantiles[i], col = "red", lwd = 2, lty = 2)
  legend("topright", "Observed Value", col = "red", lty = 2, lwd = 2, bty = "n")
}

# Reset plotting parameters
par(mfrow = c(1, 1))
```




```r
y_range <- range(c(obs_data, unlist(posterior_predictive_samples_deaths)))

# Plot the first 100 replicated trajectories as semi-transparent lines
plot(NULL,
     xlim = c(1, n_data),
     ylim = y_range,
```

```
     main = "Observed Deaths vs. Posterior Predictive Trajectories",
     xlab = "Time Step",
     ylab = "Value")
grid()

# Add replicated data lines
for (i in 1:100) {
  lines(posterior_predictive_samples_deaths[[i]], col = rgb(0.5, 0.5, 0.5, 0.25))
}

# Superimpose the observed data as a bold, colored line
lines(obs_data, col = "firebrick", lwd = 2)

# Add a legend
legend("topright",
       legend = c("Observed Deaths", "Replicated Trajectories"),
       col = c("firebrick", "gray"),
       lwd = c(2, 1),
       bty = "n")
```



Just like the previous case, AR(1) model is a poor choice to model number of deaths/change in number of deaths data.

We can try experimenting with a complex model which is polynomial or we can model different regimes (time period) with different values of ρ by splitting the data at 3 different regions - where the observations are flat, when it starts to increase and when it reaches an equilibrium.

6. Simulate posterior predictions for the cases and deaths in the United States from July 1 until August 25. Compare these posterior predictions to the actual values of the two variables, and comment on the quality of the AR(1) model for predicting future cases and deaths in this context.

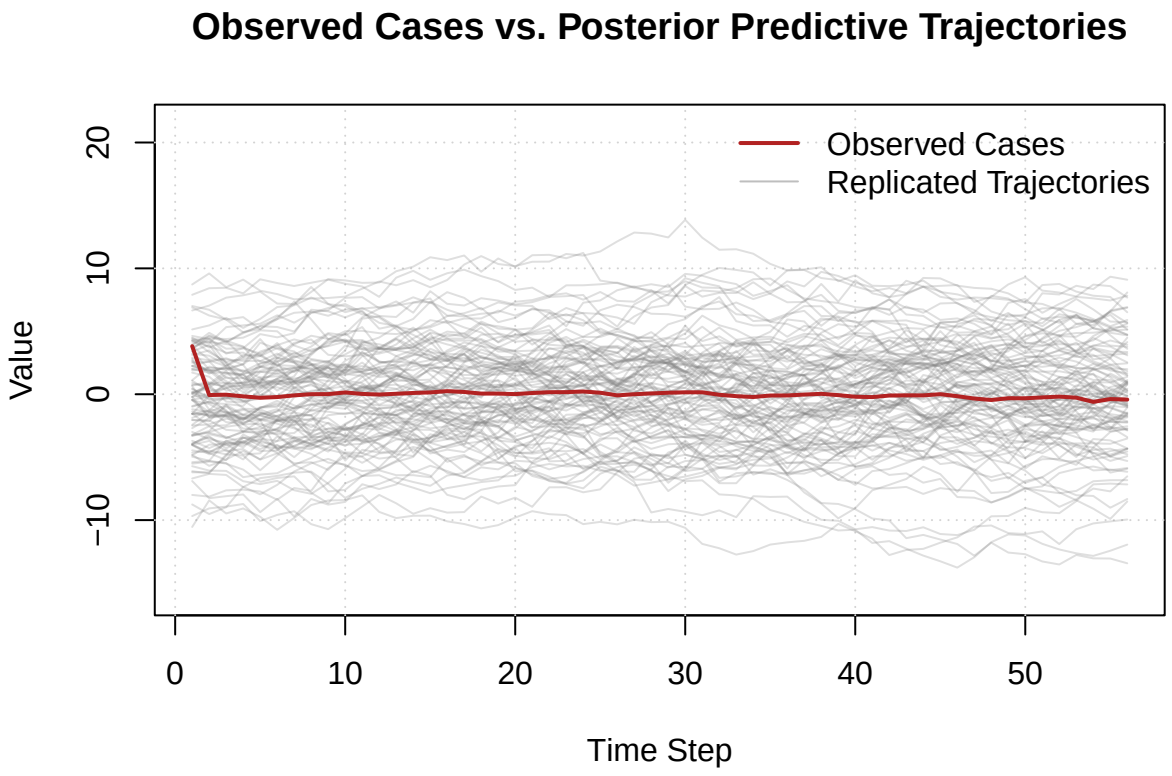
```
testing_data <- covid_us[covid_us$date >= as.Date("2020-07-01") & covid_us$date <= as.Date("2020-08-25")
head(testing_data)
```

```
##           date   cases deaths
## 163 2020-07-01 2703218 128103
## 164 2020-07-02 2758775 128826
## 165 2020-07-03 2815928 129417
## 166 2020-07-04 2865933 129678
## 167 2020-07-05 2910696 129940
## 168 2020-07-06 2958011 130331
```

```
daily_cases <- diff(testing_data$cases) # Calculate change
daily_cases <- c(testing_data$cases[1], daily_cases) # First value as initial data value
log_daily_cases <- log(daily_cases + 1) # log-transform
mean_log_daily_cases <- mean(log_daily_cases) # calculate mean
processed_data_cases <- (log_daily_cases - mean_log_daily_cases)
computation_data_cases <- data.frame(x = processed_data_cases)
```

```
obs_data <- computation_data_cases$x

n_reps <- 1000
n_data <- nrow(computation_data_cases)
posterior_predictive_samples_cases <- vector("list", n_reps)

for (i in 1:n_reps) {
  current_rho <- posterior_samples_cases$rho[i]
  current_log_sig <- posterior_samples_cases$log_sig[i]

  # Simulate a new dataset using these parameters
  posterior_predictive_samples_cases[[i]] <- simulate_ar_process(
    rho = current_rho,
    log_sig = current_log_sig,
    n = n_data
  )
}

y_range <- range(c(obs_data, unlist(posterior_predictive_samples_cases)))

# Plot the first 100 replicated trajectories as semi-transparent lines
plot(NULL,
     xlim = c(1, n_data),
     ylim = y_range,
     main = "Observed Cases vs. Posterior Predictive Trajectories",
     xlab = "Time Step",
     ylab = "Value")
grid()

# Add replicated data lines
for (i in 1:100) {
```

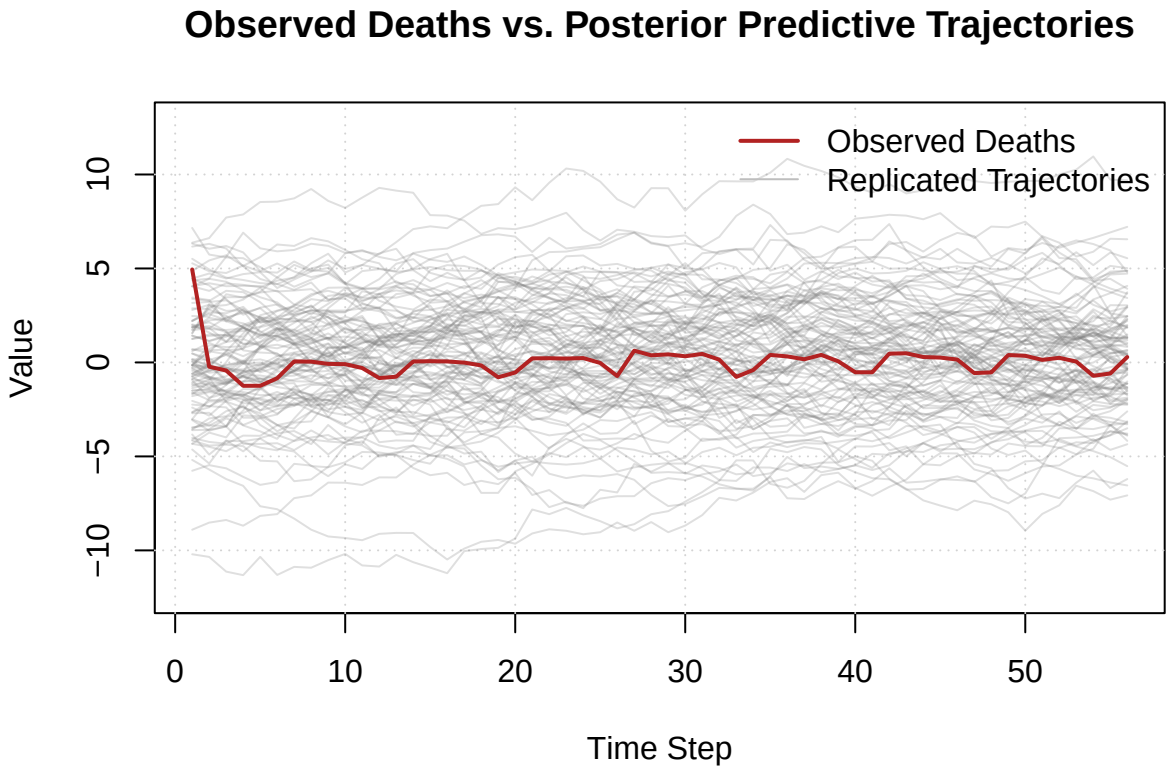
```
  lines(posterior_predictive_samples_cases[[i]], col = rgb(0.5, 0.5, 0.5, 0.25))
}

# Superimpose the observed data as a bold, colored line
lines(obs_data, col = "firebrick", lwd = 2)

# Add a legend
legend("topright",
       legend = c("Observed Cases", "Replicated Trajectories"),
       col = c("firebrick", "gray"),
       lwd = c(2, 1),
       bty = "n")
```



```
daily_deaths <- diff(testing_data$deaths) # Calculate change
daily_deaths <- c(testing_data$deaths[1], daily_deaths) # First value as initial data value
log_daily_deaths <- log(daily_deaths + 1) # log-transform
mean_log_daily_deaths <- mean(log_daily_deaths) # calculate mean
processed_data_deaths <- (log_daily_deaths - mean_log_daily_deaths)
computation_data_deaths <- data.frame(x = processed_data_deaths)
```

```
obs_data <- computation_data_deaths$x

n_reps <- 1000
n_data <- nrow(computation_data_deaths)
posterior_predictive_samples_deaths <- vector("list", n_reps)

for (i in 1:n_reps) {
  current_rho <- posterior_samples_deaths$rho[i]
```

```r
  current_log_sig <- posterior_samples_deaths$log_sig[i]

  # Simulate a new dataset using these parameters
  posterior_predictive_samples_deaths[[i]] <- simulate_ar_process(
    rho = current_rho,
    log_sig = current_log_sig,
    n = n_data
  )
}

y_range <- range(c(obs_data, unlist(posterior_predictive_samples_deaths)))

# Plot the first 100 replicated trajectories as semi-transparent lines
plot(NULL,
     xlim = c(1, n_data),
     ylim = y_range,
     main = "Observed Deaths vs. Posterior Predictive Trajectories",
     xlab = "Time Step",
     ylab = "Value")
grid()

# Add replicated data lines
for (i in 1:100) {
  lines(posterior_predictive_samples_deaths[[i]], col = rgb(0.5, 0.5, 0.5, 0.25))
}

# Superimpose the observed data as a bold, colored line
lines(obs_data, col = "firebrick", lwd = 2)

# Add a legend
legend("topright",
       legend = c("Observed Deaths", "Replicated Trajectories"),
       col = c("firebrick", "gray"),
       lwd = c(2, 1),
       bty = "n")
```



From the replicated trajectories over the unseen data we see that when the cases and deaths stabilize the model is able to generate stable trajectories. However, the AR(1) model is not able to capture the dip that is seen in the beginning of these plots. This is because of the simplicity of the AR(1) model.

7. Now replicate the analyses you performed in step 4 on the entire United States for the state of Indiana individually. Discuss if you think the results agree with the results of the entire United States.

```
covid_us_states <- read.table("./covid_us-states.txt", header = TRUE, sep = ",")
head(covid_us_states)
```

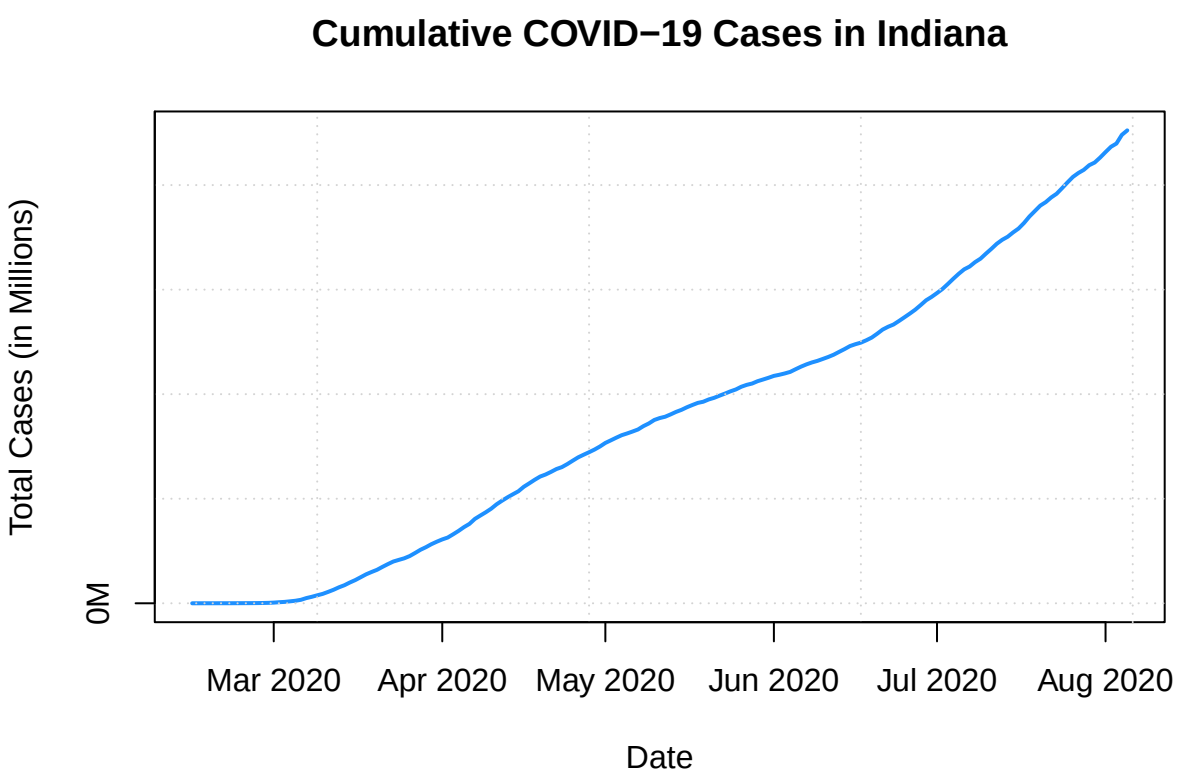
```
##   X       date      state fips cases deaths
## 1 1 2020-01-25 California    6     1      0
## 2 2 2020-01-26 California    6     2      0
## 3 3 2020-01-27 California    6     2      0
## 4 4 2020-01-28 California    6     2      0
## 5 5 2020-01-29 California    6     2      0
## 6 6 2020-01-30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

```
covid_us_states$date <- as.Date(covid_us_states$date)
# Select Indiana data
covid_us_indiana <- covid_us_states[covid_us_states$state == "Indiana",]
head(covid_us_indiana)
```

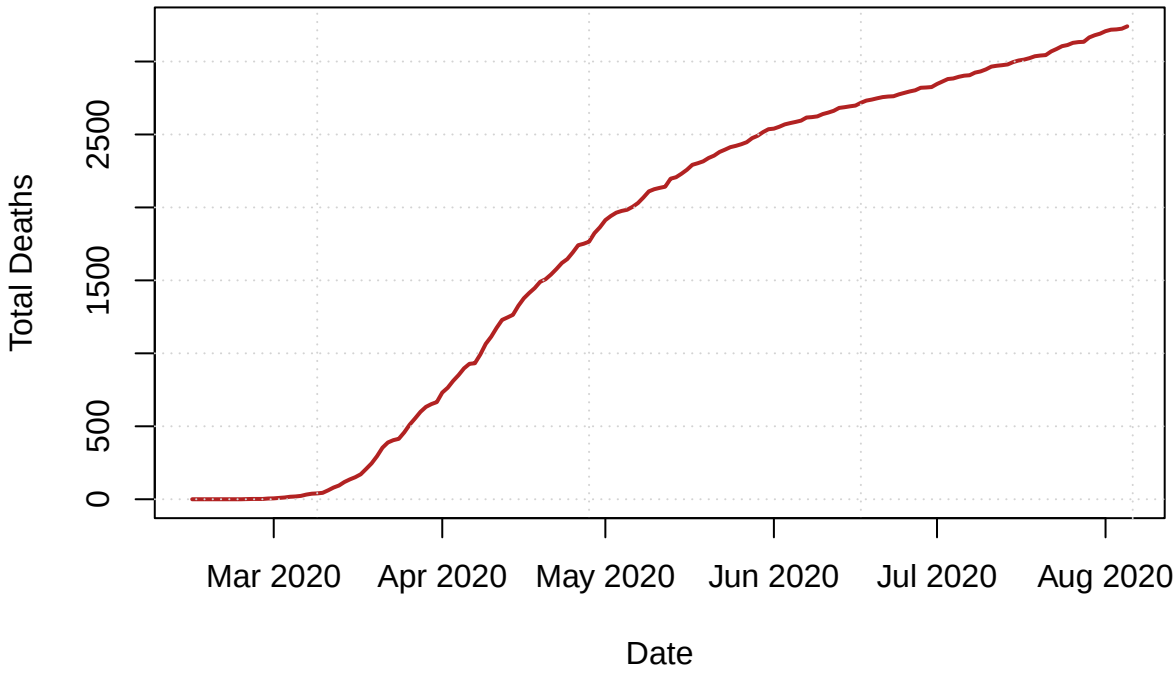
```
##     X       date   state fips cases deaths
## 66 66 2020-03-06 Indiana   18     1      0
## 69 69 2020-03-07 Indiana   18     1      0
## 72 72 2020-03-08 Indiana   18     2      0
## 75 75 2020-03-09 Indiana   18     4      0
## 78 78 2020-03-10 Indiana   18     6      0
```

```
## 81 81 2020-03-11 Indiana   18    11      0
```

```
plot(covid_us_indiana$date, covid_us_indiana$cases,
     type = 'l',
     col = 'dodgerblue',
     main = 'Cumulative COVID-19 Cases in Indiana',
     xlab = 'Date',
     ylab = 'Total Cases (in Millions)',
     lwd = 2,
     xaxt = 'n',
     yaxt = 'n'
)
grid()
axis(2, at = seq(0, max(covid_us$cases), by = 1000000),
     labels = paste0(seq(0, max(covid_us$cases), by = 1000000) / 1000000, "M"))
axis.Date(1, at = seq(min(covid_us$date), max(covid_us$date), by = "month"),
          format = "%b %Y")
```




```
plot(covid_us_indiana$date, covid_us_indiana$deaths,
     type = 'l',
     col = 'firebrick',
     main = 'Cumulative COVID-19 Deaths in Indiana',
     xlab = 'Date',
     ylab = 'Total Deaths',
     lwd = 2,
     xaxt = 'n'
)
grid()
axis.Date(1, at = seq(min(covid_us$date), max(covid_us$date), by = "month"),
          format = "%b %Y")
```



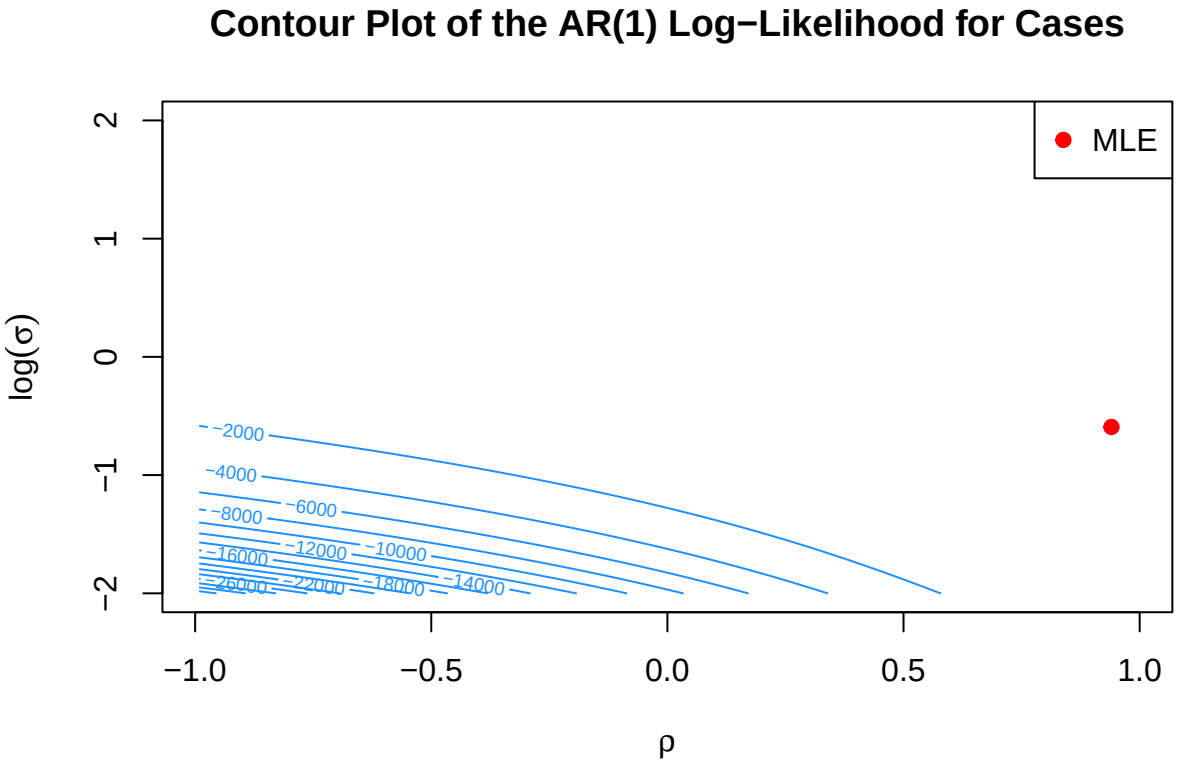

```
# Filter data for the period up to and including June 30, 2020
training_data <- covid_us_indiana[covid_us_indiana$date <= as.Date("2020-06-30"), ]
daily_cases <- diff(training_data$cases) # Calculate change
daily_cases <- c(training_data$cases[1], daily_cases) # First value as initial data value
log_daily_cases <- log(daily_cases + 1) # log-transform
mean_log_daily_cases <- mean(log_daily_cases) # calculate mean
processed_data_cases <- (log_daily_cases - mean_log_daily_cases)
computation_data_cases <- data.frame(x = processed_data_cases)
```

```
ar_loglik_cases <- function(rho, log_sig)
{
  n = length(computation_data_cases$x) # Number of rows/samples

  # sigma = exp(log_sig)
  # sigma^2 = (exp(log_sig))^2 = exp(2 * log_sig)
  sigma_sq = exp(2 * log_sig)

  y_current = computation_data_cases$x
  y_prev <- c(0, y_current[-n])
  sum_sq_err = sum((y_current - rho*y_prev)^2)

  #log_lik = -(n/2)*log(2*pi) - (n/2)*log(sigma_sq) - (1/(2*sigma_sq)) * sum_sq_err
  log_lik <- -(n-1)/2 * log(2*pi) -
             (n-1)/2 * log(sigma_sq) -
             (1/(2*sigma_sq)) * sum_sq_err
  return(log_lik)
}
```

```
rho_grid <- seq(-0.99, 0.99, length.out = 200)
log_sig_grid <- seq(-2, 2, length.out = 200)
```

```r
log_lik_surface = outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_loglik_cases))
```

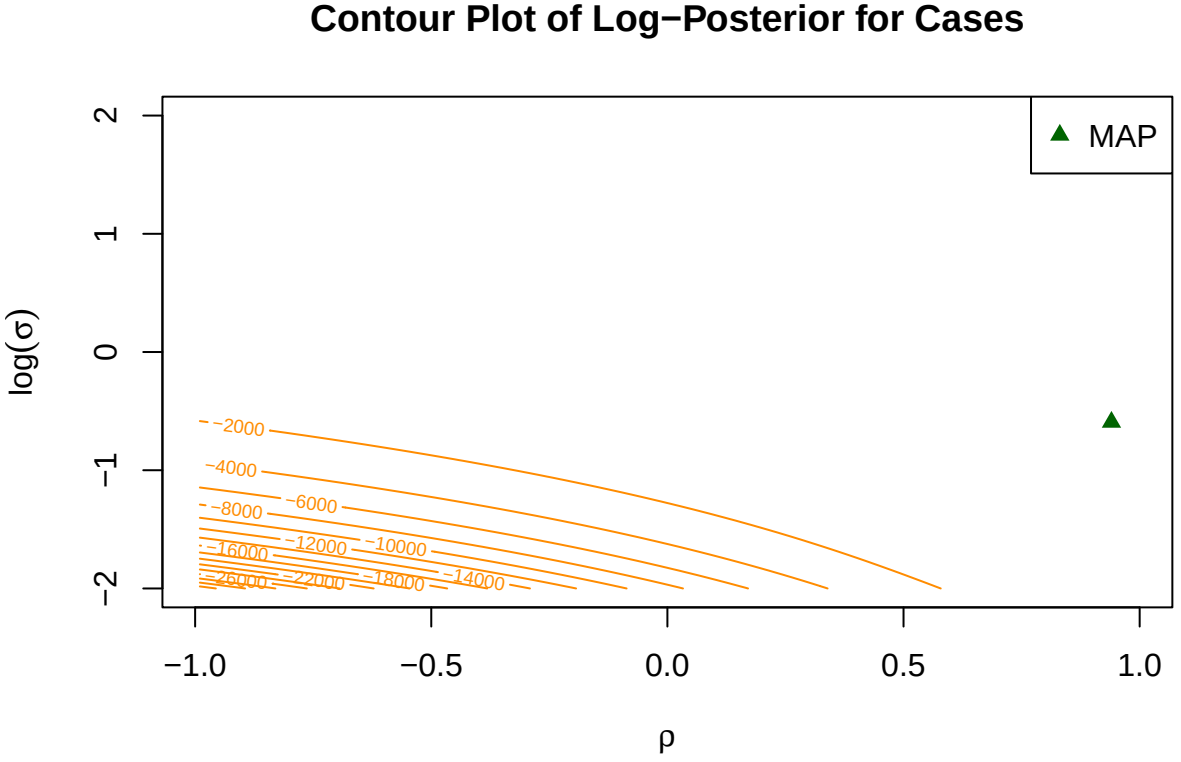
```r
contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_lik_surface,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  main = "Contour Plot of the AR(1) Log-Likelihood for Cases",
  nlevels = 20,
  col = "dodgerblue"
)

# Add a point for the maximum likelihood estimate for reference
max_lik_index <- which(log_lik_surface == max(log_lik_surface), arr.ind = TRUE)
points(rho_grid[max_lik_index[1]], log_sig_grid[max_lik_index[2]], pch = 19, col = "red")
legend("topright", "MLE", pch = 19, col = "red", bg = "white")
```




```r
ar_logposterior_cases <- function(rho, log_sig) {

  log_lik = ar_loglik_cases(rho, log_sig)

  # Log prior for rho ~ Uniform(-1, 1)
  # The density is 1/(1 - (-1)) = 1/2 for -1 < rho < 1
  # The log density is log(1/2) = -log(2) for -1 < rho < 1
  # and -Inf otherwise
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)

  # Log prior for log_sig ~ Normal(0, 10^2)
  log_prior_log_sig = -0.5 * log(2 * pi * 1) - (log_sig - 0)^2 / (2 * 1)
```

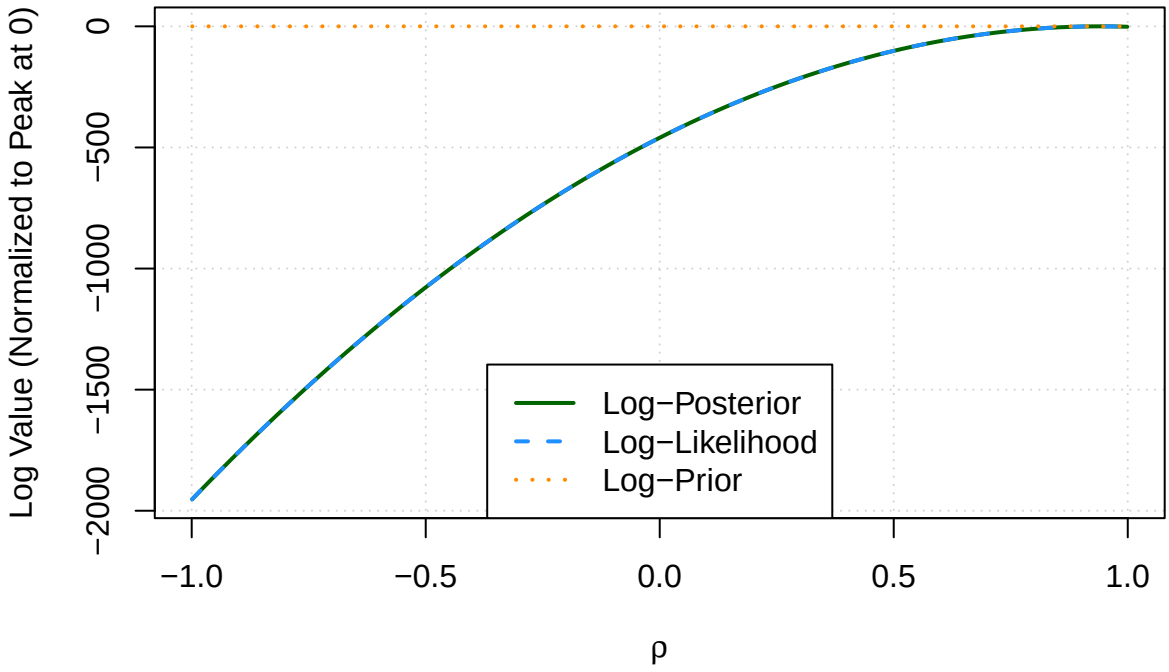
```
  log_posterior = log_lik + log_prior_rho + log_prior_log_sig

  return(log_posterior)
}
```

```
log_post_grid <- outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_logposterior_cases))

contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_post_grid,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  main = "Contour Plot of Log-Posterior for Cases",
  nlevels = 20,
  col = "darkorange"
)

# Add a point for the maximum a posteriori (MAP) estimate for reference
map_index <- which(log_post_grid == max(log_post_grid, na.rm=TRUE), arr.ind = TRUE)
points(rho_grid[map_index[1]], log_sig_grid[map_index[2]], pch = 17, col = "darkgreen")
legend("topright", "MAP", pch = 17, col = "darkgreen", bg = "white")
```



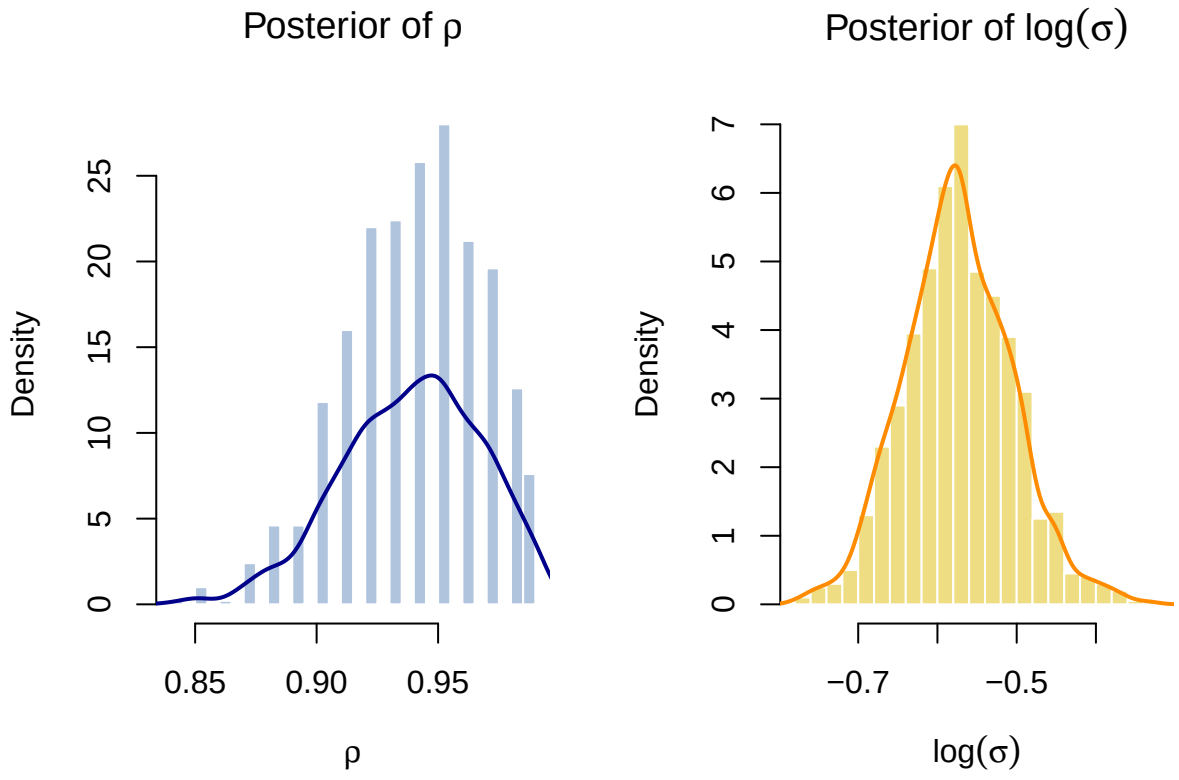

```
fixed_log_sig <- log_sig_grid[map_index[2]]

# Create a finer rho vector for a smoother plot
rho_vec <- seq(-0.999, 0.999, length.out = 500)

# Calculate log-likelihood and log-posterior along this vector
```

```r
log_lik_vals <- outer(rho_vec, fixed_log_sig, FUN = Vectorize(ar_loglik_cases))

# Define and calculate log-prior for the line plot
ar_logprior <- function(rho, log_sig) {
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)
  log_prior_log_sig = dnorm(log_sig, mean = 0, sd = 10, log = TRUE)
  return(log_prior_rho + log_prior_log_sig)
}
log_prior_vals <- ar_logprior(rho_vec, fixed_log_sig)

log_post_vals <- log_lik_vals + log_prior_vals

# Plot 3: Combined line plot
# We normalize the values to peak at 0 for easier comparison of their shapes
plot(
  rho_vec,
  log_post_vals - max(log_post_vals),
  type = 'l',
  col = "darkgreen",
  lwd = 2,
  xlab = expression(rho),
  ylab = "Log Value (Normalized to Peak at 0)",
  main = paste("Comparison at fixed log(sigma) =", round(fixed_log_sig, 2)),
  #ylim = c(-50, 5), # Adjust ylim if the plot is clipped
  panel.first = grid()
)

# Add lines for the log-likelihood and log-prior
lines(rho_vec, log_lik_vals - max(log_lik_vals), col = "dodgerblue", lwd = 2, lty = 2)
lines(rho_vec, log_prior_vals - max(log_prior_vals), col = "darkorange", lwd = 2, lty = 3)

# Add legend
legend(
  "bottom",
  legend = c("Log-Posterior", "Log-Likelihood", "Log-Prior"),
  col = c("darkgreen", "dodgerblue", "darkorange"),
  lwd = 2,
  lty = c(1, 2, 3),
  bg = "white"
)
```




```r
# For numerical stability
max_log_post <- max(log_post_grid, na.rm = TRUE)
post_grid <- exp(log_post_grid - max_log_post)

# Normalize the grid values to create a probability distribution that sums to 1.
prob_grid <- post_grid / sum(post_grid, na.rm = TRUE)

# Draw 1000 samples from posterior
n_points <- length(rho_grid) * length(log_sig_grid)
grid_indices <- 1:n_points

n_samples <- 1000

sampled_indices <- sample(
    grid_indices,
    size = n_samples,
    replace = TRUE,
    prob = as.vector(prob_grid)
)

# Convert to 2D grid
sampled_row_col <- arrayInd(sampled_indices, dim(log_post_grid))

# Extract rho and log(sigma)
sampled_rho <- rho_grid[sampled_row_col[, 1]]
sampled_log_sig <- log_sig_grid[sampled_row_col[, 2]]

# Combine the sampled parameter values into a data frame.
posterior_samples_cases <- data.frame(
    rho = sampled_rho,
```

```
    log_sig = sampled_log_sig
)

# Set up a 1x2 plotting area
par(mfrow = c(1, 2), mar = c(5, 4, 4, 2))

# Histogram for rho
hist(posterior_samples_cases$rho,
     main = expression(paste("Posterior of ", rho)),
     xlab = expression(rho),
     freq = FALSE, # Plot density instead of frequency
     col = "lightsteelblue",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples_cases$rho), col = "darkblue", lwd = 2)

# Histogram for log(sigma)
hist(posterior_samples_cases$log_sig,
     main = expression(paste("Posterior of ", log(sigma))),
     xlab = expression(log(sigma)),
     freq = FALSE, # Plot density instead of frequency
     col = "lightgoldenrod",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples_cases$log_sig), col = "darkorange", lwd = 2)
```



```r
# Reset plotting parameters
par(mfrow = c(1, 1))
```

```r
# Define the quantiles we want to compute
probs <- c(0.025, 0.25, 0.5, 0.75, 0.975)

summary_rho_cases <- c(
  mean = mean(posterior_samples_cases$rho),
  sd = sd(posterior_samples_cases$rho),
  skewness = skewness(posterior_samples_cases$rho),
  kurtosis = kurtosis(posterior_samples_cases$rho),
  quantiles = quantile(posterior_samples_cases$rho, probs = quantile_probs)
)
cat("\n--- Posterior Summary for rho ---\n")
```

```
##
## --- Posterior Summary for rho ---
```

```r
print(round(summary_rho_cases, 4))
```

```
##            mean              sd        skewness        kurtosis  quantiles.2.5%
##          0.9391          0.0284         -0.3723          2.8215          0.8806
##   quantiles.25%   quantiles.50%   quantiles.75% quantiles.97.5%
##          0.9204          0.9403          0.9602          0.9900
```

```r
summary_log_sig_cases <- c(
  mean = mean(posterior_samples_cases$log_sig),
  sd = sd(posterior_samples_cases$log_sig),
  skewness = skewness(posterior_samples_cases$log_sig),
  kurtosis = kurtosis(posterior_samples_cases$log_sig),
  quantiles = quantile(posterior_samples_cases$log_sig, probs = quantile_probs)
)
cat("\n--- Posterior Summary for log(sigma) ---\n")
```

```
##
## --- Posterior Summary for log(sigma) ---
```

```r
print(round(summary_log_sig_cases, 4))
```

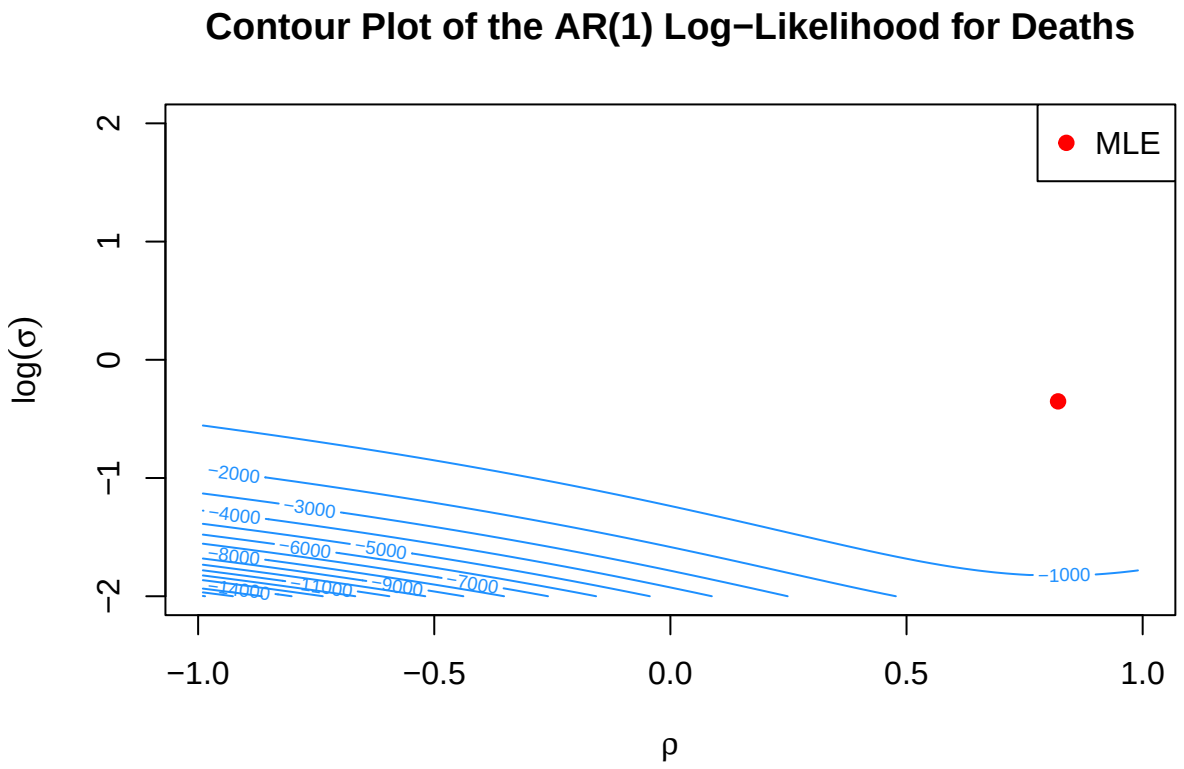
```
##            mean              sd        skewness        kurtosis  quantiles.2.5%
##         -0.5742          0.0684          0.1650          3.2269         -0.6935
##   quantiles.25%   quantiles.50%   quantiles.75% quantiles.97.5%
##         -0.6131         -0.5729         -0.5327         -0.4322
```

```r
daily_deaths <- diff(training_data$deaths)
daily_deaths <- c(training_data$deaths[1], daily_deaths)
log_daily_deaths <- log(daily_deaths + 1)
mean_log_daily_deaths <- mean(log_daily_deaths)
processed_data_deaths <- (log_daily_deaths - mean_log_daily_deaths)
computation_data_deaths <- data.frame(x = processed_data_deaths)
```

```r
ar_loglik_deaths <- function(rho, log_sig)
{
  n = length(computation_data_deaths$x) # Number of rows/samples
```

```
  # sigma = exp(log_sig)
  # sigma^2 = (exp(log_sig))^2 = exp(2 * log_sig)
  sigma_sq = exp(2 * log_sig)

  y_current = computation_data_deaths$x
  y_prev <- c(0, y_current[-n])
  sum_sq_err = sum((y_current - rho*y_prev)^2)

  #log_lik = -(n/2)*log(2*pi) - (n/2)*log(sigma_sq) - (1/(2*sigma_sq)) * sum_sq_err
  log_lik <- -(n-1)/2 * log(2*pi) -
             (n-1)/2 * log(sigma_sq) -
             (1/(2*sigma_sq)) * sum_sq_err

  return(log_lik)
}
```

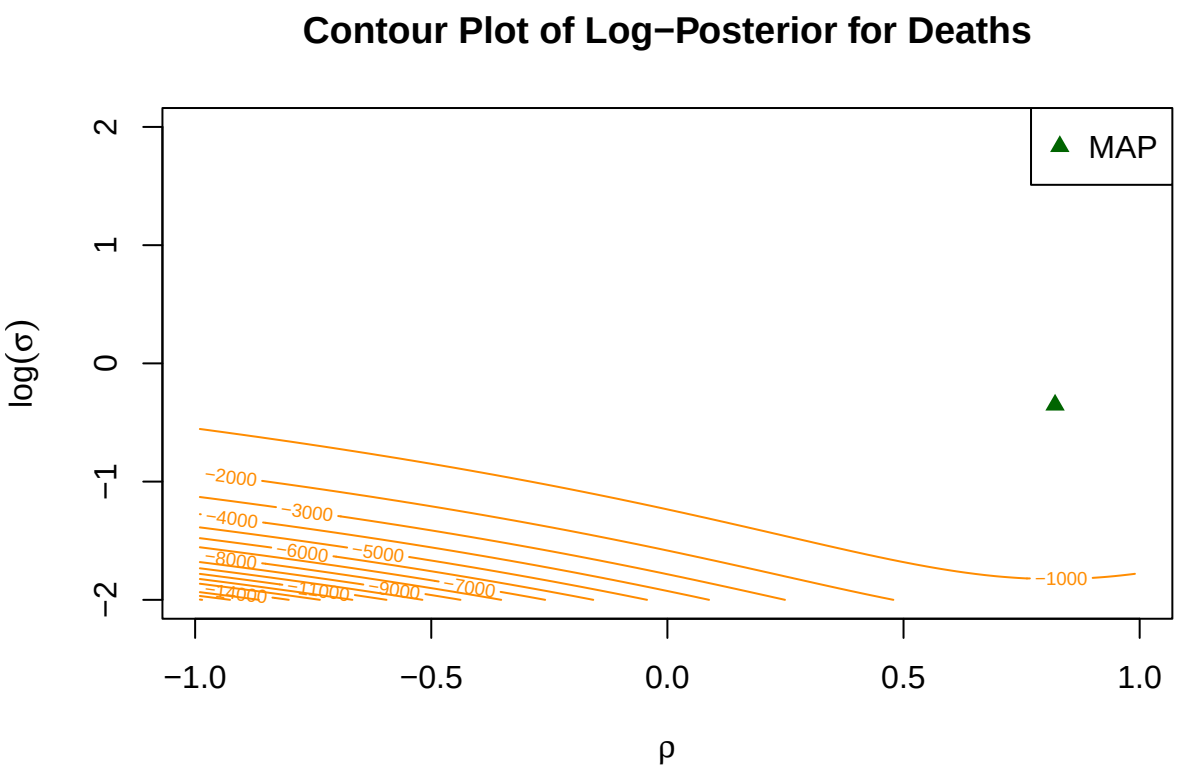
```
log_lik_surface = outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_loglik_deaths))

contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_lik_surface,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  main = "Contour Plot of the AR(1) Log-Likelihood for Deaths",
  nlevels = 20,
  col = "dodgerblue"
)

# Add a point for the maximum likelihood estimate for reference
max_lik_index <- which(log_lik_surface == max(log_lik_surface), arr.ind = TRUE)
points(rho_grid[max_lik_index[1]], log_sig_grid[max_lik_index[2]], pch = 19, col = "red")
legend("topright", "MLE", pch = 19, col = "red", bg = "white")
```




```r
ar_logposterior_deaths <- function(rho, log_sig) {

  log_lik = ar_loglik_deaths(rho, log_sig)

  # Log prior for rho ~ Uniform(-1, 1)
  # The density is 1/(1 - (-1)) = 1/2 for -1 < rho < 1
  # The log density is log(1/2) = -log(2) for -1 < rho < 1
  # and -Inf otherwise
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)

  # Log prior for log_sig ~ Normal(0, 10^2)
  log_prior_log_sig = -0.5 * log(2 * pi * 1) - (log_sig - 0)^2 / (2 * 1)

  log_posterior = log_lik + log_prior_rho + log_prior_log_sig

  return(log_posterior)
}
```

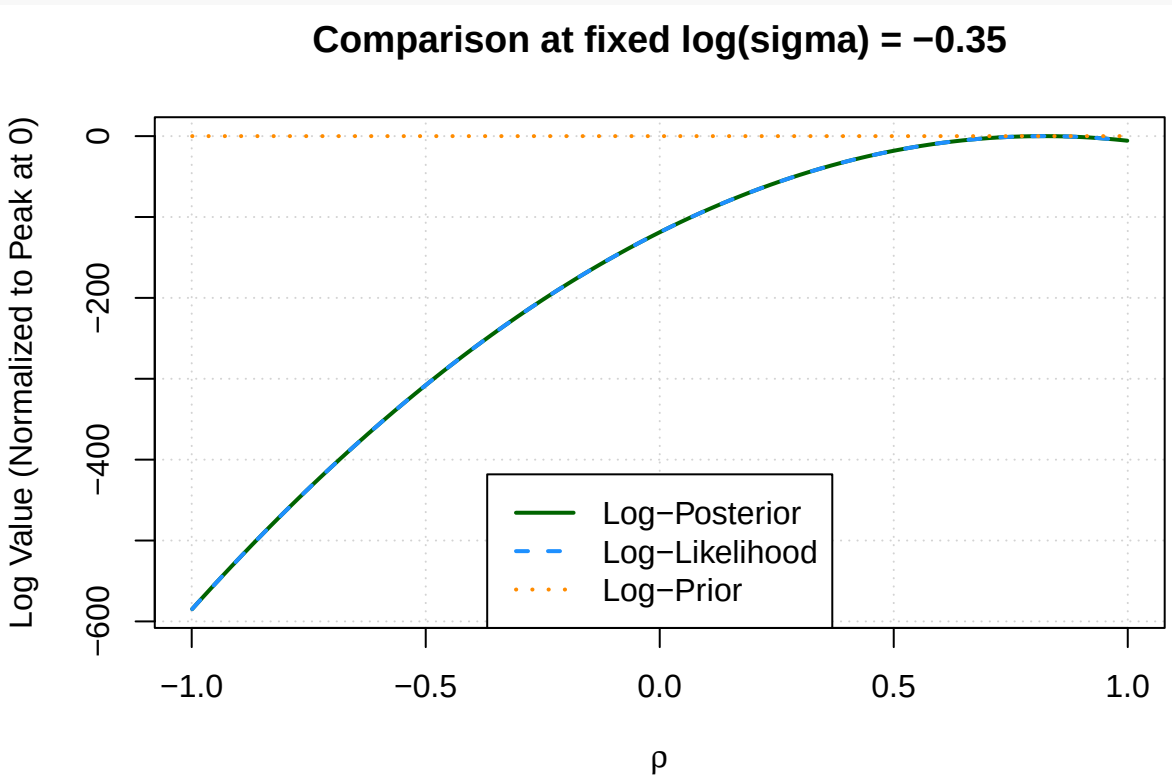
```r
log_post_grid <- outer(rho_grid, log_sig_grid, FUN = Vectorize(ar_logposterior_deaths))

contour(
  x = rho_grid,
  y = log_sig_grid,
  z = log_post_grid,
  xlab = expression(rho),
  ylab = expression(log(sigma)),
  main = "Contour Plot of Log-Posterior for Deaths",
  nlevels = 20,
  col = "darkorange"
)
```

```r
# Add a point for the maximum a posteriori (MAP) estimate for reference
map_index <- which(log_post_grid == max(log_post_grid, na.rm=TRUE), arr.ind = TRUE)
points(rho_grid[map_index[1]], log_sig_grid[map_index[2]], pch = 17, col = "darkgreen")
legend("topright", "MAP", pch = 17, col = "darkgreen", bg = "white")
```



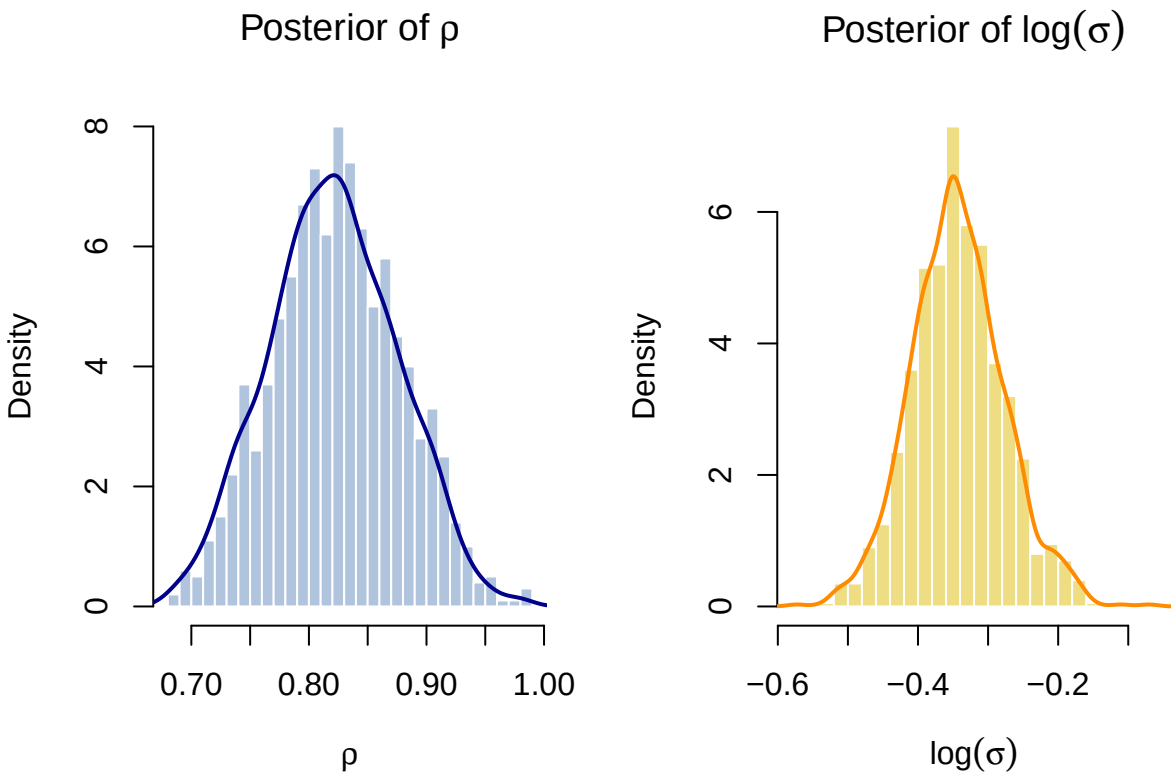

```r
fixed_log_sig <- log_sig_grid[map_index[2]]

# Create a finer rho vector for a smoother plot
rho_vec <- seq(-0.999, 0.999, length.out = 500)

# Calculate log-likelihood and log-posterior along this vector
log_lik_vals <- outer(rho_vec, fixed_log_sig, FUN = Vectorize(ar_loglik_deaths))

# Define and calculate log-prior for the line plot
ar_logprior <- function(rho, log_sig) {
  log_prior_rho = ifelse(rho > -1 & rho < 1, -log(2), -Inf)
  log_prior_log_sig = dnorm(log_sig, mean = 0, sd = 10, log = TRUE)
  return(log_prior_rho + log_prior_log_sig)
}
log_prior_vals <- ar_logprior(rho_vec, fixed_log_sig)

log_post_vals <- log_lik_vals + log_prior_vals

# Plot 3: Combined line plot
# We normalize the values to peak at 0 for easier comparison of their shapes
plot(
  rho_vec,
  log_post_vals - max(log_post_vals),
  type = 'l',
```

```
  col = "darkgreen",
  lwd = 2,
  xlab = expression(rho),
  ylab = "Log Value (Normalized to Peak at 0)",
  main = paste("Comparison at fixed log(sigma) =", round(fixed_log_sig, 2)),
  #ylim = c(-50, 5), # Adjust ylim if the plot is clipped
  panel.first = grid()
)

# Add lines for the log-likelihood and log-prior
lines(rho_vec, log_lik_vals - max(log_lik_vals), col = "dodgerblue", lwd = 2, lty = 2)
lines(rho_vec, log_prior_vals - max(log_prior_vals), col = "darkorange", lwd = 2, lty = 3)

# Add legend
legend(
  "bottom",
  legend = c("Log-Posterior", "Log-Likelihood", "Log-Prior"),
  col = c("darkgreen", "dodgerblue", "darkorange"),
  lwd = 2,
  lty = c(1, 2, 3),
  bg = "white"
)
```




```
# For numerical stability
max_log_post <- max(log_post_grid, na.rm = TRUE)
post_grid <- exp(log_post_grid - max_log_post)

# Normalize the grid values to create a probability distribution that sums to 1.
prob_grid <- post_grid / sum(post_grid, na.rm = TRUE)
```

```r
# Draw 1000 samples from posterior
n_points <- length(rho_grid) * length(log_sig_grid)
grid_indices <- 1:n_points

n_samples <- 1000

sampled_indices <- sample(
    grid_indices,
    size = n_samples,
    replace = TRUE,
    prob = as.vector(prob_grid)
)

# Convert to 2D grid
sampled_row_col <- arrayInd(sampled_indices, dim(log_post_grid))

# Extract rho and log(sigma)
sampled_rho <- rho_grid[sampled_row_col[, 1]]
sampled_log_sig <- log_sig_grid[sampled_row_col[, 2]]

# Combine the sampled parameter values into a data frame.
posterior_samples_deaths <- data.frame(
    rho = sampled_rho,
    log_sig = sampled_log_sig
)

# Set up a 1x2 plotting area
par(mfrow = c(1, 2), mar = c(5, 4, 4, 2))

# Histogram for rho
hist(posterior_samples_deaths$rho,
     main = expression(paste("Posterior of ", rho)),
     xlab = expression(rho),
     freq = FALSE, # Plot density instead of frequency
     col = "lightsteelblue",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples_deaths$rho), col = "darkblue", lwd = 2)

# Histogram for log(sigma)
hist(posterior_samples_deaths$log_sig,
     main = expression(paste("Posterior of ", log(sigma))),
     xlab = expression(log(sigma)),
     freq = FALSE, # Plot density instead of frequency
     col = "lightgoldenrod",
     border = "white",
     breaks = 30)
# Add Kernel Density Estimate (KDE) curve
lines(density(posterior_samples_deaths$log_sig), col = "darkorange", lwd = 2)
```




```
# Reset plotting parameters
par(mfrow = c(1, 1))
```

```
summary_rho_deaths <- c(
  mean = mean(posterior_samples_deaths$rho),
  sd = sd(posterior_samples_deaths$rho),
  skewness = skewness(posterior_samples_deaths$rho),
  kurtosis = kurtosis(posterior_samples_deaths$rho),
  quantiles = quantile(posterior_samples_deaths$rho, probs = quantile_probs)
)
cat("\n--- Posterior Summary for rho ---\n")
```

```
##
## --- Posterior Summary for rho ---
```

```
print(round(summary_rho_deaths, 4))
```

```
##            mean              sd        skewness        kurtosis  quantiles.2.5%
##          0.8205          0.0543          0.0634          2.6956          0.7214
##   quantiles.25%   quantiles.50%   quantiles.75% quantiles.97.5%
##          0.7811          0.8209          0.8607          0.9204
```

```
summary_log_sig_deaths <- c(
  mean = mean(posterior_samples_deaths$log_sig),
  sd = sd(posterior_samples_deaths$log_sig),
  skewness = skewness(posterior_samples_deaths$log_sig),
  kurtosis = kurtosis(posterior_samples_deaths$log_sig),
  quantiles = quantile(posterior_samples_deaths$log_sig, probs = quantile_probs)
)
cat("\n--- Posterior Summary for log(sigma) ---\n")
```

```
##
## --- Posterior Summary for log(sigma) ---
```

```
print(round(summary_log_sig_deaths, 4))
```

```
##            mean              sd        skewness        kurtosis  quantiles.2.5%
##         -0.3434          0.0663          0.1837          3.3906         -0.4724
##   quantiles.25%   quantiles.50%   quantiles.75% quantiles.97.5%
##         -0.3920         -0.3518         -0.3116         -0.2106
```

From the raw data of Indiana, we see that the trend of number of cases and deaths is similar to that of the US. However as the AR(1) model did not provide a good fit for the US data, we cannot expect it to do well with Indiana.

8. Provide a non-technical explanation of your findings for an audience with minimal statistical training. Specifically, describe the AR(1) model in lay terms, explain whether or not this model provides good predictions in this context, and provide intuitive justifications for why the AR(1) model succeeds or fails in this context.

We created a simple statistical model to predict future COVID cases. We trained the model on the first 6 months of data from the US and then asked our model to predict what would happen in July and August. The model we used is called an AR(1) model. It works under the assumption that the future will be just like the past. Think of it like predicting the local weather or temperature. We expect that there wouldn't be a sudden spike in temperature. That's how the AR(1) model works.

But due to this assumption, it's a poor choice to model covid cases. This is because as we see in the data the is a sudden increase in the number of cases and deaths before becoming stable. The model failed to predict the future. It predicted that the number of cases and deaths would be stable. But as we know, the pandemic came in waves and had sudden surges in the number of cases and deaths. Thus the AR(1) model is not a good choice for predicting covid cases.