STAT 656: Bayesian Data Analysis
Fall 2024
Homework 1

Juanwu Lu*

Synthetic Data

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

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

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

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

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

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

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

   The code implementation is as follows:

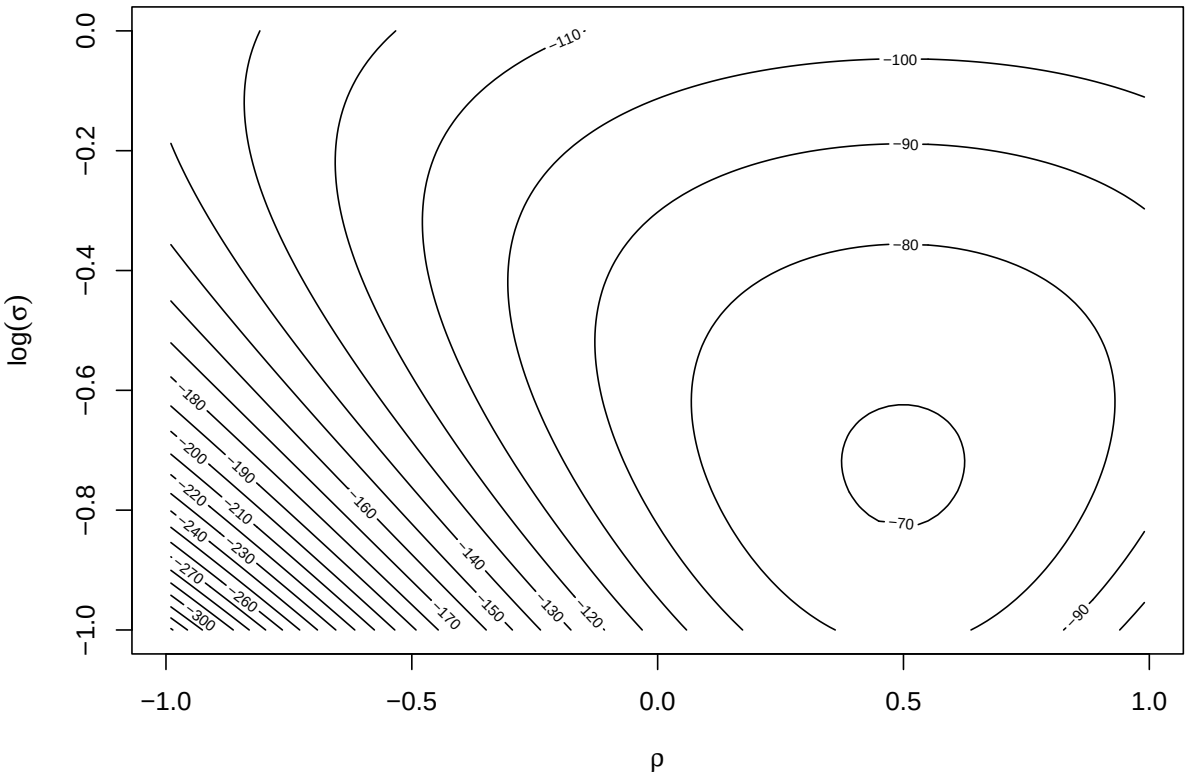
```
# Read data
if (file.exists("computation_data_hw_1.csv")) {
    data <- read.csv("computation_data_hw_1.csv")
    y <- data[['x']]
} else {
    stop("Cannot find the data file 'computation_data_hw_1.csv' at ", getwd())
}

# Define the log-likelihood function
ar_loglik <- function(rho, log_sig) {
    n <- length(y)
    sig <- exp(log_sig)
    rho <- rep(as.numeric(rho), times = n - 1)
    log_lik <- -0.5 * (
```

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

```
        n * log(2 * pi)
        + 2 * n * log_sig
        + (y[1]^2 + sum((y[2:n] - rho * y[1:(n-1)])^2)) / sig^2
    )
    return(log_lik)
}

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



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

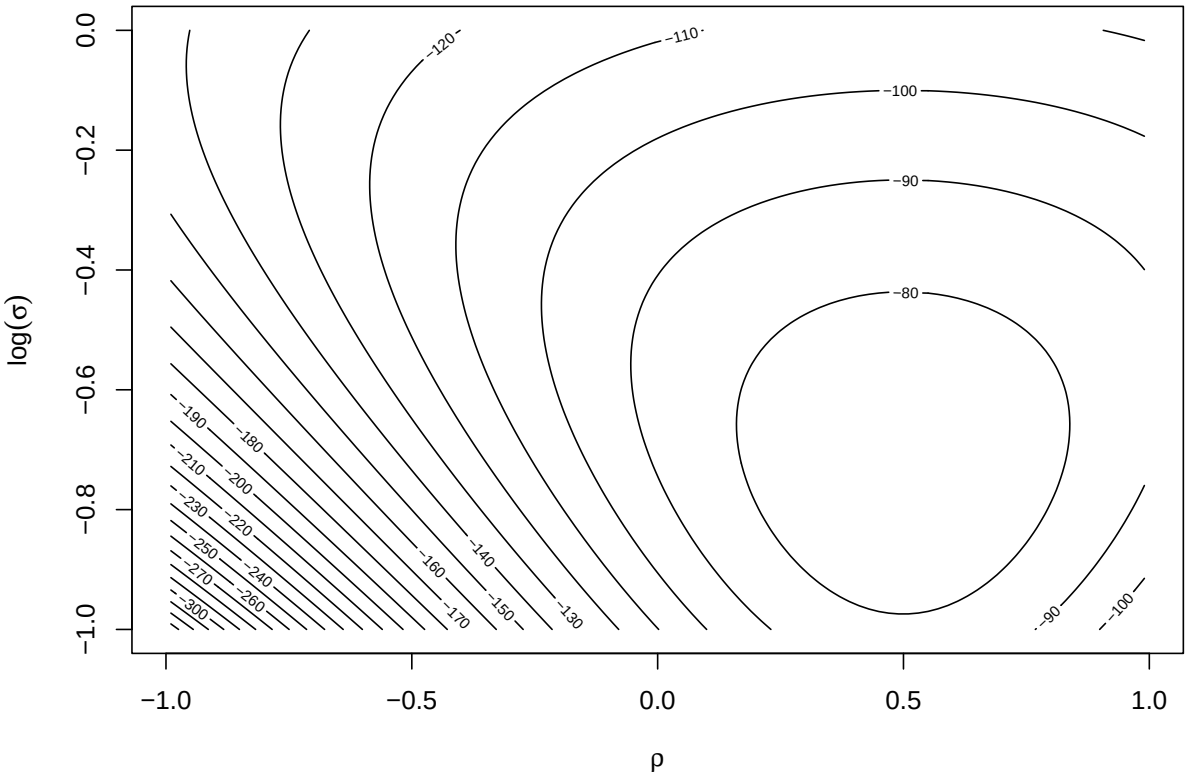
Since the prior is independent, we have $p(\rho, \log(\sigma)) = p(\rho)p(\log(\sigma)) = \frac{1}{2} \cdot \frac{1}{10\sqrt{2\pi}} \exp\left\{-\frac{\log(\sigma)^2}{200}\right\}$. Therefore, the posterior density is proportional to the product of the likelihood and the prior, and then the log of the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

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

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

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

# Visualization
logpost <- outer(rho, log_sig, Vectorize(ar_logpost))
contour(
    x=rho,
    y=log_sig,
    z=logpost,
    xlab=expression(rho),
    ylab=expression(log(sigma)),
    nlevels=20,
)
```



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

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

   From the visualization in 3., we can see that the posterior density is concentrated around $0.0 \leq \rho \leq 1.0$ and $-0.9 \leq \log(\sigma) \leq 0.0$. Therefore, we can choose a grid of ρ and $\log(\sigma)$ as follows:

```r
set.seed(42)
rho_grid <- sample(x=seq(0.0, 1.0, length=5000), size=1000)
log_sig_grid <- sample(x=seq(-0.9, 0.0, length=5000), size=1000)
```

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

   The code implementation is as follows:

```r
library(moments)

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

```
##       2.5%        25%        50%        75%      97.5%
## 0.02039408 0.24469894 0.49449890 0.75040008 0.97120424
```

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

```
## [1] "Mean: 0.4961"
```

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

```
## [1] "Standard Deviation: 0.2897"
```

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

```
## [1] "Skewnewss: 0.0052"
```

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

```
## [1] "Kurtosis: 1.7739"
```

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

```
##        2.5%         25%         50%         75%       97.5%
## -0.87465993 -0.66842869 -0.44720944 -0.21356771 -0.01871924
```

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

```
## [1] "Mean: -0.4438"
```

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

```
## [1] "Standard Deviation: 0.2620"
```

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

```
## [1] "Skewness: 0.0002"
```

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

```
## [1] "Kurtosis: 1.7764"
```

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

   The code implementation is as follows:

```r
ar_post_predictive <- function(rho, log_sig) {
    # Sample from the posterior predictive distribution
    n <- length(y)
    new_y <- rep(0.0, times = n)
    sig <- exp(log_sig)
```

```
    for (i in 2:n) {
        new_y[i] <- rnorm(n=1, mean=rho * new_y[i-1], sd=sig)
    }
    return(new_y)
}

params <- data.frame(rho=rho_grid, log_sig=log_sig_grid)
samples <- matrix(NA, nrow=nrow(params), ncol=length(y))
for (i in 1:nrow(params)) {
    samples[i,] <- ar_post_predictive(params[i, 'rho'], params[i, 'log_sig'])
}
```

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

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

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

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

```
##   [1]  0.0000000000 -0.0031875437 -0.0119846621  0.0240939651  0.0187420046
##   [6] -0.0088082768 -0.0289766955 -0.0375556621 -0.0395850767 -0.0372231542
##  [11] -0.0761318346 -0.0768494390 -0.0715749590 -0.0459076469  0.0026211557
##  [16] -0.0163020252  0.0137938285  0.0053478166 -0.0045317396 -0.0158680785
##  [21] -0.0032805275 -0.0348364792 -0.0180801002 -0.0088148512  0.0166475231
##  [26] -0.0181050505 -0.0159232358  0.0010458481 -0.0168768428  0.0339922065
##  [31]  0.0091446846 -0.0049843377 -0.0244396845 -0.0240874507  0.0027630216
##  [36] -0.0164662989 -0.0383118338 -0.0316245291 -0.0222601807  0.0080696784
##  [41] -0.0049466028 -0.0163425035 -0.0093627548 -0.0581964283 -0.0775784617
##  [46] -0.0786127125 -0.1101090524 -0.0707871260 -0.0854296289 -0.0690213897
##  [51] -0.0529924619 -0.0498759084 -0.0734066612 -0.0288514612 -0.0315544925
##  [56] -0.0745860260 -0.0400153836 -0.0304046207 -0.0365895940 -0.0593479286
##  [61] -0.0503962281 -0.0217175920 -0.0415310622 -0.0376051628 -0.0828520090
##  [66] -0.1118352919 -0.0876871955 -0.0911817387 -0.0973458004 -0.1108418324
##  [71] -0.0675015732 -0.0587272089 -0.0909219530 -0.0561468360 -0.0717478087
##  [76] -0.0604103202 -0.0729157312 -0.0740911025 -0.0612794474 -0.0605965102
##  [81] -0.0653237473 -0.0849150762 -0.0355167860  0.0090595363 -0.0026425977
##  [86] -0.0003404731 -0.0143654785 -0.0361299202  0.0080423459  0.0056462280
##  [91]  0.0069017208  0.0082684773  0.0103926752 -0.0103737261  0.0598787876
##  [96]  0.0314636405 -0.0070527633  0.0340970571  0.0486960830  0.0272661360
```

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

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

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

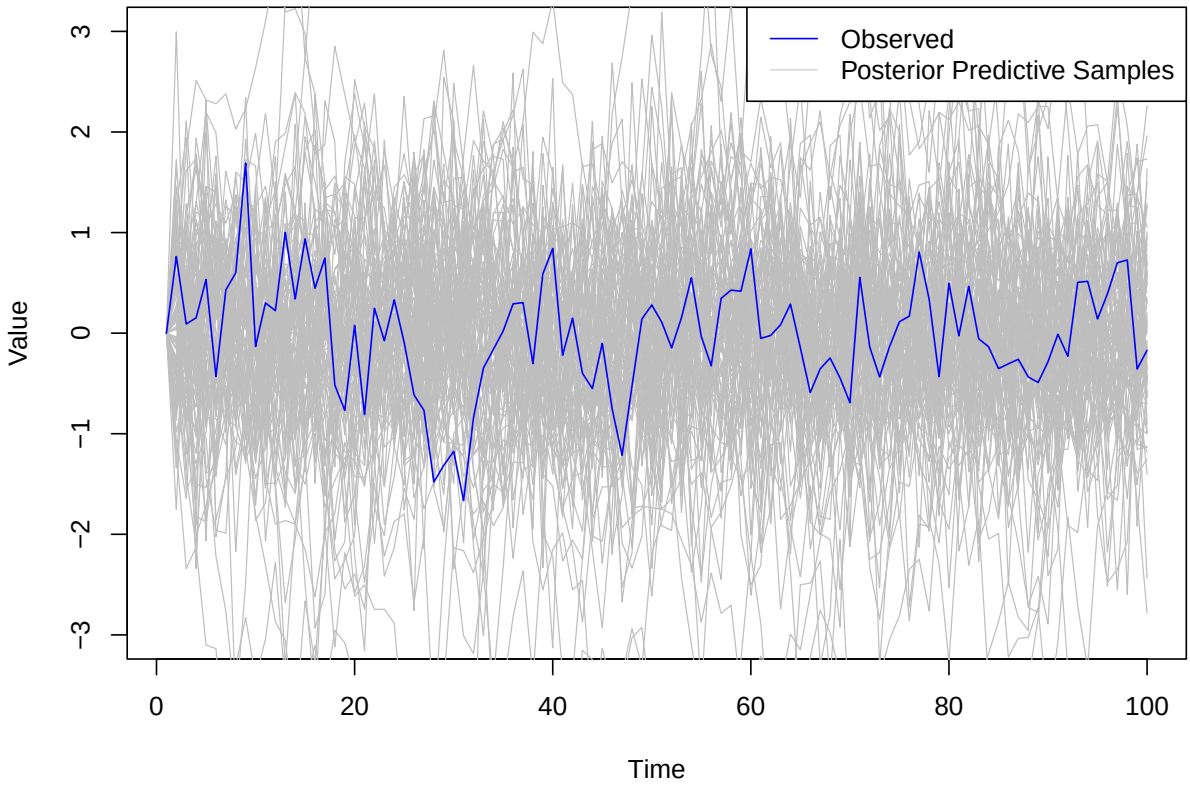
```
##   [1] 0.0000000 0.6800868 0.8085618 0.8560047 0.8934981 0.8914606 0.8942376
##   [8] 0.9514328 0.9679373 0.9468258 0.9725048 0.9786030 1.0193521 1.0123199
##  [15] 0.9656909 0.9901524 1.0002605 0.9694573 1.0173110 1.0529548 1.0376654
##  [22] 1.0445598 1.0405291 1.0697708 1.0792462 1.0738456 1.0965141 1.0954564
##  [29] 1.1375857 1.1435963 1.1068939 1.1152165 1.1346099 1.1104811 1.1534850
##  [36] 1.1260849 1.1755581 1.1503553 1.1483156 1.1626453 1.1541862 1.1611693
##  [43] 1.1634377 1.1593565 1.1771083 1.2193605 1.2138418 1.2240727 1.1986682
##  [50] 1.1879993 1.2303566 1.2132209 1.2031022 1.2293575 1.1977091 1.2054920
```

```
##  [57] 1.1678305 1.1439297 1.1448539 1.1497848 1.1655418 1.1840338 1.1609326
##  [64] 1.1433708 1.1886212 1.1973919 1.2055565 1.1972511 1.1516102 1.1468456
##  [71] 1.1679882 1.2015700 1.1891387 1.1418216 1.1255290 1.1480573 1.1467357
##  [78] 1.1194994 1.1459664 1.1463916 1.1498872 1.1355445 1.1454236 1.1298158
##  [85] 1.1693619 1.1382310 1.1432368 1.1447570 1.1589832 1.1604316 1.1638586
##  [92] 1.1827594 1.1305361 1.1588163 1.1942917 1.2291056 1.2165072 1.2114300
##  [99] 1.2454818 1.2505021
```

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

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

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



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