Bayesian Learning - Lab 04

Lakshidaa Saigiridharan (laksa656) and Maximilian Pfundstein (maxpf364)

2019-05-18

Contents

1 Time series models in Stan

Exercise:

(a) Write a function in R that simulates data from the AR(1)-process

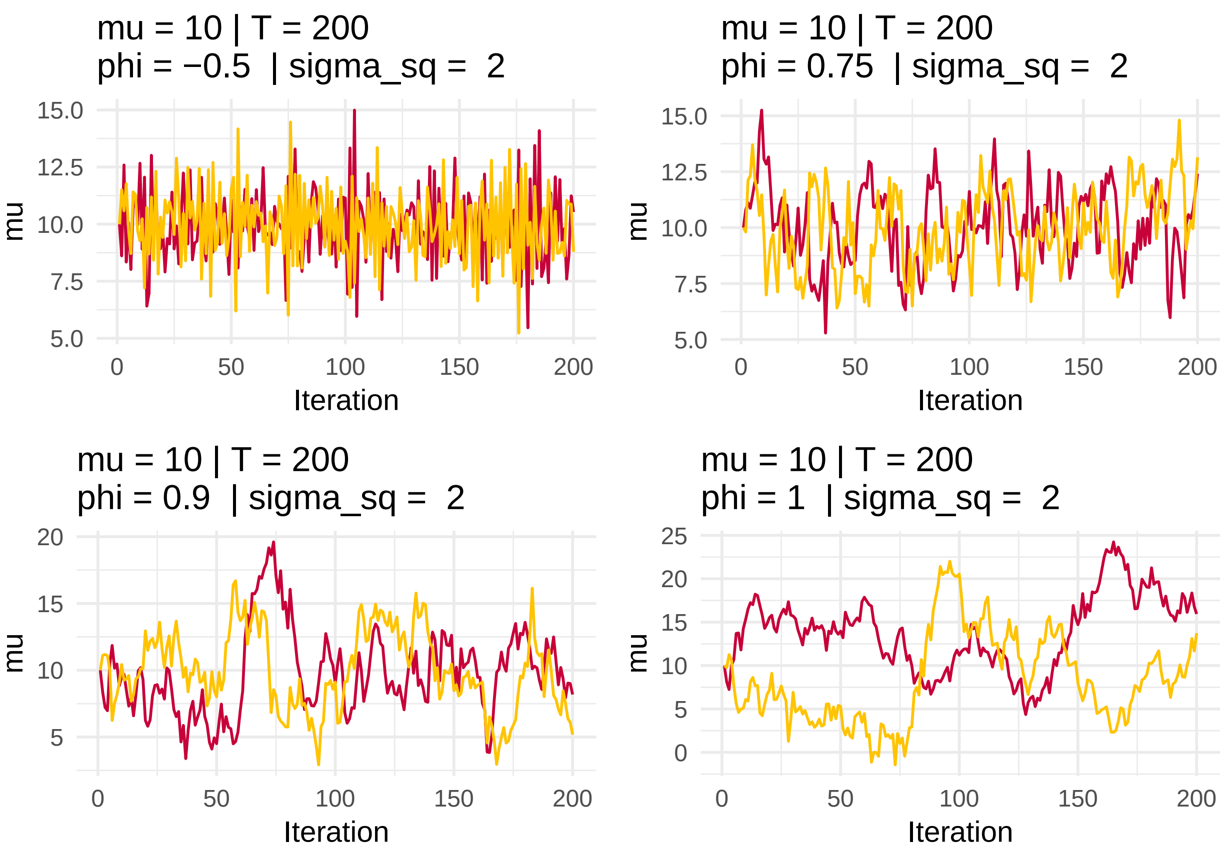
$$x_t = \mu + \phi(x_{t-1} - \mu) + \epsilon_t, \epsilon_t \sim N(0, \sigma^2)$$

for given values of μ, ϕ and σ^2. Start the process at $x_1 = \mu$ and then simulate values for x_t for t = 2, 3 . . . , T and return the vector $x_{1:T}$ containing all time points. Use $\mu = 10$, $\sigma^2 = 2$ and T = 200 and look at some different realizations (simulations) of $x_{1:T}$ for values of ϕ between -1 and 1 (this is the interval of ϕ where the AR(1)-process is stable). Include a plot of at least one realization in the report. What effect does the value of ϕ have on $x_{1:T}$?

```
ar_process = function(mu, tau, phi, sigma_sq) {

  # storing the values
  X = rep(NaN, tau)

  X[1] = mu

  for (i in 1:(tau - 1)) {
    X[i+1] = mu + phi *(X[i] - mu) + rnorm(n = 1, mean = 0, sd = sqrt(sigma_sq))
  }

  return(X)
}
```

```
simulate_ar_process = function(mu, tau, phi, sigma_sq) {

  res_1 = ar_process(mu, tau, phi, sigma_sq)
  res_2 = ar_process(mu, tau, phi, sigma_sq)

  df = data.frame(x = 1:tau,
                  y1 = res_1,
                  y2 = res_2)

  p = ggplot(df) +
    geom_line(aes(x = x, y = y1), color = "#C70039") +
    geom_line(aes(x = x, y = y2), color = "#FFC300") +
    labs(title = paste("mu =", mu, "| T =", tau, "\nphi =", phi, " | sigma_sq = ", sigma_sq) , y = "mu"
```

```
    x = "Iteration", color = "Legend") +
    theme_minimal()

  return(p)
}

p1 = simulate_ar_process(mu = 10, tau = 200, phi = -0.5, sigma_sq = 2)
p2 = simulate_ar_process(mu = 10, tau = 200, phi = 0.75, sigma_sq = 2)
p3 = simulate_ar_process(mu = 10, tau = 200, phi = 0.9, sigma_sq = 2)
p4 = simulate_ar_process(mu = 10, tau = 200, phi = 1, sigma_sq = 2)

grid.arrange(p1, p2, p3, p4, nrow = 2)
```



Answer: Phi = 1 is random walk, if abs(phi) < 1 it's wide-sense stationary and if abs(phi) > 1 it's not stationary.

(b) Use your function from a) to simulate two AR(1)-processes, $x_{1:T}$ with $\phi = 0.3$ and $y_{1:T}$ with $\phi = 0.95$. Now, treat the values of μ, ϕ and σ^2 as unknown and estimate them using MCMC. Implement Stan-code that samples from the posterior of the three parameters, using suitable non-informative priors of your choice. [Hint: Look at the time-series models examples in the Stan reference manual, and note the different parameterization used here.]

  (i) Report the posterior mean, 95% credible intervals and the number of effective posterior samples for the three inferred parameters for each of the simulated AR(1)-process. Are you able to estimate the true values?

  (ii) For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of μ and ϕ. Comments?

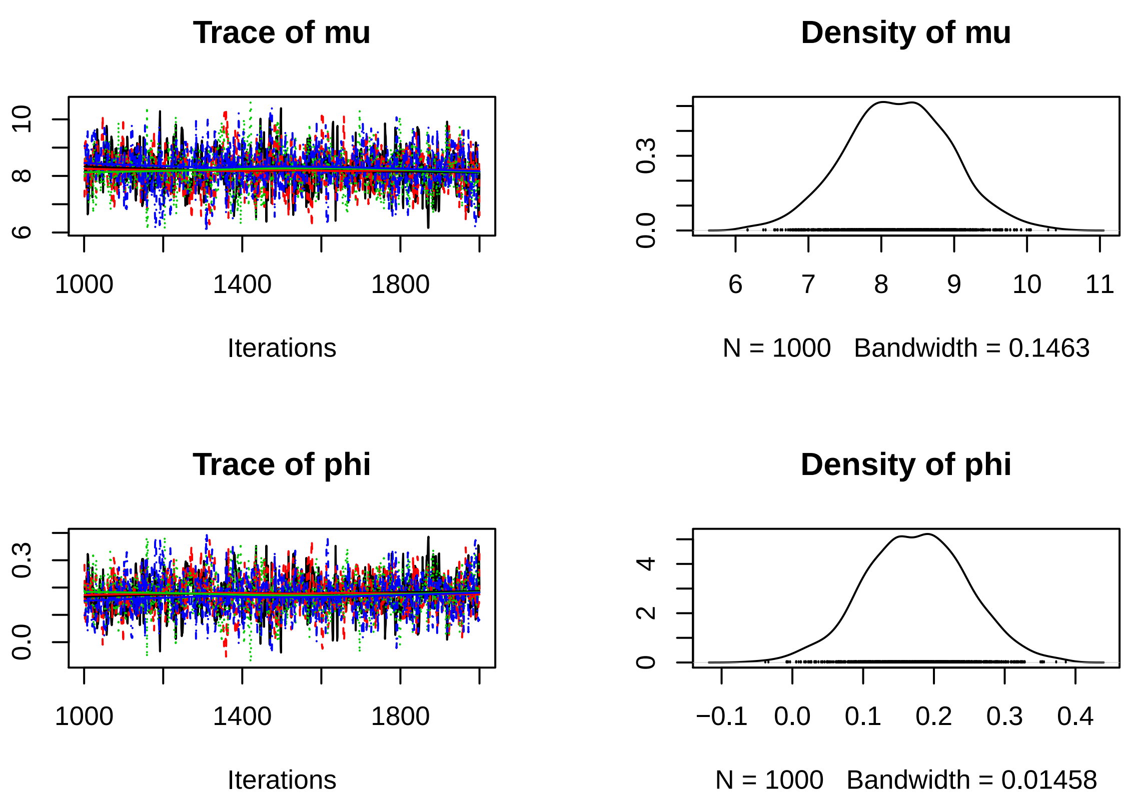
```r
stanModelX
```

```
## Inference for Stan model: AR_X.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## mu       8.23    0.02 0.73    6.81    7.74    8.23    8.73    9.68  1012
## phi      0.18    0.00 0.07    0.03    0.12    0.18    0.22    0.32  1023
## sigma    1.11    0.00 0.03    1.06    1.09    1.11    1.13    1.17  1730
## lp__  -142.20    0.03 1.17 -145.10 -142.79 -141.90 -141.31 -140.80  1481
##       Rhat
## mu       1
## phi      1
## sigma    1
## lp__     1
##
## Samples were drawn using NUTS(diag_e) at Sat May 18 16:09:28 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

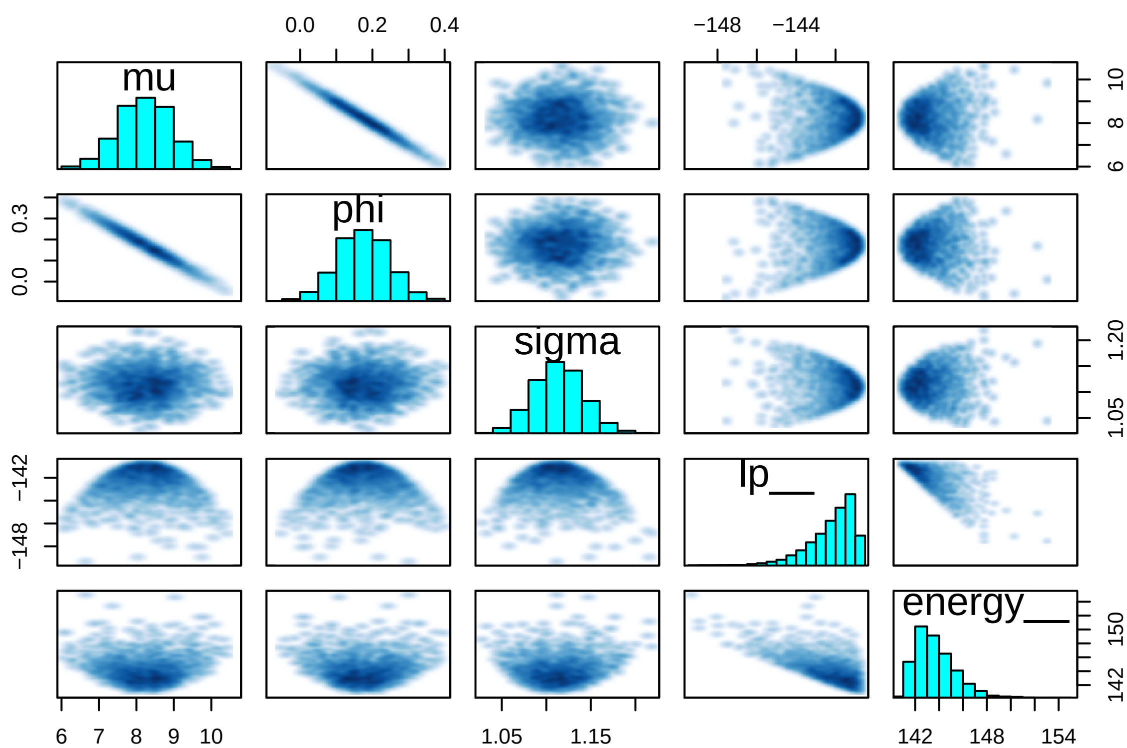
```r
summary(stanModelX, pars = c("mu", "phi", "sigma"), probs = c(0.025, 0.975))$summary
```

```
##            mean     se_mean         sd       2.5%     97.5%    n_eff
## mu    8.2282283 0.022790673 0.72519031 6.80978775 9.6761063 1012.487
## phi   0.1750827 0.002258961 0.07224726 0.02852201 0.3181852 1022.882
## sigma 1.1141405 0.000665824 0.02769335 1.06137429 1.1707386 1729.944
##           Rhat
## mu    1.002057
## phi   1.001997
## sigma 1.000610
```

```r
plot(posterior_paramsX)
```




```r
pairs(stanModelX)
```




```r
stanModelY
```

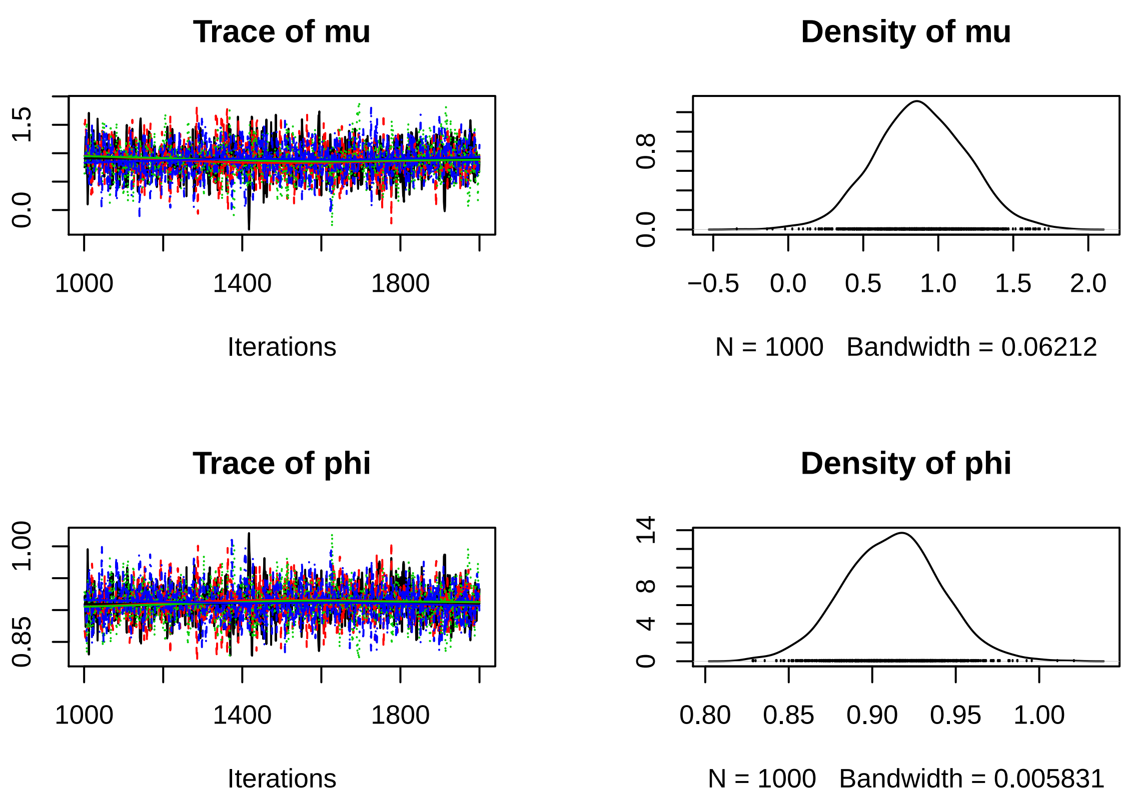
```
## Inference for Stan model: AR_X.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
```

```
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## mu       0.88    0.01 0.31    0.27    0.68    0.88    1.09    1.48  1699
## phi      0.91    0.00 0.03    0.85    0.89    0.91    0.93    0.97  1682
## sigma    1.20    0.00 0.03    1.14    1.17    1.19    1.21    1.26  1952
## lp__  -170.24    0.03 1.23 -173.43 -170.75 -169.93 -169.36 -168.85  1429
##       Rhat
## mu       1
## phi      1
## sigma    1
## lp__     1
##
## Samples were drawn using NUTS(diag_e) at Sat May 18 16:09:29 2019.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

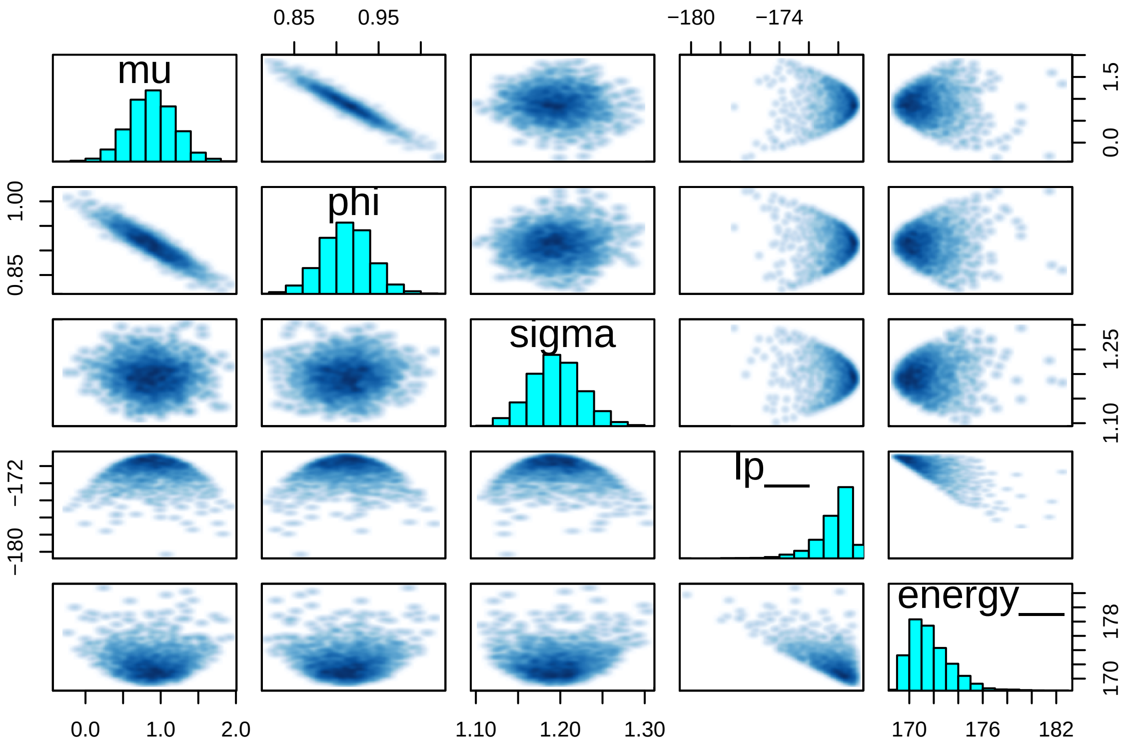
```r
summary(stanModelY, pars = c("mu", "phi", "sigma"), probs = c(0.025, 0.975))$summary
```

```
##            mean      se_mean         sd      2.5%     97.5%    n_eff
## mu    0.8805876 0.0075128202 0.30970850 0.2708691 1.4837954 1699.418
## phi   0.9119667 0.0007065881 0.02898183 0.8546124 0.9691476 1682.359
## sigma 1.1953664 0.0006814864 0.03010566 1.1367789 1.2560276 1951.560
##           Rhat
## mu    1.000606
## phi   1.000854
## sigma 1.000377
```

```r
plot(posterior_paramsY)
```



```r
pairs(stanModelY)
```



(c) The data `campy.dat` contain the number of cases of campylobacter infections in the north of the province Quebec (Canada) in four week intervals from January 1990 to the end of October 2000. It has 13 observations per year and 140 observations in total. Assume that the number of infections c_t at each time point follows an independent Poisson distribution when conditioned on a latent AR(1)-process x_t, that is

$$c_t|x_t \sim Poisson(exp(x_t))$$

where x_t is an AR(1)-process as in a). Implement and estimate the model in Stan, using suitable priors of your choice. Produce a plot that contains both the data and the posterior mean and 95% credible intervals for the latent intensity $\theta_t = exp(x_t)$ over time. [Hint: Should x_t be seen as data or parameters?]

(d) Now, assume that we have a prior belief that the true underlying intensity θ_t varies more smoothly than the data suggests. Change the prior for σ^2 so that it becomes informative about that the AR(1)-process increments ϵ_t should be small. Re-estimate the model using Stan with the new prior and produce the same plot as in c). Has the posterior for θ_t changed?

2 Source Code

```r
knitr::opts_chunk$set(echo = TRUE)
library(knitr)
library(ggplot2)
library(gridExtra)
library(coda)
library(rstan)

# RStan Setup
## Use multi-cores
```

```
options(mc.cores = parallel::detectCores())
## Reuse compiled binary
rstan_options(auto_write = TRUE)

ar_process = function(mu, tau, phi, sigma_sq) {

  # storing the values
  X = rep(NaN, tau)

  X[1] = mu

  for (i in 1:(tau - 1)) {
    X[i+1] = mu + phi *(X[i] - mu) + rnorm(n = 1, mean = 0, sd = sqrt(sigma_sq))
  }

  return(X)
}


simulate_ar_process = function(mu, tau, phi, sigma_sq) {

  res_1 = ar_process(mu, tau, phi, sigma_sq)
  res_2 = ar_process(mu, tau, phi, sigma_sq)

  df = data.frame(x = 1:tau,
                  y1 = res_1,
                  y2 = res_2)

  p = ggplot(df) +
    geom_line(aes(x = x, y = y1), color = "#C70039") +
    geom_line(aes(x = x, y = y2), color = "#FFC300") +
    labs(title = paste("mu =", mu, "| T =", tau, "\nphi =", phi, " | sigma_sq = ", sigma_sq) , y = "mu"
    x = "Iteration", color = "Legend") +
    theme_minimal()

  return(p)
}

p1 = simulate_ar_process(mu = 10, tau = 200, phi = -0.5, sigma_sq = 2)
p2 = simulate_ar_process(mu = 10, tau = 200, phi = 0.75, sigma_sq = 2)
p3 = simulate_ar_process(mu = 10, tau = 200, phi = 0.9, sigma_sq = 2)
p4 = simulate_ar_process(mu = 10, tau = 200, phi = 1, sigma_sq = 2)

grid.arrange(p1, p2, p3, p4, nrow = 2)


X = ar_process(mu = 10, tau = 200, phi = 0.3, sigma_sq = 2)
Y = ar_process(mu = 10, tau = 200, phi = 0.95, sigma_sq = 2)

stanModel = '

data {
  int<lower=0> N;
```

```
  vector[N] y;
}
parameters {
  real mu;
  real phi;
  real<lower=0> sigma;
}
model {
  y[2:N] ~ normal(mu + phi * y[1:(N - 1)], sigma*sigma);
}

'

stanModelX = stan(model_code = stanModel,
                  model_name = "AR_X",
                  data = list(N = length(X), y = X),
                  warmup = 1000,
                  iter = 2000)

stanModelY = stan(model_code = stanModel,
                  model_name = "AR_Y",
                  data = list(N = length(Y), y = Y),
                  warmup = 1000,
                  iter = 2000)

posteriorX = extract(stanModelX)
posterior_paramsX = As.mcmc.list(stanModelX, c("mu", "phi"))

posteriorY = extract(stanModelY)
posterior_paramsY = As.mcmc.list(stanModelY, c("mu", "phi"))


stanModelX
summary(stanModelX, pars = c("mu", "phi", "sigma"), probs = c(0.025, 0.975))$summary

plot(posterior_paramsX)
pairs(stanModelX)


stanModelY
summary(stanModelY, pars = c("mu", "phi", "sigma"), probs = c(0.025, 0.975))$summary

plot(posterior_paramsY)
pairs(stanModelY)
```