

# Lab3 report

Yi Hung Chen, Jonathan Dorairaj

2023-05-14

## Q1 Gibbs sampler for a normal model

(a Implement (code!) a Gibbs sampler that simulates from the joint posterior

```
#####Q1
library(ggplot2)
library(mvtnorm)
#library(extraDistr)
precipitation <- readRDS("Precipitation.rds")
log_prec <- log(precipitation)

mu_post <- function(mu_0,tau_0_sqr,sigma_square_current,y,n){

  tau_n_sqr = 1 / (1/tau_0_sqr + n/sigma_square_current)
  mu_n <- tau_n_sqr * (mu_0/tau_0_sqr + sum(y)/sigma_square_current)
  mupost <- rnorm(1, mu_n, sqrt(tau_n_sqr))
  return(mupost)

}

sigma_square_post <- function(mu_current, v_0, sigma_square_0, y,n, use_myinvchi=TRUE) {

  v_n <- v_0 + n
  elem1 <- v_0*sigma_square_0
  elem2 <- sum((y - mu_current)^2)
  elem3 <- n+v_0
  elem_comb <- (elem1+elem2)/elem3
  if (use_myinvchi){
    sigma_square_post <- my_inv_chi(v_n,elem_comb)
  }
  else {
    sigma_square_post <- rinvchisq(1, v_n, elem_comb) #Requires package extraDistr
  }

  return(sigma_square_post)
}

my_inv_chi<- function(df,tau_sqr) {
  X <- rchisq(1,df)
```

```

    inv_chi <- (df*tau_sqr)/X
    return(inv_chi)
}

#init
mu_0 <- 0
tau_0_sqr <- 1
sigma_square_0 <- 1
v_0 <- 1 #degree of freedom for chi square

gibbs_sampler <- function(nstep, data, mu_0, tau_0_sqr, v_0, sigma_square_0) {
  # Init parameters
  mu_current <- 0
  sigma_square_current <- 1

  mu_samples <- rep(0,nstep)
  sigma_square_samples <- rep(0,nstep)

  for (i in 1:nstep) {

    mu_current <- mu_post(mu_0, tau_0_sqr, sigma_square_current, y=data, length(data))
    #print(mu_current)
    sigma_square_current <- sigma_square_post(mu_current, v_0, sigma_square_0, y=data,
                                              length(data),use_myinvchi=TRUE)
    #print(sigma_square_current)

    mu_samples[i] <- mu_current
    sigma_square_samples[i] <- sigma_square_current
  }

  output_df <- data.frame(mu_sample = mu_samples, sigma_sample = sigma_square_samples)
  return(output_df)
}

sample_gibbs <- gibbs_sampler(nstep=10000, data=log_prec, mu_0, tau_0_sqr, v_0, sigma_square_0)

```

In lecture slide  $IF = 1 + 2 \sum_{k=1}^{\infty} \rho_k$  where  $\rho_k$  is autocorrelation at lag k

```

my_acf <- acf(sample_gibbs$mu_sample,plot = F) # getting the autocorrelation
if_mu <- 1 + 2 * sum(my_acf$acf[-1])
my_acf <- acf(sample_gibbs$sigma_sample,plot = F)
if_sigma <- 1 + 2 * sum(my_acf$acf[-1])

```

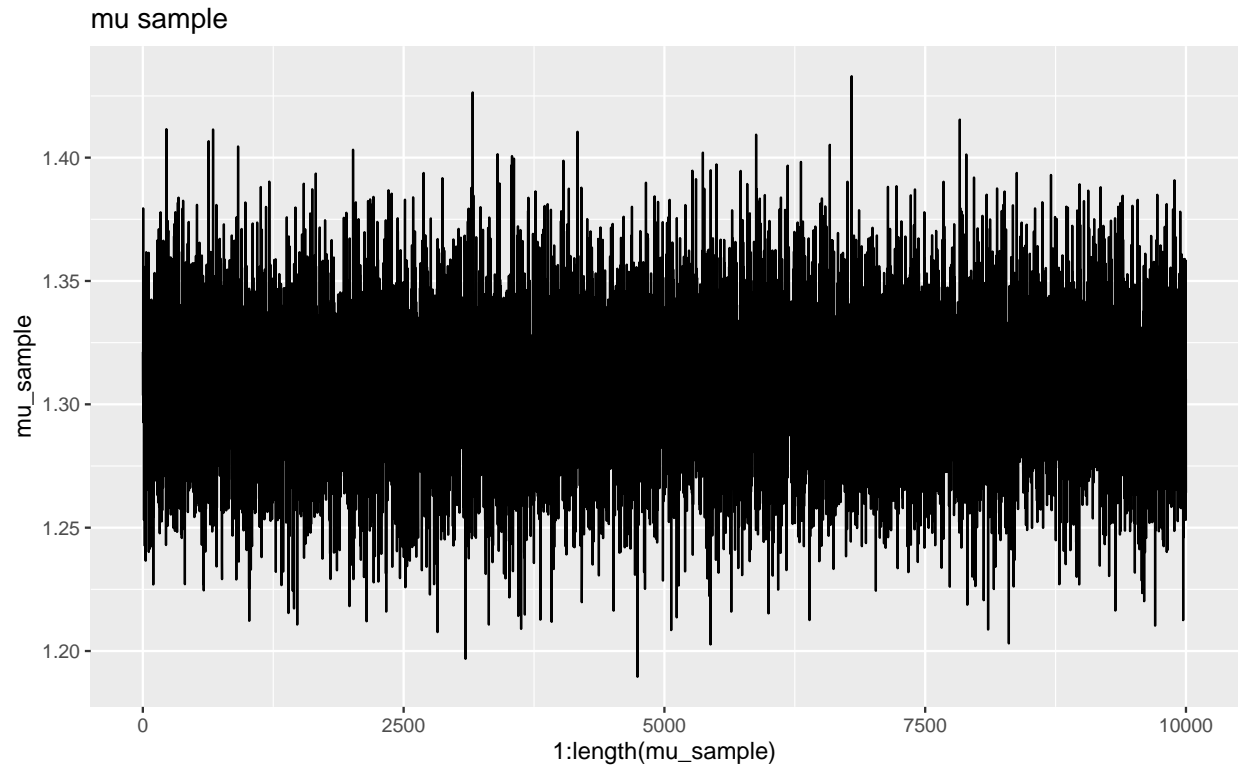
The the Inefficiency Factors (IF) of  $\mu = 0.9982076$  and IF of  $\sigma = 0.9522093$ . The value of IF is around 1 indicates that the chain has converged to the stationary distribution.

Below, we present two trace plot that also confirm the samples' convergence.

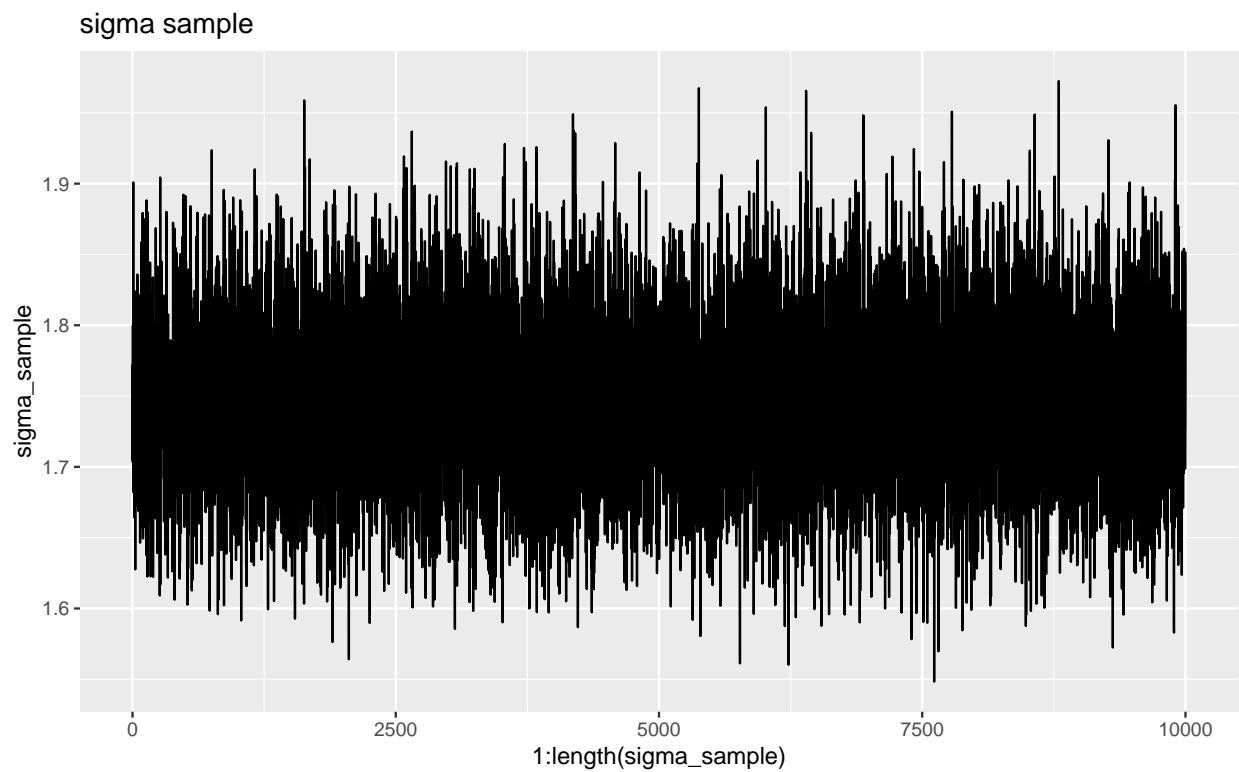
```

ggplot(data=sample_gibbs, aes(x = 1:length(mu_sample), y = mu_sample)) +
  geom_line()+labs(title = "mu sample")

```



```
ggplot(data=sample_gibbs, aes(x = 1:length(sigma_sample), y = sigma_sample)) +  
  geom_line() +labs(title = "sigma sample")
```



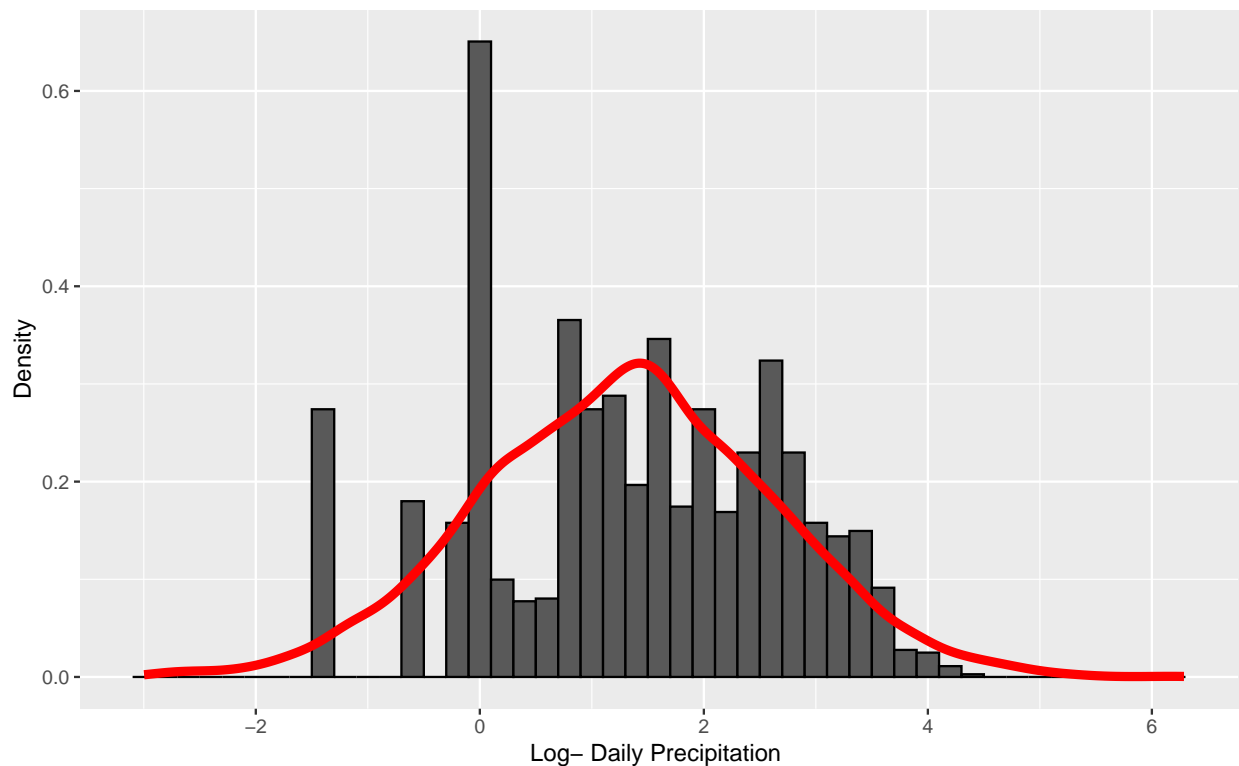
(b)

Plot the following in one figure:

- 1) a histogram or kernel density estimate of the daily precipitation
- 2) The resulting posterior predictive density using the simulated posterior draws from How well does the posterior predictive density agree with this data?

```
# Using Gibbs sample's mu and sigma to sample posterior prediction
post_pred_samples <- rnorm(n = 10000, mean = sample_gibbs$mu_sample,
                           sd = sqrt(sample_gibbs$sigma_sample))

ggplot(data = data.frame(y = log_prec), aes(x = y)) +
  geom_histogram(aes(y = after_stat(density)), color = "black", binwidth = 0.2) +
  geom_density(data = data.frame(y = post_pred_samples), aes(x = y, y = after_stat(density)),
              color = "red", linewidth = 2) +
  labs(x = "Log- Daily Precipitation ", y = "Density")
```



Above is the plot of the histogram of the daily precipitation, and the resulting posterior predictive density (red line). We can say that the posterior prediction does not totally agree with the existing data, especially towards to left-hand side of the mode.

### Q3 Time series models in Stan

(a)

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

```
#####Q3
mu <- 13
sigma_square <- 3
t <- 300
phi <- seq(from = -1 ,to = 1, by =0.25 )

ar_func <- function(phi,mu,sigma_square,t){
  counter <- 0
  result_vector <- rep(0,t)
  result_vector[1] <- mu
  for(i in 2:t){
    epsilon <- rnorm(1,0,sqrt(sigma_square))
    x_i <- mu+phi*(result_vector[i-1]-mu)+ epsilon
    result_vector[i] <- x_i
  }

  return(result_vector)
}

test_phi_func <- function(phi,mu,sigma_square,t){

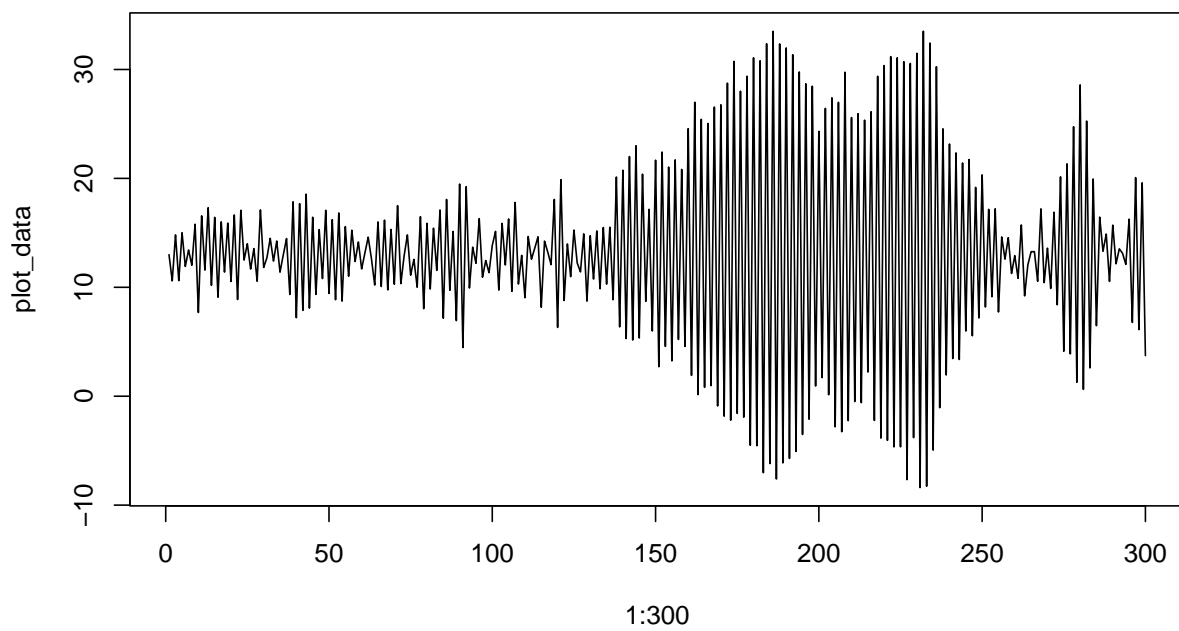
  phi_test_df <- data.frame(matrix(0, nrow = t, ncol = length(phi)))
  colnames(phi_test_df) <- phi

  for (j in 1:length(phi)) {
    phi_test <- ar_func(phi[j], mu, sigma_square, t)
    phi_test_df[, j] <- phi_test
  }
  return(phi_test_df)
}

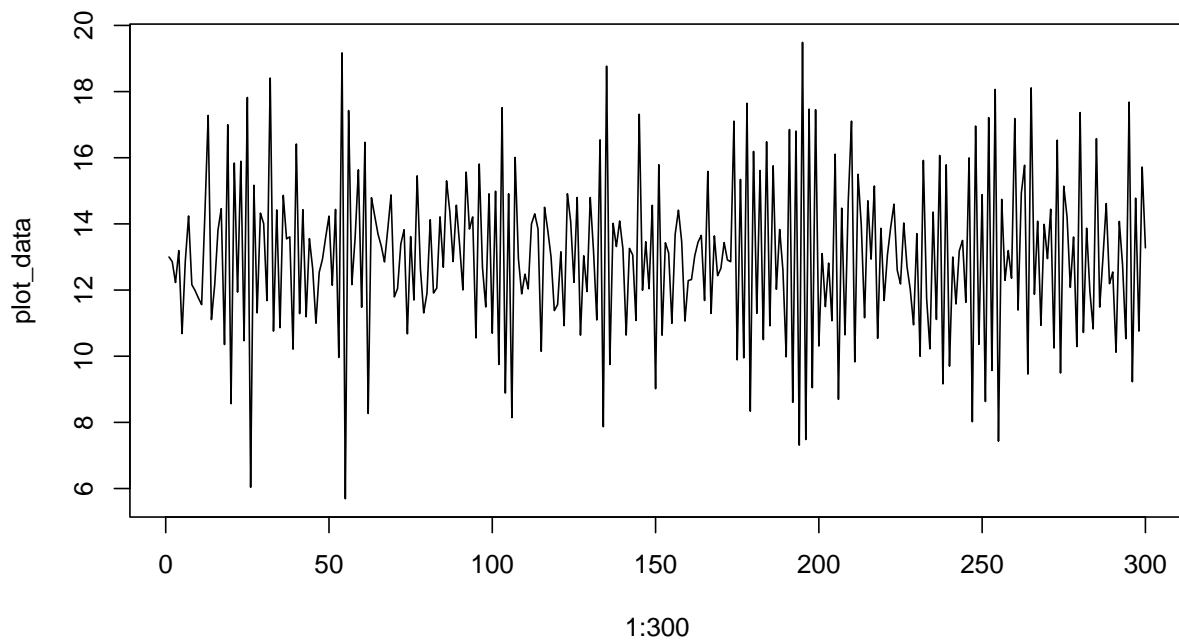
phi_df <- test_phi_func(phi,mu,sigma_square,t)

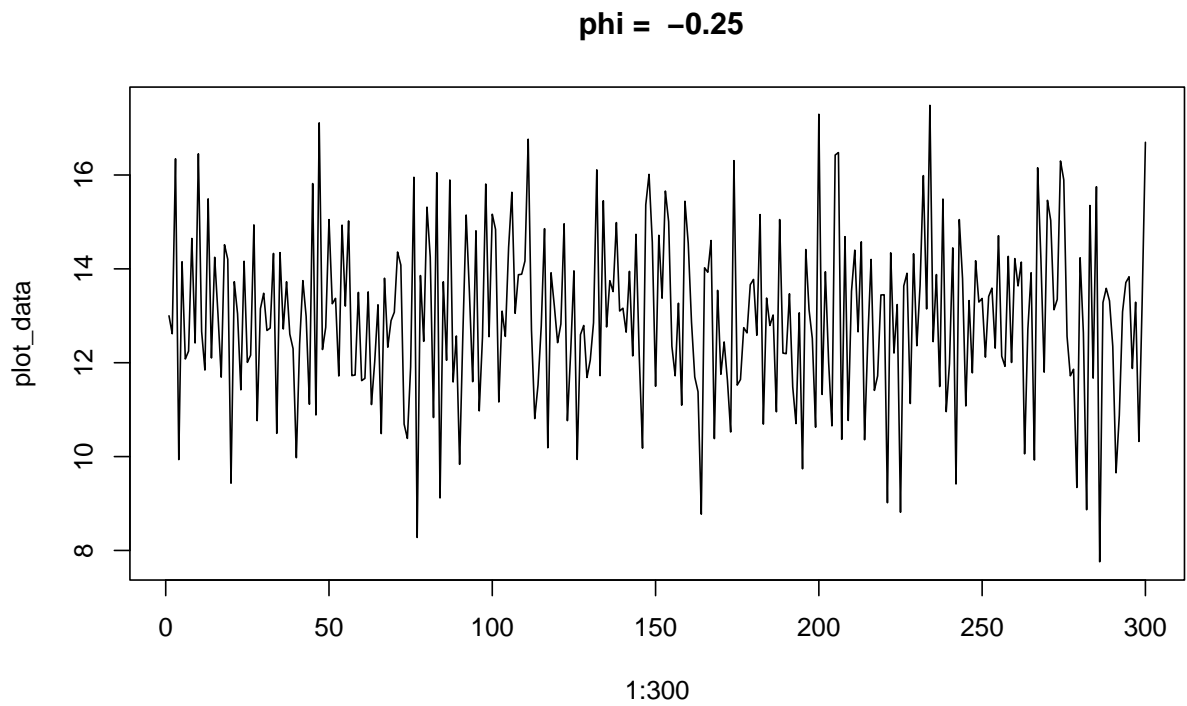
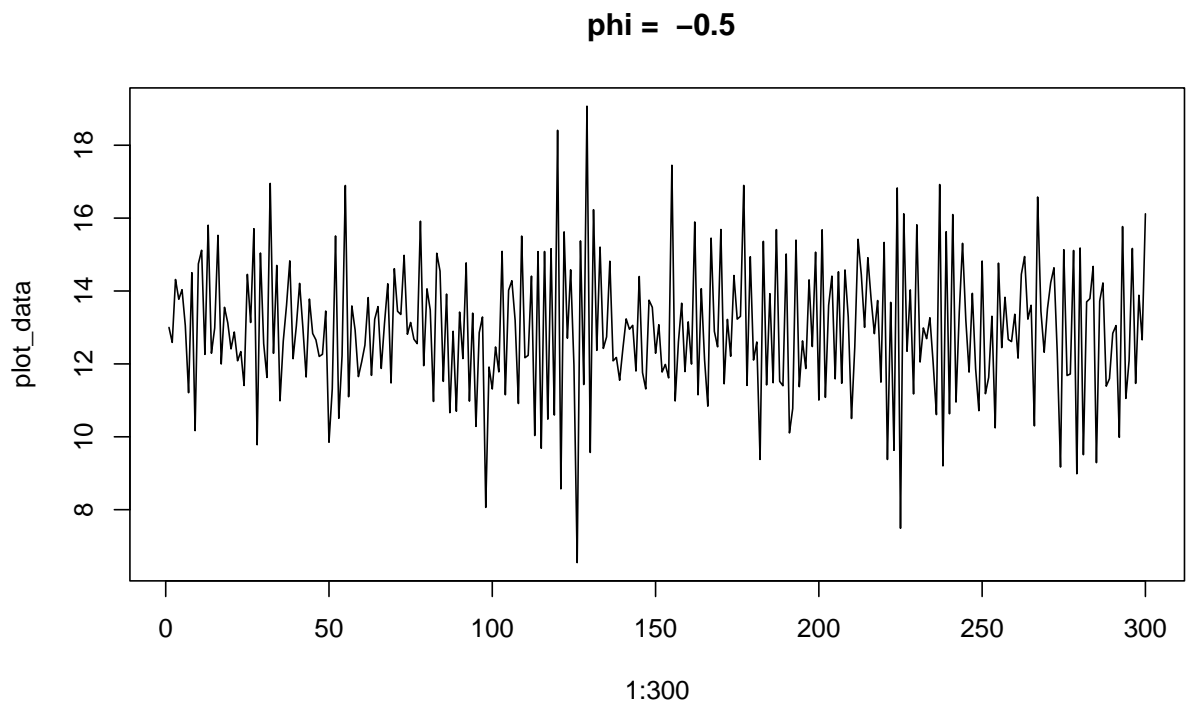
for(k in 1:length(phi)){
  plot_data <- phi_df[,k]
  plot(x=1:300, plot_data, type = "l", main = paste(" phi = ", phi[k]))
  #Sys.sleep(1)
}
```

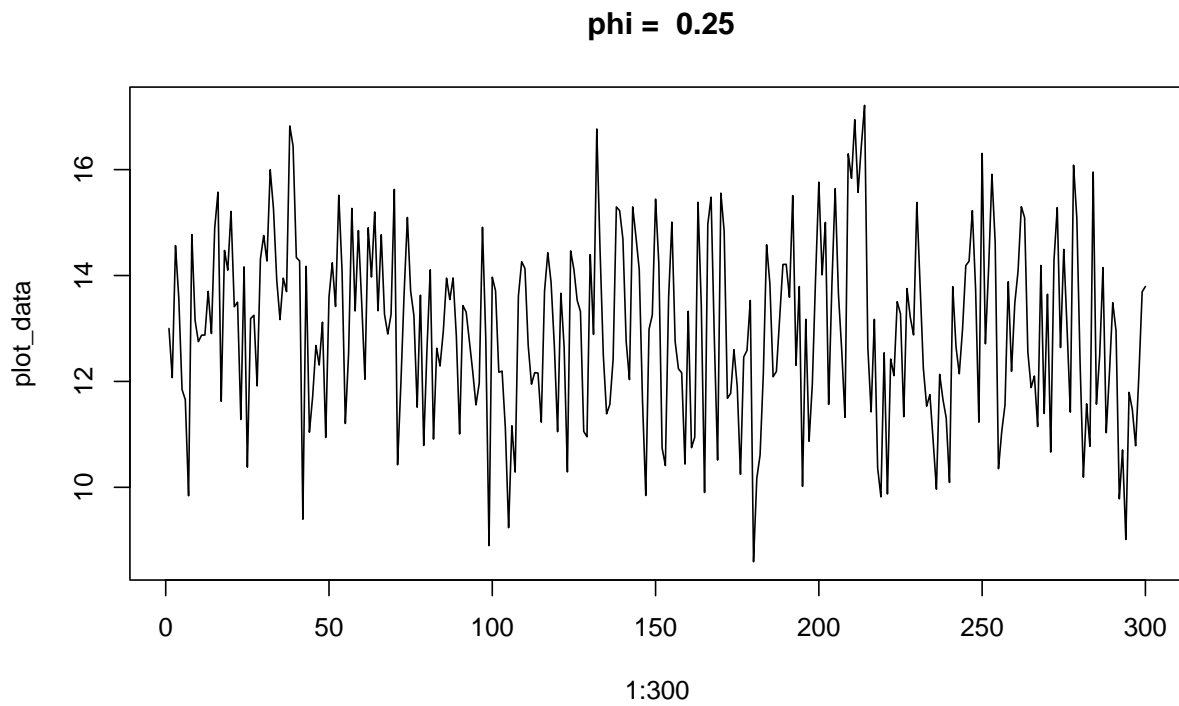
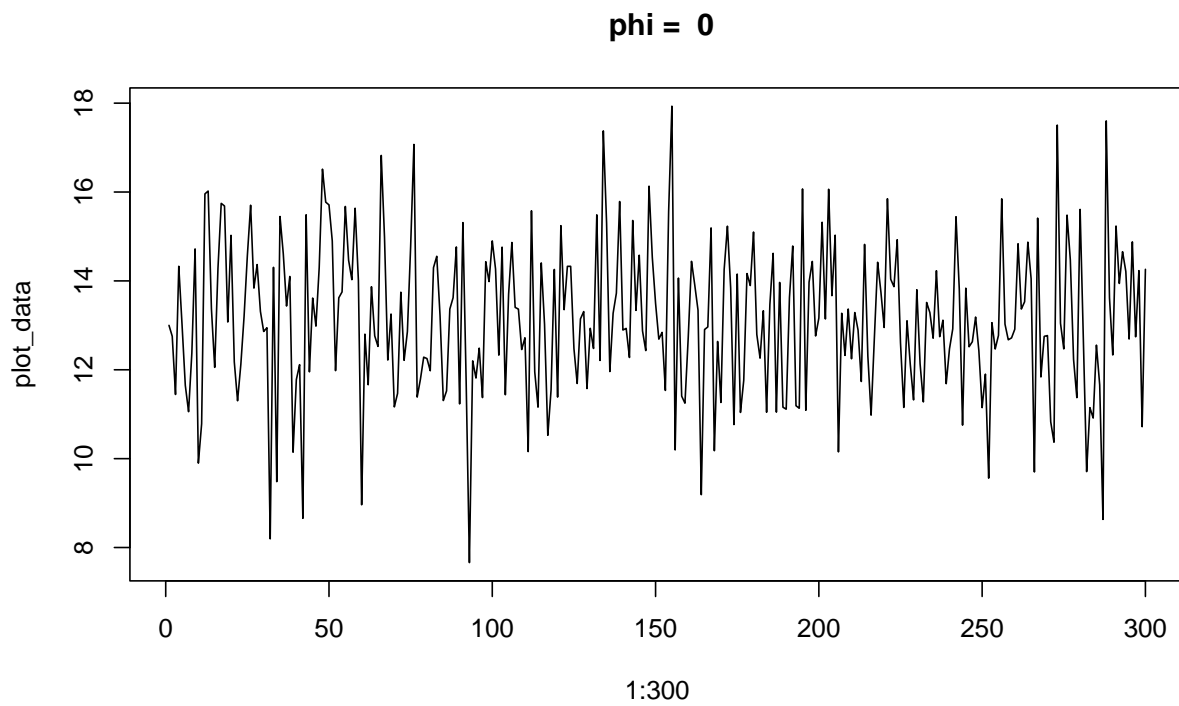
**$\phi = -1$**



**$\phi = -0.75$**

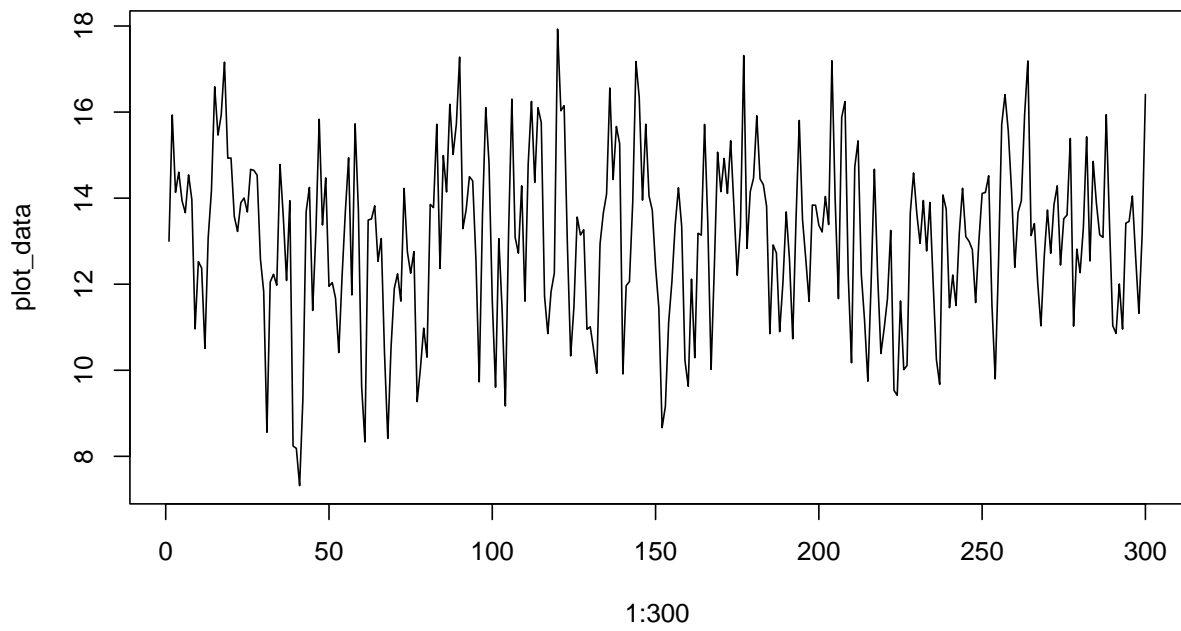




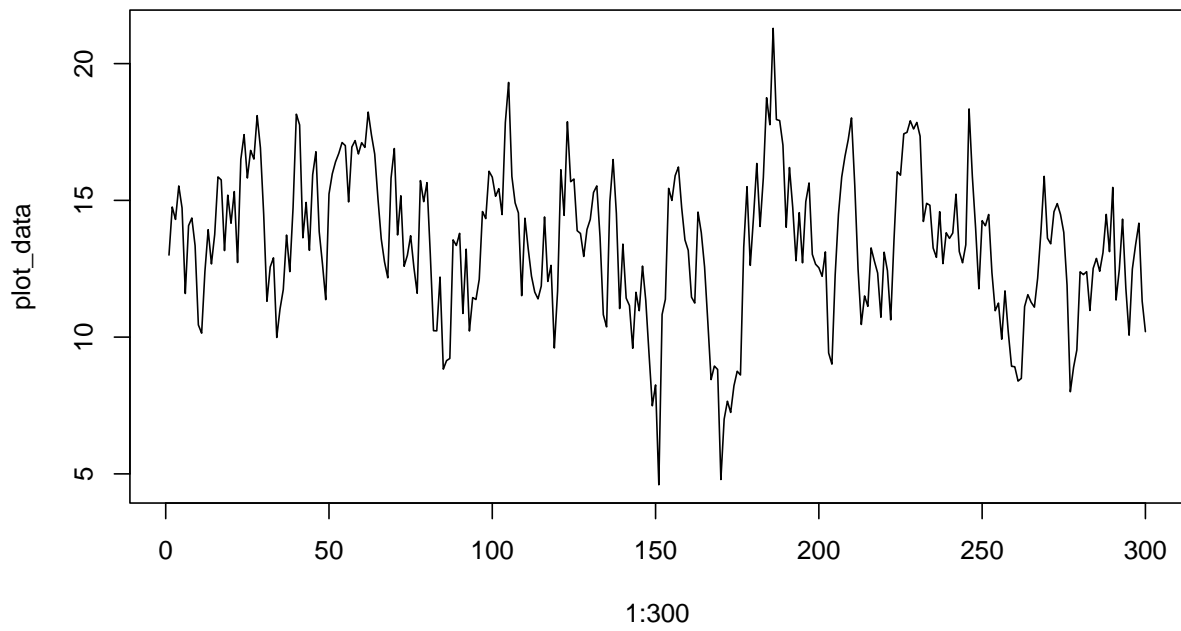


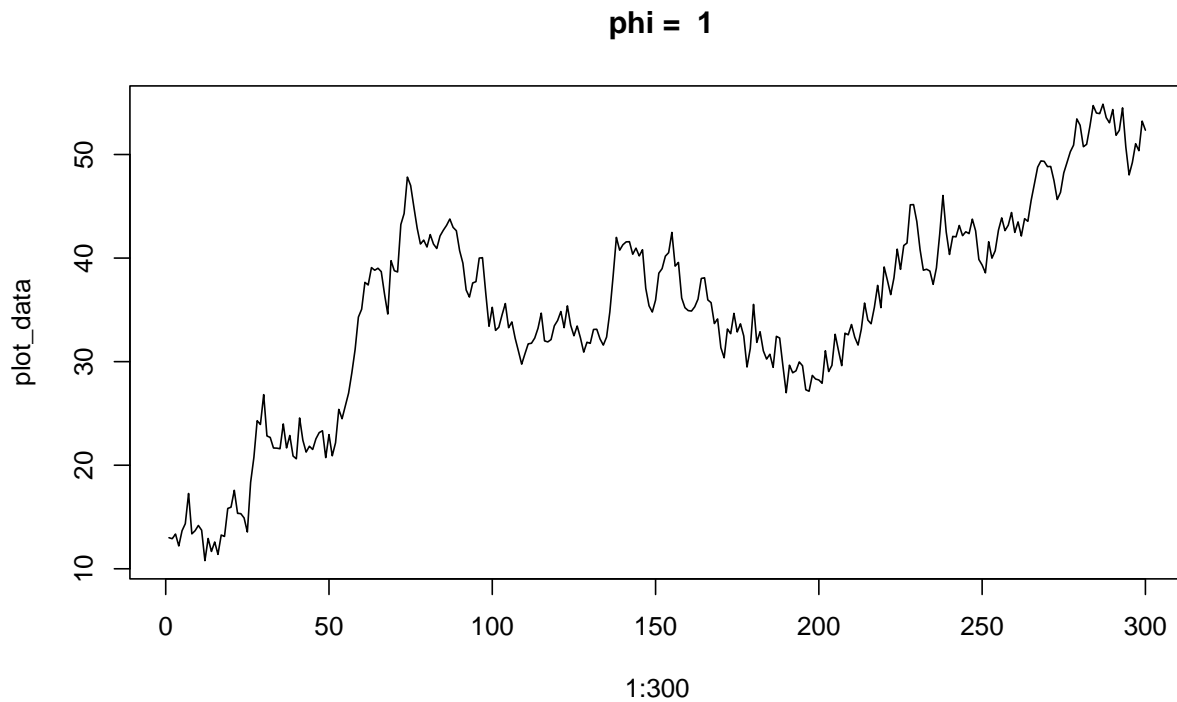


**phi = 0.5**



**phi = 0.75**





As we can observe with different  $\phi$ , as  $\phi$  increases, the gap between peaks is getting larger (not as fluctuate as lower  $\phi$ ). This indicates that the data tends to have higher autocorrelation, which means each data point is more correlated with preceding data.

(b)

Rstan

```
library(rstan)
set.seed(12345)
x <- ar_func(phi=0.2, mu, sigma_square, t)
y <- ar_func(phi=0.95, mu, sigma_square, t)

stan_code <- "
data {
  int<lower=0> T;          // Number of time points
  vector[T] x;
  vector[T] y;
}
parameters {
  real mu_x;
  real mu_y;
  real phi_x;
  real phi_y;
  real sigma_x;
  real sigma_y;
}
```

```

model {
  // After some research, it is common to use a flat prior or a vague prior as
  // non-informative prior
  // Therefore, we pick normal distribution with higher variance as prior.
  mu_x ~ normal(0, 50);
  mu_y ~ normal(0, 50);
  phi_x ~ normal(0, 10);
  phi_y ~ normal(0, 10);

  sigma_x ~ normal(0, 50);
  sigma_y ~ normal(0, 50);

  x[2:T] ~ normal(mu_x + phi_x * (x[1:(T - 1)] - mu_x), sigma_x);
  y[2:T] ~ normal(mu_y + phi_y * (y[1:(T - 1)] - mu_y), sigma_y);
}
"

data_list <- list(
  T = t,
  x = x,
  y = y
)

# Set the MCMC settings
niter <- 5000
warmup <- 500

# Compile the Stan model
model <- stan_model(model_code = stan_code)

# Fit the Stan model to the data
# Set the control to avoid too many divergent after warmup
control <- list(adapt_delta = 0.90, stepsize = 0.0001)
fit<- sampling(model, data = data_list, warmup = warmup, iter = niter, chains = 5, control=control, refresh=
#refresh = 0 to mute the printout
summary(fit)$summary

```

##	mean	se_mean	sd	2.5%	25%
## mu_x	13.1736454	0.0009810772	0.12894646	12.92120195	13.0866009
## mu_y	14.7662220	0.4216001425	9.66646731	-12.59440997	12.9360194
## phi_x	0.1886121	0.0004666932	0.05768479	0.07426121	0.1492954
## phi_y	0.9713154	0.0003255917	0.01843371	0.93472438	0.9583243
## sigma_x	1.7892259	0.0005562130	0.07331174	1.65097259	1.7390022
## sigma_y	1.7228587	0.0006095451	0.07117800	1.58966456	1.6741621
## lp__	-634.2138672	0.0264929349	1.83419194	-638.47628106	-635.3056310
##	50%	75%	97.5%	n_eff	Rhat
## mu_x	13.1734372	13.2605943	13.4247425	17274.7783	1.000210
## mu_y	15.5310107	18.0162790	34.1507642	525.6953	1.006105
## phi_x	0.1889787	0.2277933	0.2991747	15277.7597	1.000192
## phi_y	0.9713980	0.9854067	1.0021845	3205.3759	1.003303

```
## sigma_x    1.7867845    1.8374897    1.9404140 17372.5969 1.000122
## sigma_y    1.7208153    1.7689548    1.8698400 13635.7782 1.000495
## lp__      -633.9305759 -632.8203154 -631.5496630 4793.2427 1.001257
```

```
post_mean <- summary(fit)$summary[, "mean"]
interval_025 <- summary(fit)$summary[, "25%"]
interval_975 <- summary(fit)$summary[, "97.5%"]
n_eff <- summary(fit)$summary[, "n_eff"]
Rhat <- summary(fit)$summary[, "Rhat"]
```

(i) Based on the summary, we obtain:  
Posterior mean:

```
##      mu_x      mu_y      phi_x      phi_y      sigma_x      sigma_y
## 13.1736454 14.7662220 0.1886121 0.9713154 1.7892259 1.7228587
##      lp__
## -634.2138672
```

95% credible interval are  
2.5%:

```
##      mu_x      mu_y      phi_x      phi_y      sigma_x      sigma_y
## 13.0866009 12.9360194 0.1492954 0.9583243 1.7390022 1.6741621
##      lp__
## -635.3056310
```

97.5%:

```
##      mu_x      mu_y      phi_x      phi_y      sigma_x      sigma_y
## 13.4247425 34.1507642 0.2991747 1.0021845 1.9404140 1.8698400
##      lp__
## -631.5496630
```

number of effective posterior sample:

```
##      mu_x      mu_y      phi_x      phi_y      sigma_x      sigma_y      lp__
## 17274.7783 525.6953 15277.7597 3205.3759 17372.5969 13635.7782 4793.2427
```

The true values are  $\mu_x = \mu_y = 13$ ,  $\phi_y = 0.95$ ,  $\phi_x = 0.2$ ,  $\sigma_x = \sigma_y = \sqrt{3} \approx 1.73$

For posterior mean, all the inferred parameter are very close to their true value. Hence we can say that we are able to estimate the true value. In terms of 95% credible interval and number of effective posterior sample, we would say that for estimated  $\mu_x$ ,  $\phi_y$ ,  $\sigma_x$  and  $\sigma_y$  has narrower credible intervals, while estimated  $\mu_x$ ,  $\phi_x$ ,  $\sigma_x$  and  $\sigma_y$  has larger number of effective posterior samples. This indicates the estimate of these inferred parameters are more precise.

(ii) For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of  $\mu$  and  $\phi$ . Comments?

We can use Rhat to evaluate the convergence. The Rhat are as below:

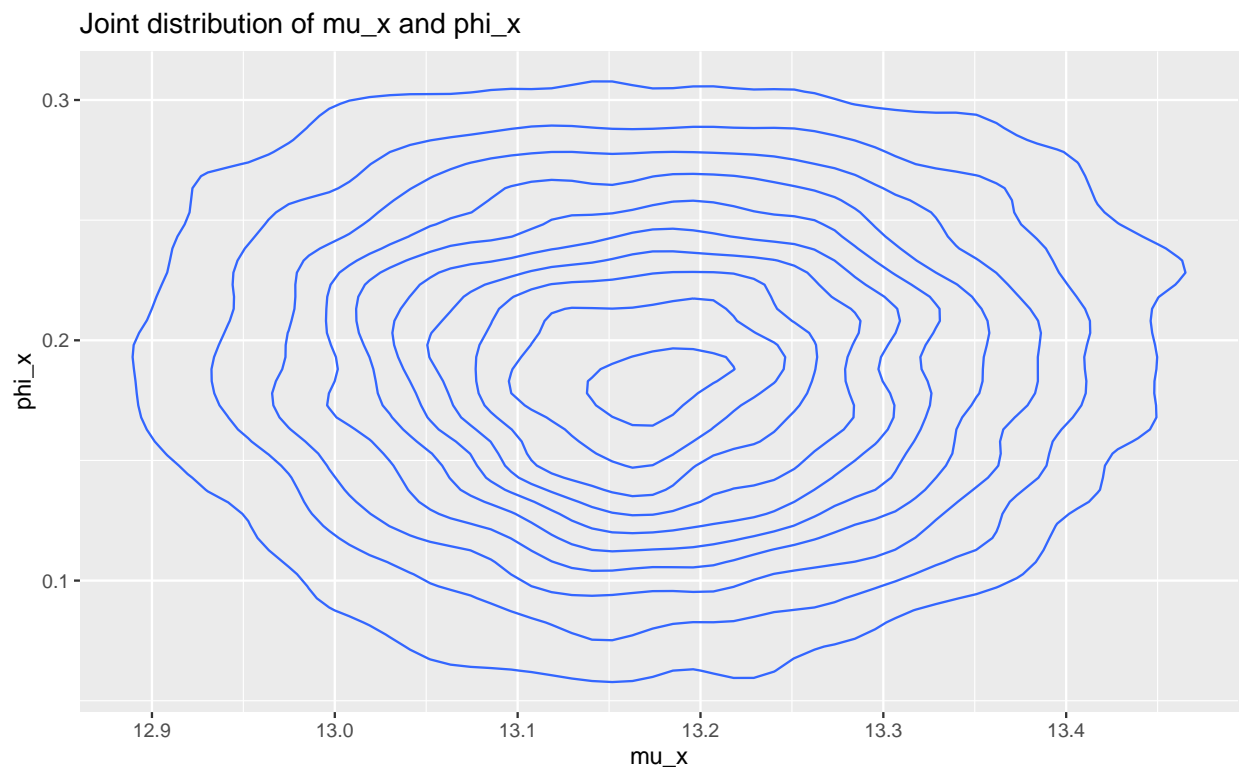
```
##      mu_x      mu_y      phi_x      phi_y sigma_x sigma_y      lp__
## 1.000210 1.006105 1.000192 1.003303 1.000122 1.000495 1.001257
```

Since all inferred parameters has Rhat close to 1, this imply our sampler converged well.

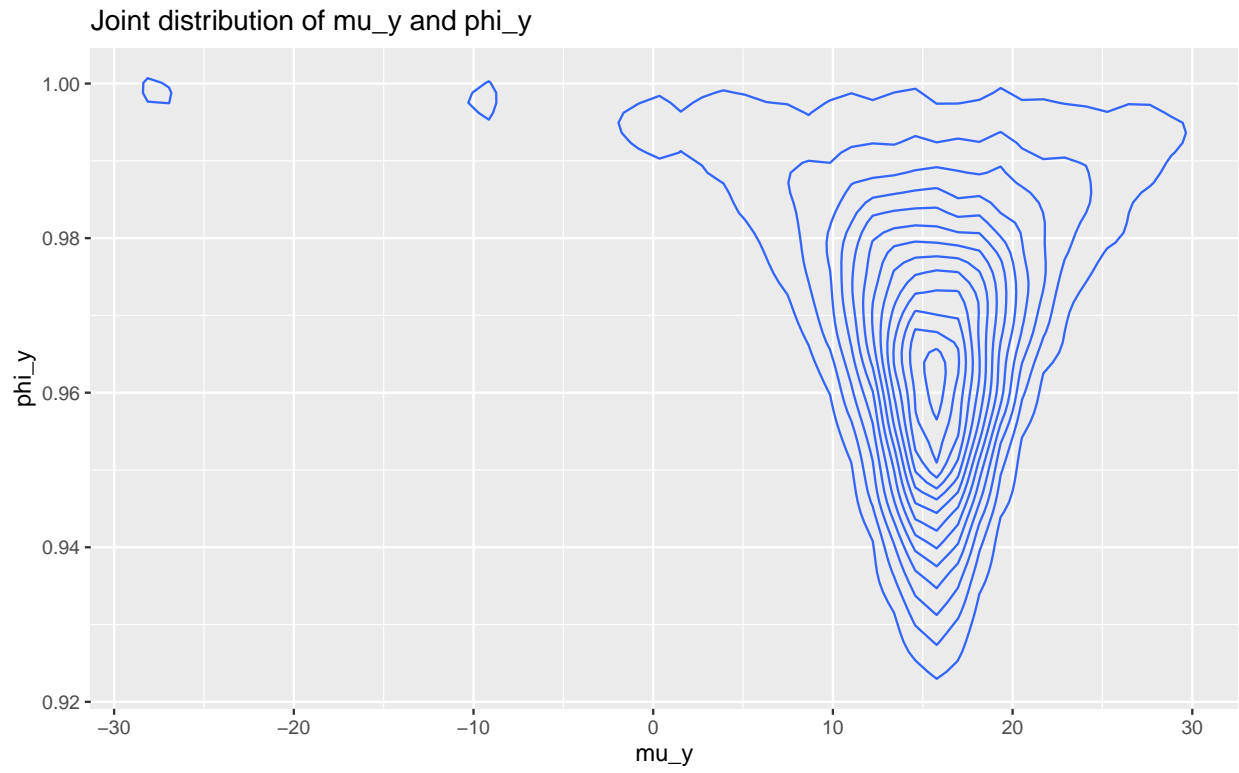
Below are the plot of the joint posterior of  $\mu$  and  $\phi$

For x, the shape of joint distribution are rather symmetric along two axis, this means  $\mu$  and  $\phi$  has higher correlation. For y, the different shape means, mu and phi has lower correlation hence make it harder to understand the relationship.

```
posterior_x <- extract(fit, pars = c("mu_x", "phi_x"))
posterior_df_x <- data.frame(posterior_x)
ggplot(data = posterior_df_x, aes(x = mu_x, y = phi_x)) +
  stat_density_2d() +
  xlab("mu_x") +
  ylab("phi_x") +
  ggtitle("Joint distribution of mu_x and phi_x")
```



```
posterior_y <- extract(fit, pars = c("mu_y", "phi_y"))
posterior_df_y <- data.frame(posterior_y)
ggplot(data = posterior_df_y, aes(x = mu_y, y = phi_y)) +
  stat_density_2d() +
  xlab("mu_y") +
  ylab("phi_y") +
  ggtitle("Joint distribution of mu_y and phi_y")
```



## Appendix

```
rm(list=ls())
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(fig.align = "center")
knitr::opts_chunk$set(fig.width=8, fig.height=5)
knitr::opts_chunk$set(warning=FALSE)

set.seed(12345)
#####Q1
library(ggplot2)
library(mvtnorm)
#library(extraDistr)
precipitation <- readRDS("Precipitation.rds")
log_prec <- log(precipitation)

mu_post <- function(mu_0,tau_0_sqr,sigma_square_current,y,n){

  tau_n_sqr = 1 / (1/tau_0_sqr + n/sigma_square_current)
  mu_n <- tau_n_sqr * (mu_0/tau_0_sqr + sum(y)/sigma_square_current)
  mupost <- rnorm(1, mu_n, sqrt(tau_n_sqr))
  return(mupost)

}

sigma_square_post <- function(mu_current, v_0, sigma_square_0, y,n, use_myinvchi=TRUE) {

  v_n <- v_0 + n
  elem1 <- v_0*sigma_square_0
  elem2 <- sum((y - mu_current)^2)
  elem3 <- n+v_0
  elem_comb <- (elem1+elem2)/elem3
  if (use_myinvchi){
    sigma_square_post <- my_inv_chi(v_n,elem_comb)
  }
  else {
    sigma_square_post <- rinvchisq(1, v_n, elem_comb) #Requires package extraDistr
  }

  return(sigma_square_post)
}

my_inv_chi<- function(df,tau_sqr) {
  X <- rchisq(1,df)
  inv_chi <- (df*tau_sqr)/X
  return(inv_chi)
}

#init
mu_0 <- 0
tau_0_sqr <- 1
```

```

sigma_square_0 <- 1
v_0 <- 1 #degree of freedom for chi square

gibbs_sampler <- function(nstep, data, mu_0, tau_0_sqr, v_0, sigma_square_0) {
  # Init parameters
  mu_current <- 0
  sigma_square_current <- 1

  mu_samples <- rep(0,nstep)
  sigma_square_samples <- rep(0,nstep)

  for (i in 1:nstep) {

    mu_current <- mu_post(mu_0, tau_0_sqr, sigma_square_current, y=data, length(data))
    #print(mu_current)
    sigma_square_current <- sigma_square_post(mu_current, v_0, sigma_square_0, y=data,
                                              length(data),use_myinvchi=TRUE)
    #print(sigma_square_current)

    mu_samples[i] <- mu_current
    sigma_square_samples[i] <- sigma_square_current
  }

  output_df <- data.frame(mu_sample = mu_samples, sigma_sample = sigma_square_samples)
  return(output_df)
}

sample_gibbs <- gibbs_sampler(nstep=10000, data=log_prec, mu_0, tau_0_sqr, v_0, sigma_square_0)
my_acf <- acf(sample_gibbs$mu_sample,plot = F) # getting the autocorrelation
if_mu <- 1 + 2 * sum(my_acf$acf[-1])
my_acf <- acf(sample_gibbs$sigma_sample,plot = F)
if_sigma <- 1 + 2 * sum(my_acf$acf[-1])
ggplot(data=sample_gibbs, aes(x = 1:length(mu_sample), y = mu_sample)) +
  geom_line()+labs(title = "mu sample")

ggplot(data=sample_gibbs, aes(x = 1:length(sigma_sample), y = sigma_sample)) +
  geom_line() +labs(title = "sigma sample")
# Using Gibbs sample's mu and sigma to sample posterior prediction
post_pred_samples <- rnorm(n = 10000, mean = sample_gibbs$mu_sample,
                          sd = sqrt(sample_gibbs$sigma_sample))

ggplot(data = data.frame(y = log_prec), aes(x = y)) +
  geom_histogram(aes(y =after_stat(density)), color = "black", binwidth = 0.2) +
  geom_density(data = data.frame(y = post_pred_samples), aes(x = y, y = after_stat(density)),
              color = "red",linewidth = 2) +
  labs(x = "Log- Daily Precipitation ", y = "Density")

#####Q3
mu <- 13
sigma_square <- 3
t <- 300
phi <- seq(from = -1 ,to = 1, by =0.25 )

```



```

ar_func <- function(phi,mu,sigma_square,t){
  counter <- 0
  result_vector <- rep(0,t)
  result_vector[1] <- mu
  for(i in 2:t){
    epsilon <- rnorm(1,0,sqrt(sigma_square))
    x_i <- mu+phi*(result_vector[i-1]-mu)+ epsilon
    result_vector[i] <- x_i
  }

  return(result_vector)
}

test_phi_func <- function(phi,mu,sigma_square,t){

  phi_test_df <- data.frame(matrix(0, nrow = t, ncol = length(phi)))
  colnames(phi_test_df) <- phi

  for (j in 1:length(phi)) {
    phi_test <- ar_func(phi[j], mu, sigma_square, t)
    phi_test_df[, j] <- phi_test
  }
  return(phi_test_df)
}

phi_df <- test_phi_func(phi,mu,sigma_square,t)

for(k in 1:length(phi)){
  plot_data <- phi_df[,k]
  plot(x=1:300, plot_data, type = "l", main = paste(" phi = ", phi[k]))
  #Sys.sleep(1)
}
library(rstan)
set.seed(12345)
x <- ar_func(phi=0.2, mu, sigma_square, t)
y <- ar_func(phi=0.95, mu, sigma_square, t)

stan_code <- "
data {
  int<lower=0> T;          // Number of time points
  vector[T] x;
  vector[T] y;
}
parameters {
  real mu_x;
  real mu_y;
  real phi_x;
  real phi_y;
  real sigma_x;
  real sigma_y;
}
model {

```

```

// After some research, it is common to use a flat prior or a vague prior as
// non-informative prior
// Therefore, we pick normal distribution with higher variance as prior.
mu_x ~ normal(0, 50);
mu_y ~ normal(0, 50);
phi_x ~ normal(0, 10);
phi_y ~ normal(0, 10);

sigma_x ~ normal(0, 50);
sigma_y ~ normal(0, 50);

x[2:T] ~ normal(mu_x + phi_x * (x[1:(T - 1)] - mu_x), sigma_x);
y[2:T] ~ normal(mu_y + phi_y * (y[1:(T - 1)] - mu_y), sigma_y);
}
"

data_list <- list(
  T = t,
  x = x,
  y = y
)

# Set the MCMC settings
niter <- 5000
warmup <- 500

# Compile the Stan model
model <- stan_model(model_code = stan_code)

# Fit the Stan model to the data
# Set the control to avoid too many divergent after warmup
control <- list(adapt_delta = 0.90, stepsize = 0.0001)
fit <- sampling(model, data = data_list, warmup = warmup, iter = niter, chains = 5, control = control, refresh = 0)
#refresh = 0 to mute the printout
summary(fit)$summary

post_mean <- summary(fit)$summary[, "mean"]
interval_025 <- summary(fit)$summary[, "25%"]
interval_975 <- summary(fit)$summary[, "97.5%"]
n_eff <- summary(fit)$summary[, "n_eff"]
Rhat <- summary(fit)$summary[, "Rhat"]
post_mean
interval_025
interval_975
n_eff
Rhat
posterior_x <- extract(fit, pars = c("mu_x", "phi_x"))
posterior_df_x <- data.frame(posterior_x)
ggplot(data = posterior_df_x, aes(x = mu_x, y = phi_x)) +
  stat_density_2d() +

```

```
xlab("mu_x") +  
ylab("phi_x") +  
ggtitle("Joint distribution of mu_x and phi_x")  
  
posterior_y <- extract(fit, pars = c("mu_y", "phi_y"))  
posterior_df_y <- data.frame(posterior_y)  
ggplot(data = posterior_df_y, aes(x = mu_y, y = phi_y)) +  
  stat_density_2d() +  
  xlab("mu_y") +  
  ylab("phi_y") +  
  ggtitle("Joint distribution of mu_y and phi_y")
```