Lab4

Naveen Gabriel(navga709) Sridhar Adhikarla(sriad858)

2019-05-29

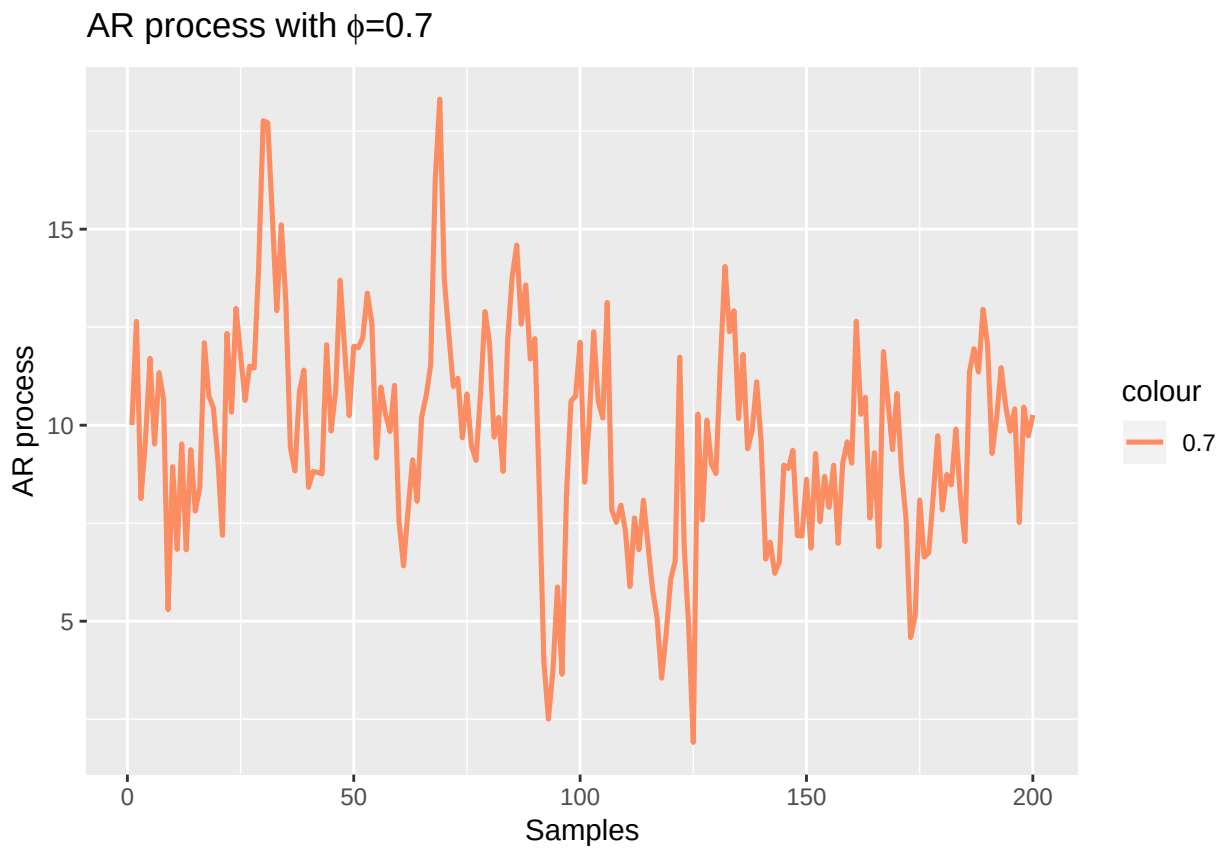
Contents

1 Time series models in Stan

Question 1A : Function in R that simulates data from the AR(1)-process

Given AR process :

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



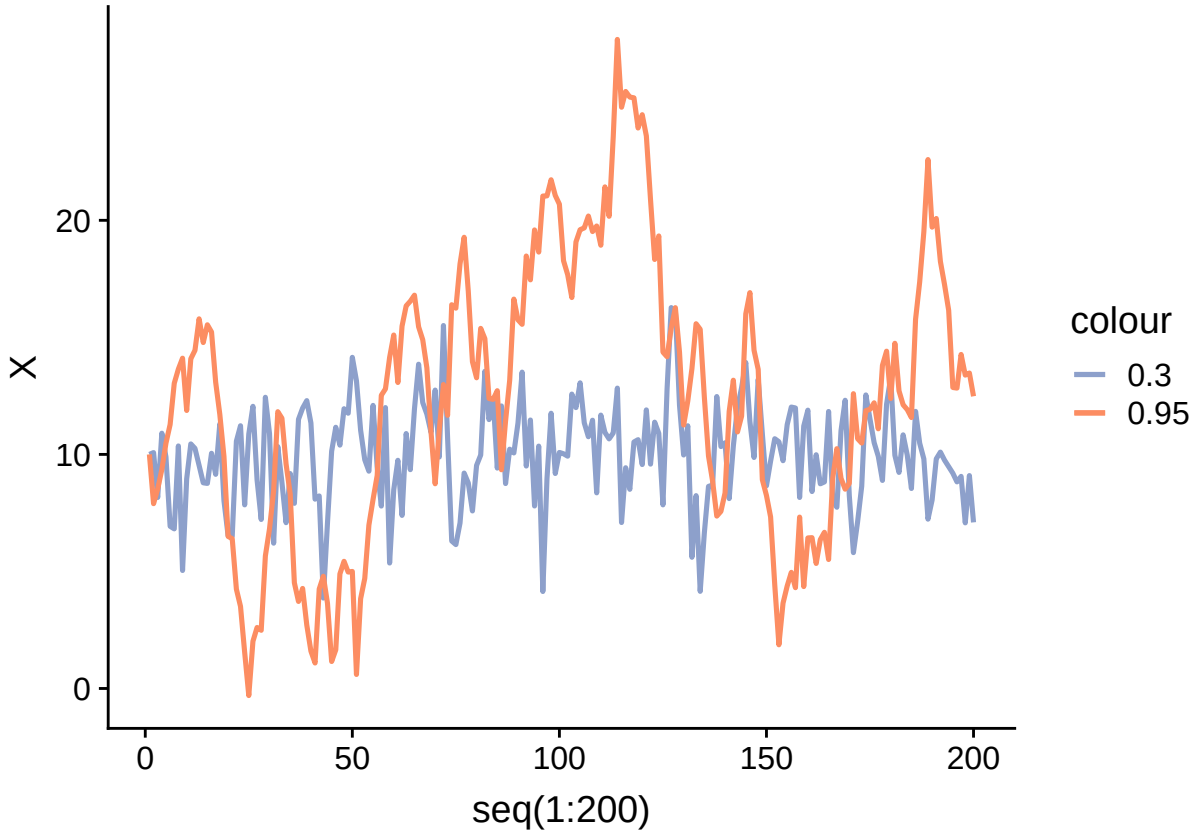
Different ϕ has different effect on AR process. For -1 the AR process oscillates the most going over point again and again. As the value of phi increases towards 1 the AR process oscillates decreases and goes through varied different points. We have chosen *phi* as 0.7 for further analysis

Question 1B : Stan Model using non-informative prior on X and Y

Compiling the model and generating data required

```
#1B
stan_code1 = "data {
  int<lower=0> T;
  vector[T] y;
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(0.00001);

  y[2:T] ~ normal(alpha + beta*y[1:(T-1)], sigma);
}
"
#compile model
model <- stan_model(model_code = stan_code1)
```

This is the AR1 model implemented in stan. The priors used for this model are non-informative as it is almost flat in the space of the parameter. All the possible values for the parameter are equally likely as we have no prior knowledge about the data.



This plot shows plots for the data generated for this model, X and Y.

We sample 4 chains of size 1000 from the model for each of the data X and Y. The burn-in preiod is 500, so the first 500 samples will be discarded by the sampler.

According to the AR1 model

beta = phi, and

alpha = mu * (1 - beta)

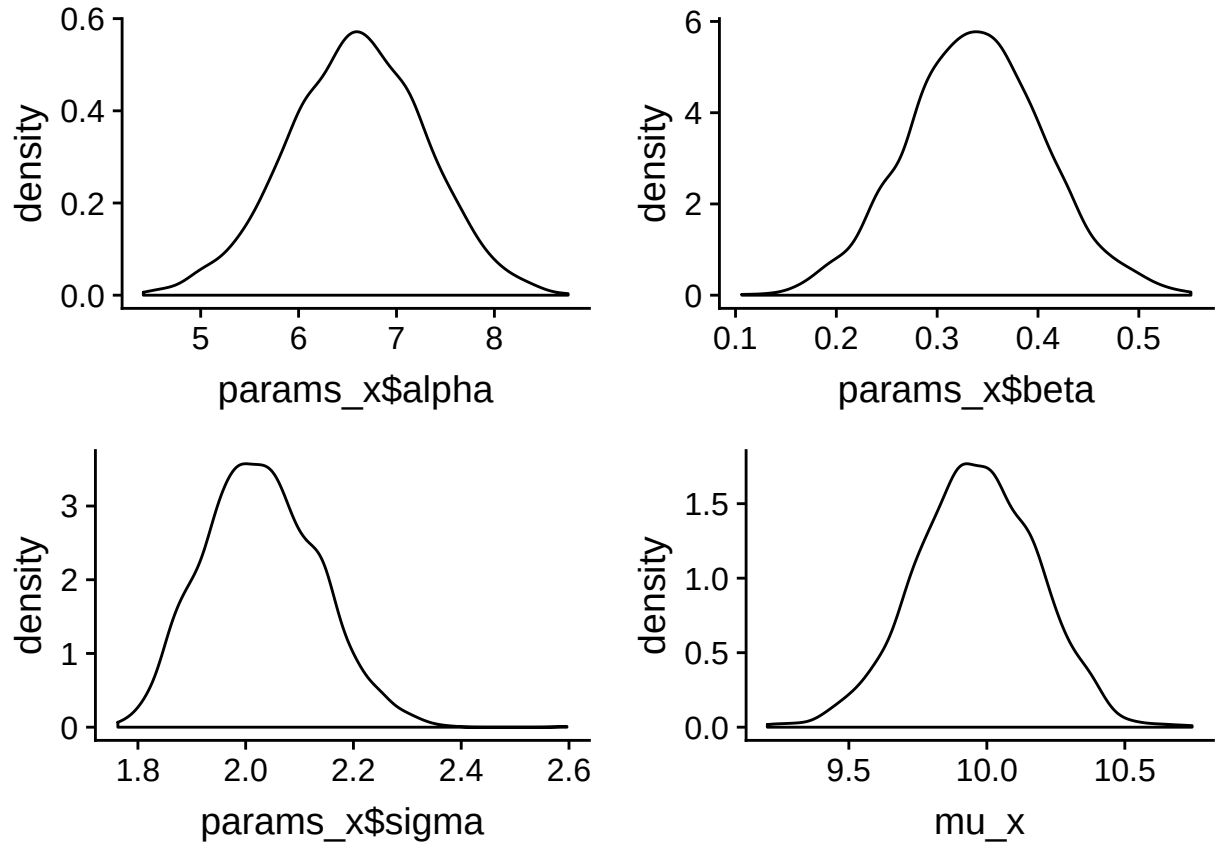
This is the reason we recompute mu from alpha using the formula mu = alpha / (1 - beta)

Summary from model fit on data X

```
## Mean for mu : 9.962597
```

```
## [1] "95% confidence interval for mu"
```

```
##      2.5%     97.5%
##  9.517727 10.379458
```




```
## [1] "95% confidence intervals for alpha, beta, sigma"
```

```
##         mean   2.5%  97.5% n_eff
## alpha 6.5930 5.1651 7.9403 720.0
## beta  0.3383 0.2023 0.4778 721.1
## sigma 2.0289 1.8453 2.2455 956.0
```

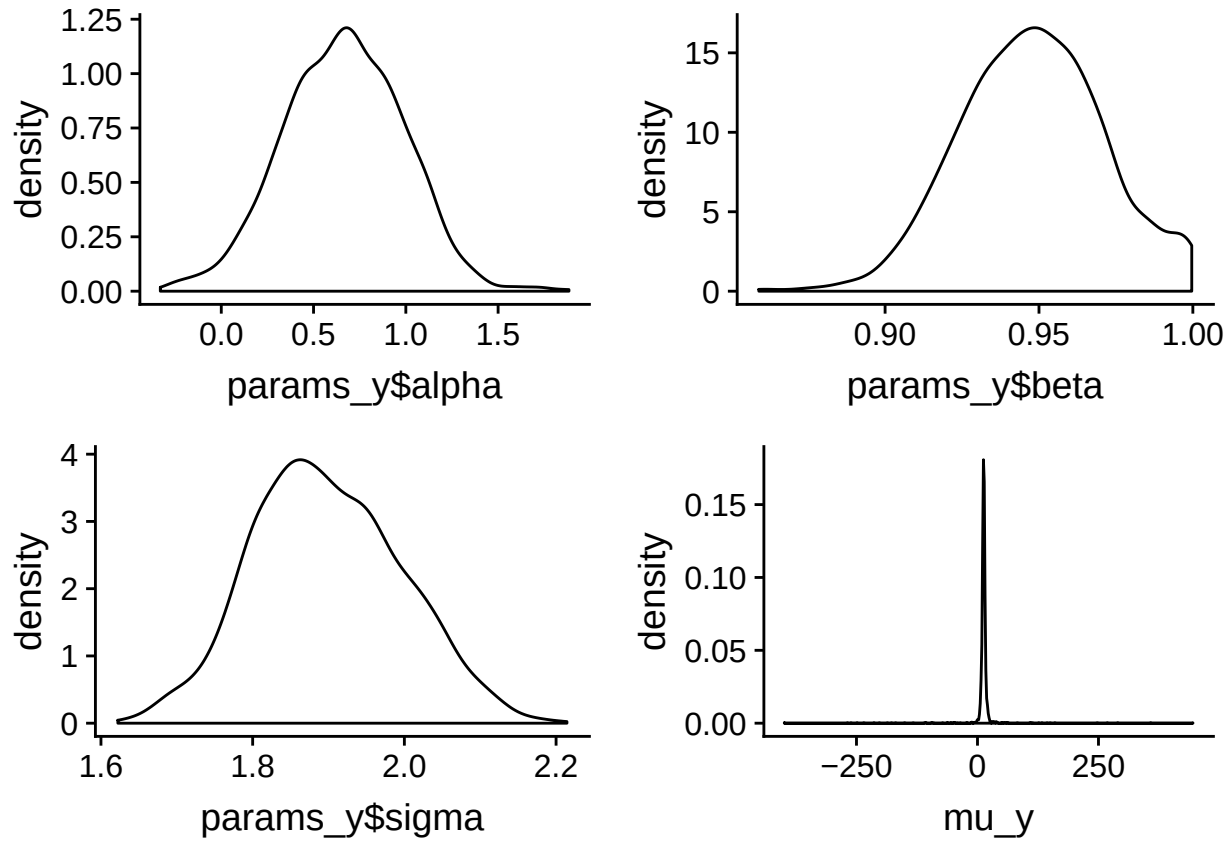
The estimated parameters are close to the original parameters. The estimated value for mu is almost exactly the same as the true value. The true values for all the parameters lie in the 95% credible interval calculated for that parameter.

Summary from model fit on data Y

```
## Mean for mu : 12.2213
```

```
## [1] "95% confidence interval for mu"
```

```
##      2.5%     97.5%
##  1.443359 23.446087
```



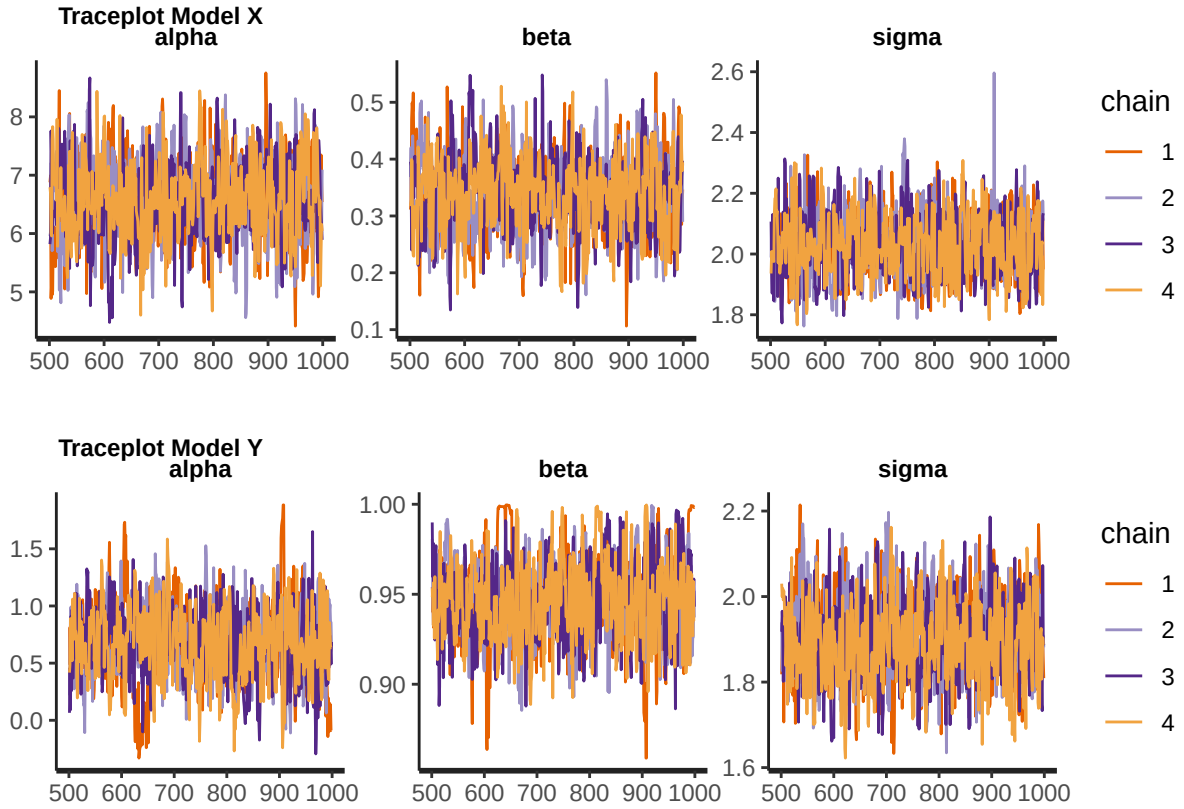

```
## [1] "95% confidence intervals for alpha, beta, sigma"
```

```
##         mean    2.5%  97.5% n_eff
## alpha 0.6639 0.02273 1.2730 456.5
## beta  0.9479 0.90268 0.9966 455.2
## sigma 1.8983 1.71097 2.0944 902.5
```

The estimated parameters are close to the original parameters. The estimated value for mu is a little deviated from the true value. The true values for all the parameters still lie in the 95% credible interval calculated for that parameter.

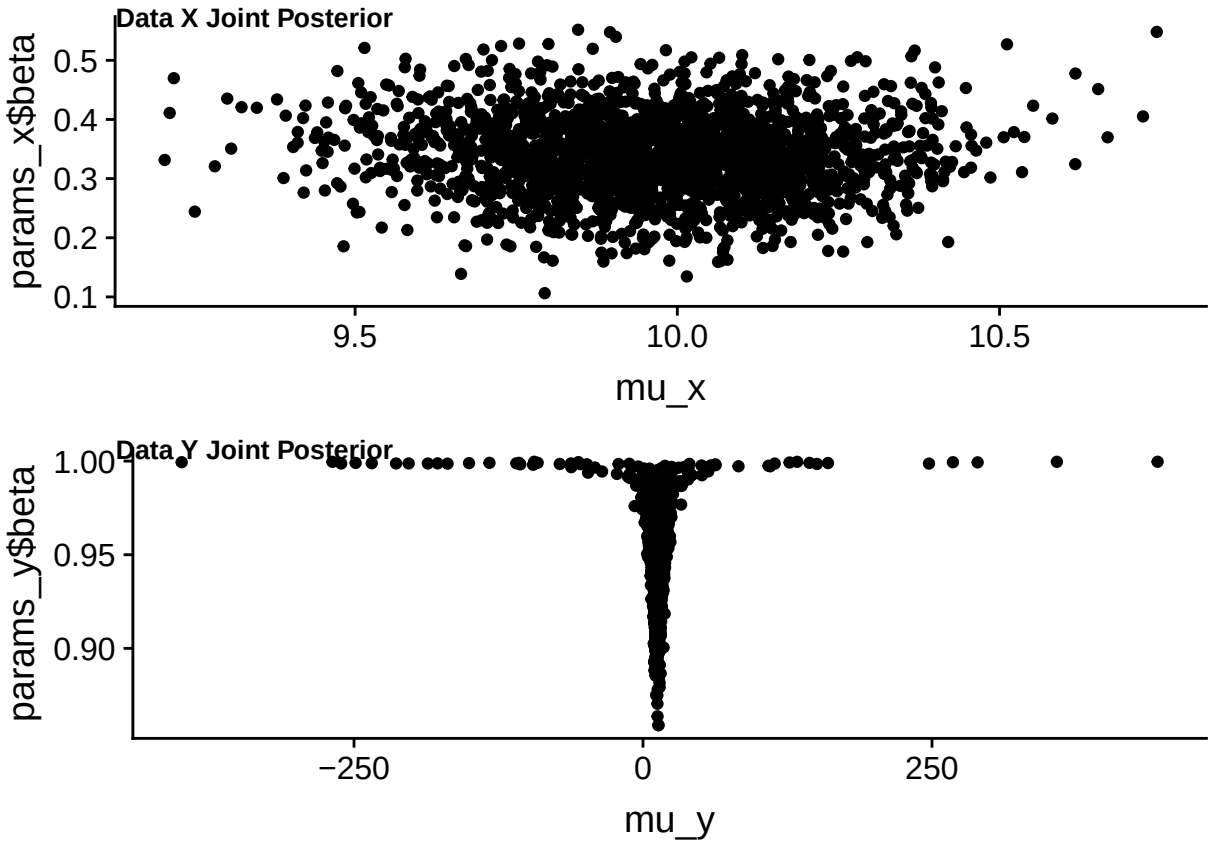
The effective sample size is large for all the simulated parameters, which is good showing that the parameters are sampled from a stationary distributions.

Traceplots for the fit models



We can see from the trace plots that the parameters for both the data models have reached convergence.

Joint Posterior for mu and phi



From the plot for the joint posterior of mu and phi we can see that the model fit on data X has reached convergence as the values are randomly distributed in the samples. But in the second model, fit on data Y, it looks like it requires some more burn in period. phi looks to have reached convergence but mu has a lot of deviation in it. Simulations for phi for both the models are centered near the true value for the model, which is good.

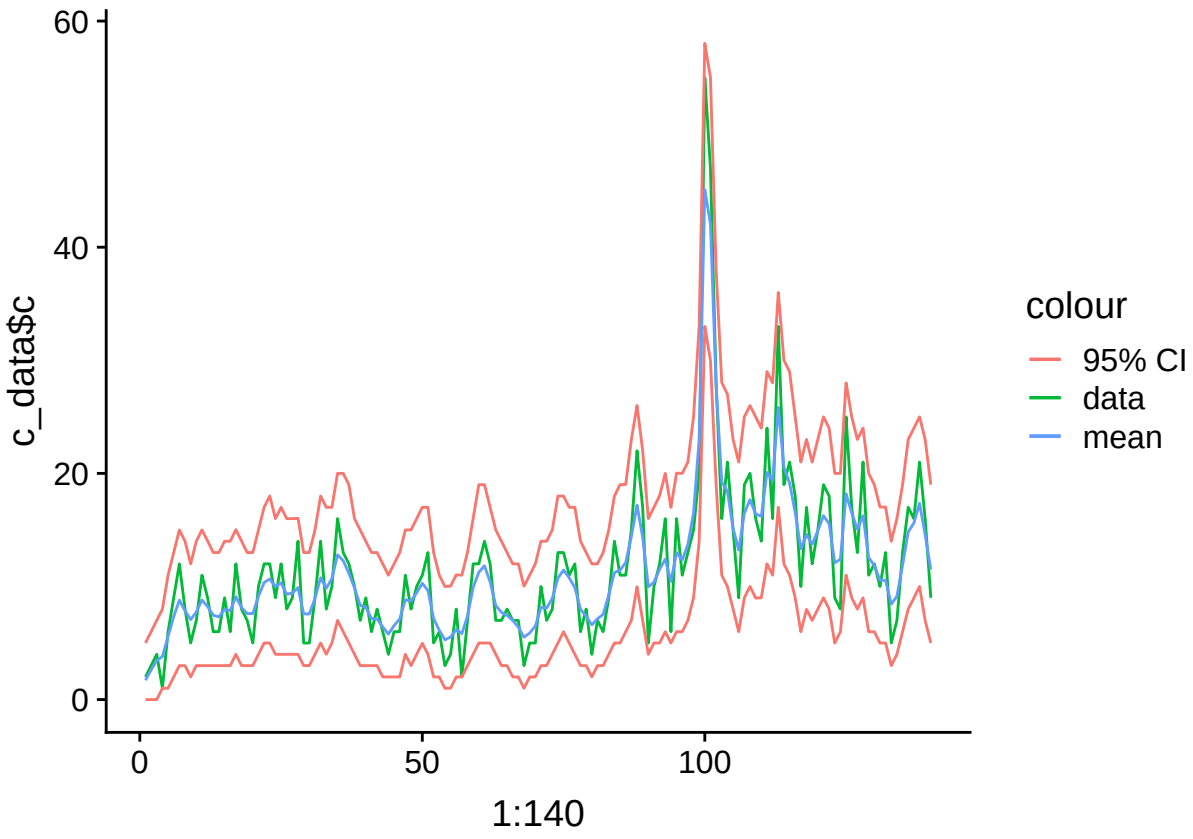
Question 1C : Stan Model using non-informative prior on campy data

```
stan_code2 = "data {
  int<lower=0> T;
  int c[T];
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
  vector[T] x;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(0.00001);

  x[2:T] ~ normal(alpha + beta*x[1:(T-1)], sigma);
  c ~ poisson(exp(x));
}
"
model <- stan_model(model_code = stan_code2)

c_data <- as.data.frame(read.table("data.txt", header = T))
#hist(c_data$c)
T = length(c_data$c)
```

We are reading the campy data, and creating the stan model for the problem. We are then compiling the model and storing into the variable model. The priors used for this model are non-informative as it is almost flat in the space of the parameter. All the possible values for the parameter are equally likely as we have no prior knowledge about the data.

We are then sampling from this compiled model. We are taking 2 markov chains of 10000 simulations from this posterior with a burnin period of 1000.

Summary from the model fit on campy data



Here we calculate the confidence interval for the parameter theta.

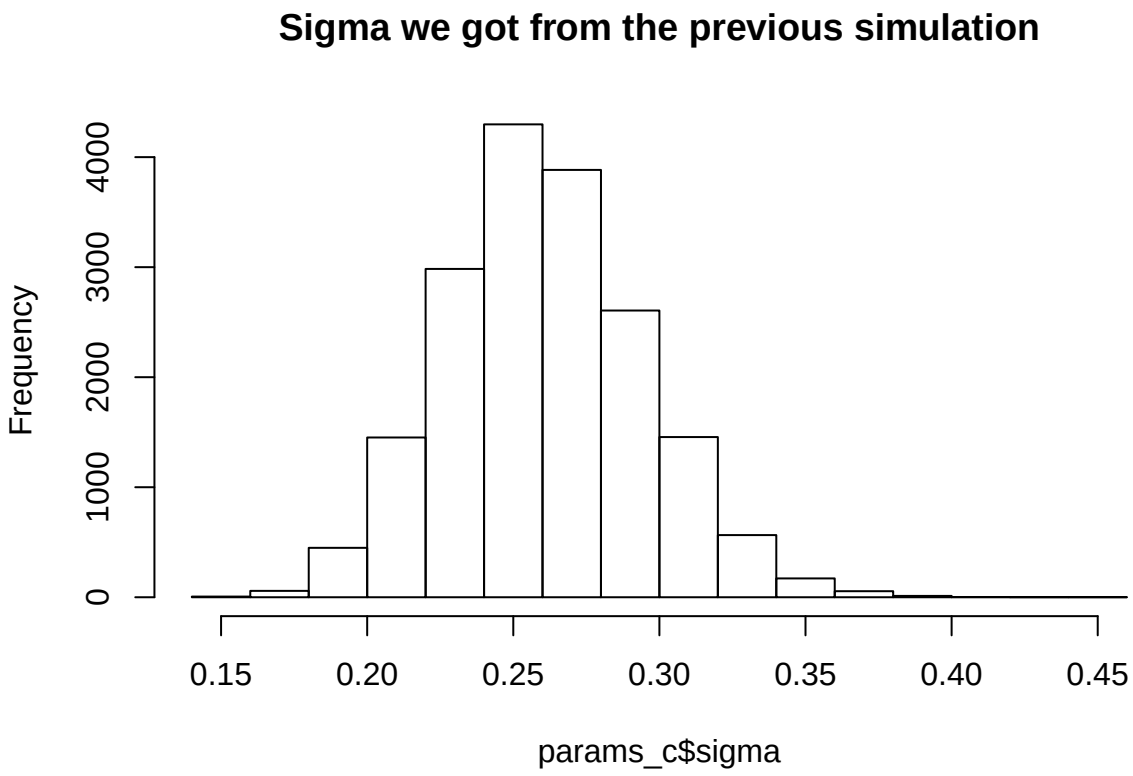
theta[t] = exp(x[t])

For the calculation of the confidence interval, we simulate from the possion distribution using the theta values. We take 1000 randomly generated values from the poission distribution and calculate the 95% credible interval on it. We do this for all the 140 theta values to get the 95% interval.

For the posterior mean, we take the mean of the randomly generated 1000 samples from the poission distribution and take the mean of that.

Question 1D : Stan Model using Informative prior on X and Y

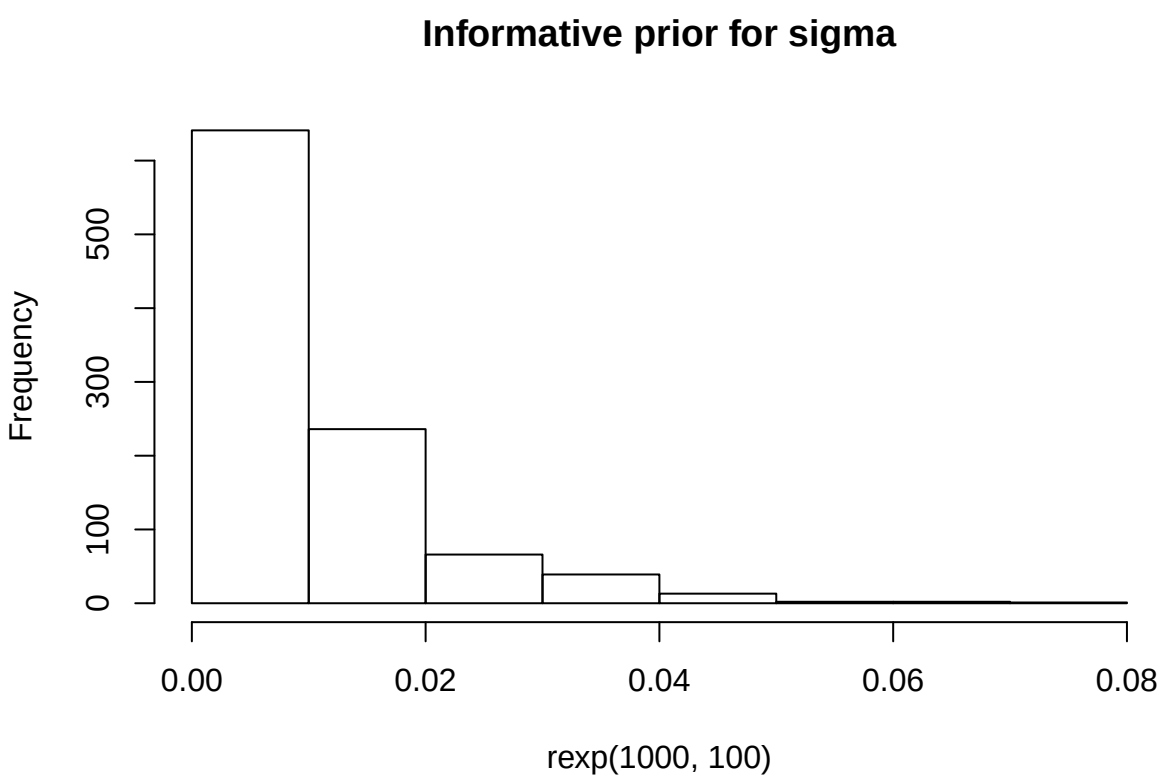
```
stan_code3 = "data {
  int<lower=0> T;
  int c[T];
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
  vector[T] x;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(100);

  x[2:T] ~ normal(alpha + beta*x[1:(T-1)], sigma);
  c ~ poisson(exp(x));
}

"
model <- stan_model(model_code = stan_code3)

hist(params_c$sigma, main="Sigma we got from the previous simulation")
```




```
mean(params_c$sigma)
```

```
## [1] 0.2606058
```

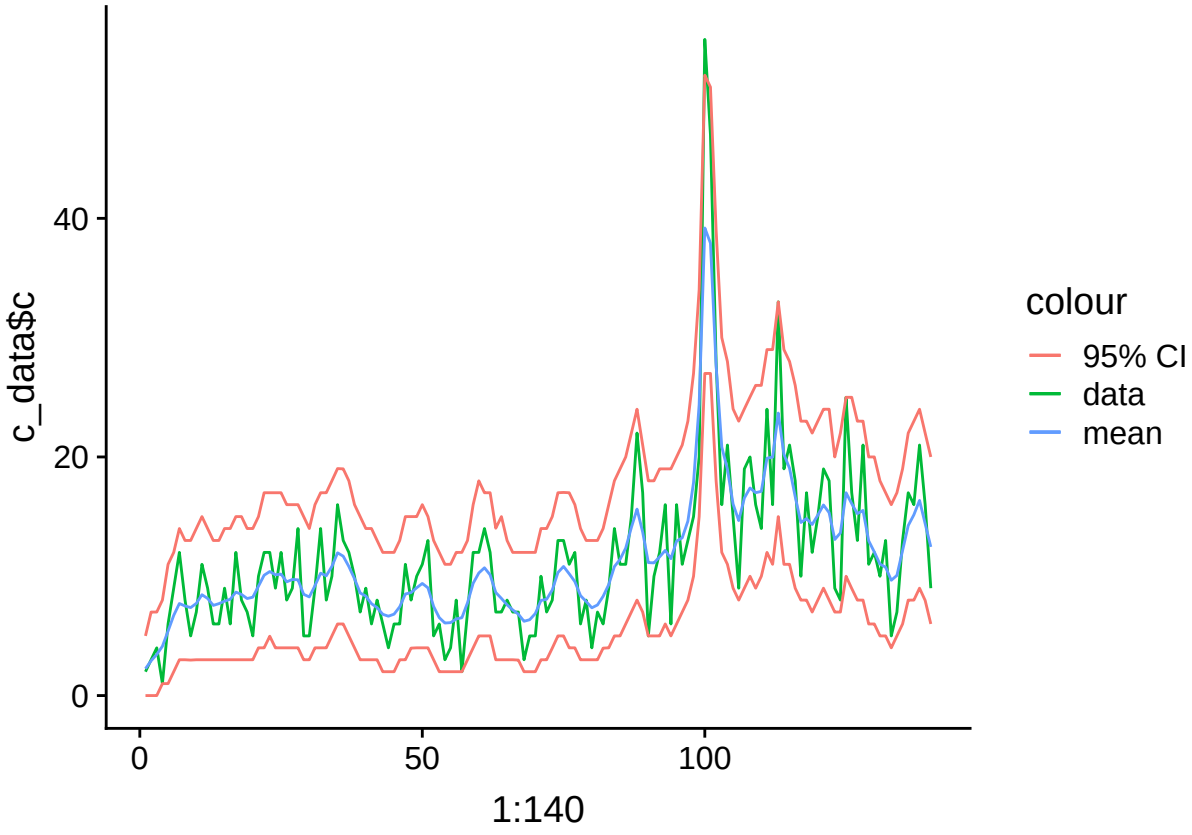
```
hist(rexp(1000, 100), main="Informative prior for sigma")
```

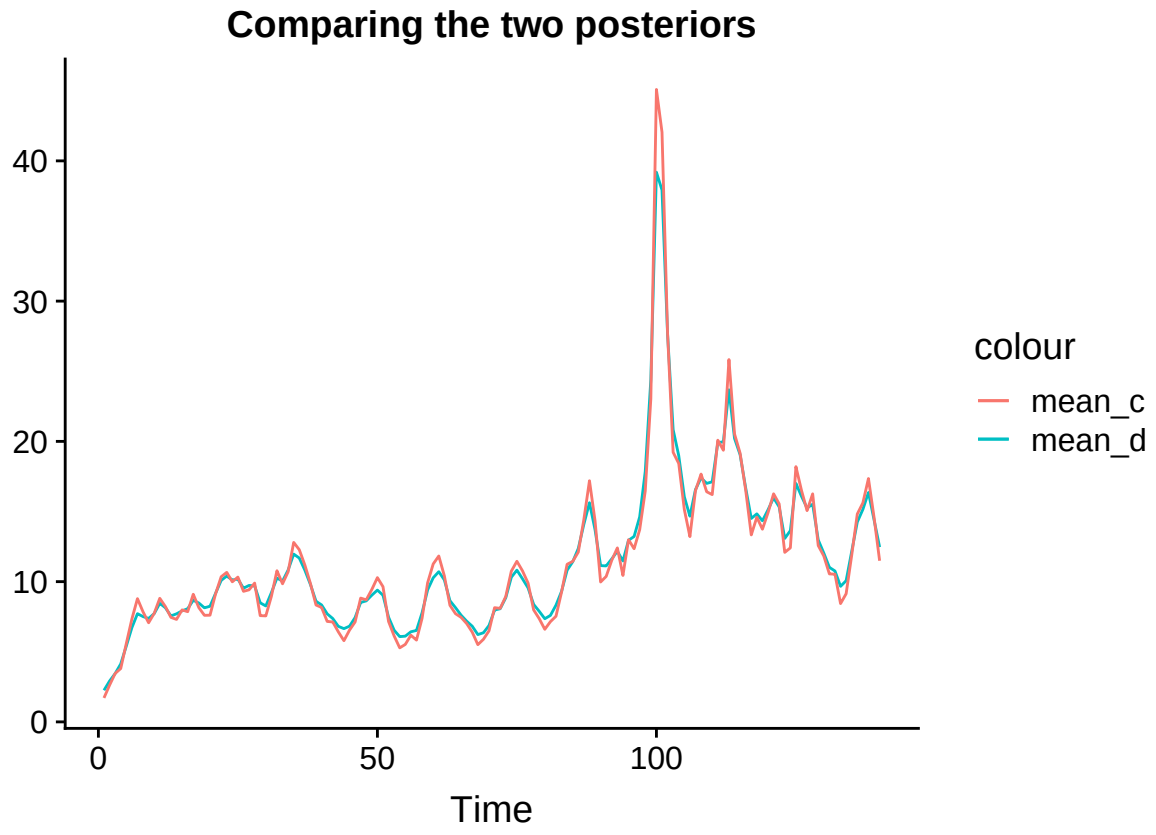


We are creating the stan model for the problem. We are then compiling the model and storing into the variable model. The priors used for this model are informative as we are told that the variance sigma for the data is smaller than the data suggests. The value for sigma we got from the previous model was around 0.25, and we are told that it is smoother than the data suggests. So our prior knowledge tells us that the values of sigma smaller than 0.2 should be given higher probability. We use a exponential distribution to specify the informative prior for sigma. We use a large rate for the exponential distribution so the density is centered close to zero.

We are then sampling from this compiled model. We are taking 4 markov chains of 10000 simulations from this posterior with a burnin period of 1000.

Summary from fitted model


c_data$c
40
20
0
0
50
100
1:140
colour
95% CI
data
mean



The posterior means and CI are calculated in the same manner as we did in the previous question. The posterior mean plot we get using an informative prior is similar, but it is smoother from the one we got in the previous question.

To make a comparision of both, we made a plot of the posterior mean we got from the previous question and the posterior mean calculated using an informative prior in this question. The posterior mean for the informative prior for sigma is a little smoother. This is the only difference we found using a informative prior.

Appendix

```
library(ggplot2)
library(ggplot2)
library(rstan)
library(cowplot)
options(mc.cores=4)

#Given intial values
mu <- 10
sg_sq <- 2
t_p <- 200   #time points. Number of variables


#function for creating Ar process according to given equation
ar_process <- function(phi,t_p,mu,sg_sq) {
  x <- c()
  x[1] <- mu
  for(i in 2:t_p) {
    x[i] <- mu + phi*(x[i-1]-mu) + rnorm(1,0,sg_sq)
  }

  return(x)
}

#phi value between -1 and 1
phi <- c(-1,0.7,1)

#Creating data for three phi values
ar_data <- do.call(cbind,lapply(phi,ar_process,t_p=200,mu=10,sg_sq=2))
colnames(ar_data) <- c("samp1","samp2","samp3")
ar_data <- as.data.frame(ar_data)

#plotting to check the effect of phi on Ar process. We need not show these three plots.
ggplot(ar_data,aes(x=seq(1:200))) +
  geom_line(aes(y=samp1,color="samp1"),size=0.9) +
  geom_line(aes(y=samp2,color="samp2"),size=0.9) +
  geom_line(aes(y=samp3,color="samp3"),size=0.9) +
  theme_grey() + scale_color_manual(values = c("#8da0cb", "#fc8d62", "#1b9e77"),
                                    labels = c(phi[1], phi[2], phi[3]))


#choosing 0.7 as  avalue to work further on.
ggplot(ar_data,aes(x=seq(1:200))) +
  geom_line(aes(y=samp2,color="samp2"),size=0.9)+ theme_grey() +
  scale_color_manual(values = "#fc8d62",labels = phi[2]) +
  ggtitle(expression(paste("AR process with ",phi,"=0.7"))) +
  xlab("Samples") + ylab("AR process")

#1B
stan_code1 = "data {
  int<lower=0> T;
  vector[T] y;
```

```
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(0.00001);

  y[2:T] ~ normal(alpha + beta*y[1:(T-1)], sigma);
}
"
#compile model
model <- stan_model(model_code = stan_code1)

#phi value between -1 and 1
phi <- c(0.3,0.95)
T = 200
mu = 10

#Creating data for three phi values
ar_data <- do.call(cbind,lapply(phi,ar_process,t_p=T,mu=mu,sg_sq=2))
colnames(ar_data) <- c("X","Y")
ar_data <- as.data.frame(ar_data)
#head(ar_data)

#plotting to check the effect of phi on Ar process. We need not show these three plots.
ggplot(ar_data,aes(x=seq(1:200))) +
  geom_line(aes(y=X,color="samp1"),size=0.9) +
  geom_line(aes(y=Y,color="samp2"),size=0.9) +
  scale_color_manual(values = c("#8da0cb", "#fc8d62"),
                     labels = c(phi[1], phi[2]))

#model run on X
fit_x <- sampling(model, list(T=T, y=ar_data$X), iter=1000, warmup=500, chains=4)

#model run on Y
fit_y <- sampling(model, list(T=T, y=ar_data$Y), iter=1000, warmup=500, chains=4)

params_x <- extract(fit_x)
mu_x = params_x$alpha/(1 - params_x$beta)
cat("Mean for mu :", mean(mu_x), "\n\n")
print("95% confidence interval for mu")
quantile(mu_x, probs = c(0.025, 0.975))
cat("\n\n")
plot_grid(ggplot() + geom_density(aes(params_x$alpha)),
          ggplot() + geom_density(aes(params_x$beta)),
          ggplot() + geom_density(aes(params_x$sigma)),
          ggplot() + geom_density(aes(mu_x)))
```

```r
#print(fit_x)
a = summary(fit_x)
print("95% confidence intervals for alpha, beta, sigma")
print(a$summary[1:3,c(1, 4, 8, 9)], digits = 4)


params_y <- extract(fit_y)
mu_y = params_y$alpha/(1 - params_y$beta)
cat("Mean for mu :", mean(mu_y), "\n\n")
print("95% confidence interval for mu")
quantile(mu_y, probs = c(0.025, 0.975))
cat("\n\n")
plot_grid(ggplot() + geom_density(aes(params_y$alpha)),
          ggplot() + geom_density(aes(params_y$beta)),
          ggplot() + geom_density(aes(params_y$sigma)),
          ggplot() + geom_density(aes(mu_y)))

#print(fit_y)
b = summary(fit_y)
print("95% confidence intervals for alpha, beta, sigma")
print(b$summary[1:3,c(1, 4, 8, 9)], digits = 4)

plot_grid(nrow = 2, ncol = 1,
          traceplot(fit_x),
          traceplot(fit_y),
          labels = c("Traceplot Model X", "Traceplot Model Y"),label_size = 9)

plot_grid(nrow = 2, ncol = 1,
          ggplot() + geom_point(aes(mu_x, params_x$beta)),
          ggplot() + geom_point(aes(mu_y, params_y$beta)),
          labels = c("Data X Joint Posterior", "Data Y Joint Posterior"),
          label_size = 10)


#1C
stan_code2 = "data {
  int<lower=0> T;
  int c[T];
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
  vector[T] x;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(0.00001);

  x[2:T] ~ normal(alpha + beta*x[1:(T-1)], sigma);
```

```
  c ~ poisson(exp(x));
}
"
model <- stan_model(model_code = stan_code2)

c_data <- as.data.frame(read.table("data.txt", header = T))
#hist(c_data$c)
T = length(c_data$c)

fit_c <- sampling(model, list(T=T, c=c_data$c), iter=10000, warmup=1000, chains=2)

params_c <- extract(fit_c)

x_post = colMeans(params_c$x)
theta_x = exp(x_post)

post_ci = matrix(0, nrow = 140, ncol = 2)
post_mean = matrix(0, nrow = 140, ncol = 1)
#simulate using this theta_x
for(i in 1:140){
  post_theta = rpois(1000, theta_x[i])
  post_ci[i, ] = quantile(post_theta, probs = c(0.025, 0.975))
  post_mean[i,1] = mean(post_theta)
}

ggplot() +
  geom_line(aes(1:140, c_data$c, col="data")) +
  geom_line(aes(1:140, post_ci[,1], col="95% CI")) +
  geom_line(aes(1:140, post_ci[,2], col="95% CI"))+
  geom_line(aes(1:140, post_mean[,1], col="mean"))


#1D
stan_code3 = "data {
  int<lower=0> T;
  int c[T];
}

parameters {
  real alpha;
  real<lower=-1,upper=1> beta;
  real<lower=0> sigma;
  vector[T] x;
}

model {
  alpha ~ normal(0, 10000);
  beta ~ uniform(-1, 1);
  sigma ~ exponential(100);

  x[2:T] ~ normal(alpha + beta*x[1:(T-1)], sigma);
  c ~ poisson(exp(x));
}
```

```
"
model <- stan_model(model_code = stan_code3)

hist(params_c$sigma, main="Sigma we got from the previous simulation")
mean(params_c$sigma)

hist(rexp(1000, 100), main="Informative prior for sigma")

#model run for d
fit_d <- sampling(model, list(T=T, c=c_data$c), iter=10000, warmup=1000, chains=4)

params_d <- extract(fit_d)
x_post_d = colMeans(params_d$x)
theta_x_d = exp(x_post_d)

post_ci_d = matrix(0, nrow = 140, ncol = 2)
post_mean_d = matrix(0, nrow = 140, ncol = 1)
#simulate using this theta_x
for(i in 1:140){
  post_theta = rpois(1000, theta_x_d[i])
  post_ci_d[i, ] = quantile(post_theta, probs = c(0.025, 0.975))
  post_mean_d[i,1] = mean(post_theta)
}

ggplot() +
  geom_line(aes(1:140, c_data$c, col="data")) +
  geom_line(aes(1:140, post_ci_d[,1], col="95% CI")) +
  geom_line(aes(1:140, post_ci_d[,2], col="95% CI"))+
  geom_line(aes(1:140, post_mean_d[,1], col="mean"))


ggplot() +
  geom_line(aes(1:140, post_mean_d[,1], col="mean_d")) +
  geom_line(aes(1:140, post_mean[,1], col="mean_c")) +
  labs(x="Time", y="", title="Comparing the two posteriors")
```