Bayesian Learning (732A91) Lab4

Prudhvi Peddmalluprupe690

29 May, 2019

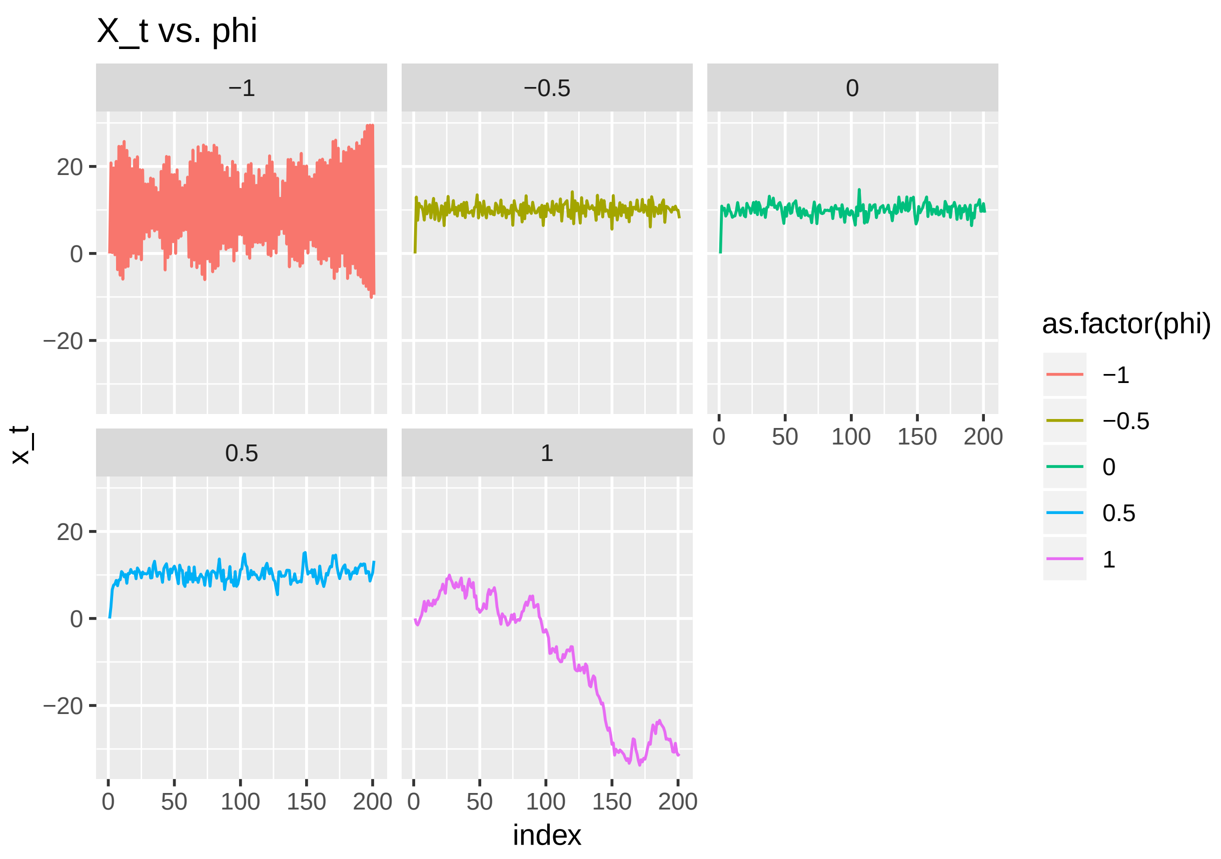
Question 1: Time series models in Stan

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 μ=10, σ^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 $x_{1:T}$?

```r
set.seed(12345)
time_series_function <- function(mu,phi,sigma_sq, T){
  x <-rep(0, T)
  x[0] <- mu
  for(i in 1:T){
    x[i+1] <- mu + phi * (x[i]-mu) + rnorm(n=1,mean=0, sd= sqrt(sigma_sq))
  }
  x <- as.matrix(x)
  temp <- cbind(x, index=1:length(x))
  return(temp)
}

final <- NULL
phi <- seq(-1,1,0.5)
for(i in 1:length(phi)){
  temp <- time_series_function(mu=10, phi=phi[i], sigma_sq=2, T=200)
  temp <- cbind(temp, phi=phi[i])
  final <- rbind(final,temp)
}
final <- final %>% as.data.frame()
colnames(final) <- c("x_t", "index", "phi")

ggplot(data=final, aes(x=index, y=x_t)) +
  geom_line(aes(color=as.factor(phi))) +
  facet_wrap(~phi) +
  ggtitle("X_t vs. phi")
```



Analysis: Varying phi acts as the trend for the time series, if phi is close to 1 the series is increasing time series, while if the phi close to -1 its a decreasing time series, phi values close to 0 makes the series a very constant one.

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 of 95 percentage 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?

```
set.seed(12345)
T = 201
burnin = 1000
niter = 2000


x_t <- time_series_function(mu=10, phi=0.3, sigma_sq=2, T=200)
x_t <- as.vector(x_t[,1])
y_t <- time_series_function(mu=10, phi=0.95, sigma_sq=2, T=200)
y_t <- y_t[,1]
```

```r
ARStanModel = 'data {
  int<lower=1> N;
  real x[N];
}
parameters {
  real mu;
  real<lower=0> sigma_sq;
  real<lower=-1,upper=1> phi;
}
transformed parameters {
  real sigma;
  sigma = sqrt(sigma_sq);
}
model {
 for (n in 2:N)
   x[n] ~ normal(mu + phi * (x[n-1] - mu), sigma);
}'

# perform MCMC
x_t_fit = stan(model_code=ARStanModel, data=list(x=x_t, N=T),
               control = list(adapt_delta = 0.99), warmup=burnin,iter=niter,chains=5)
y_t_fit = stan(model_code=ARStanModel, data=list(x=y_t, N=T),
               control = list(adapt_delta = 0.99), warmup=burnin,iter=niter,chains=5)

posterior_mean_x_t <- get_posterior_mean(x_t_fit)
posterior_mean_y_t <- get_posterior_mean(y_t_fit)

# view the series
posterior_mean_x_t %>% xtable() %>% kable()
```

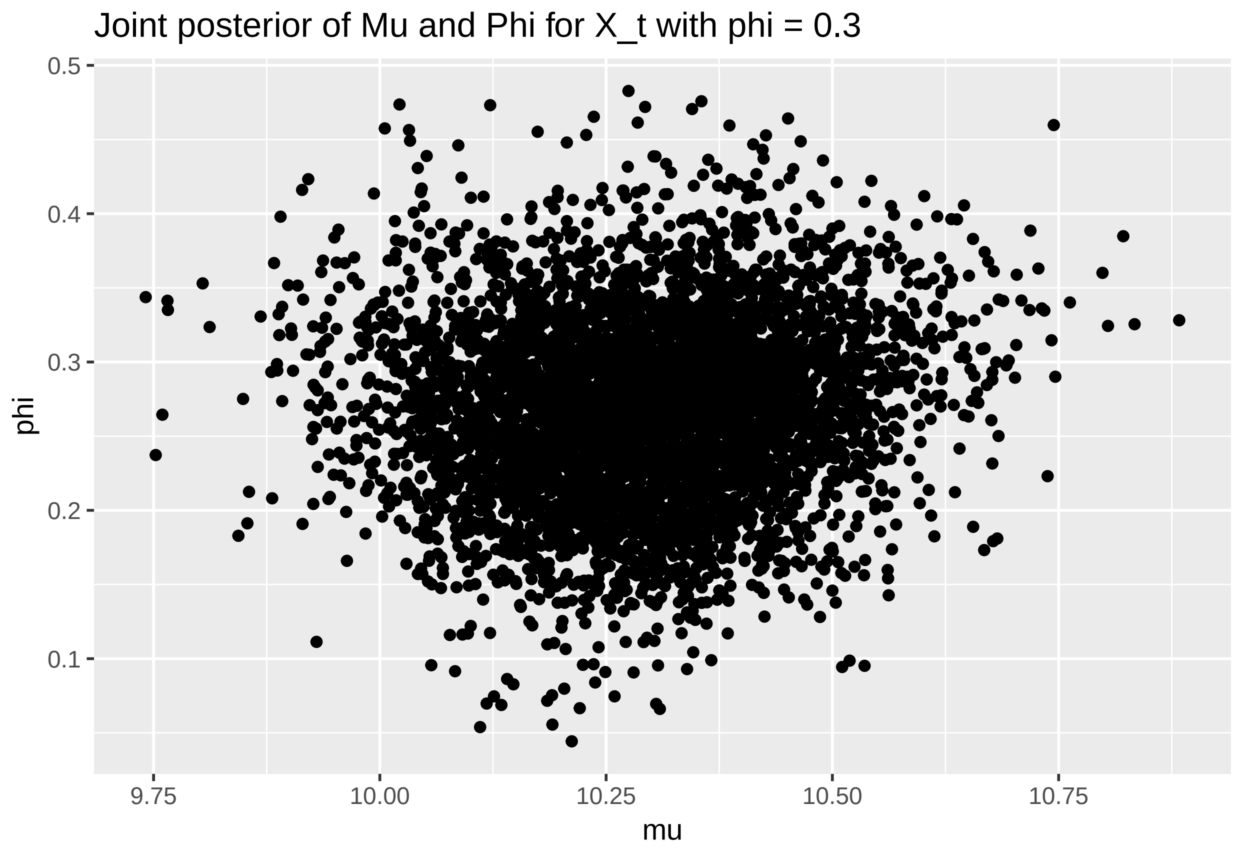
	mean-chain:1	mean-chain:2	mean-chain:3	mean-chain:4	mean-chain:5	mean-all chains
mu	10.3003116	10.291882	10.2888822	10.2879861	10.2910254	10.2920175
sigma_sq	2.3317024	2.335793	2.3277056	2.3140533	2.3081254	2.3234758
phi	0.2713264	0.267846	0.2675556	0.2696746	0.2666851	0.2686175
sigma	1.5251371	1.526381	1.5236411	1.5195305	1.5171208	1.5223620
lp__	-182.8577150	-183.013887	-182.8605489	-182.8045461	-183.0461312	-182.9165656

```r
posterior_mean_y_t %>% xtable() %>% kable()
```

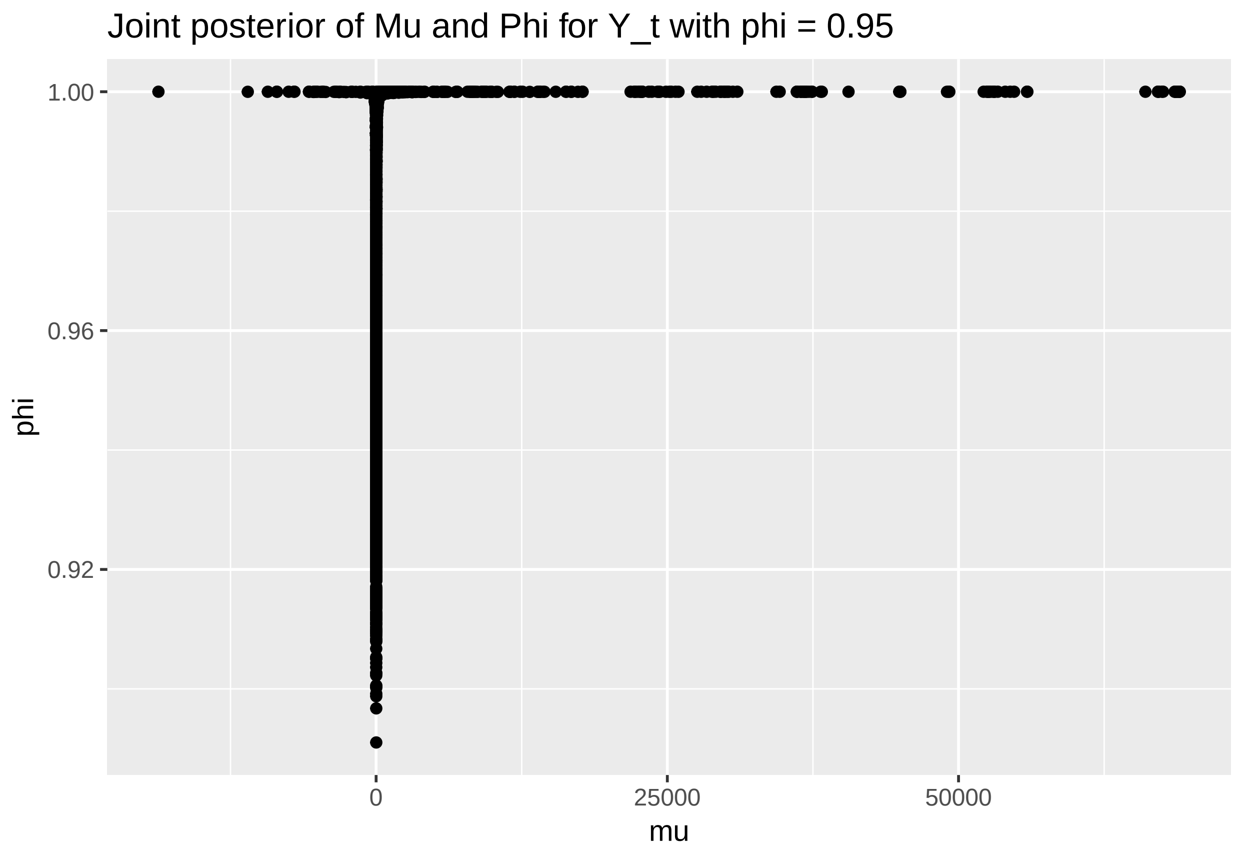
	mean-chain:1	mean-chain:2	mean-chain:3	mean-chain:4	mean-chain:5	mean-all chains
mu	34.8837169	4828.2144277	14.0261741	15.8836576	96.2921132	997.8600179
sigma_sq	1.8251916	1.9918558	1.8883744	1.8901587	1.8868261	1.8964813
phi	0.9676703	0.9980641	0.9651566	0.9611996	0.9632391	0.9710659
sigma	1.3486473	1.4094019	1.3741448	1.3730874	1.3719778	1.3754518
lp__	-165.8314328	-172.0849971	-164.6375419	-165.1513214	-165.5695638	-166.6549714

```r
# mu_x_t_post <- posterior_mean_x_t[1, 5]
# sigma_x_t_post <- posterior_mean_x_t[2, 5]
# phi_x_t_post <- posterior_mean_x_t[3, 5]
```

```r
x_t_params <- extract(x_t_fit, pars = c("mu", "phi")) %>% as.data.frame()
y_t_params <- extract(y_t_fit, pars = c("mu", "phi")) %>% as.data.frame()


ggplot(data=x_t_params, aes(x=mu, y=phi)) +
  geom_point() +
  ggtitle("Joint posterior of Mu and Phi for X_t with phi = 0.3")
```



```r
ggplot(data=y_t_params, aes(x=mu, y=phi)) +
  geom_point() +
  ggtitle("Joint posterior of Mu and Phi for Y_t with phi = 0.95")
```



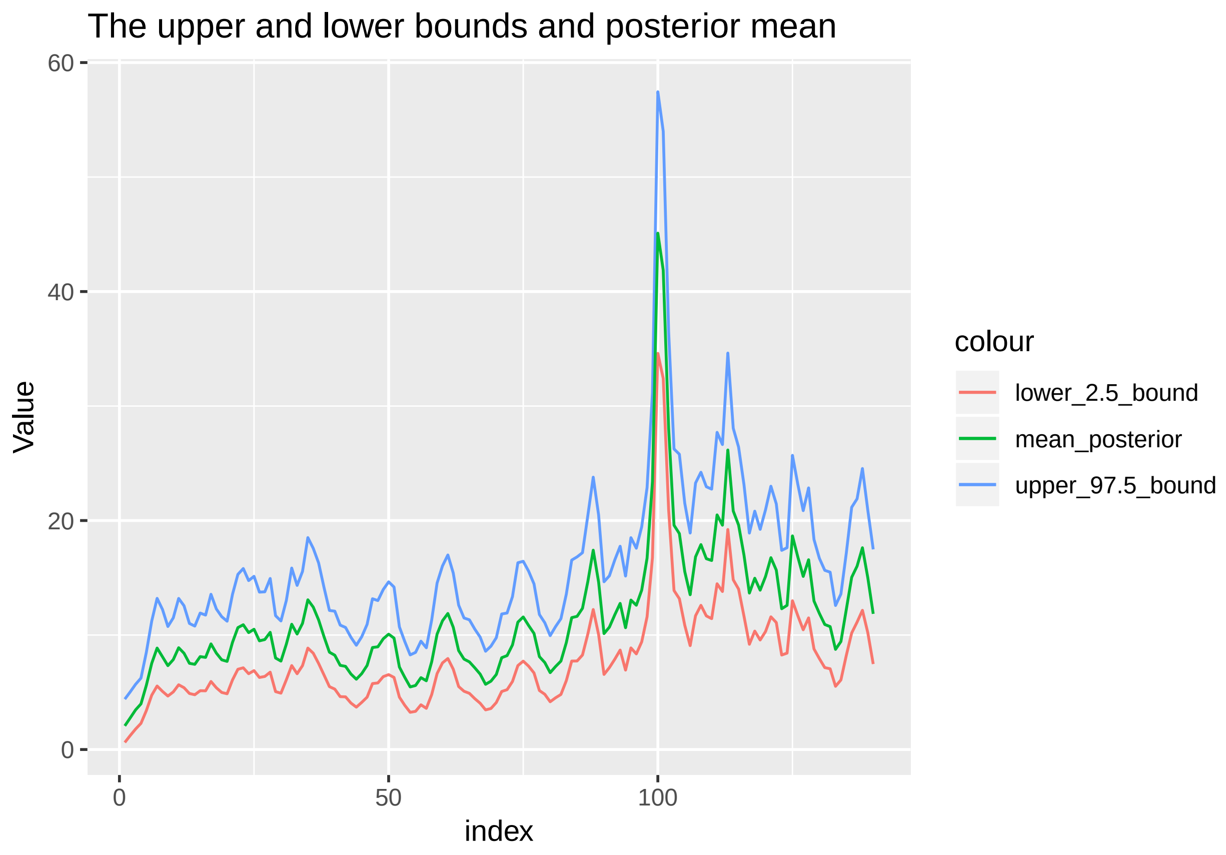
Analysis: For the 1st series(phi=0.35) the estimates of mu and sigma_sq are very close to the true values while for the second series(phi=0.95) the estimates mu and sigma_sq are way off from the true values. For the case of estimation of phi, the second's series(phi=0.95) estimation from stan model are closer to true value when compared to the 1st series(phi=0.35)

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 ct at each time point follows an independent Poisson distribution when conditioned on a latent AR(1)-process xt, 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

```
campy_data <- read.csv("campy.txt", sep="")

x_campy <- as.vector(campy_data$c)
N <- length(x_campy)


Stanpoisson = 'data {
  int<lower=0> N;
  int c[N];
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real phi;
  vector[N] x;
}
transformed parameters {
  real sigma;
  sigma = sqrt(sigma2);
}
model {
  for (n in 2:N)
    x[n] ~ normal(mu + phi * (x[n-1] - mu), sigma);
  for (n in 1:N)
    c[n] ~ poisson(exp(x[n]));
}'

c.fit <- stan(model_code = Stanpoisson, data = list (c = x_campy, N = N))

params <- extract(c.fit, pars = c("mu", "sigma"))
x <- extract(c.fit, pars = "x")

theta_t <- exp(x$x)
x_mean <- apply(theta_t, 2, mean)
x_quant <- apply(theta_t, 2, quantile, probs=c(0.025, 0.975))

x_quant_transpose <- t(x_quant) %>% as.matrix()
x_mean <- x_mean %>% as.matrix()

data_frame_plot <- cbind(x_quant_transpose, x_mean) %>% as.data.frame()
colnames(data_frame_plot) <- c("lower_2.5_bound", "upper_97.5_bound", "mean_posterior")
data_frame_plot$index <- seq(1:NROW(data_frame_plot))

ggplot(data=data_frame_plot, aes(x=index)) +
  geom_line(aes(y=lower_2.5_bound, color = "lower_2.5_bound")) +
  geom_line(aes(y=upper_97.5_bound, color = "upper_97.5_bound"))  +
  geom_line(aes(y=mean_posterior, color = "mean_posterior"))  +
  ggtitle("The upper and lower bounds and posterior mean") +
  ylab("Value")
```

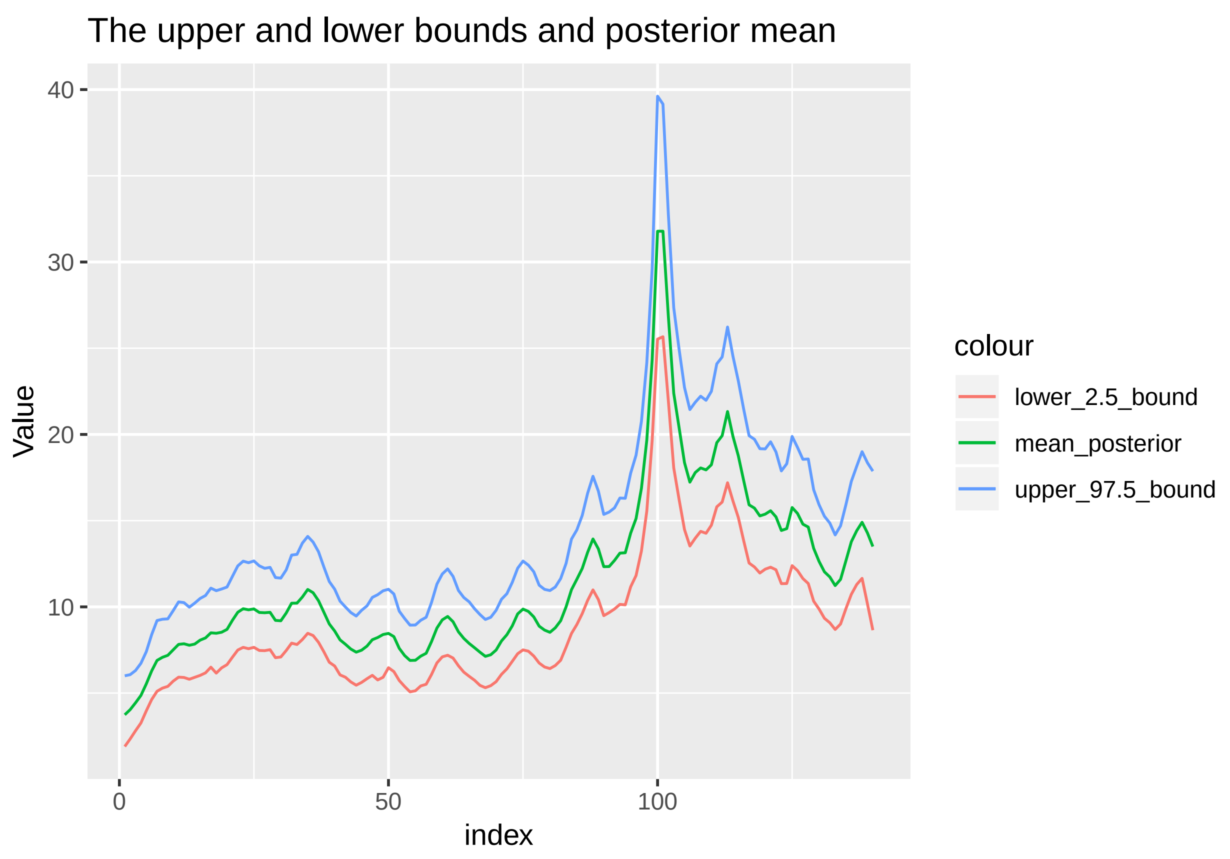


d) Now, assume that we have a prior belief that the true underlying intensity θ varies more smoothly than the data suggests. Change the prior for σ^2 so that it becomes informative about that the AR(1)-process increments ϵ should be small. Re-estimate the model using Stan with the new prior and produce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

```
Stanpoisson = 'data {
  int<lower=0> N;
  int c[N];
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real phi;
  vector[N] x;
}
transformed parameters {
  real sigma;
  sigma = sqrt(sigma2);
}
model {
  sigma  ~ normal(0, 0.02);
  phi    ~ normal(0, 0.6);
  for (n in 2:N)
    x[n] ~ normal(mu + phi * (x[n-1] - mu), sigma);
  for (n in 1:N)
    c[n] ~ poisson(exp(x[n]));
}'

c.fit <- stan(model_code = Stanpoisson, data = list (c = x_campy, N = N))
```

```
## DIAGNOSTIC(S) FROM PARSER:
## Info (non-fatal):
## Left-hand side of sampling statement (~) may contain a non-linear transform of a parameter or local
## If it does, you need to include a target += statement with the log absolute determinant of the Jacob
## Left-hand-side of sampling statement:
##     sigma ~ normal(...)

## Warning: There were 619 divergent transitions after warmup. Increasing adapt_delta above 0.8 may hel
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Warning: There were 105 transitions after warmup that exceeded the maximum treedepth. Increase max_t
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
params <- extract(c.fit, pars = c("mu", "sigma"))
x <- extract(c.fit, pars = "x")

theta_t <- exp(x$x)
x_mean <- apply(theta_t, 2, mean)
x_quant <- apply(theta_t, 2, quantile, probs=c(0.025, 0.975))

x_quant_transpose <- t(x_quant) %>% as.matrix()
x_mean <- x_mean %>% as.matrix()

data_frame_plot <- cbind(x_quant_transpose, x_mean) %>% as.data.frame()
colnames(data_frame_plot) <- c("lower_2.5_bound", "upper_97.5_bound", "mean_posterior")
data_frame_plot$index <- seq(1:NROW(data_frame_plot))

ggplot(data=data_frame_plot, aes(x=index)) +
  geom_line(aes(y=lower_2.5_bound, color = "lower_2.5_bound")) +
  geom_line(aes(y=upper_97.5_bound, color = "upper_97.5_bound"))  +
  geom_line(aes(y=mean_posterior, color = "mean_posterior"))  +
  ggtitle("The upper and lower bounds and posterior mean") +
  ylab("Value")
```



Analysis: The posterior does appear to have changed, its a lot more smoother than the one with no prior set.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
#options(tinytex.verbose = TRUE)
options(scipen=999)

Sys.setenv(USE_CXX14 = 1)

library("ggplot2")
library("tidyverse")
library("rstan")
library("gridExtra") # combine plots
library("xtable") # model summary as table
library("knitr")


# The palette with black:
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
              "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
set.seed(12345)
```

```r
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
Sys.setenv(LOCAL_CPPFLAGS = '-march=native')


set.seed(12345)
time_series_function <- function(mu,phi,sigma_sq, T){
  x <-rep(0, T)
  x[0] <- mu
  for(i in 1:T){
    x[i+1] <- mu + phi * (x[i]-mu) + rnorm(n=1,mean=0, sd= sqrt(sigma_sq))
  }
  x <- as.matrix(x)
  temp <- cbind(x, index=1:length(x))
  return(temp)
}

final <- NULL
phi <- seq(-1,1,0.5)
for(i in 1:length(phi)){
  temp <- time_series_function(mu=10, phi=phi[i], sigma_sq=2, T=200)
  temp <- cbind(temp, phi=phi[i])
  final <- rbind(final,temp)
}
final <- final %>% as.data.frame()
colnames(final) <- c("x_t", "index", "phi")

ggplot(data=final, aes(x=index, y=x_t)) +
  geom_line(aes(color=as.factor(phi))) +
  facet_wrap(~phi) +
  ggtitle("X_t vs. phi")



set.seed(12345)
T = 201
burnin = 1000
niter = 2000


x_t <- time_series_function(mu=10, phi=0.3, sigma_sq=2, T=200)
x_t <- as.vector(x_t[,1])
y_t <- time_series_function(mu=10, phi=0.95, sigma_sq=2, T=200)
y_t <- y_t[,1]

ARStanModel = 'data {
  int<lower=1> N;
  real x[N];
}
parameters {
  real mu;
  real<lower=0> sigma_sq;
  real<lower=-1,upper=1> phi;
```

```
}
transformed parameters {
  real sigma;
  sigma = sqrt(sigma_sq);
}
model {
 for (n in 2:N)
   x[n] ~ normal(mu + phi * (x[n-1] - mu), sigma);
}'

# perform MCMC
x_t_fit = stan(model_code=ARStanModel, data=list(x=x_t, N=T),
               control = list(adapt_delta = 0.99), warmup=burnin,iter=niter,chains=5)
y_t_fit = stan(model_code=ARStanModel, data=list(x=y_t, N=T),
               control = list(adapt_delta = 0.99), warmup=burnin,iter=niter,chains=5)

posterior_mean_x_t <- get_posterior_mean(x_t_fit)
posterior_mean_y_t <- get_posterior_mean(y_t_fit)

# view the series
posterior_mean_x_t %>% xtable() %>% kable()
posterior_mean_y_t %>% xtable() %>% kable()

# mu_x_t_post <- posterior_mean_x_t[1, 5]
# sigma_x_t_post <- posterior_mean_x_t[2, 5]
# phi_x_t_post <- posterior_mean_x_t[3, 5]

x_t_params <- extract(x_t_fit, pars = c("mu", "phi")) %>% as.data.frame()
y_t_params <- extract(y_t_fit, pars = c("mu", "phi")) %>% as.data.frame()


ggplot(data=x_t_params, aes(x=mu, y=phi)) +
  geom_point() +
  ggtitle("Joint posterior of Mu and Phi for X_t with phi = 0.3")

ggplot(data=y_t_params, aes(x=mu, y=phi)) +
  geom_point() +
  ggtitle("Joint posterior of Mu and Phi for Y_t with phi = 0.95")


campy_data <- read.csv("campy.txt", sep="")

x_campy <- as.vector(campy_data$c)
N <- length(x_campy)


Stanpoisson = 'data {
  int<lower=0> N;
  int c[N];
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real phi;
  vector[N] x;
}
transformed parameters {
  real sigma;
  sigma = sqrt(sigma2);
}
model {
  for (n in 2:N)
    x[n] ~ normal(mu + phi * (x[n-1] - mu), sigma);
  for (n in 1:N)
    c[n] ~ poisson(exp(x[n]));
}'

c.fit <- stan(model_code = Stanpoisson, data = list (c = x_campy, N = N))

params <- extract(c.fit, pars = c("mu", "sigma"))
x <- extract(c.fit, pars = "x")

theta_t <- exp(x$x)
x_mean <- apply(theta_t, 2, mean)
x_quant <- apply(theta_t, 2, quantile, probs=c(0.025, 0.975))

x_quant_transpose <- t(x_quant) %>% as.matrix()
x_mean <- x_mean %>% as.matrix()

data_frame_plot <- cbind(x_quant_transpose, x_mean) %>% as.data.frame()
colnames(data_frame_plot) <- c("lower_2.5_bound", "upper_97.5_bound", "mean_posterior")
data_frame_plot$index <- seq(1:NROW(data_frame_plot))

ggplot(data=data_frame_plot, aes(x=index)) +
  geom_line(aes(y=lower_2.5_bound, color = "lower_2.5_bound")) +
  geom_line(aes(y=upper_97.5_bound, color = "upper_97.5_bound"))  +
  geom_line(aes(y=mean_posterior, color = "mean_posterior"))  +
  ggtitle("The upper and lower bounds and posterior mean") +
  ylab("Value")



Stanpoisson = 'data {
  int<lower=0> N;
  int c[N];
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real phi;
  vector[N] x;
}
transformed parameters {
  real sigma;
  sigma = sqrt(sigma2);
}
```

```
model {
  sigma  ~ normal(0, 0.02);
  phi    ~ normal(0, 0.6);
  for (n in 2:N)
    x[n] ~ normal(mu + phi * (x[n-1] - mu), sigma);
  for (n in 1:N)
    c[n] ~ poisson(exp(x[n]));
}'

c.fit <- stan(model_code = Stanpoisson, data = list (c = x_campy, N = N))

params <- extract(c.fit, pars = c("mu", "sigma"))
x <- extract(c.fit, pars = "x")

theta_t <- exp(x$x)
x_mean <- apply(theta_t, 2, mean)
x_quant <- apply(theta_t, 2, quantile, probs=c(0.025, 0.975))

x_quant_transpose <- t(x_quant) %>% as.matrix()
x_mean <- x_mean %>% as.matrix()

data_frame_plot <- cbind(x_quant_transpose, x_mean) %>% as.data.frame()
colnames(data_frame_plot) <- c("lower_2.5_bound", "upper_97.5_bound", "mean_posterior")
data_frame_plot$index <- seq(1:NROW(data_frame_plot))

ggplot(data=data_frame_plot, aes(x=index)) +
  geom_line(aes(y=lower_2.5_bound, color = "lower_2.5_bound")) +
  geom_line(aes(y=upper_97.5_bound, color = "upper_97.5_bound"))  +
  geom_line(aes(y=mean_posterior, color = "mean_posterior"))  +
  ggtitle("The upper and lower bounds and posterior mean") +
  ylab("Value")
```