Bayesian Learning LAB 04

Rakhshanda Jabeen

17/05/2020

Contents

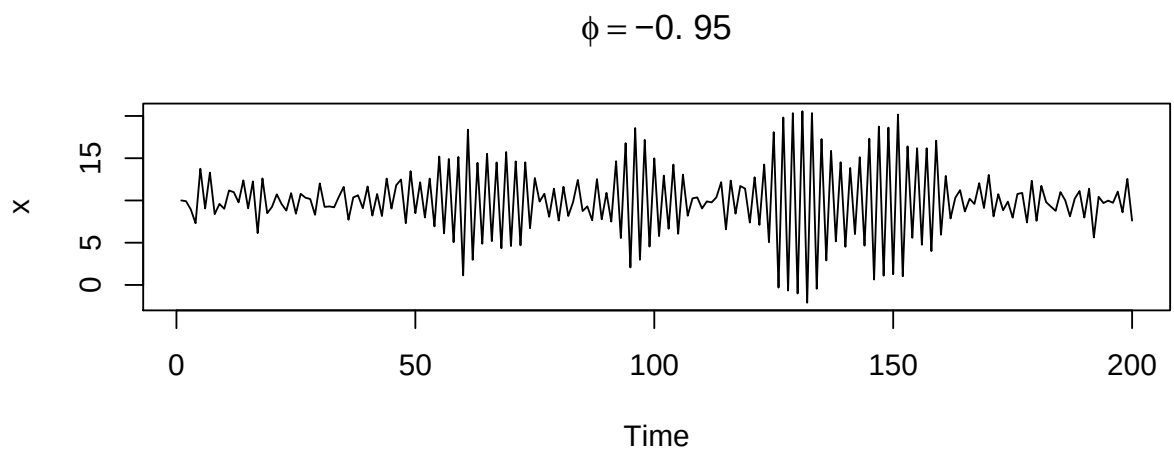
1. Time Series Models in Stan

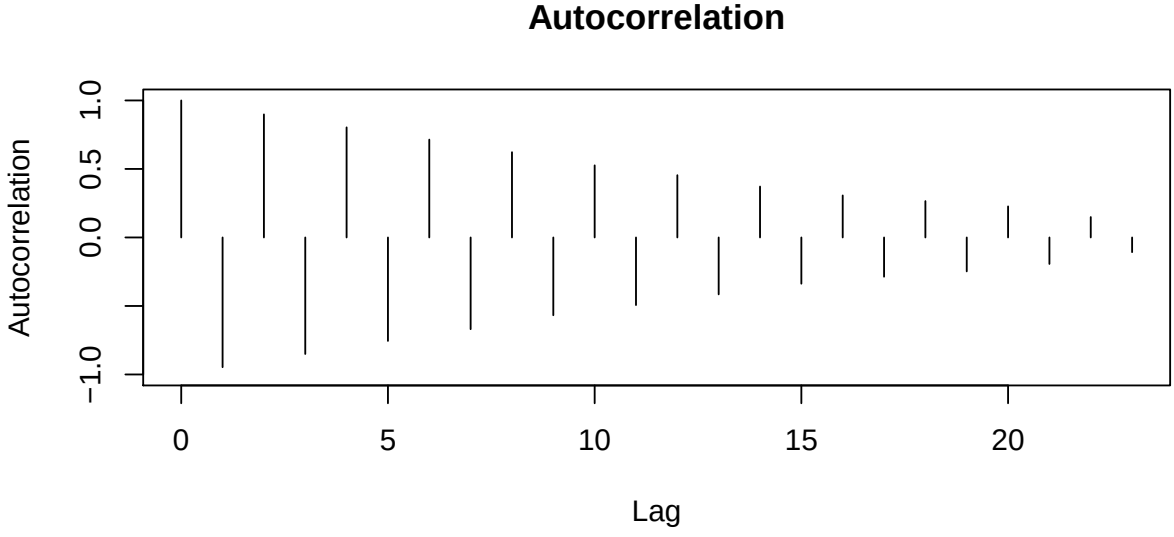
1.1 AR(1) Process

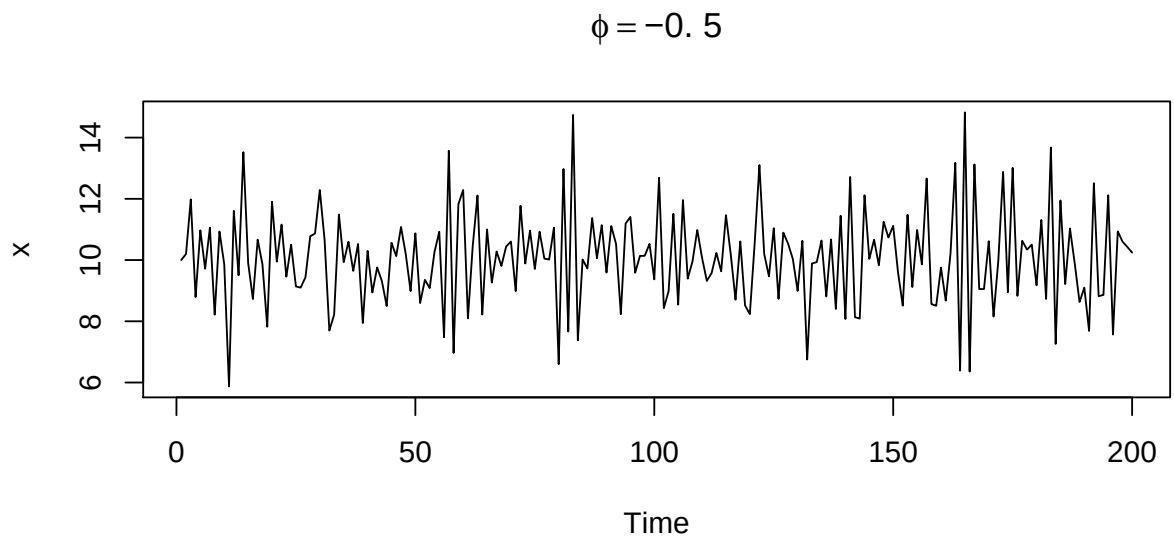
In this task we are simmulating data from autoregressive process of order 1 for $\mu = 10$, $\sigma^2 = 2$.

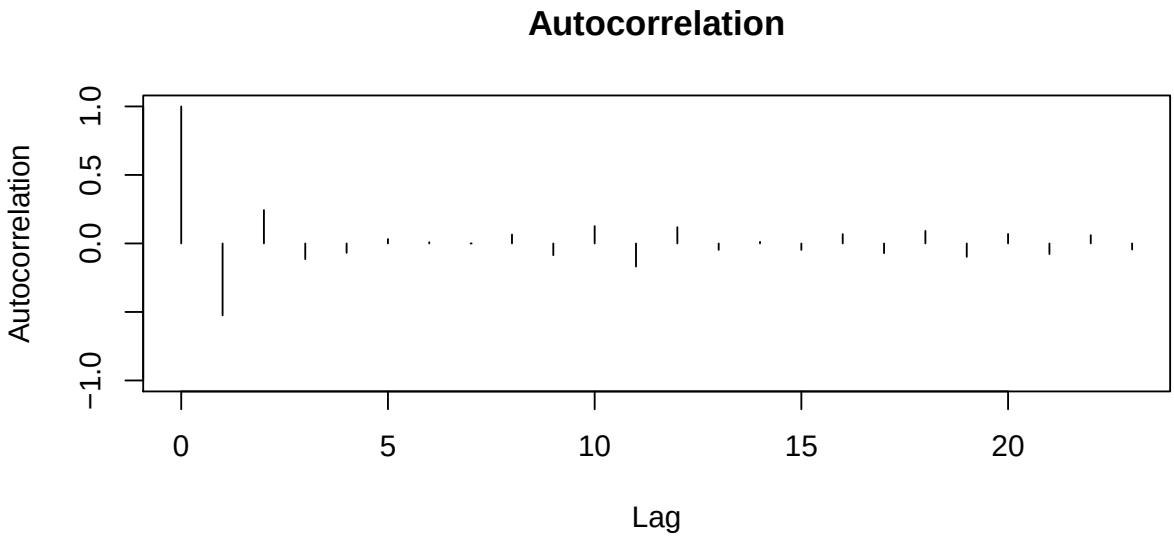
$$x_t = \mu + \phi\big(x_{t-1} - \mu\big) + \epsilon_t, \qquad \epsilon_t \sim \mathcal{N}(0, \sigma^2)$$

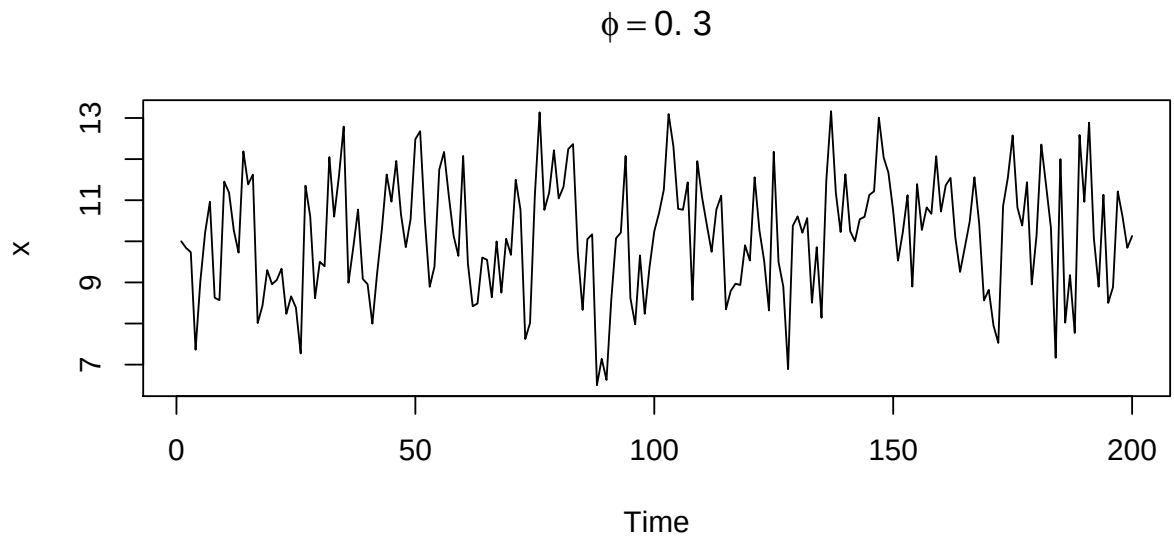
The following plots represents the time series plots and their respective autocorrelation plots simmulated from $x_1 : T$ where, $T = 200$.

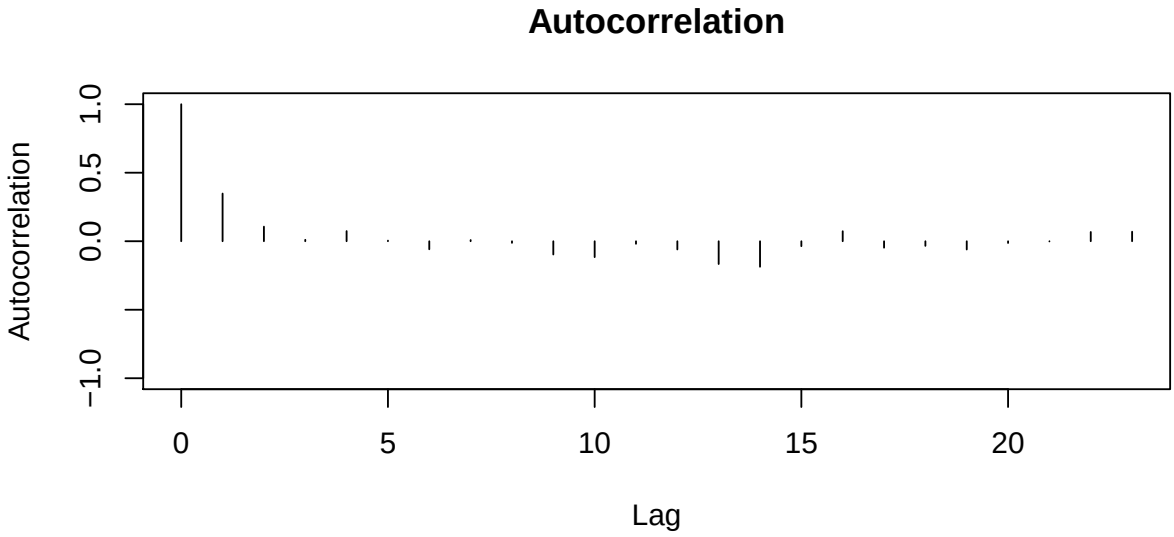


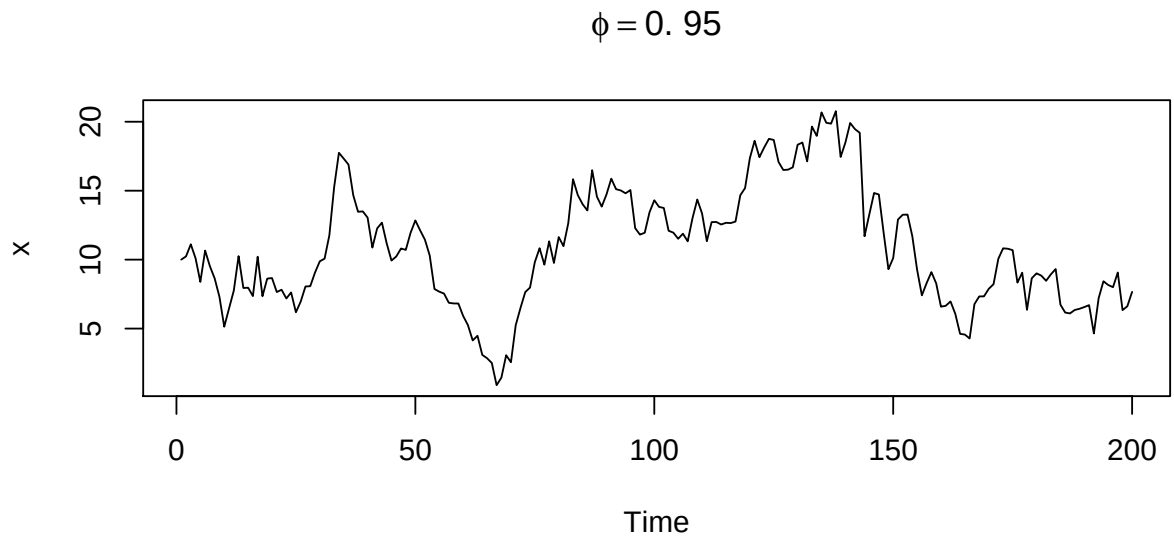


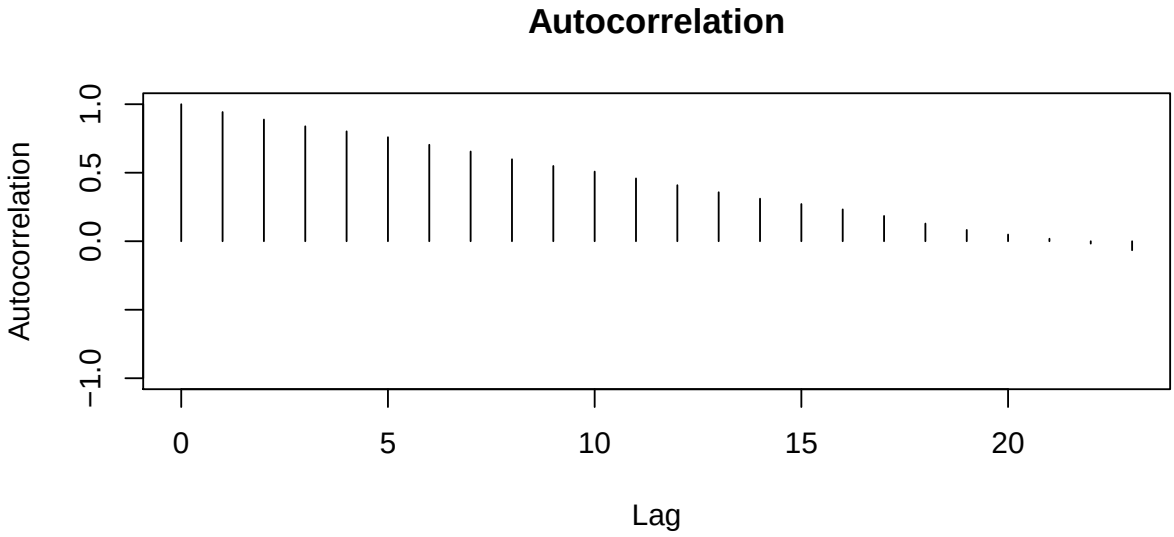

$\phi = -0.5$
x
6
8
10
12
14
0
50
100
150
200
Time


Autocorrelation
Autocorrelation
−1.0
0.0
0.5
1.0
0
5
10
15
20
Lag


$\phi = 0.3$
x
7
9
11
13
0
50
100
150
200
Time


Autocorrelation
Autocorrelation
−1.0
0.0
0.5
1.0
0
5
10
15
20
Lag





Autocorrelation is values between -1 to 1 which measure how the previous value is linearly dependent on the current values in a chain called lag. We can see from the above plots that for higher absolute values of parameter ϕ yields highly correlated chains.
As higly correlated chains yields less information about the stationary distribution. Thus we can infer that values of ϕ with in a range of 0 to 0.5 can give us less correlated values.

1.2. Posterior of 3 Parameters

In this task we are simmulating two AR(1) vectors, $x_1 : T$ with $\phi = 0.3$ and $y_1 : T$ with $\phi = 0.95$ and treating them as synthetic data. Here we are considering that μ, ϕ and σ^2 and estimating them using MCMC by implementing stan code that samples from posterior of these three unknown parameters.

Time-Series Model of $x_1 : T$

```
Inference for Stan model: be8c0f3a34c9d2b18ff4b46c20f0bc03.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
mu       10.16    0.00 0.16    9.85   10.06   10.16   10.27   10.47  1653    1
sigma2    1.88    0.00 0.20    1.53    1.74    1.87    2.01    2.28  1725    1
phi       0.35    0.00 0.07    0.22    0.31    0.35    0.40    0.49  1546    1
lp__   -160.12    0.04 1.29 -163.48 -160.74 -159.78 -159.17 -158.65   952    1

Samples were drawn using NUTS(diag_e) at Wed Aug 05 10:47:32 2020.
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).
```

Time-Series Model of $y_1 : T$

```
Inference for Stan model: be8c0f3a34c9d2b18ff4b46c20f0bc03.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
mu       10.16    0.00 0.15    9.87   10.06   10.15   10.26   10.47  1860 1.00
sigma2    1.88    0.00 0.19    1.55    1.75    1.87    2.00    2.29  1597 1.00
phi       0.35    0.00 0.07    0.22    0.31    0.35    0.40    0.49  1428 1.00
lp__   -160.08    0.05 1.28 -163.25 -160.70 -159.74 -159.15 -158.63   786 1.01

Samples were drawn using NUTS(diag_e) at Wed Aug 05 10:47:41 2020.
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).
```

1.2.1 Posterior Means and 95% Credible Intervals of the parameters

In this task we are calculating the posterior mean and 95% credible intervals of the number of effective posterior samples for the three inferred parameters for each of the simulated AR(1)-process.

Posterior Means and 95% Credible Intervals of $x_1 : T$ **With** $\phi = 0.3$

Table 1: Posterior Mean of Parameters

Mu	Sigma2	Phi
10.16383	1.879654	0.3536385

Table 2: Credible Interval

	Mu	Sigma2	Phi
Lower	9.85454	1.545293	0.2228316
Upper	10.46769	2.278967	0.4855015

Posterior Means and 95% Credible Intervals of $y_1 : T$ **With** $\phi = 0.95$

Table 3: Posterior Mean of Parameters

Mu	Sigma2	Phi
10.15961	1.884998	0.3523245

Table 4: Credible Interval

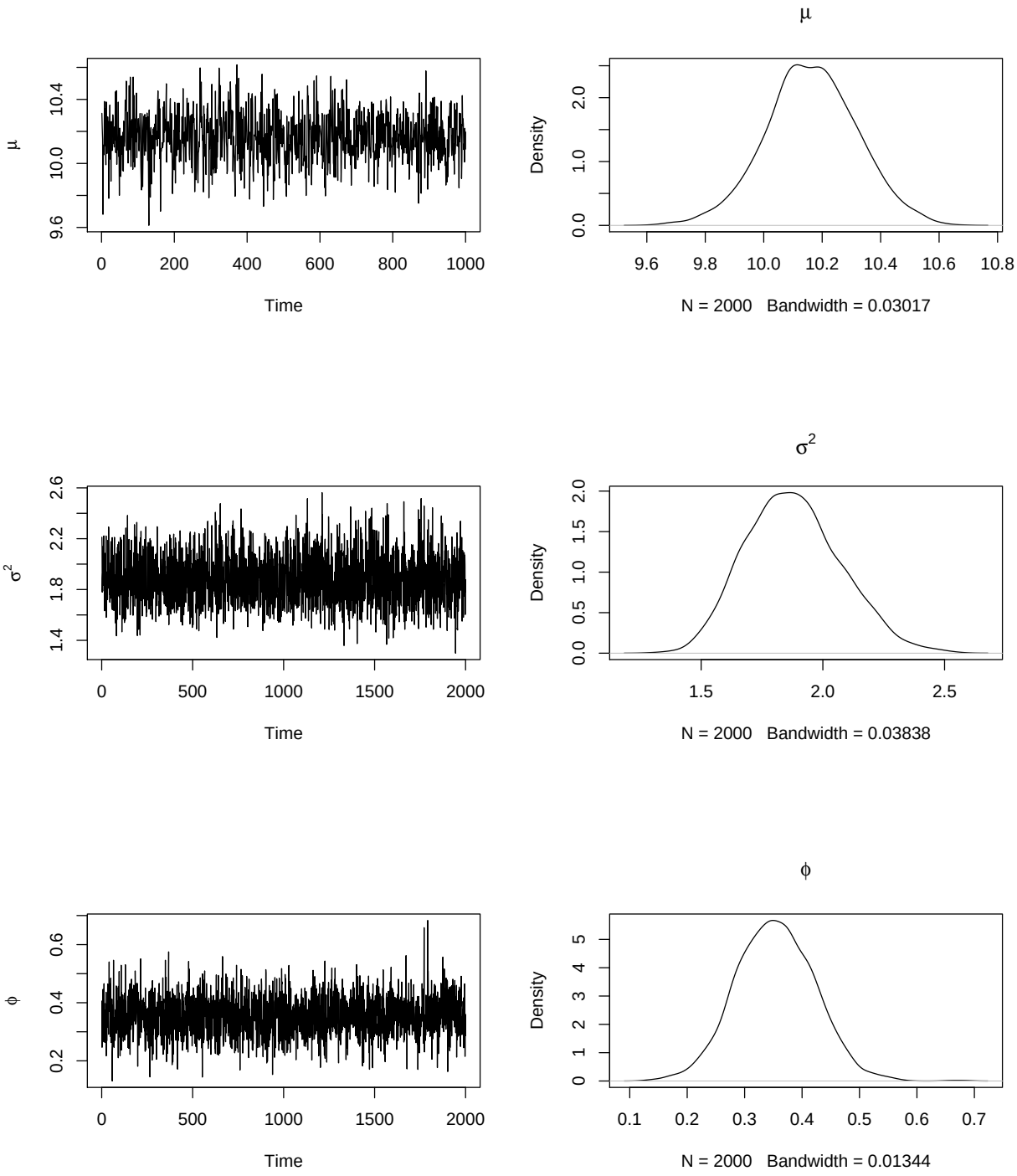
	Mu	Sigma2	Phi
Lower	9.868933	1.545293	0.2174277
Upper	10.470198	2.289158	0.4908914

We can see from above tables that the posterior means of the three parameters are colser to te actual assignes values of the three parameters and all the three parameters actual avalues lies in the 95% credible interval of the posterior draws.

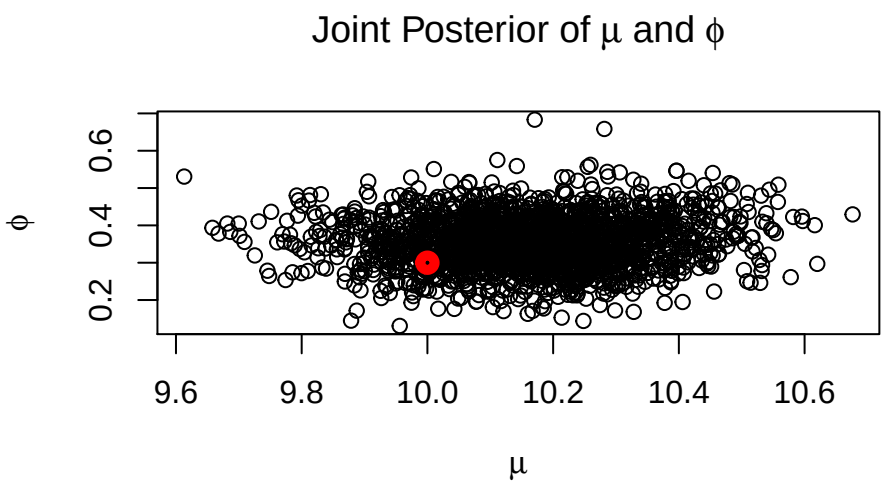
Thus we can say that we are able to estimate the true values of parameters.

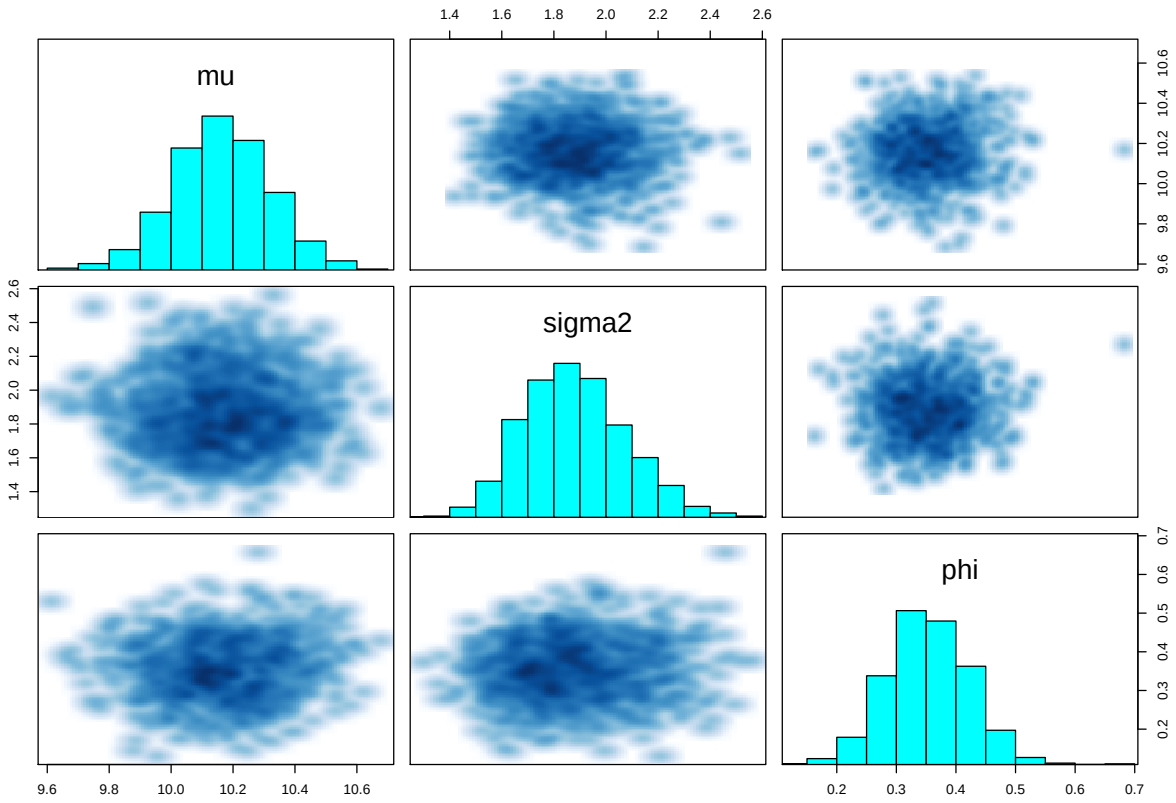
1.2.2 Convergence of the Sampler

Convergence of the sample of $x_1 : T$ **With** $\phi = 0.3$



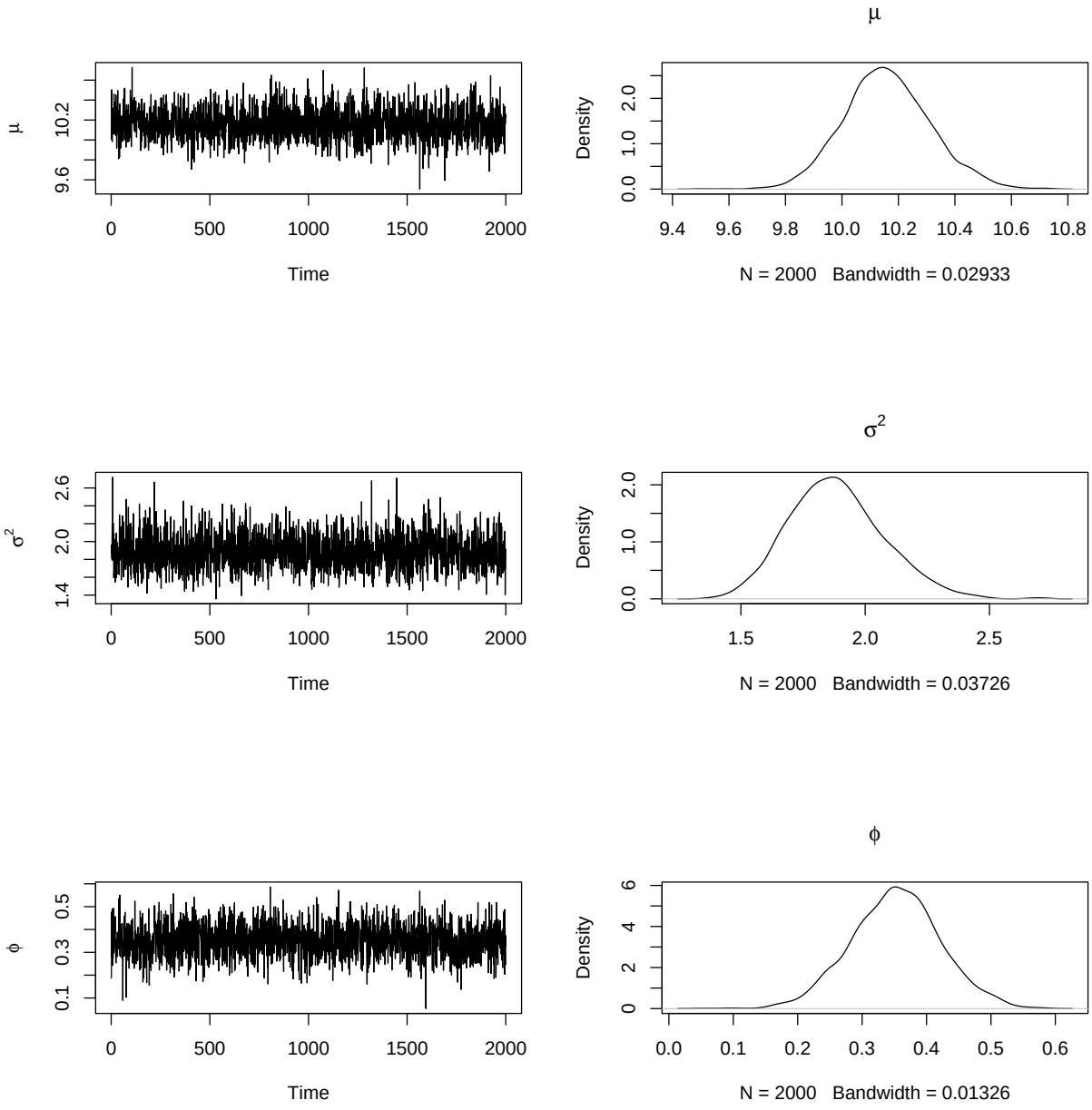
In the above shown trajectories of the parameters we can see that markov chain is traversing in a similar fashion thus we can infer that it is showing convergence to a stationary distribution.



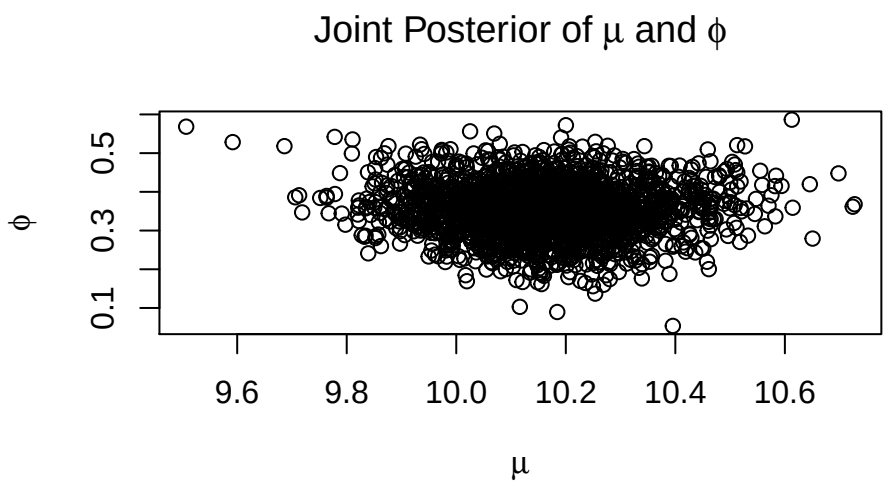


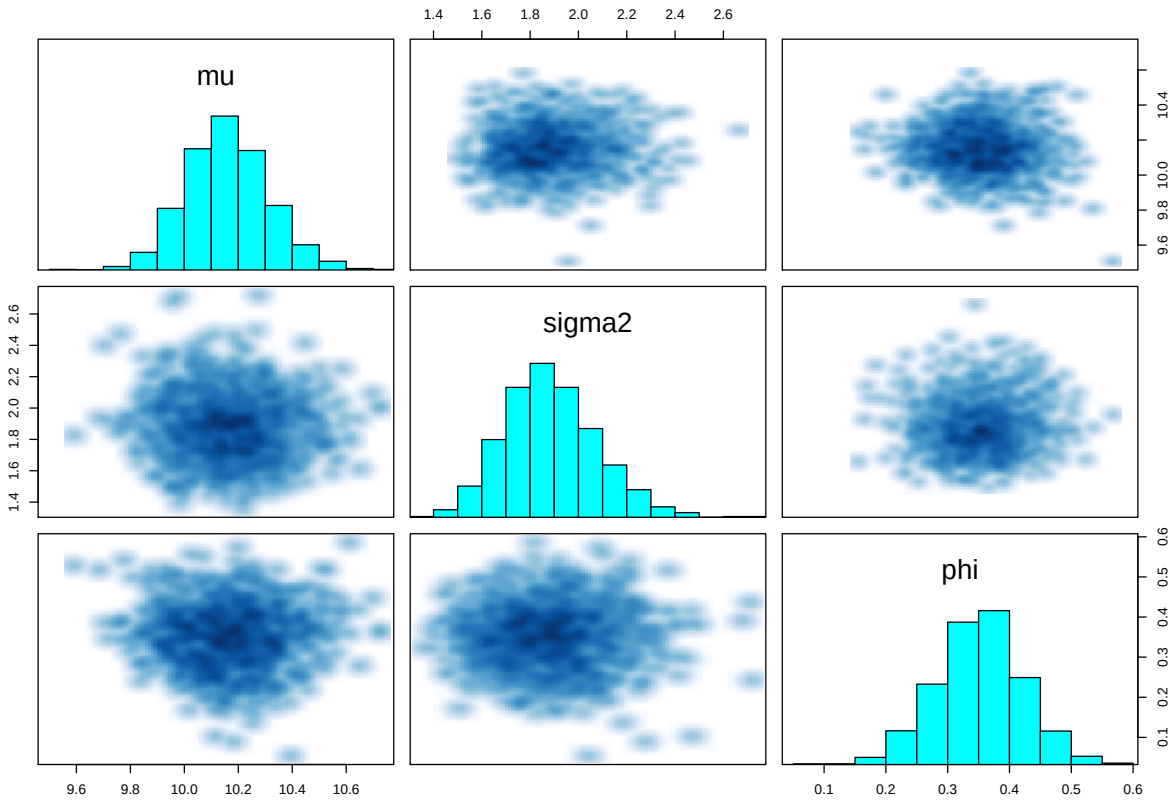
In the above shown joint posterior We can see that the drawn simmulation are centered around the initial values of the parameters which were $mu = 10$ and $\phi = 0.3$.

Convergence of the sample of $y_1 : T$ **With** $\phi = 0.95$



In the above shown trajectories of the parameters we can see that markov chain is traversing in a similar fashion thus we can infer that it is showing convergence to a stationary distribution.





In the above shown joint posterior We can see that the drawn simmulation are not centered around the initial values of the parameters which were $mu = 10$ and $\phi = 0.95$.

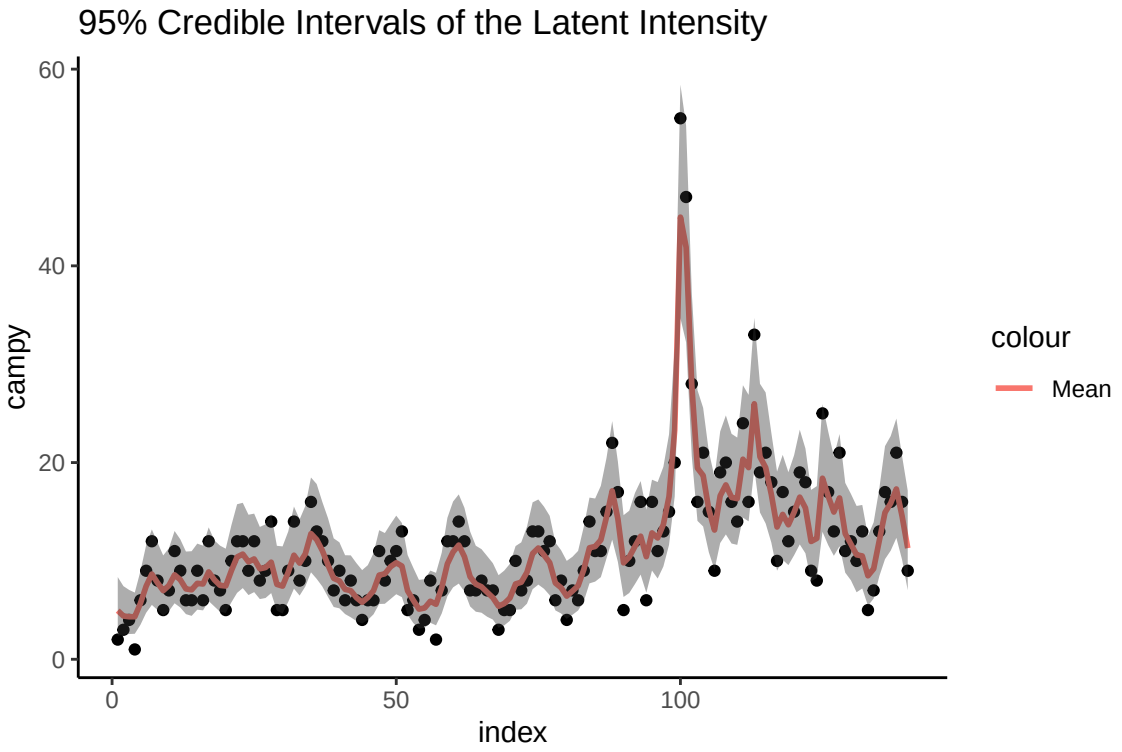
1.3. Implementing & Estimating A Poisson Model in Stan

In this task we are analyzing campy data which contains 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. Assuming 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

$$c_t|x_t \sim Poisson(e^{x_t})$$

Here x_t is an AR(1) process as in first part of the question.

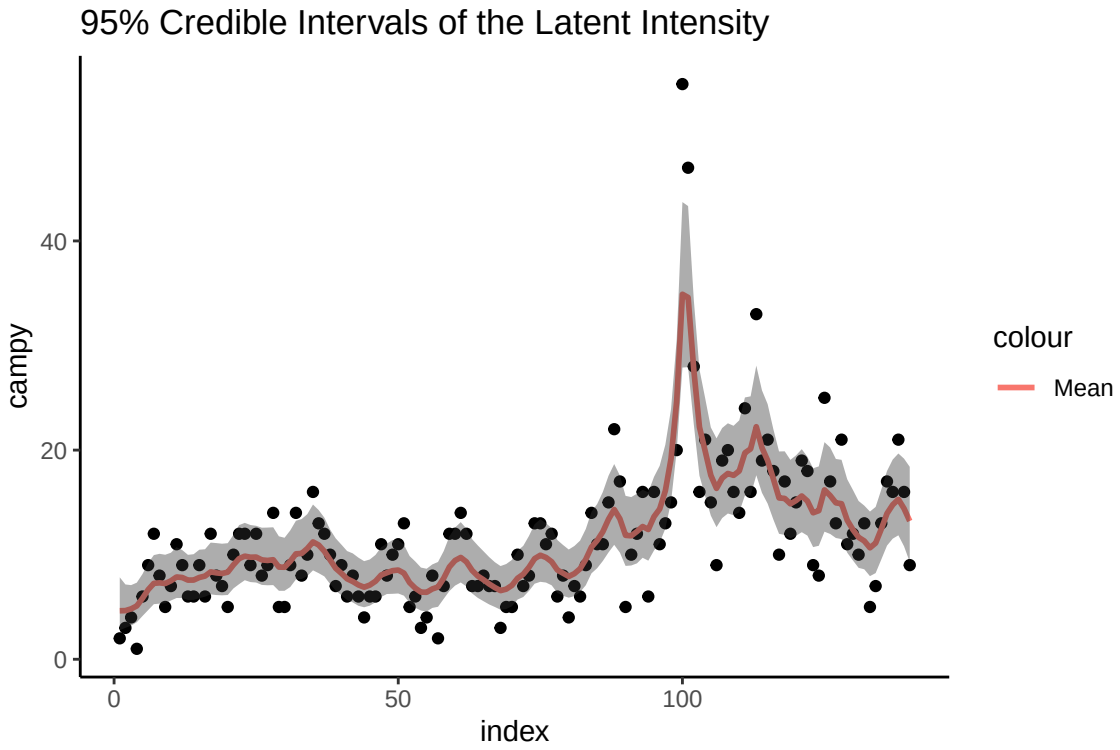
In the followin plot the 95% credible intervals for the latent intensity $\theta_t = e^{x_t}$ over time is represented.



It is evident from the plot that almost all the data points lies in the 95% credibe interval of the posterior estimates. The credible interval is growing rapidly to capture all the data points.

1.4. Implementing & Estimating A Poisson Model Wit Prior in Stan

In this task we are assuming that the true underlying latent intensity θ_t varies more smoothly than the data suggests. Thus we are setting prior $\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$ so that it becomes informative about that the AR(1)-process. We are using small increment for ϵ_t.



It is evident from the plot that many of the data points does not lie in the 95% credibe interval of the posterior estimates. The credible interval is not growing rapidly to capture all the data points.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
library(rstan)
library(latex2exp)
library(coda)
library(knitr)
library(kableExtra)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
setwd("E:/Bayesian Learning/LABS/lab_02")
AR_1 = function(nDraws=200,mu=10,sigma2=2,phi){
  x = vector(length = nDraws)
  x[1] = mu
  for (i in 2:nDraws) {
    x[i] = mu + phi*(x[i-1]-mu) + rnorm(n = 1, mean = 0, sd = sqrt(sigma2))
  }
  ts.plot(x,ylab = 'x',
          main = TeX(paste0("$\\phi\ = \ $",phi)))

  return(x)
}
a = AR_1(phi = -0.95)
autocorr.plot(as.mcmc(a), main = 'Autocorrelation')
b = AR_1(phi = -0.5)
autocorr.plot(as.mcmc(b), main = 'Autocorrelation')
x = AR_1(phi = 0.3)
autocorr.plot(as.mcmc(x), main = 'Autocorrelation')
y = AR_1(phi = 0.95)
autocorr.plot(as.mcmc(y), main = 'Autocorrelation')
#_____(b)_____#
# The input data is a vector 'y' of length 'N'
# MOdel accepts three unknown parameters

MCMC_posterior = '
data {
  int<lower=0> N;
  vector[N] X;

}

parameters {
  real mu;
  real<lower=0> sigma2;
  real phi;

}

model {
  for(i in 2:N) {
    X[i] ~  normal(mu + phi*(X[i-1] - mu), sqrt(sigma2));
```

```
  }
}
'

x_t = list(N = length(x), X=x)
y_t = list(N = length(y), Y=y)

fit1 = stan(model_code = MCMC_posterior, data = x_t, iter = 1000, warmup = 500)
print(fit1)
#_____________#
fit2 = stan(model_code = MCMC_posterior, data = x_t, iter = 1000, warmup = 500)
print(fit2)
sm2 = summary(fit2)$summary
sm = summary(fit1)$summary
#____(i)____#
#95% credible interval of x_t
CI_mu1 = c(sm[1,4], sm[1,8])
CI_sig1 = c(sm2[2,4], sm[2,8])
CI_phi1 = c(sm[3,4], sm[3,8])
CI_1 = data.frame(CI_mu1, CI_sig1, CI_phi1)
rownames(CI_1) = c('Lower', 'Upper')
colnames(CI_1) = c('Mu', 'Sigma2', 'Phi')
# posterior means
post_means1 = data.frame(sm[1,1], sm[2,1], sm[3,1])
colnames(post_means1) = c('Mu', 'Sigma2', 'Phi')
kable(post_means1, "latex",caption ='Posterior Mean of Parameters', booktabs = T) %>%
kable_styling(latex_options = "HOLD_position")
kable(CI_1, "latex", caption = 'Credible Interval', booktabs = T) %>%
kable_styling(latex_options = "HOLD_position")
sm2 = summary(fit2)$summary
#95% credible interval of  y_t
CI_mu2 = c(sm2[1,4], sm2[1,8])
CI_sig2 = c(sm2[2,4], sm2[2,8])
CI_phi2 = c(sm2[3,4], sm2[3,8])
CI_2 = data.frame(CI_mu2, CI_sig2, CI_phi2)
rownames(CI_2) = c('Lower', 'Upper')
colnames(CI_2) = c('Mu', 'Sigma2', 'Phi')
#posterior means
post_means2 = data.frame(sm2[1,1], sm2[2,1], sm2[3,1])
colnames(post_means2) = c('Mu', 'Sigma2', 'Phi')
kable(post_means2, "latex",caption ='Posterior Mean of Parameters', booktabs = T) %>%
kable_styling(latex_options = "HOLD_position")
kable(CI_2, "latex",caption ='Credible Interval', booktabs = T) %>%
kable_styling(latex_options = "HOLD_position")
#____(ii)_____#
set.seed(123456)
draws1 = extract(fit1)
mu1 = draws1$mu
phi1 = draws1$phi
sig1 = draws1$sigma2

# convergence of the sample x_t
par(mfrow =c(1,2))
```

```
ts.plot(mu1[1:1000], ylab = expression(mu))
plot(density(mu1), main = expression(mu))

par(mfrow=c(1,2))
ts.plot(sig1, ylab = expression(sigma^2))
plot(density(sig1), main = expression(sigma^2))

par(mfrow=c(1,2))
ts.plot(phi1, ylab = expression(phi))
plot(density(phi1), main = expression(phi))
# joint posterior of mu and phi x_t
plot(x = mu1, y = phi1, xlab = expression(mu), ylab = expression(phi),
     main = TeX(paste("Joint Posterior of $\\mu$ and $\\phi$")))
points(x = 10, y = 0.3, col = 'red', lwd= 5)
pairs(fit1,pars = c("mu","sigma2", "phi"))
#_____(ii)____#
set.seed(123456)
draws2 = extract(fit2)
mu2 = draws2$mu
phi2 = draws2$phi
sig2 = draws2$sigma2

# convergence of the sample y_t
par(mfrow=c(1,2))
ts.plot(mu2, ylab = expression(mu))
plot(density(mu2), main = expression(mu))

par(mfrow=c(1,2))
ts.plot(sig2, ylab = expression(sigma^2))
plot(density(sig2), main = expression(sigma^2))

par(mfrow=c(1,2))
ts.plot(phi2, ylab = expression(phi))
plot(density(phi2), main = expression(phi))
# joint posterior of mu and phi y_t
plot(x = mu2, y = phi2, xlab = expression(mu), ylab = expression(phi),
     main = TeX(paste("Joint Posterior of $\\mu$ and $\\phi$")))
pairs(fit2,pars = c("mu","sigma2", "phi"))
#_______(c)______#
data = read.table("campy.dat", header = T)
campy = as.vector(data$c)
N = length(campy)
postData = list(N = N, c = campy)

poisson_stan = '
data {
  int<lower=0> N;
  int  c[N];
}

parameters {
  real x[N];
  real mu;
```

```
  real phi;
  real<lower=0> sigma2;

}

model {
  x[1] ~ normal(mu, sqrt(sigma2));
  c[1] ~ poisson(exp(x[1]));
  for(i in 2:N) {
    x[i] ~ normal(mu + phi*(x[i-1] - mu), sqrt(sigma2));
    c[i] ~ poisson(exp(x[i]));

  }
}
'
fit = stan(model_code = poisson_stan, data = postData,
           iter = 1000, warmup = 500)
sm = summary(fit)$summary
Mean = unlist(sapply(c(sm[1:N,1]), function(x) {exp(x)}))
lower = unlist(sapply(c(sm[1:N,4]), function(x) {exp(x)}))
upper = unlist(sapply(c(sm[1:N,8]), function(x) {exp(x)}))
df = cbind(index = 1:N,campy, Mean, lower, upper)
ggplot(as.data.frame(df))+ theme_classic()+
  geom_point(mapping = aes(x = index, y = campy))+
  geom_line(mapping = aes(x = index, y = Mean, col = 'Mean'), size = 1)+
  geom_ribbon(mapping = aes(x= index, ymin= lower, ymax =upper),alpha = 0.4)+          ggtitle('95% Cre
#________(d)_________#
poissonPosterior_prior = '

data {
  int<lower=0> N;
  int  c[N];
  real<lower=0> v0;
  real<lower=0> sigma20;
}

parameters {
  real x[N];
  real mu;
  real phi;
  real<lower=0> sigma2;

}

model {

  sigma2 ~ scaled_inv_chi_square(v0, sqrt(sigma20));
  x[1] ~ normal(mu, sqrt(sigma2));
  c[1] ~ poisson(exp(x[1]));
  for(i in 2:N) {
    x[i] ~ normal(mu + phi*(x[i-1] - mu), sqrt(sigma2));
    c[i] ~ poisson(exp(x[i]));
```

```
  }
}
'
priorData = list(N = N, c = campy, v0 = 50, sigma20 = 0.01)

fit1 = stan(model_code = poissonPosterior_prior,data = priorData,
            iter = 1000, warmup = 500)
sm1 = summary(fit1)$summary
Mean1 = unlist(sapply(c(sm1[1:N,1]), function(x) {exp(x)}))
lower1 = unlist(sapply(c(sm1[1:N,4]), function(x) {exp(x)}))
upper1 = unlist(sapply(c(sm1[1:N,8]), function(x) {exp(x)}))
df = cbind(index = 1:N,campy, Mean1, lower1, upper1)
ggplot(as.data.frame(df))+ theme_classic()+
  geom_point(mapping = aes(x = index, y = campy))+
  geom_line(mapping = aes(x = index, y = Mean1, col = 'Mean'), size = 1)+
  geom_ribbon(mapping = aes(x= index, ymin= lower1, ymax =upper1),alpha = 0.4)+
  ggtitle('95% Credible Intervals of the Latent Intensity')
```