Bayesian Learning - lab4

Joris van Doorn , Weng Hang Wong

4/15/2020

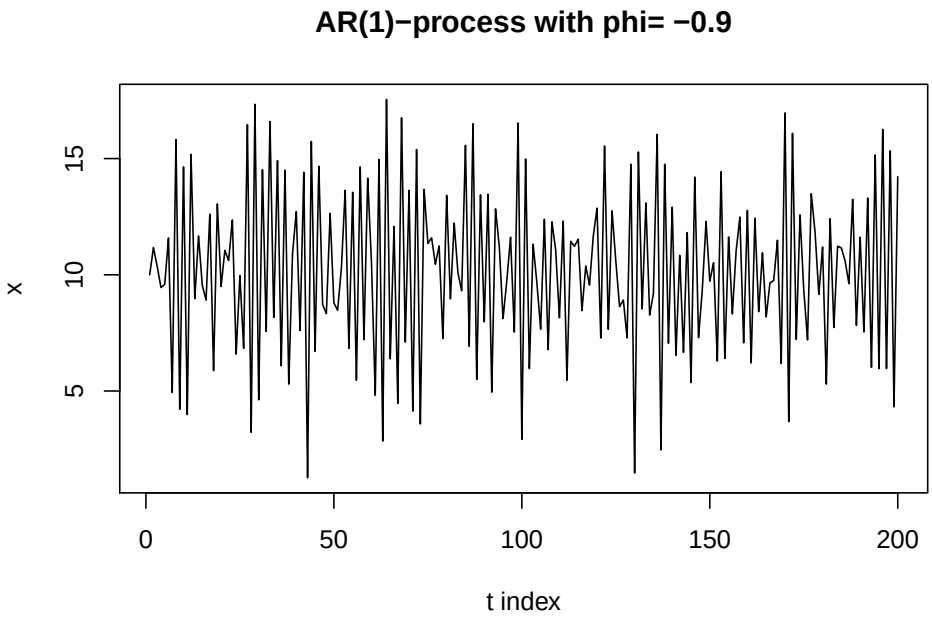
1. Time series models in Stan

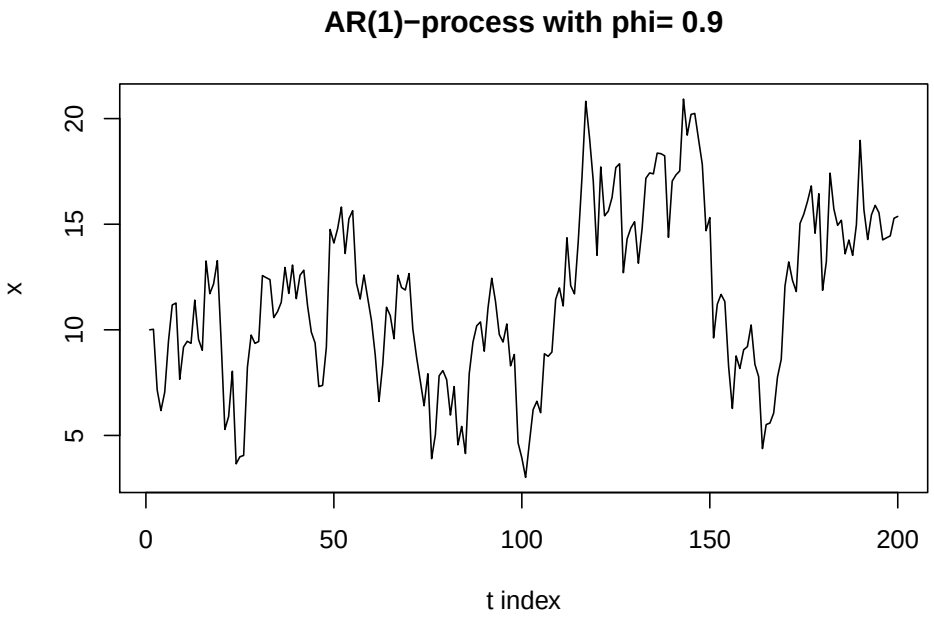
(a) Write a function in R that simulates data from the AR(1)-process

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

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* $T = 2000$ 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 $x_{1:T}$?

Here, we using the given value to simulate the AR(1)-process, where the values of ϕ is between -1 and 1, so we take the ϕ as -0.9 and 0.9, so that to observe the difference between two plots.ϕ is a momentum parameters that can be used to ajust the algorithm, According to the below two plots, it's very obviously to see that with a low value of $\phi = -0.9$ is moving on the plot very intensively. On the other hand, with a higher value $\phi = 0.9$, it comes with a less difference in each iteration so it moves more slowly.





(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 your simulated vectors as synthetic data, and 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: Lookat the time-series models examples in the Stan user's guide/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?

```
## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.19.3, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)

## Inference for Stan model: 4fd8f8e983e46e9858faca2f63abcb18.
## 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       10.00    0.00 0.24    9.52    9.84    9.99   10.16   10.48  3412
## phi       0.41    0.00 0.07    0.27    0.36    0.41    0.45    0.53  3511
```

```
## sigma2    3.98    0.01 0.40    3.29    3.70    3.95    4.22    4.86  3844
## lp__   -238.57    0.03 1.24 -241.77 -239.18 -238.25 -237.65 -237.15  1929
##        Rhat
## mu        1
## phi       1
## sigma2    1
## lp__      1
##
## Samples were drawn using NUTS(diag_e) at Tue May 26 15:12:01 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).
```

Table 1: Posterior of x

parameters	mean	X95CI	EffectSamples	TrueValue
mu	10.43	(9.98,10.89)	3609	9.996674
phi	0.35	(0.22,0.48)	3575	0.300000
sigma2	4.26	(3.52,5.16)	3550	4.637813

```
## Inference for Stan model: 4fd8f8e983e46e9858faca2f63abcb18.
## 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       17.79    1.19 22.08   -5.68   11.90   14.26   17.65   73.92   347
## phi       0.96    0.00  0.03    0.90    0.94    0.96    0.98    1.00   421
## sigma2    4.17    0.01  0.43    3.40    3.87    4.14    4.42    5.11   835
## lp__   -245.38    0.14  2.23 -251.03 -246.49 -244.71 -243.71 -242.88   259
##        Rhat
## mu     1.01
## phi    1.01
## sigma2 1.00
## lp__   1.03
##
## Samples were drawn using NUTS(diag_e) at Tue May 26 15:12:04 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).
```

Table 2: Posterior of y

parameters	mean	CI	EffectSamples	TrueValue
mu	15.67	(-41.45,81.54)	588	12.47552
phi	0.98	(0.93,1.00)	430	0.95000
sigma2	4.89	(3.99,6.07)	342	29.43775

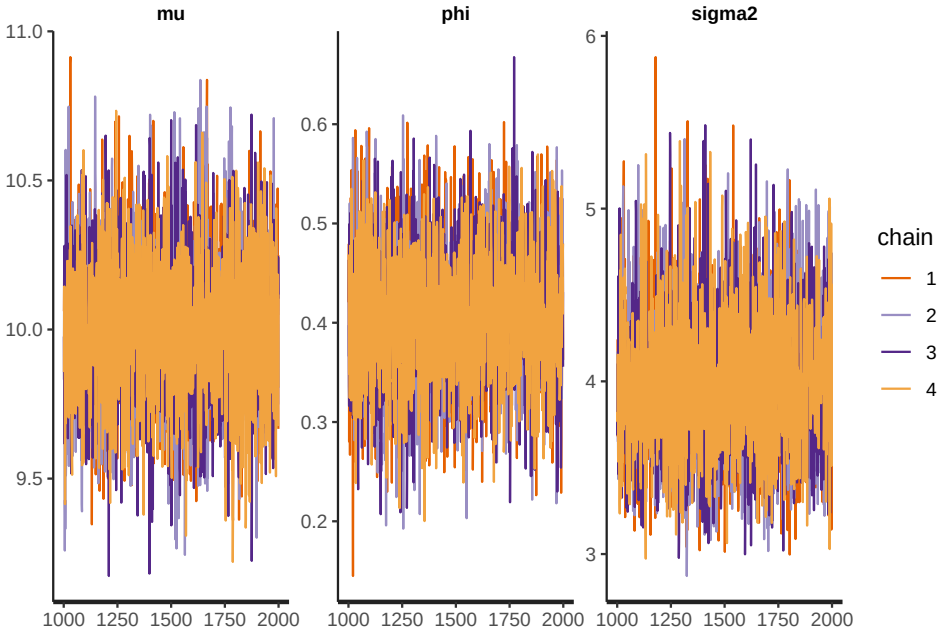
In 2000 iterations, we use rtan to simulate the posterior parameters μ, ϕ *and* σ^2. Look at the table above we can see the posterior mean, 95% CI and effective samples from the simulations. Comparing the posterior mean with the true value from (a), we can see that the simulation from $x_{1:T}$ has the better estimation then

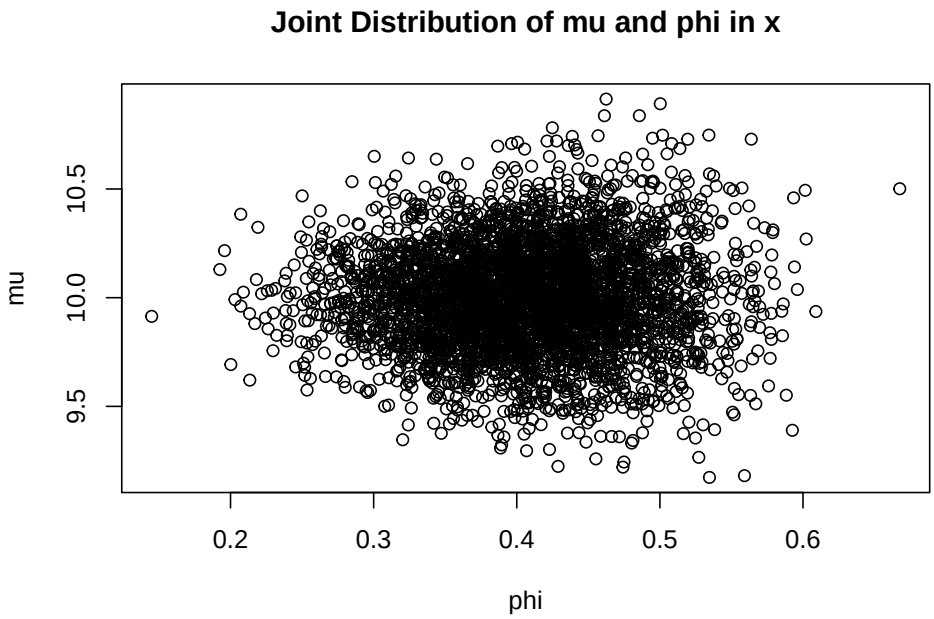
from $y_{1:T}$, especially the σ^2 of y true value is 64.196, compare to the simulation which is 4.89, it is because we set the prior of $\sigma^2 \sim$ Normal(0,10). That's the reason why it lower down the accurracy.

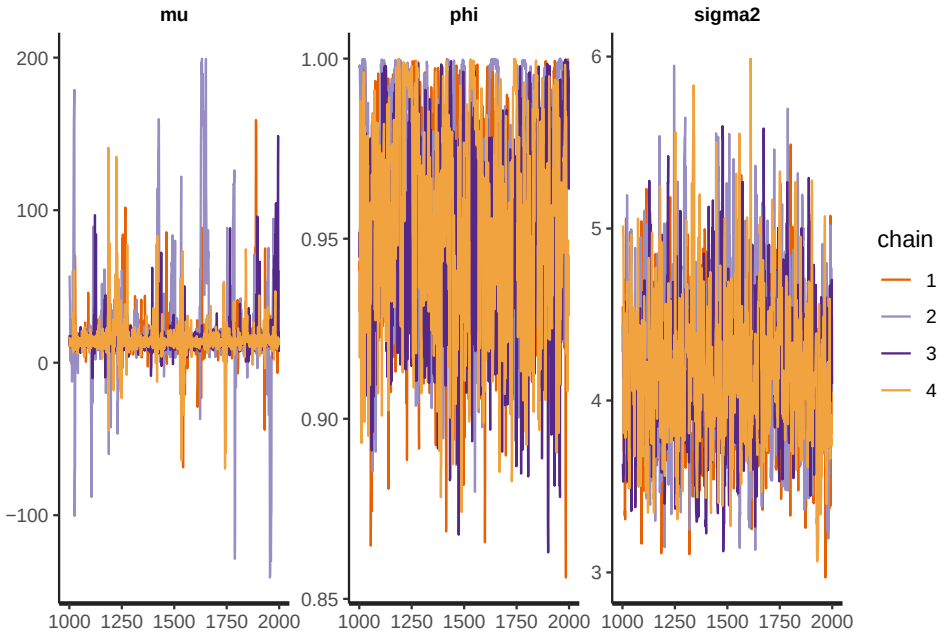
ii. For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of μ and ϕ. Comments?

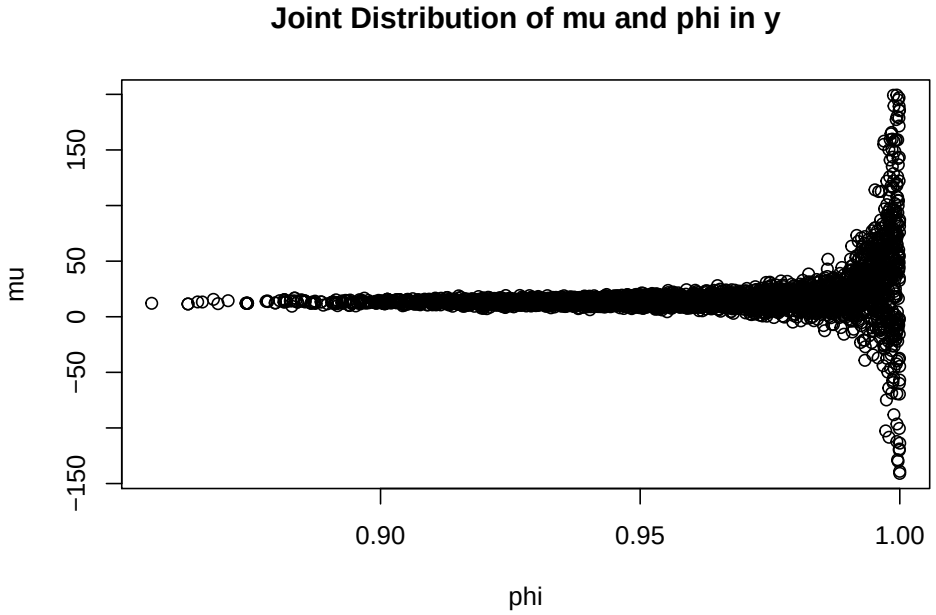
First, we look at the x_T traceplot and it's joint distribution. The traceplot within 4 chains are converaged well and looks good. Along with the joint distribution with μ and ϕ, the scatter points are concentrated.

However, when we look at the y_T traceplot, we can see the parameters are not converaged well, especially in the μ one, along the 4 chains iterations, the lines are not covered each other. As well as the joint distribution plot between μ and ϕ, which tells us the simulations are not varied a lot and concentrate to the same value for each parameter.




Joint Distribution of mu and phi in x
mu
10.5
10.0
9.5
0.2
0.3
0.4
0.5
0.6
phi


mu
phi
sigma2
200
100
0
−100
1.00
0.95
0.90
0.85
6
5
4
3
1000 1250 1500 1750 2000
chain
1
2
3
4



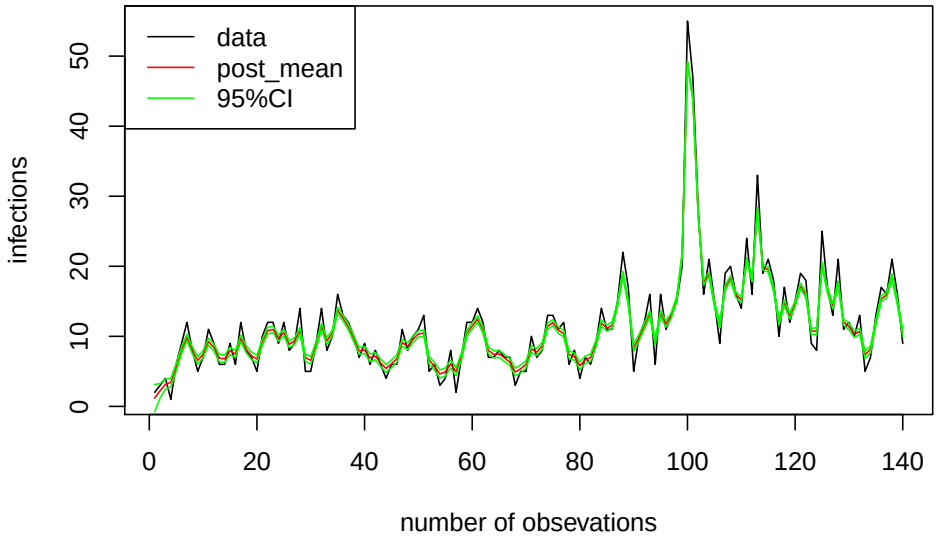
(c) The data campy.dat contain the number of cases of campylobacter infections in the north of the province Quebec (Canada) in four week intervals from January 1990 to the end of October 2000. It has 13 observations per year and 140 observations in total. Assume that the number of infections c_t at each 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?]

We implement the possion model in Stan, after we compute the posterior mean, posterior sd from the x_t parameter, than we get the 95% CI interval here in the plot. Since the prior is still non-informative in here, we use the same setting for the parameters as above. Which are: mu ~ normal(10,100); phi ~ normal(0,10); sigma2 ~ scaled_inv_chi_square(1,2);

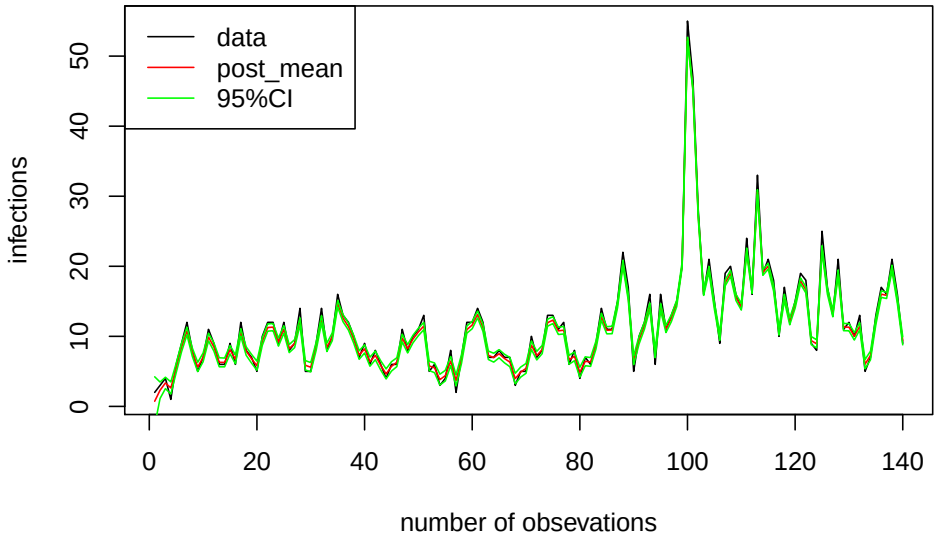
Under non-informative parameters setting, the Posterior mean and the 95% credible interval are not nicely fitting the data. we can see the CI are out of the data and the mean is very high.



(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?

Here, the parameters become informative so that we adjust the parameter setting in the stan model, which are: mu ~ normal(10,100); phi ~ normal(0,10); sigma2 ~ scaled_inv_chi_square(50,1);

Comparing the plot with (c), the Posterior Mean is more accurrate and the Credible interval are relatively narrow down within the data.



Appendix

```
knitr::opts_chunk$set(echo = TRUE)
#1.a

set.seed(12345)

#given values
mu=10
sigma2=2
t=200

AR_1_process = function(mu, sigma2, t, phi){
  x = c()
  x[1] = mu
  for(i in 2:t){
    eps = rnorm(1,0,sigma2)
    x[i] = mu +phi*(x[i-1]-mu)+eps
  }
  return(x)
}


# plot and compare with different phi
for(i in c(-0.9,0.9)){
  AR_1 = AR_1_process(mu,sigma2,t,phi=i)
  plot(AR_1,type="l", main=paste("AR(1)-process with phi=",i), xlab="t index", ylab="x"  )

}
```

```r
#1.b
library(rstan)

AR_x = AR_1_process(mu, sigma2,t,phi = 0.3)
AR_y = AR_1_process(mu,sigma2,t,phi = 0.95)

#plot(AR_x,type="l")
#plot(AR_y,type="l")


#Stan to simulate mu, sigam2,phi
#AR(1)model
#https://mc-stan.org/docs/2_23/stan-users-guide/autoregressive-section.html

StanModel = '
data {
  int<lower=0> N;
  vector[N] y;
}
parameters {
  real mu;
  real<lower=-1,upper=1> phi;
  real<lower=0> sigma2;
}
model {
  mu ~ normal(10,100);  //non-informative, larger sigma2
  phi ~ normal(0,10); //-1 1
  sigma2 ~ scaled_inv_chi_square(1,2);
  for (n in 2:N)
    y[n] ~ normal(mu + phi * (y[n-1]-mu), sqrt(sigma2));
}'
# i

## fit the AR x samples
N_x = length(AR_x)
dataX = list(N=N_x, y=AR_x)
burnin=1000
niter=2000
fit_x=stan(model_code = StanModel,data=dataX,warmup=burnin, iter=niter, chains = 4)

## fit the AR y samples
N_y = length(AR_y)
dataY = list(N=N_y,y=AR_y)
fit_y = stan(model_code = StanModel,data=dataY,warmup=burnin, iter=niter, chains = 4)

#print(fit, digits_summary=3)
# Extract posterior samples
postDraws_x = extract(fit_x)
postDraws_y = extract(fit_y)

print(fit_x)
```

```r
library(knitr)
post_x=data.frame(parameters=c("mu","phi","sigma2"),mean=c(10.43,0.35,4.26),"95CI"=c("(9.98,10.89)","(0
kable(post_x,caption="Posterior of x")


print(fit_y)
post_y = data.frame(parameters=c("mu","phi","sigma2"),mean=c(15.67,0.98,4.89),"CI"=c("(-41.45,81.54)","
kable(post_y,caption="Posterior of y")

# ii
traceplot(fit_x, main="Posterior of x")

plot(mu~phi,data=fit_x, main="Joint Distribution of mu and phi in x")


traceplot(fit_y, main="posterior of y")
plot(mu~phi, data=fit_y, main="Joint Distribution of mu and phi in y")


# 1.c

data = read.table("campy.dat", header=T)

#Poisson stan model

StanModel_poi = '
data {
  int<lower=0> N;
  int<lower=0> c[N];
}
parameters {
  real mu;
  real<lower=-1, upper=1> phi;
  real<lower=0> sigma2;
  real x[N]; //para in possion
}
model {
  mu ~ normal(10,100);
  phi ~ normal(0,10);
  sigma2 ~ scaled_inv_chi_square(1,2);
  for (n in 2:N){
    x[n] ~ normal(mu+phi*(x[n-1]-mu),sqrt(sigma2));
    c[n] ~ poisson(exp(x[n]));
  }
}'

## fit the Possion conditioned on
N = length(data$c)
data= list(N=N, c=data$c)
burnin=1000
niter=2000

library(rstan)
```

```
fit_poi = stan(model_code = StanModel_poi, data = data, warmup = burnin, iter = niter, chains = 4)

#print(fit_poisson)

#Extract posterior samples
post_poi = extract(fit_poi)

# Posterior mean/sd of exp(x_t)
post_mean = exp(colMeans(post_poi$x))
post_sd = apply(post_poi$x, 2, sd)

# 95% CI% mean +_ 1.96*sd
upperCI= post_mean + 1.96*post_sd
lowerCI= post_mean - 1.96*post_sd

#plot data + posterior mean + 95%CI
plot(data$c,type="l", xlab="number of obsevations",ylab="infections")
lines(post_mean, col="red")
lines(upperCI, col="green")
lines(lowerCI, col="green")
legend("topleft", legend=c("data","post_mean","95%CI"), col=c("black","red","green"), lwd=1)


#1.d


##with prior belief

StanModel_poi2 = '
data {
  int<lower=0> N;
  int<lower=0> c[N];
}
parameters {
  real mu;
  real<lower=-1, upper=1> phi;
  real<lower=0> sigma2;
  real x[N]; //para in possion
}
model {
  mu ~ normal(10,100);
  phi ~ normal(0,10);
  sigma2 ~ scaled_inv_chi_square(50,1); //informative
  for (n in 2:N){
    x[n] ~ normal(mu+phi*(x[n-1]-mu),sqrt(sigma2));
    c[n] ~ poisson(exp(x[n]));
  }
}'

## fit the Possion conditioned on
N = length(data$c)
data= list(N=N, c=data$c)
burnin=1000
```

```
niter=2000

library(rstan)
fit_poi2 = stan(model_code = StanModel_poi2, data = data, warmup = burnin, iter = niter, chains = 4)




#Extract posterior samples
post_poi = extract(fit_poi2)

# Posterior mean/sd of exp(x_t)
post_mean = exp(colMeans(post_poi$x))
post_sd = apply(post_poi$x, 2, sd)

# 95% CI% mean +_ 1.96*sd
upperCI= post_mean + 1.96*post_sd
lowerCI= post_mean - 1.96*post_sd

#plot data + posterior mean + 95%CI
plot(data$c,type="l", xlab="number of obsevations",ylab="infections")
lines(post_mean, col="red")
lines(upperCI, col="green")
lines(lowerCI, col="green")
legend("topleft", legend=c("data","post_mean","95%CI"), col=c("black","red","green"), lwd=1)
```

‘‘‘