Lab4

Andreas C Charitos

24 May 2019

1. Time series models in Stan

a)

Write a function in R that simulates data from the $AR(1)$-process $x_t \sim= \mu+\phi(x_{t-1}-\mu)+\epsilon_t,\ \epsilon_t \sim^{iid} N(0,\sigma^2)$, 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 timepoints.Use $\mu = 10$,$\sigma^2 = 2$ and $T = 200$ and look at some different realizations (simulations) of $x_{1:T}$ for values of ϕ between and 1 (this is the interval where the $AR(1)-process$ is stable).Include a plot at least one realization in the report.What effect does the value of ϕ have on $x_{1:T}$

```
# a) ------------------------------------------------------------------
require(rstan)
```

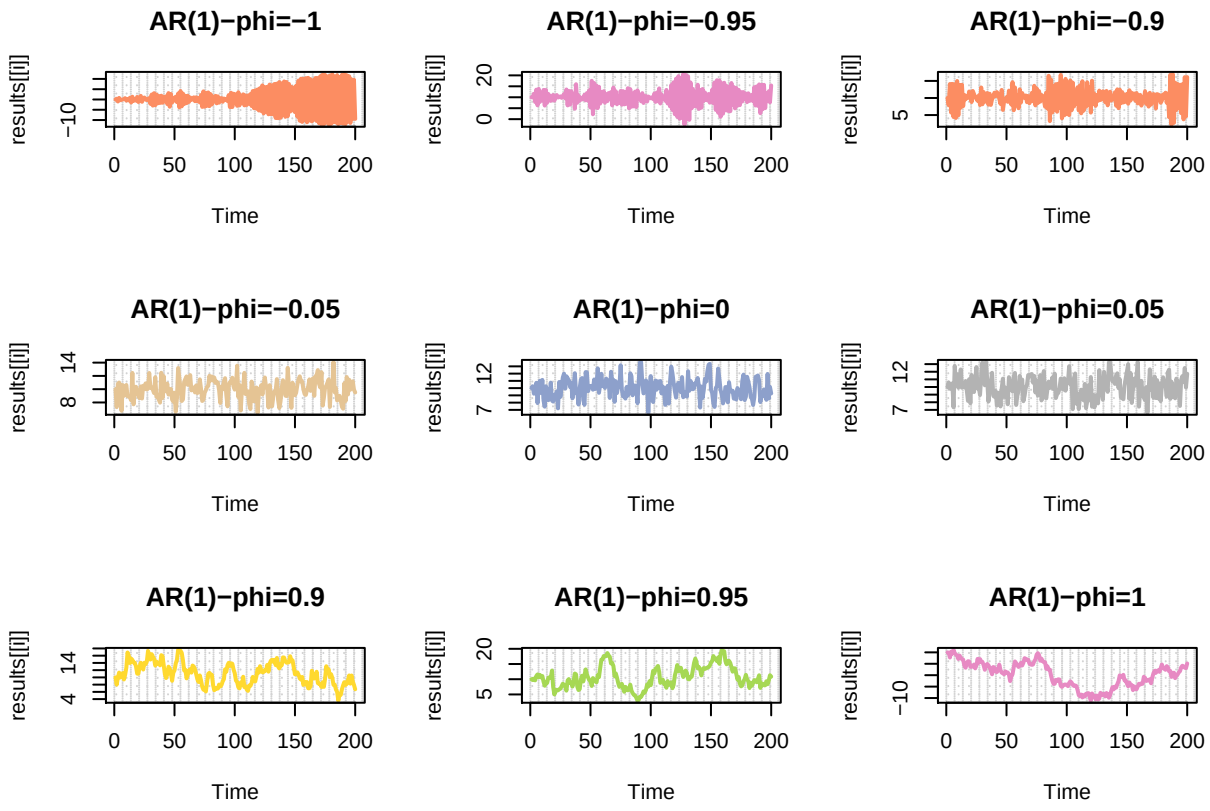
```
## Warning: package 'rstan' was built under R version 3.5.3
```

```
## Warning: package 'StanHeaders' was built under R version 3.5.3
```

```
require(RColorBrewer)
pal<-brewer.pal(8,"Set2")

# set prior values
mu=10
Sigma2=2
phi_grid=seq(-1,1,by=0.05);phi_grid=round(phi_grid,2)
Taf=200

# AR(1)-process simulation
ARSim<-function(mu,Sigma2,phi,nIter){
  xt=double(Taf)
  xt[1]=mu
  for(i in 2:nIter){
  xt[i]=mu+phi*(xt[i-1]-mu)+rnorm(1,0,sqrt(Sigma2))
  }
  L=list(xt)
  names(L)[1] <- paste("AR(1)-phi=",phi,sep="")
  return(L)
}

# apply the AR function at phi_grid vector
results<-sapply(phi_grid, function(phi) ARSim(mu,Sigma2=Sigma2,phi=phi,nIter=Taf))
nPhi<-length(results)
# Take the index for some values of phi
indx<-which(phi_grid%in%c(-1.00,-0.95,-0.90,-0.05,0.00,0.05,0.90,0.95,1.00) )
```

```
# # plot AR(1)-process
# par(mfrow=c(round(length(indx)/3),3))
# for(i in indx){
#   plot(results[[i]],main=names(results)[i],type="l",
#        col=sample(pal,1),panel.first=grid(25,25),lwd=2)
```

```
# }

# plot AR(1)-process as Time series
par(mfrow=c(round(length(indx)/3),3))
for(i in indx){
  plot.ts(results[[i]],main=names(results)[i],type="l",
          col=sample(pal,1),panel.first=grid(25,25),lwd=2)
}
```



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 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.

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?

```
# b) ------------------------------------------------------------

MCMC_model='
data{
int<lower=0> N;
vector[N] y;
```

```
}
parameters{
real mu;
real<lower=0> sigma;
real phi;
}
model{
for(n in 2:N)
y[n] ~ normal(mu + phi*(y[n-1]-mu), sqrt(sigma));

}'

# use ARSim with the 2 values of phi
ar1=ARSim(mu,Sigma2,0.3,Taf)
ar2=ARSim(mu,Sigma2,0.95,Taf)

#N1=length(ar1)
# create lists for the 2 ARSim
data1<-list(N = Taf,
           y=ar1[[1]] )

data2<-list(N=Taf,
              y=ar2[[1]])
# fit models
fit.mod1<-stan(model_code = MCMC_model, # Stan model
              data = data1,              # named list of data
              chains = 1,              # number of Markov chains
              warmup = 1000,           # number of warmup iterations per chain
              iter = 2000,             # total number of iterations per chain
              cores = 2,               # number of cores (could use one per chain)
              refresh = 0 )


fit.mod2<-stan(model_code = MCMC_model, # Stan model
               data = data2,              # named list of data
               chains = 1,              # number of Markov chains
               warmup = 1000,           # number of warmup iterations per chain
               iter = 2000,             # total number of iterations per chain
               cores = 2,               # number of cores (could use one per chain)
               refresh = 0 )
```

```
## Warning: There were 10 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

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

```
# extract informations from models
model_stats1<-extract(fit.mod1)
#
model_stats2<-extract(fit.mod2)
```

Table with 95% for $\phi = 0.3$

```r
# i)
## 95% credible intervals
library(knitr)
# apply quantile function
q1_mu=quantile(model_stats1$mu,probs=c(0.025,0.975)) # for mu
q1_sigma=quantile(model_stats1$sigma,probs=c(0.025,0.975)) # for sigma
q1_phi=quantile(model_stats1$phi,probs=c(0.025,0.975)) # for phi


q2_mu=quantile(model_stats2$mu,probs=c(0.025,0.975)) # for mu
q2_sigma=quantile(model_stats2$sigma,probs=c(0.025,0.975)) # for sigma
q2_phi=quantile(model_stats2$phi,probs=c(0.025,0.975)) # for phi

# create df
dataFrame1=data.frame( rbind(q1_mu,q1_sigma,q1_phi) )
colnames(dataFrame1)<-c("2.5%","97.5%")
rownames(dataFrame1)<-NULL
rownames(dataFrame1)<-c("mu","sigma","phi")
#
kable(dataFrame1,caption = "Table for phi=0.3")
```

Table 1: Table for phi=0.3

	2.5%	97.5%
mu	9.6501400	10.2661785
sigma	1.7059787	2.4691775
phi	0.2370603	0.5266308

Table with 95% CI for $\phi = 0.95$

```r
#
dataFrame2=data.frame( rbind(q2_mu,q2_sigma,q2_phi) )
colnames(dataFrame2)<-c("2.5%","97.5%")
rownames(dataFrame2)<-NULL
rownames(dataFrame2)<-c("mu","sigma","phi")
#
kable(dataFrame2,caption = "Table for phi=0.95")
```

Table 2: Table for phi=0.95

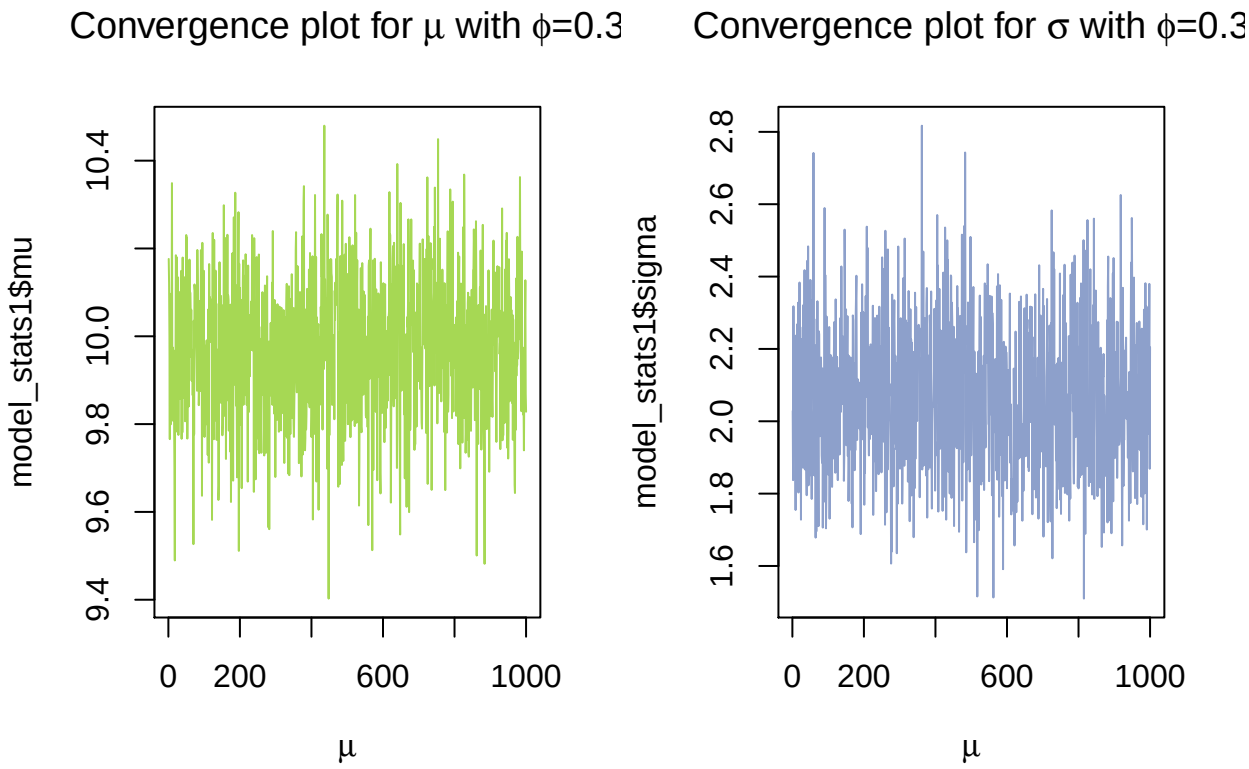
	2.5%	97.5%
mu	-22.5157535	48.398548
sigma	1.9297084	2.883460
phi	0.8996814	1.005743

ii)

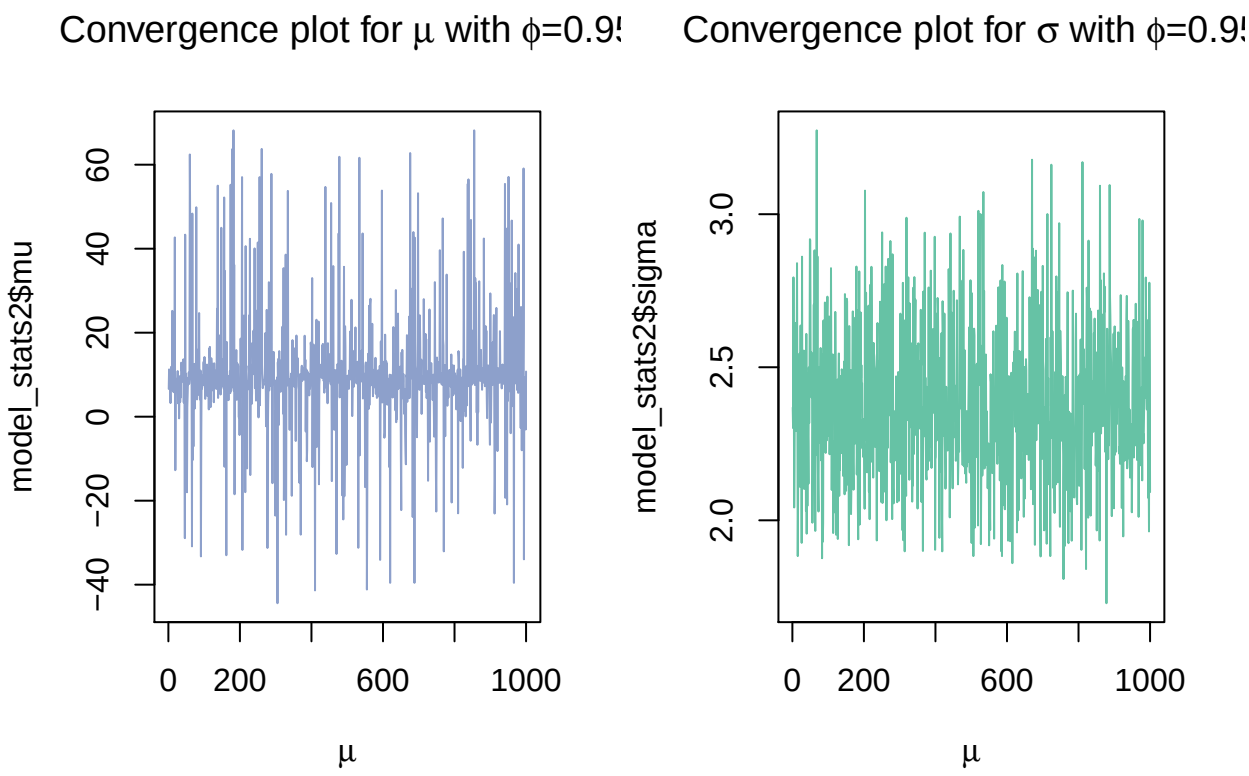
For each of the two data sets, evaluate the convergence of the samplersand plot the joint posterior of μ and ϕ.Comments?

Convergence Plots

```
# ii)

# convergence plots

par(mfrow=c(1,2))
plot(model_stats1$mu,col=sample(pal,1),
     main=expression(paste("Convergence plot for ",mu," with ",phi,"=0.3")),
     xlab=expression(mu),type="l")
plot(model_stats1$sigma,col=sample(pal,1),
     main=expression(paste("Convergence plot for ",sigma," with ",phi,"=0.3")),
     xlab=expression(mu),type="l")
```

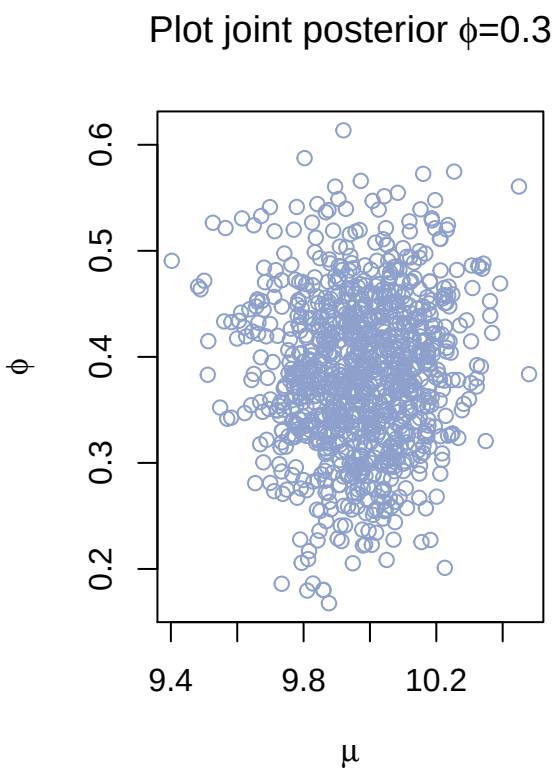


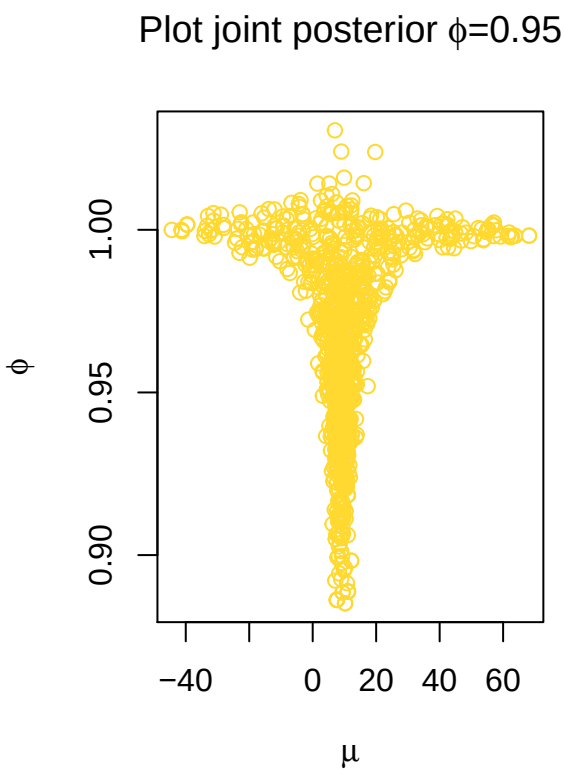

```
par(mfrow=c(1,2))
plot(model_stats2$mu,col=sample(pal,1),
     main=expression(paste("Convergence plot for ",mu," with ",phi,"=0.95")),
     xlab=expression(mu),type="l")
plot(model_stats2$sigma,col=sample(pal,1),
     main=expression(paste("Convergence plot for ",sigma," with ",phi,"=0.95")),
     xlab=expression(mu),type="l")
```



Posterior Plots

```
# posterior plots
par(mfrow=c(1,2))
plot(model_stats1$mu,model_stats1$phi,col=sample(pal,1),
     main=expression(paste("Plot joint posterior ",phi,"=0.3")),
     xlab=expression(mu),ylab=expression(phi))
plot(model_stats2$mu,model_stats2$phi,col=sample(pal,1),
     main=expression(paste("Plot joint posterior ",phi,"=0.95")),
     xlab=expression(mu),ylab=expression(phi))
```





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

$$c_t|x_t \sim Poisson(exp(x_t))$$

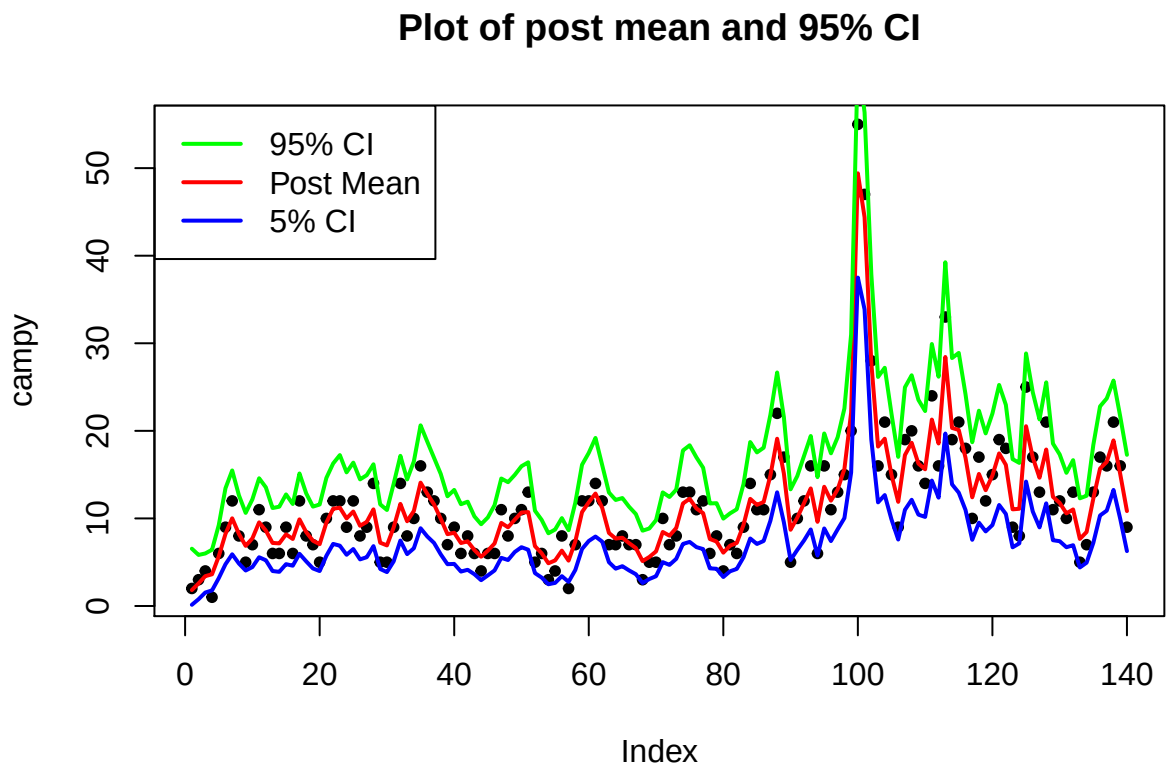
where x_t is an AR(1)-process as in a). Implement and estimate the model in Stan, using suitable priors of your choice.Produce a plot that contains both the data and the posterior mean and 95% credible intervals for the latent intensity $\theta_t = exp(x_t)$ over time.

```
# read data
campy<-read.table("campy.dat",head=T)

# poisson stan model
poisson_model='
data{
int<lower=0> N;
int y[N];
}
parameters{
real mu ;
real<lower=0> sigma;
real<lower=-1, upper=1> phi;
vector[N] xt;
```

```
}
model{
mu ~ normal(0,100); // Normal with mean 0, st.dev. 100
sigma ~ scaled_inv_chi_square(1,2); // Scaled-inv-chi2 with nu 1, sigma 2
for(n in 2:N){
xt[n]~normal(mu + phi*(xt[n-1]-mu), sqrt(sigma));
y[n]~poisson(exp(xt[n]));
}
}'

# list of the data
dataP=list(N=length(campy$c),y=campy$c)
# fit model
fit.poisson=stan(model_code = poisson_model ,
                 data = dataP,              # named list of data
                 chains = 1,              # number of Markov chains
                 warmup = 1000,           # number of warmup iterations per chain
                 iter = 2000,             # total number of iterations per chain
                 cores = 2,               # number of cores (could use one per chain)
                 refresh = 0 )
# extracct model
p<-extract(fit.poisson)
# take the exp(xt)
theta_post=exp(p$xt)
# apply mean
theta_post_mean=apply(theta_post,2,mean)
# apply quantile
CI<-apply(theta_post,2,function(x) quantile(x,prob=c(0.025,0.975)))
```

Plot of posterior mean and 95% CI with our priors

```
plot(campy$c,main="Plot of post mean and 95% CI",
     pch=20,ylab="campy")
lines(theta_post_mean,col="red",lwd=2)
lines(CI[2,],col="green",lwd=2)
lines(CI[1,],col="blue",lwd=2)
legend("topleft",legend=c("95% CI","Post Mean","5% CI"),col=c("green","red","blue"),
       lty=1,lwd=2)
```



d)

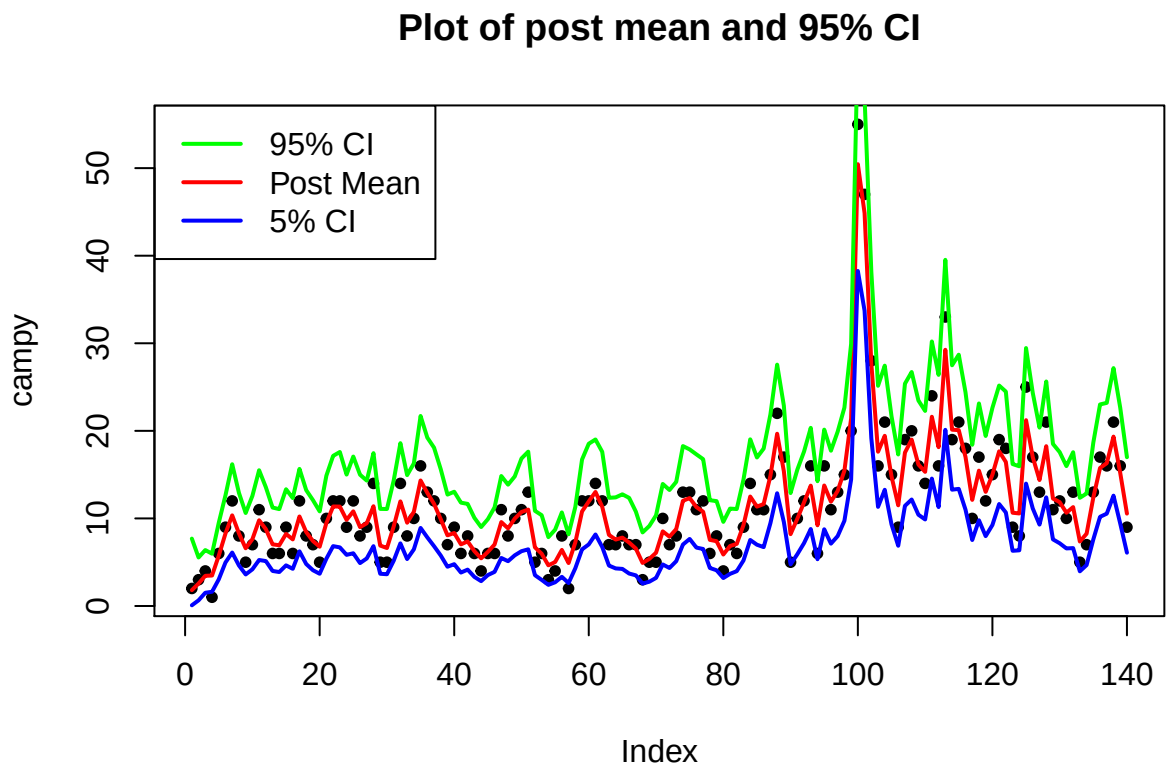
Now, assume that we have a prior belief that the true underlying intensity θ_t varies more smoothly than the data suggests.Change the prior forσ^2 so that it becomes informative about that the AR(1)-process increments ϵ_t should be small.Re-estimate the model using Stan with the new prior and produce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

```
poisson_model2='
data{
int<lower=0> N;
int y[N];
}
parameters{
real mu ;
real<lower=0> sigma;
real<lower=-1, upper=1> phi;
vector[N] xt;
}
model{
mu ~ normal(0,100); // Normal with mean 0, st.dev. 100
sigma ~ scaled_inv_chi_square(100,0.5); // Scaled-inv-chi2 with nu 100, sigma 0.5
for(n in 2:N){
xt[n]~normal(mu + phi*(xt[n-1]-mu), sqrt(sigma));
y[n]~poisson(exp(xt[n]));
}
}'
```

```
fit.poisson2=stan(model_code = poisson_model2 ,
                  data = dataP,              # named list of data
                  chains = 1,              # number of Markov chains
                  warmup = 1000,           # number of warmup iterations per chain
                  iter = 2000,             # total number of iterations per chain
                  cores = 2,               # number of cores (could use one per chain)
                  refresh = 0 )

p2<-extract(fit.poisson2)
theta_post2=exp(p2$xt)
theta_post_mean2=apply(theta_post2,2,mean)
#
CI2<-apply(theta_post2,2,function(x) quantile(x,prob=c(0.025,0.975)))
```

Plot of posterior mean and 95% CI with informative σ^2

```
plot(campy$c,main="Plot of post mean and 95% CI",
     pch=20,ylab="campy")
lines(theta_post_mean2,col="red",lwd=2)
lines(CI2[2,],col="green",lwd=2)
lines(CI2[1,],col="blue",lwd=2)
legend("topleft",legend=c("95% CI","Post Mean","5% CI"),col=c("green","red","blue"),
       lty=1,lwd=2)
```


Plot of post mean and 95% CI
95% CI
Post Mean
5% CI
campy
0
10
20
30
40
50
0
20
40
60
80
100
120
140
Index