

L4-phiho267-zijfe244

phiho267 & zijfe244

2019/5/23

(a) Write an AR(1)-process function in R

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

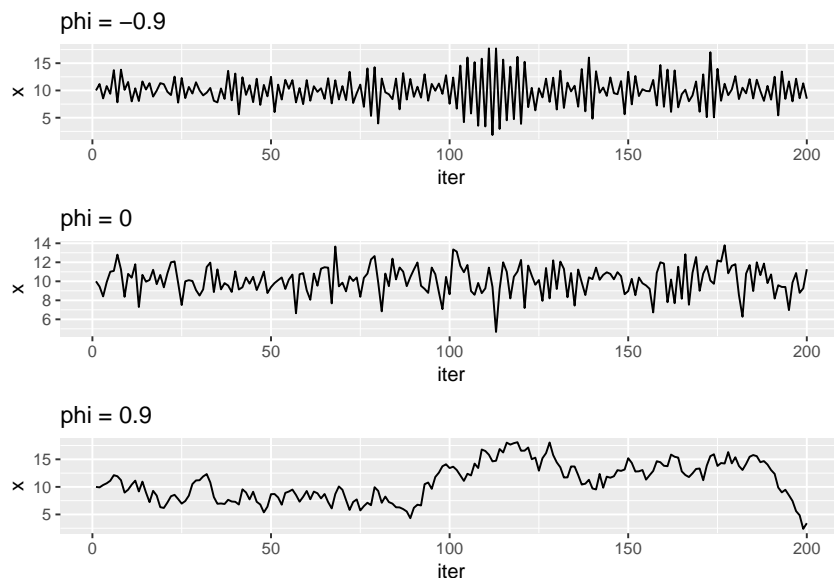
$$x_t = \mu + \phi(x_{t-1} - \mu) + \varepsilon_t, \varepsilon_t \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$$

for given values of μ , ϕ and σ^2 . Start the process at $x_1 = \mu$ and then simulate values for x_t for $t = 2, 3, \dots, T$ and return the vector $x_{1:T}$ containing all time points. 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 -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 $x_{1:T}$?

```
# setting
library(rstan)
library(latex2exp)
library(gridExtra)
library(ggplot2)

rm(list = ls())
set.seed(123456)

# AR(1) process
AR1 <- function(phi, n=200, mu=10, sigma2=2){
  x <- c()
  x[1] <- mu
  for (i in 2:n) {
    x[i] <- mu + phi*(x[i-1]-mu) + rnorm(1, 0, sqrt(sigma2))
  }
  x
}
```



Based on the formula of AR(1)-process, ϕ is the parameter representing the impact from previous iterations. when $\phi = 0$ all the terms of seem to be independent mutually. On the other hand, there seems to be a positive correlation among simulated terms when $\phi > 0$ and negative correlation when $\phi < 0$.

(b) Simulate two AR(1)-processes

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.]

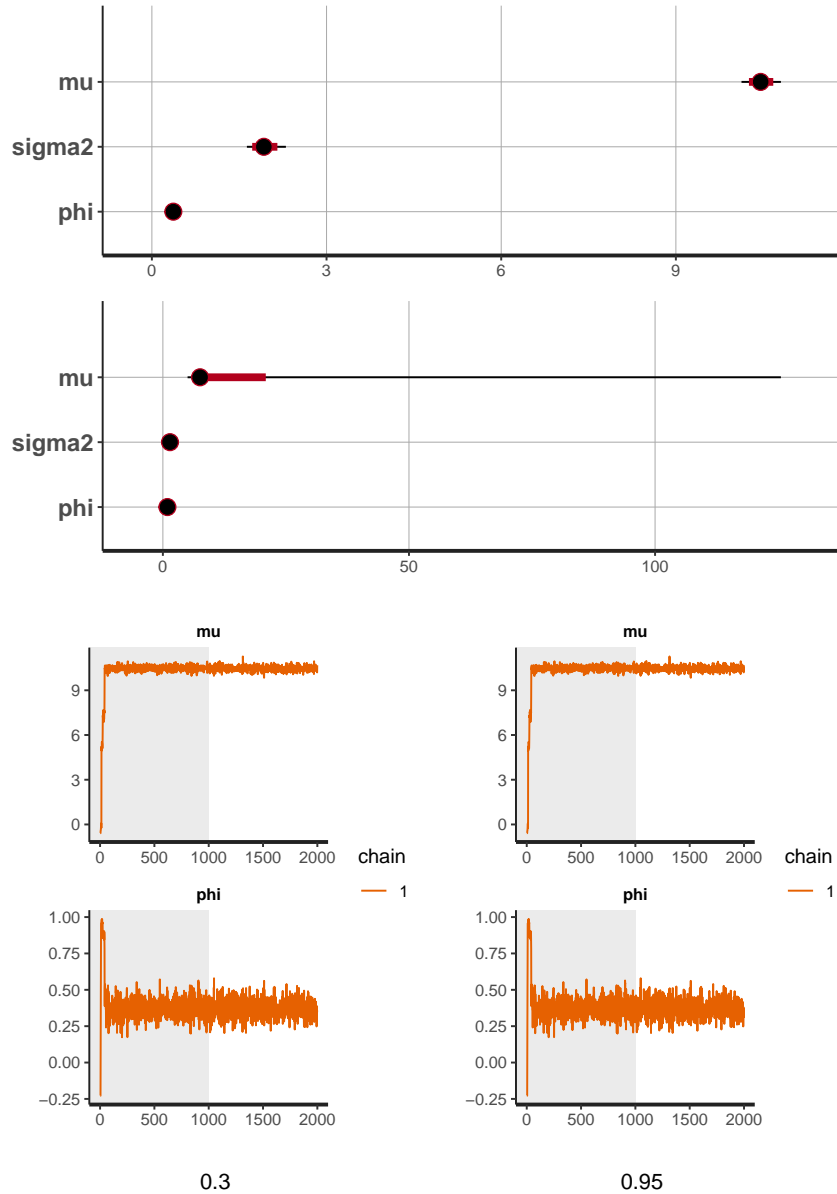
- i. Report the posterior mean, 95% credible intervals and the number of effective posterior samples for the three inferred parameters for each of the simulated AR(1)-process. Are you able to estimate the true values?
- ii. For each of the two data sets, evaluate the convergence of the samplers and plot the joint posterior of μ and ϕ . Comments?

```
## Inference for Stan model: ca7ee6f33420f715c5b4c5ae05c4ca31.
## 1 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=1000.
##
##           mean se_mean   sd    2.5%    25%    50%    75%   97.5% n_eff
## mu         10.46     0.01 0.17    10.13   10.35   10.46   10.56   10.80   860
## sigma2      1.93     0.01 0.17     1.63    1.81    1.92    2.04    2.30   785
## phi         0.37     0.00 0.06     0.26    0.33    0.37    0.41    0.49   953
## lp__      -248.06     0.07 1.39   -251.60 -248.61 -247.71 -247.06 -246.54  411
##
##           Rhat
## mu             1
## sigma2         1
## phi            1
## lp__           1
##
## Samples were drawn using NUTS(diag_e) at Fri May 24 18:48:39 2019.
## 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).

## Inference for Stan model: ca7ee6f33420f715c5b4c5ae05c4ca31.
## 1 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=1000.
##
##           mean se_mean   sd    2.5%    25%    50%    75%   97.5% n_eff
## mu         16.33     8.46 31.47     5.02    6.83    7.56    8.44  125.54   14
## sigma2      1.45     0.01 0.11     1.23    1.37    1.45    1.52    1.70  238
## phi         0.93     0.01 0.04     0.86    0.90    0.92    0.95    1.00   28
## lp__      -207.74     0.88 3.36   -217.71 -208.09 -206.49 -205.63 -205.08  15
##
##           Rhat
## mu          1.08
## sigma2      1.01
## phi         1.03
## lp__        1.05
##
## Samples were drawn using NUTS(diag_e) at Fri May 24 18:48:40 2019.
## 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).

The mean's in the results for all parameters are quite close to our defined values (besides μ for $\phi = 0.95$). But the 95% credible intervals and number of effective posterior samples are different when ϕ are different. The result with $\phi = 0.3$ has smaller credible intervals and more effective samples. Anyway, it is okay to estimate sigma2 and phi by `stan`.



All parameters converge to the actual values before 300 iterations. However, since the original y is a positive correlated process, a continuing increasing/decreasing instance always results in the explosion of estimation for μ . Such phenomena could make the estimation for μ problematic. Even if μ does not explode, its 95% credible interval would be larger compared with the one with original $\phi = 0.3$.

(c) Cases of campylobacter infections

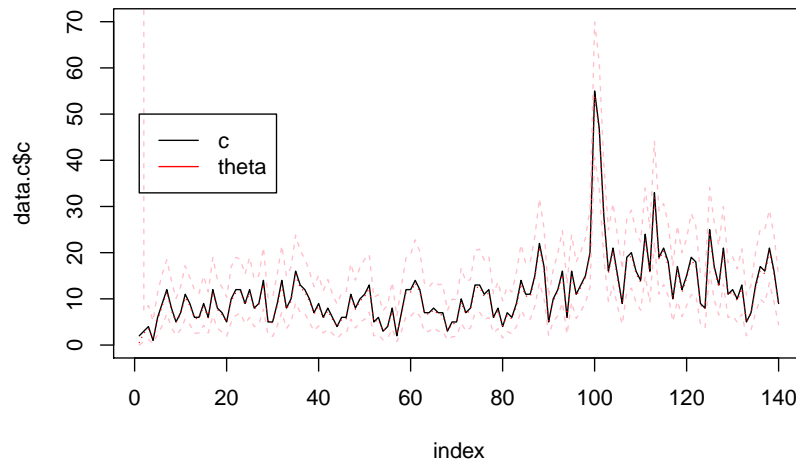
The data *campy.dat* contain the number of cases of campylobacter infections in the north of the province Quebec (Canada) in four week intervals from January 1990 to the end of October 2000. It has 13 observations

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

$$c_t | x_t \sim \text{Poisson}(\exp(x_t)),$$

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?]

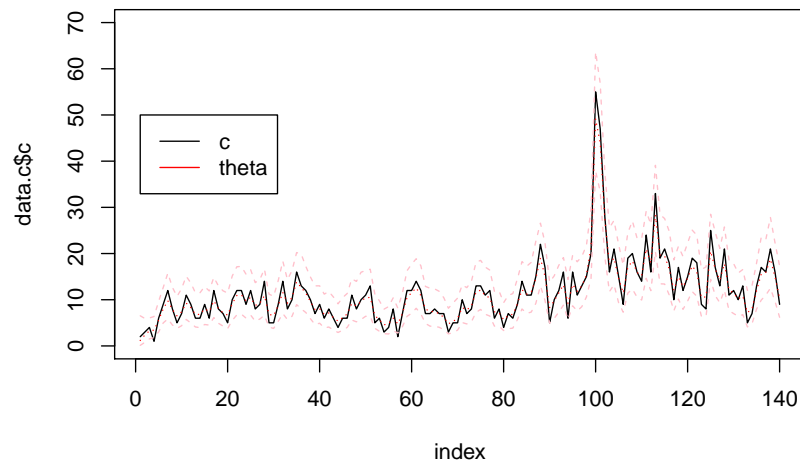
```
data.c <- read.csv("campy.dat", header = TRUE)
N <- dim(data.c)[1]
```



(d) Change the prior for σ^2

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?

Change σ^2 from $\text{Inv } \chi^2(100, 2)$ to $\text{Inv } \chi^2(50, 0.5)$



By changing the parameters of prior distribution for σ^2 , true underlying intensity θ_t is smoother a little bit. However, It is obvious that the 95% credible intervals are smoother compared with σ^2 with big-variance prior distribution, which is reasonable.

Lecture code

Appendix

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE, out.width = "320px")
# setting
library(rstan)
library(latex2exp)
library(gridExtra)
library(ggplot2)

rm(list = ls())
set.seed(123456)
# AR(1) process
AR1 <- function(phi, n=200, mu=10, sigma2=2){
  x <- c()
  x[1] <- mu
  for (i in 2:n) {
    x[i] <- mu + phi*(x[i-1]-mu) + rnorm(1, 0, sqrt(sigma2))
  }
  x
}
iter <- 1:200
phis <- c(-0.9, 0, 0.9)
plot.a <- lapply(phis, function(phi){
  x <- AR1(phi = phi)
  title <- paste0("phi = ",phi)
  data <- data.frame(iter = iter, x=x)
  ggplot(data) +
    geom_line(aes(x=iter, y=x)) +
    ggtitle(title)
})

plot(arrangeGrob(grobs = plot.a))

set.seed(1111)

x <- AR1(phi=0.3)
y <- AR1(phi=0.95)
N = 200

StanModel='
data {
  int<lower=0> N;
  vector[N] x;
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real<lower=-1, upper=1> phi;
}
model {
  mu ~ normal(0,100);
  sigma2 ~ scaled_inv_chi_square(100,1);
  x[2:N] ~ normal( mu + phi*(x[1:(N-1)]-mu), sqrt(sigma2));
```

```

}'

fit1 <- stan(
  model_code = StanModel, # Stan program
  data = list(N=N, x=x), # named list of data
  chains = 1,
  warmup = 1000, # number of warmup iterations per chain
  iter = 2000, # total number of iterations per chain
  refresh = 0
)

fit2 <- stan(
  model_code = StanModel, # Stan program
  data = list(N=N, x=y), # named list of data
  chains = 1,
  warmup = 1000, # number of warmup iterations per chain
  iter = 2000, # total number of iterations per chain
  refresh = 0
)

fit1
re1 <- extract(fit1)
fit2
re2 <- extract(fit2)
grid.arrange(plot(fit1), plot(fit2), ncol = 1)
trace1 <- traceplot(fit1, pars = c("mu", "phi"), inc_warmup = TRUE, nrow = 2)
trace2 <- traceplot(fit2, pars = c("mu", "phi"), inc_warmup = TRUE, nrow = 2)
grid.arrange(arrangeGrob(trace1, bottom = "0.3"), arrangeGrob(trace1, bottom = "0.95"), ncol = 2)
data.c <- read.csv("campy.dat", header = TRUE)
N <- dim(data.c)[1]
set.seed(1111)
StanModel.c='
data {
  int<lower=0> N;
  int c[N];
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real phi;
  real x[N];
}
transformed parameters {
  real theta[N];
  theta = exp(x);
}
model {
  // prior
  mu ~ normal(0,50);
  sigma2 ~ scaled_inv_chi_square(100,2);

  // model
  for (n in 2:N){
    x[n] ~ normal(mu+phi*(x[n-1]-mu),sqrt(sigma2));
  }
}

```



```

    c[n] ~ poisson(theta[n]);
  }
}
'

set.seed(1111)
datapoi=list(N=N,c=data.c$c)
fit.c<-stan(model_code=StanModel.c,
            data=datapoi,
            warmup=1000,
            iter=2000,
            chains=1,
            refresh=0
            )

re.c <- extract(fit.c)
index <- 1:N
plot(x=index, y=data.c$c, col="black",type="l",ylim=c(0,70))

theta <- exp(colMeans(re.c$x))
lines(x=index, y=theta, col="red", lty=3)

q <- matrix(nrow=2, ncol=140)
for (i in index) {
  q[,i] <- quantile(re.c$x[,i],probs = c(0.025, 0.975))
}
lines(x=index, y=exp(q[1,]), lty=2, col = "pink")
lines(x=index, y=exp(q[2,]), lty=2, col = "pink")
legend(x=1,y=50, legend=c("c","theta"), col=c("black", "red"),lty=1, cex=1)
set.seed(1111)
StanModel.c='
data {
  int<lower=0> N;
  int c[N];
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real phi;
  real x[N];
}
transformed parameters {
  real theta[N];
  theta = exp(x);
}
model {
  // prior
  mu ~ normal(0,50);
  sigma2 ~ scaled_inv_chi_square(50,0.5);

  // model
  for (n in 2:N){
    x[n] ~ normal(mu+phi*(x[n-1]-mu),sqrt(sigma2));
  }
}
'

```

```

    c[n] ~ poisson(theta[n]);
  }
}
'

set.seed(1111)
datapoi=list(N=N,c=data.c$c)
fit.c<-stan(model_code=StanModel.c,
            data=datapoi,
            warmup=1000,
            iter=2000,
            chains=1,
            refresh=0
            )

re.c <- extract(fit.c)

index <- 1:N
plot(x=index, y=data.c$c, col="black",type="l",ylim=c(0,70))

theta <- exp(colMeans(re.c$x))
lines(x=index, y=theta, col="red", lty=3)

q <- matrix(nrow=2, ncol=140)
for (i in index) {
  q[,i] <- quantile(re.c$x[,i],probs = c(0.025, 0.975))
}
lines(x=index, y=exp(q[1,]), lty=2, col = "pink")
lines(x=index, y=exp(q[2,]), lty=2, col = "pink")
legend(x=1,y=50, legend=c("c","theta"), col=c("black", "red"),lty=1, cex=1)

```