L4-phiho267-zijfe244

phiho267 & zijfe244

2019/5/23

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

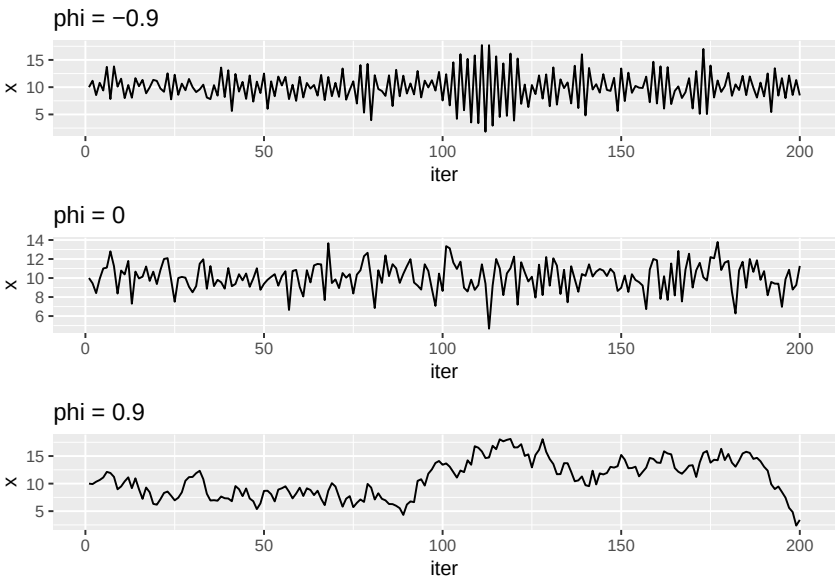
Write a in R that simulates data from the AR(1)-process

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

for given values of μ, ϕ and σ^2. Start the process at $x1 = \mu$ 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 $T = 200$ and look at some different realizations (simulations) of x1: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?

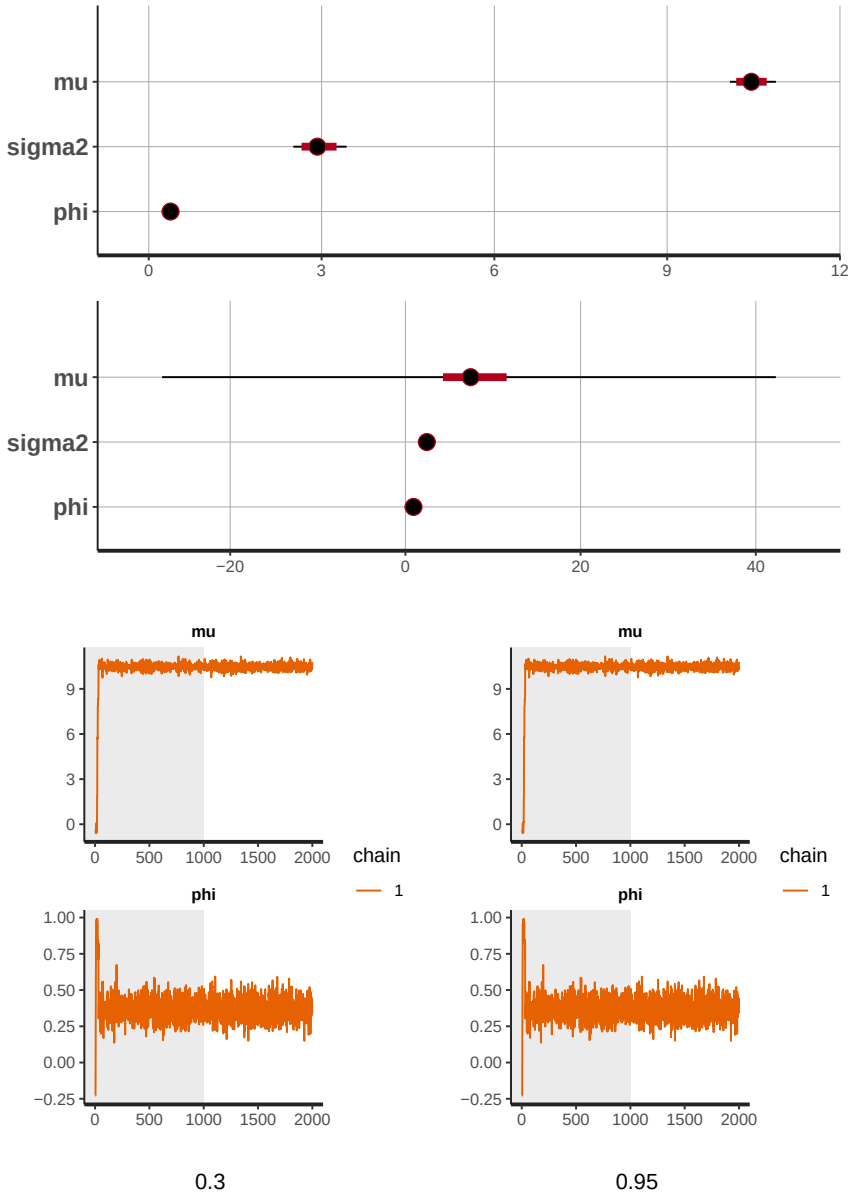
We assume $\mu \sim \mathcal{N}(0, 100)$ and $\sigma^2 \sim Inv\ \chi^2(100, 2)$$

```
## Inference for Stan model: 08745450845184f6aa34186b8a30d453.
## 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       10.47    0.01 0.21   10.09   10.34   10.46   10.61   10.89   780
## sigma2    2.95    0.01 0.23    2.51    2.78    2.93    3.10    3.44   753
## phi       0.38    0.00 0.07    0.23    0.33    0.38    0.43    0.52  1063
## lp__   -311.31    0.05 1.19 -314.52 -311.92 -311.02 -310.40 -309.90   519
##        Rhat
## mu        1
## sigma2    1
## phi       1
## lp__      1
##
## Samples were drawn using NUTS(diag_e) at Mon May 27 18:04:36 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: 08745450845184f6aa34186b8a30d453.
## 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        8.02    3.15 18.39  -27.76    6.31    7.45    8.64   42.27    34
## sigma2    2.47    0.01  0.21    2.10    2.33    2.45    2.58    2.91   358
## phi       0.93    0.01  0.04    0.85    0.90    0.93    0.96    1.00    45
## lp__   -287.21    0.48  2.54 -293.84 -288.09 -286.33 -285.38 -284.70    28
##        Rhat
## mu     1.00
## sigma2 1.00
## phi    1.06
## lp__   1.09
##
## Samples were drawn using NUTS(diag_e) at Mon May 27 18:04:37 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`.



0.3 0.95

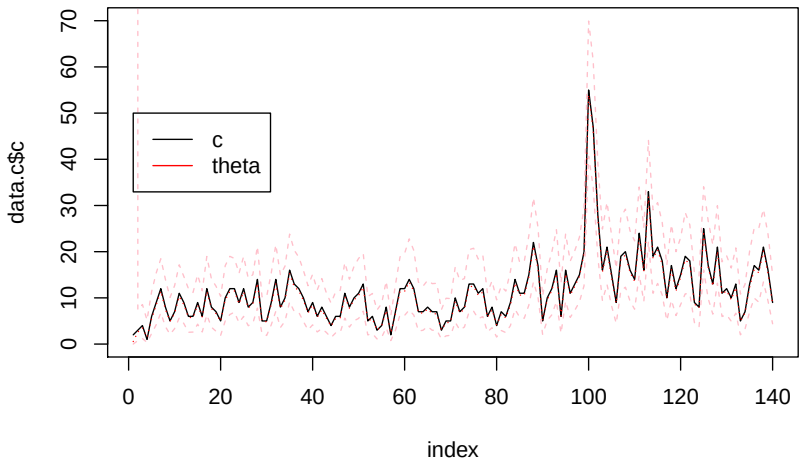
All parameters converge to the actual values before 300 iterations. However, since the original y is a positive correlated process, a continuing increasing/decreasing instance always results in the explosion of estimation for μ. Such phenomena could make the estimation for μ problematic. Even if mu does 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 ct at each time point follows an independent Poisson distribution when conditioned on a latent AR(1)-process x_t, that is

$$c_t|x_t \sim \mathcal{P}oisson(\exp(x_t)),$$

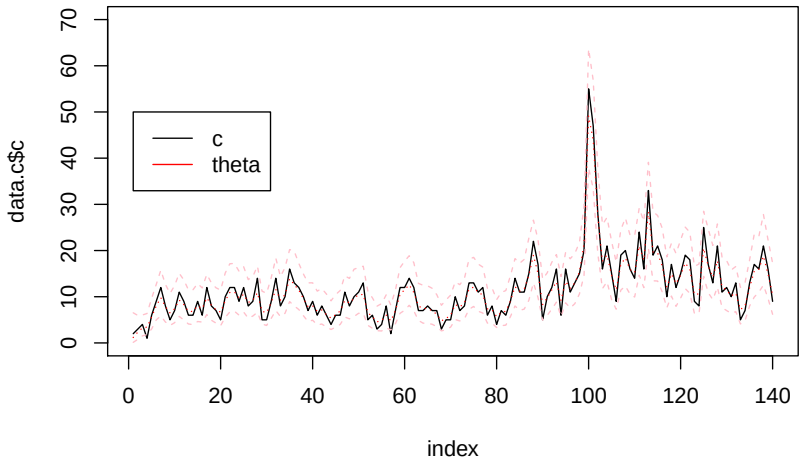
where x_t is an AR(1)-process as in a). Implement and estimate the model in Stan, using suitable priors of your choice. Produce a plot that contains both the data and the posterior mean and 95% credible intervals for the latent intensity $\theta_t = \exp(x_t)$ over time. [Hint: Should xt be seen as data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



(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 $Inv\ \chi^2(100, 2)$ to $Inv\ \chi^2(50, 0.5)$



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

```
# BRugs code for analyzing a simple Bernoulli model with a Beta(a,b) prior
# Author: Mattias Villani, Linkoping University
# Ported to RStan by Mans Magnusson
# Date:    2013-10-24
# Updated: 2018-04-16 by Per Siden
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
# Data
x = c(1,1,0,0,1,1,1,1,0,1)
n = length(x)
a = 1
b = 1
BernBetaData <- list(n = length(x), x=x, a=a, b=b)
# Model
BernBetaStanModel <- '
data {
int<lower=0> n;
int<lower=0,upper=1> x[n];
real<lower=0> a;
real<lower=0> b;
}
parameters {
real<lower=0,upper=1> theta;
}
model {
theta ~ beta(a,b);
for (i in 1:n)
x[i] ~ bernoulli(theta);
}
'
# Do the fitting of the model
burnin = 1000
niter = 2000
fit1<-stan(model_code=BernBetaStanModel,
           data=BernBetaData,
           warmup=burnin,
           iter=niter,
           chains=4)
# Print the fitted model
print(fit1,digits_summary=3)
# Extract posterior samples
postDraws <- extract(fit1)
# Do traceplots of the first chain
par(mfrow = c(1,1))
plot(postDraws$theta[1:(niter-burnin)],type="l",ylab="theta",main="Traceplot")
# Do automatic traceplots of all chains
traceplot(fit1)
# Plot posterior histogram and compare with analytical posterior
thetaSeq <- seq(0,1,by=0.01)
```

```r
par(mfrow = c(1,1))
hist(postDraws$theta, 40, freq = FALSE, main = 'Posterior of theta - all chains', xlab ='theta') # hist
lines(thetaSeq, dbeta(thetaSeq, shape1 = sum(x) + a, shape2 = n - sum(x) + b), col = "red")
legend("topleft", inset=.05, legend = c('MCMC approximation','True density'), lty =c(1,1),col=c('black'
############
# My own version of the example from http://mathstat.helsinki.fi/openbugs/ExamplesFrames.html
# Author: Mattias Villani, Statistics, Linkoping University, Sweden
#         mattias.villani@liu.se
# Date:    2012-12-05
# Ported to RStan by Mans Magnusson
# Date:    2013-10-24
# Updated: 2018-04-16 by Per Siden
rm(list=ls())
library(rstan)
nBurnin <- 1000
nIter <- 2000
rstanSeedModel<-'
data {
  int<lower=0> N; // Number of observations
  int<lower=0> r[N]; // Number of successes
  int<lower=0> n[N]; // Numer of trials
  vector[N] x1; // Covariate 1
  vector[N] x2; // Covariate 2
}
parameters {
  real alpha0;
  real alpha1;
  real alpha2;
  real<lower=0> tau;
  vector[N] b;
}
transformed parameters {
  real<lower=0> sigma;
  sigma <- 1.0 / sqrt(tau);
}
model {
  // Priors
  alpha0 ~ normal(0.0,1.0E3);
  alpha1 ~ normal(0.0,1.0E3);
  alpha2 ~ normal(0.0,1.0E3);
  tau ~ gamma(1.0E-3,1.0E-3);
  // Model
  b ~ normal(0.0, sigma);
  r ~ binomial_logit(n, alpha0 + alpha1 * x1 + alpha2 * x2 + b);
}
'
# Data
seedsData <- list(N = 21,
                  r = c(10, 23, 23, 26, 17, 5, 53, 55, 32, 46, 10, 8, 10, 8, 23, 0, 3, 22, 15, 32, 3),
                  n = c(39, 62, 81, 51, 39, 6, 74, 72, 51, 79, 13, 16, 30, 28, 45, 4, 12, 41, 30, 51, 7
                  x1 = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
                  x2 = c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1))
fit1<-stan(model_code=rstanSeedModel,
```

```
            data=seedsData,
            warmup=nBurnin,
            iter=(nBurnin+nIter),
            chains=2)
fit1
print(fit1,digits_summary=3)
plot(fit1)
# Extract parameters samples
fit1ParSamples<-extract(fit1,permuted=FALSE)
dim(fit1ParSamples)
# Reuse the model
fit2 <- stan(fit=fit1,data=seedsData,warmup=nBurnin,iter=(nIter+nBurnin)*10,thin=10)
print(fit2,digits_summary=3)
fit2ParSamples<-extract(fit2,permuted=FALSE)
dim(fit2ParSamples)
########
# My own version of the example Roaches
# Author: Mattias Villani, Statistics, Linkoping University, Sweden
#         mattias.villani@liu.se
# Date:    2012-12-05
# Ported to RStan by Mans Magnusson
# Date:    2013-10-24
# Updated: 2018-04-16 by Per Siden
#install.packages("ggplot2")
library(ggplot2)
#install.packages("rstan")
library(rstan)
roachesData<-read.csv2("../Data/roachdata.csv")
head(roachesData)
roachDataRStan<-
  list(N = nrow(roachesData),
       exposure2 = roachesData$exposure2,
       senior = roachesData$senior,
       treatment = roachesData$treatment,
       y = roachesData$y)
# The poisson regression model
roachModel = '
data {
  int<lower=0> N;
  vector[N] exposure2;
  vector[N] senior;
  vector[N] treatment;
  int y[N];
}
transformed data {
  vector[N] log_expo;
  log_expo = log(exposure2);
}
parameters {
  vector[3] beta;
}
model {
  // Prior
```

```
  beta ~ normal(0,1000.0);
  // Model/likelihood
  y ~ poisson_log(log_expo + beta[1] + beta[2] * treatment
                  + beta[3] * senior);
}
generated quantities {
  int<lower=0> pred_treat;
  int<lower=0> pred_notreat;
  vector[3] exp_beta;
  exp_beta = exp(beta);
  pred_treat = poisson_rng(exp_beta[1]*exp_beta[2]);
  pred_notreat = poisson_rng(exp_beta[1]);
}
'
fitRoach<-stan(model_code=roachModel,
           data=roachDataRStan,
           par=c("exp_beta","pred_treat","pred_notreat"),
           warmup=1000,
           iter=10000,
           chains=2)
print(fitRoach,digits=2)
# Plot some results
res<-extract(fitRoach)
# Plot overlapping histograms
par(mfrow=c(2,1))
str(res$exp_beta[,2])
hist(res$exp_beta[,2])
hist(res$pred_treat,main="Treated",xlim=c(0,70),xlab="Predicted number of roaches")
hist(res$pred_notreat,main="Not treated",xlim=c(0,70),xlab="Predicted number of roaches")
smoothScatter(res$pred_treat,res$pred_notreat,xlim=c(0,70),ylim=c(0,70))
# Whats the probability that there will be more roaches in a treated house?
mean(res$pred_treat > res$pred_notreat)
lines(x=c(0,70),y=c(0,70))
#####
# Toy RStan example: Measuring the number of flowers of 3 plants for 4 months
# Per Siden 2018-04-23
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
y = c(4,5,6,4,0,2,5,3,8,6,10,8)
plot(y,col=c(1,1,1,1,2,2,2,2,3,3,3,3))
N = length(y)
P = 3
StanModel = '
data {
  int<lower=0> N; // Number of observations
  int<lower=0> y[N]; // Number of flowers
  int<lower=0> P; // Number of plants
}
transformed data {
  int<lower=0> M; // Number of months
  M = N / P;
```

```
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real mup[P];
  //real sigmap2[P];
}
transformed parameters {
  //real sigma;
  //sigma = sqrt(sigma2);
}
model {
  mu ~ normal(0,100); // Normal with mean 0, st.dev. 100
  sigma2 ~ scaled_inv_chi_square(1,2); // Scaled-inv-chi2 with nu 1, sigma 2
  //sigma ~ exponential(.2); // Exponential with mean 5 = 1/.2

  // Model 1: iid
  //for(i in 1:N)
  //  y[i] ~ normal(mu,sqrt(sigma2));
  // Model 2: Multilevel normal
  //for(p in 1:P){
  //  mup[p] ~ normal(mu,sqrt(sigma2));
  //  for(m in 1:M)
  //    y[M*(p-1)+m] ~ normal(mup[p],sqrt(sigmap2[p]));
  //}
  // Model 3: Multilevel poisson
  for(p in 1:P){
    mup[p] ~ lognormal(mu,sqrt(sigma2)); // Log-normal
    for(m in 1:M)
      y[M*(p-1)+m] ~ poisson(mup[p]); // Poisson
  }
}'
data = list(N=N, y=y, P=P)
burnin = 1000
niter = 2000
fit = stan(model_code=StanModel,
           data=data,
           warmup=burnin,
           iter=niter,
           chains=4)
# Print the fitted model
print(fit,digits_summary=3)
# Extract posterior samples
postDraws <- extract(fit)
# Do traceplots of the first chain
par(mfrow = c(1,1))
plot(postDraws$mu[1:(niter-burnin)],type="l",ylab="mu",main="Traceplot")
# Do automatic traceplots of all chains
traceplot(fit)
# Bivariate posterior plots
pairs(fit)
```

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(100,2);
  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 {
  mu ~ normal(0,50);
  sigma2 ~ scaled_inv_chi_square(100,2);
  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$x[,i],probs = c(0.025, 0.975))
}
lines(x=index, y=exp(q[1,]), lty=2, col = "pink")
lines(x=index, y=exp(q[2,]), lty=2, col = "pink")
legend(x=1,y=50, legend=c("c","theta"), col=c("black", "red"),lty=1,  cex=1)
set.seed(1111)
StanModel.c='
data {
  int<lower=0> N;
  int c[N];
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real phi;
  real x[N];
}
transformed parameters {
  real theta[N];
  theta = exp(x);
}
model {
  mu ~ normal(0,50);
  sigma2 ~ scaled_inv_chi_square(50,0.5);
  for (n in 2:N){
    x[n] ~ normal(mu+phi*(x[n-1]-mu),sqrt(sigma2));
    c[n] ~ poisson(theta[n]);
  }
}
'
set.seed(1111)
datapoi=list(N=N,c=data.c$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)
# BRugs code for analyzing a simple Bernoulli model with a Beta(a,b) prior
# Author: Mattias Villani, Linkoping University
# Ported to RStan by Mans Magnusson
# Date:    2013-10-24
# Updated: 2018-04-16 by Per Siden
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
# Data
x = c(1,1,0,0,1,1,1,1,0,1)
n = length(x)
a = 1
b = 1
BernBetaData <- list(n = length(x), x=x, a=a, b=b)
# Model
BernBetaStanModel <- '
data {
int<lower=0> n;
int<lower=0,upper=1> x[n];
real<lower=0> a;
real<lower=0> b;
}
parameters {
real<lower=0,upper=1> theta;
}
model {
theta ~ beta(a,b);
for (i in 1:n)
x[i] ~ bernoulli(theta);
}
'
# Do the fitting of the model
burnin = 1000
niter = 2000
fit1<-stan(model_code=BernBetaStanModel,
           data=BernBetaData,
           warmup=burnin,
           iter=niter,
           chains=4)
# Print the fitted model
print(fit1,digits_summary=3)
# Extract posterior samples
postDraws <- extract(fit1)
# Do traceplots of the first chain
```

```r
par(mfrow = c(1,1))
plot(postDraws$theta[1:(niter-burnin)],type="l",ylab="theta",main="Traceplot")
# Do automatic traceplots of all chains
traceplot(fit1)
# Plot posterior histogram and compare with analytical posterior
thetaSeq <- seq(0,1,by=0.01)
par(mfrow = c(1,1))
hist(postDraws$theta, 40, freq = FALSE, main = 'Posterior of theta - all chains', xlab ='theta') # hist
lines(thetaSeq, dbeta(thetaSeq, shape1 = sum(x) + a, shape2 = n - sum(x) + b), col = "red")
legend("topleft", inset=.05, legend = c('MCMC approximation','True density'), lty =c(1,1),col=c('black'
###########
# My own version of the example from http://mathstat.helsinki.fi/openbugs/ExamplesFrames.html
# Author: Mattias Villani, Statistics, Linkoping University, Sweden
#         mattias.villani@liu.se
# Date:    2012-12-05
# Ported to RStan by Mans Magnusson
# Date:    2013-10-24
# Updated: 2018-04-16 by Per Siden
rm(list=ls())
library(rstan)
nBurnin <- 1000
nIter <- 2000
rstanSeedModel<-'
data {
  int<lower=0> N; // Number of observations
  int<lower=0> r[N]; // Number of successes
  int<lower=0> n[N]; // Numer of trials
  vector[N] x1; // Covariate 1
  vector[N] x2; // Covariate 2
}
parameters {
  real alpha0;
  real alpha1;
  real alpha2;
  real<lower=0> tau;
  vector[N] b;
}
transformed parameters {
  real<lower=0> sigma;
  sigma <- 1.0 / sqrt(tau);
}
model {
  // Priors
  alpha0 ~ normal(0.0,1.0E3);
  alpha1 ~ normal(0.0,1.0E3);
  alpha2 ~ normal(0.0,1.0E3);
  tau ~ gamma(1.0E-3,1.0E-3);
  // Model
  b ~ normal(0.0, sigma);
  r ~ binomial_logit(n, alpha0 + alpha1 * x1 + alpha2 * x2 + b);
}
'
# Data
```

```
seedsData <- list(N = 21,
                  r = c(10, 23, 23, 26, 17, 5, 53, 55, 32, 46, 10, 8, 10, 8, 23, 0, 3, 22, 15, 32, 3),
                  n = c(39, 62, 81, 51, 39, 6, 74, 72, 51, 79, 13, 16, 30, 28, 45, 4, 12, 41, 30, 51, 7
                  x1 = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
                  x2 = c(0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1))
fit1<-stan(model_code=rstanSeedModel,
           data=seedsData,
           warmup=nBurnin,
           iter=(nBurnin+nIter),
           chains=2)
fit1
print(fit1,digits_summary=3)
plot(fit1)
# Extract parameters samples
fit1ParSamples<-extract(fit1,permuted=FALSE)
dim(fit1ParSamples)
# Reuse the model
fit2 <- stan(fit=fit1,data=seedsData,warmup=nBurnin,iter=(nIter+nBurnin)*10,thin=10)
print(fit2,digits_summary=3)
fit2ParSamples<-extract(fit2,permuted=FALSE)
dim(fit2ParSamples)
########
# My own version of the example Roaches
# Author: Mattias Villani, Statistics, Linkoping University, Sweden
#         mattias.villani@liu.se
# Date:    2012-12-05
# Ported to RStan by Mans Magnusson
# Date:    2013-10-24
# Updated: 2018-04-16 by Per Siden
#install.packages("ggplot2")
library(ggplot2)
#install.packages("rstan")
library(rstan)
roachesData<-read.csv2("../Data/roachdata.csv")
head(roachesData)
roachDataRStan<-
  list(N = nrow(roachesData),
       exposure2 = roachesData$exposure2,
       senior = roachesData$senior,
       treatment = roachesData$treatment,
       y = roachesData$y)
# The poisson regression model
roachModel = '
data {
  int<lower=0> N;
  vector[N] exposure2;
  vector[N] senior;
  vector[N] treatment;
  int y[N];
}
transformed data {
  vector[N] log_expo;
  log_expo = log(exposure2);
```

```
}
parameters {
  vector[3] beta;
}
model {
  // Prior
  beta ~ normal(0,1000.0);
  // Model/likelihood
  y ~ poisson_log(log_expo + beta[1] + beta[2] * treatment
                  + beta[3] * senior);
}
generated quantities {
  int<lower=0> pred_treat;
  int<lower=0> pred_notreat;
  vector[3] exp_beta;
  exp_beta = exp(beta);
  pred_treat = poisson_rng(exp_beta[1]*exp_beta[2]);
  pred_notreat = poisson_rng(exp_beta[1]);
}
'
fitRoach<-stan(model_code=roachModel,
           data=roachDataRStan,
           par=c("exp_beta","pred_treat","pred_notreat"),
           warmup=1000,
           iter=10000,
           chains=2)
print(fitRoach,digits=2)
# Plot some results
res<-extract(fitRoach)
# Plot overlapping histograms
par(mfrow=c(2,1))
str(res$exp_beta[,2])
hist(res$exp_beta[,2])
hist(res$pred_treat,main="Treated",xlim=c(0,70),xlab="Predicted number of roaches")
hist(res$pred_notreat,main="Not treated",xlim=c(0,70),xlab="Predicted number of roaches")
smoothScatter(res$pred_treat,res$pred_notreat,xlim=c(0,70),ylim=c(0,70))
# Whats the probability that there will be more roaches in a treated house?
mean(res$pred_treat > res$pred_notreat)
lines(x=c(0,70),y=c(0,70))
#####
# Toy RStan example: Measuring the number of flowers of 3 plants for 4 months
# Per Siden 2018-04-23
rm(list=ls())
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
y = c(4,5,6,4,0,2,5,3,8,6,10,8)
plot(y,col=c(1,1,1,1,2,2,2,2,3,3,3,3))
N = length(y)
P = 3
StanModel = '
data {
  int<lower=0> N; // Number of observations
```

```
  int<lower=0> y[N]; // Number of flowers
  int<lower=0> P; // Number of plants
}
transformed data {
  int<lower=0> M; // Number of months
  M = N / P;
}
parameters {
  real mu;
  real<lower=0> sigma2;
  real mup[P];
  //real sigmap2[P];
}
transformed parameters {
  //real sigma;
  //sigma = sqrt(sigma2);
}
model {
  mu ~ normal(0,100); // Normal with mean 0, st.dev. 100
  sigma2 ~ scaled_inv_chi_square(1,2); // Scaled-inv-chi2 with nu 1, sigma 2
  //sigma ~ exponential(.2); // Exponential with mean 5 = 1/.2

  // Model 1: iid
  //for(i in 1:N)
  //  y[i] ~ normal(mu,sqrt(sigma2));
  // Model 2: Multilevel normal
  //for(p in 1:P){
  //  mup[p] ~ normal(mu,sqrt(sigma2));
  //  for(m in 1:M)
  //    y[M*(p-1)+m] ~ normal(mup[p],sqrt(sigmap2[p]));
  //}
  // Model 3: Multilevel poisson
  for(p in 1:P){
    mup[p] ~ lognormal(mu,sqrt(sigma2)); // Log-normal
    for(m in 1:M)
      y[M*(p-1)+m] ~ poisson(mup[p]); // Poisson
  }
}'
data = list(N=N, y=y, P=P)
burnin = 1000
niter = 2000
fit = stan(model_code=StanModel,
           data=data,
           warmup=burnin,
           iter=niter,
           chains=4)
# Print the fitted model
print(fit,digits_summary=3)
# Extract posterior samples
postDraws <- extract(fit)
# Do traceplots of the first chain
par(mfrow = c(1,1))
plot(postDraws$mu[1:(niter-burnin)],type="l",ylab="mu",main="Traceplot")
```

```
# Do automatic traceplots of all chains
traceplot(fit)
# Bivariate posterior plots
pairs(fit)
```