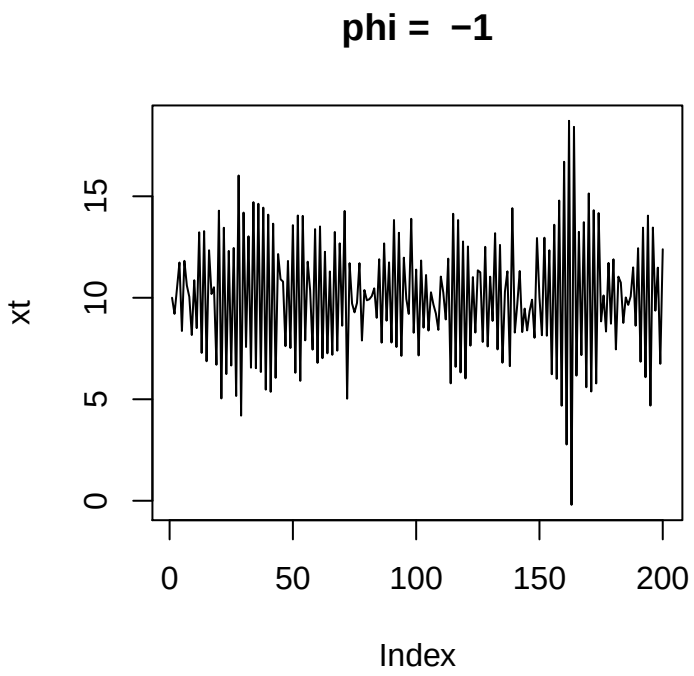
TDDE07 - Lab 4

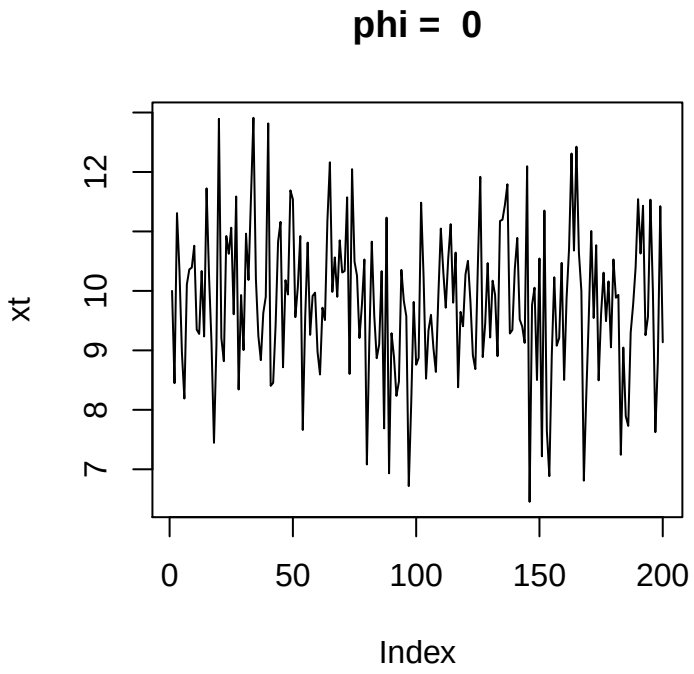
Pontus Svensson (ponsv690) & Kristian Sikiric (krisi211)

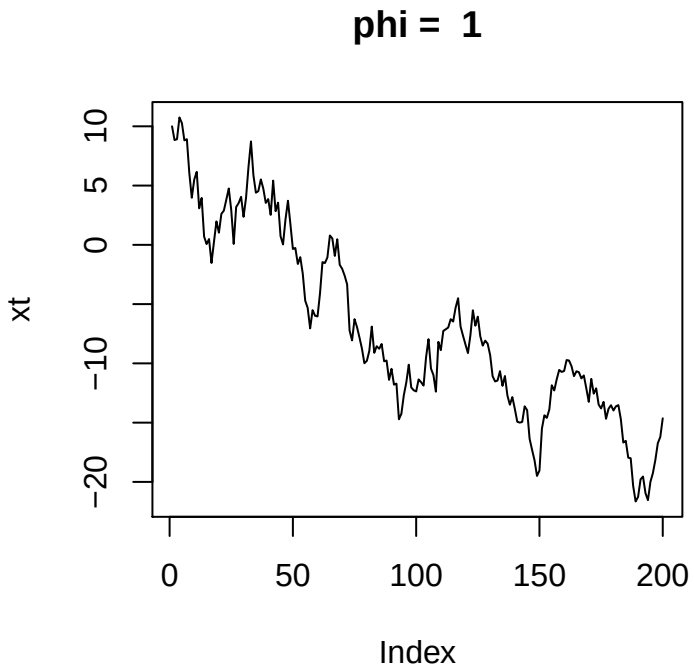
Assignment 1

Time series models in Stan

A function was written to simulate data from the AR(1)-process $x_t = \mu + \phi(x_{t-1} - \mu) + \epsilon_t, \epsilon_t \sim N(0, \sigma^2)$, with $\mu = 10, \sigma^2 = 2, T = 200$ and $t = 2, 3, ..., T$. The vector $x_{1:T}$ containing all time points was returned. Simulations for values of ϕ between -1 and 1 were performed to show the effect of ϕ. The plots below show that a high positive ϕ makes the AR(1)-process only depend on previous values, making the process follow temporary trends. When $\phi = 0$, the values oscillates around the mean. With a ϕ close to -1, the values oscillates quicker around the mean than larger values of ϕ.







Next, two AR(1)-processes were simulated, one with $\phi = 0.3$ and one with $\phi = 0.95$. These were used in Stan to estimate the values of μ, ϕ and σ^2 using MCMC.

Below is the posterior mean, 95% credible interval and number of effective samples for the three parameters along with a plot showing the convergence of the sampler. As we can see, when $\phi = 0.95$ the sampler have a

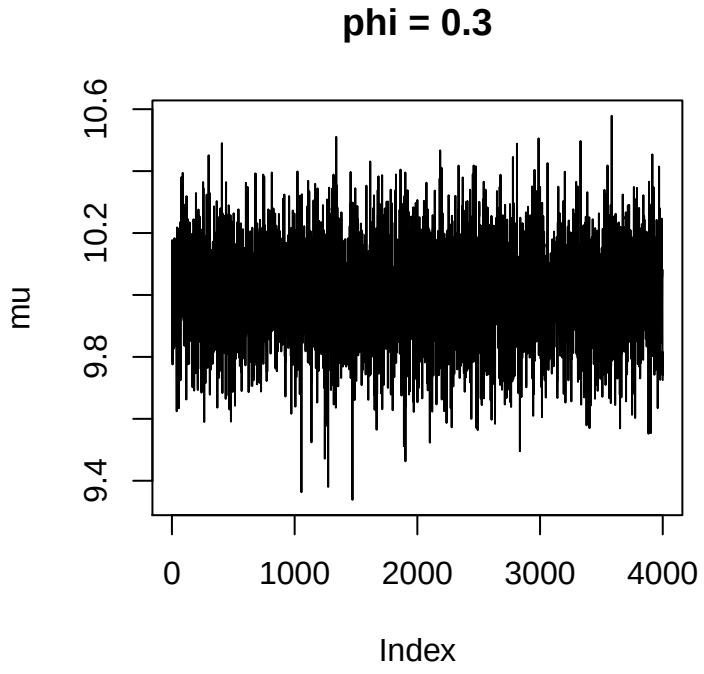
hard time converging. We also see that we are close to sampling the true values. Also worth mentioning is that the effective samples for y are very few, meaning that the samples are correlated. This makes sense since ϕ is large, so the samples depend heavily on the previous sample.

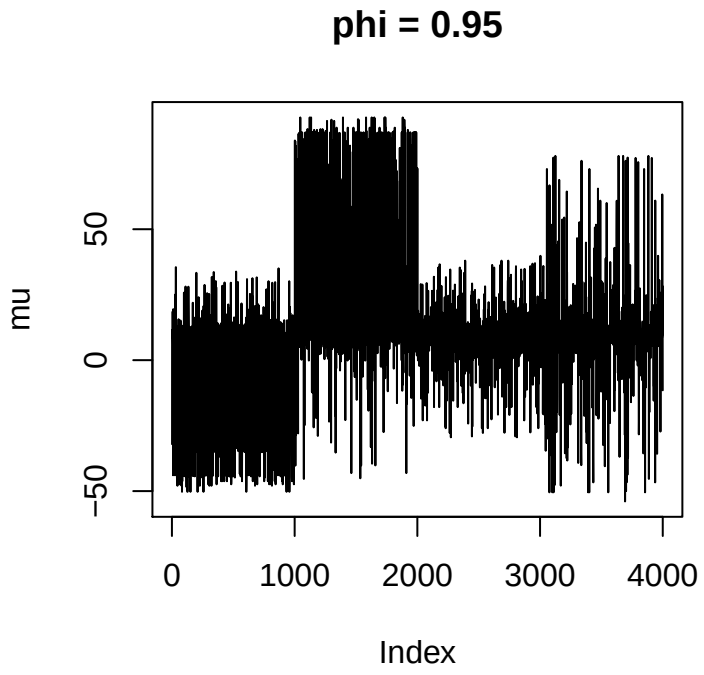
```
## [1] "Parameters for x"
```

```
## Inference for Stan model: 3118d30d9147b2cf56d6b63fb0063ccd.
## 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%  97.5% n_eff Rhat
## mu     10.002   0.003 0.158 9.683 10.309  3283    1
## sigma2  2.111   0.004 0.220 1.708  2.577  3540    1
## phi     0.341   0.001 0.071 0.201  0.478  3318    1
##
## Samples were drawn using NUTS(diag_e) at Mon May 20 16:13:23 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).
```

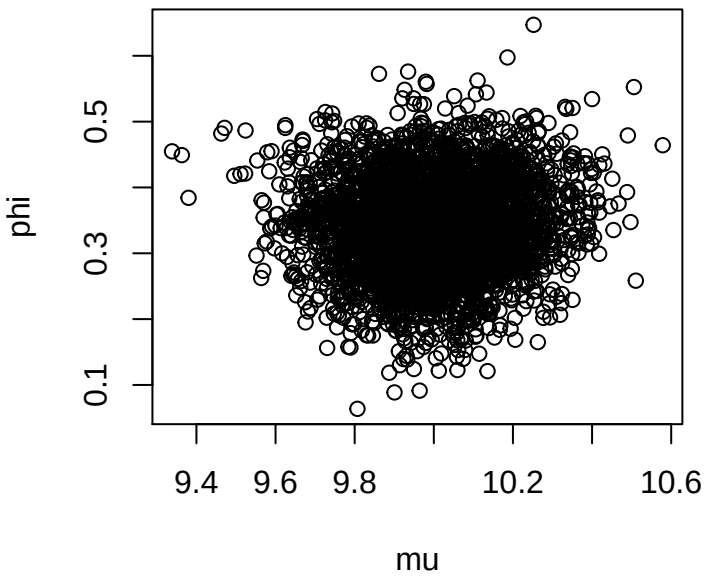
```
##
## Parameters for y
```

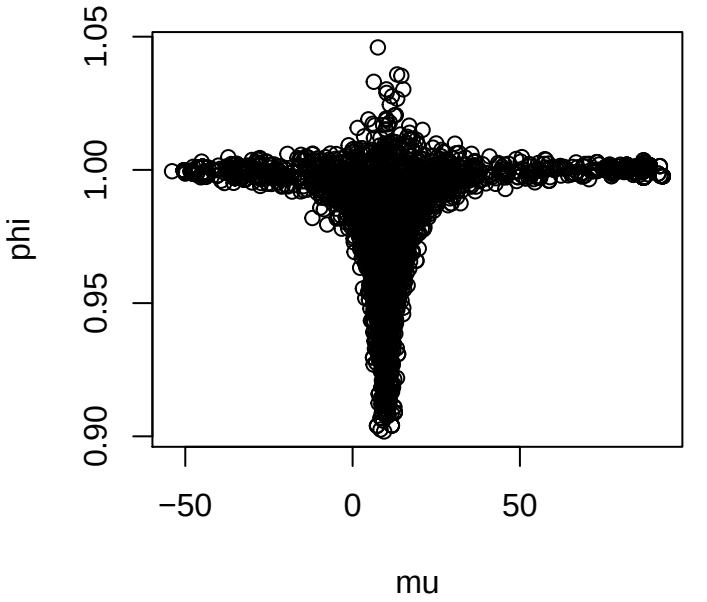
```
## Inference for Stan model: 3118d30d9147b2cf56d6b63fb0063ccd.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean     sd    2.5%  97.5% n_eff  Rhat
## mu     10.934   5.949 24.182 -44.260 86.445    17 1.299
## sigma2  2.085   0.015  0.211   1.696  2.505   205 1.020
## phi     0.974   0.003  0.025   0.917  1.005    96 1.066
##
## Samples were drawn using NUTS(diag_e) at Mon May 20 16:13:25 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).
```





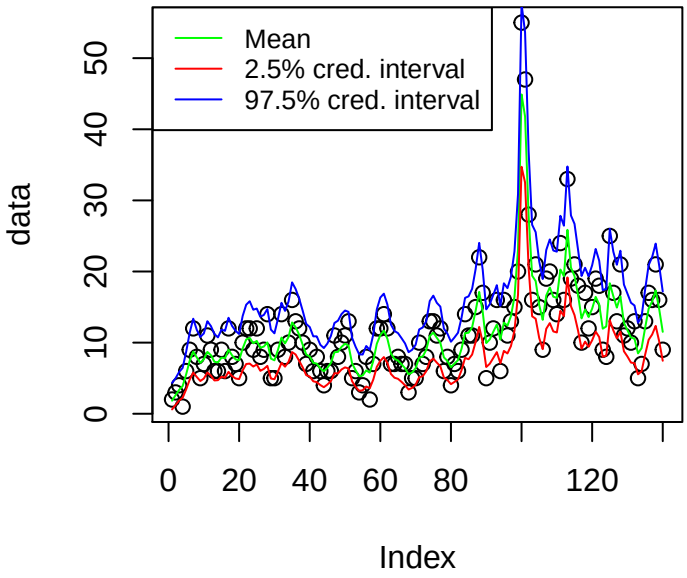
Below are two plots showing the joint posterior of ϕ and μ for the two samplers. We see that μ varies more and more when ϕ is close to one.



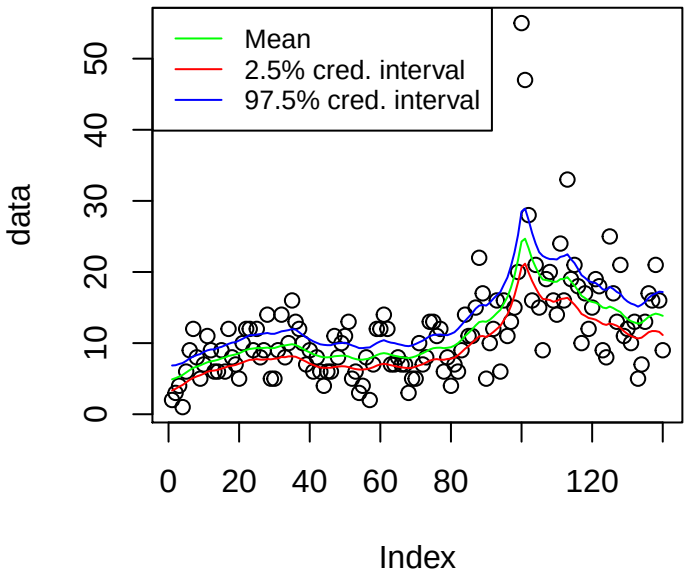


Now a dataset containing the number of cases of campylobacter infections was used. It was assumed that the number of infections c_t at each time point followed an independent Poisson distribution when conditioned on a latent AR(1)-process x_t, $c_t|x_t \sim Poisson(exp(x_t))$. The simulation was done in Stan again, we only used the prior that ϕ is between -1 and 1. Below is the plot of the posterior mean and 95% credible intervals for

the latent intensity $\theta_t = exp(x_t)$.



The assignment above was repeated but with a different prior of σ^2. σ^2 was supposed to be small, so we used a scaled inverse chi squared distribution to get a small σ^2. The following plot shows the new result.



As we can see, the intensity is much smoother, so the posterior has changed.

Appendix

```
set.seed(123)
## a)

T=200
mu = 10
sigma2 = 2

t = seq(2,T)
x1 = mu
phi = seq(-1,1,by=0.1)


ar.sim = function(phi){
  x_sample = rep(0,length=length(t))
  x_sample[1] = x1
  for(time in t){
    epsilon_t = rnorm(1,0,sqrt(sigma2))
    x_sample[time] = mu + phi*(x_sample[time-1] - mu) + epsilon_t
  }
  return (x_sample)
}

ar_sims = matrix(ncol = length(t)+1, nrow = length(phi),data = 0)

for (i in 1:length(phi)){
  ar_sims[i,] = ar.sim(phi[i])
}
#Hur ska man tolka det här?
plot(ar_sims[1,],type = 'l', main = paste("phi = ",phi[1]), ylab = "xt")
plot(ar_sims[11,],type = 'l',main = paste("phi = ",phi[11]), ylab = "xt")
plot(ar_sims[21,],type = 'l', main = paste("phi = ",phi[21]), ylab = "xt")

## b)
suppressMessages(library('rstan'))

x = ar.sim(0.3)
y = ar.sim(0.95)

StanModel = '
data{
  int<lower=1> N;
  real x[N];
}
parameters{
  real mu;
  real phi;
  real sigma2;
}
transformed parameters {
  real sigma;
  sigma = sqrt(sigma2);
```

```
}
model{
  for(n in 2:N){
    x[n] ~ normal(mu + phi*(x[n-1]-mu), sigma);
  }
}
'
x.fit = stan(model_code = StanModel, data = list(N = T, x=x),refresh = 0)
y.fit = stan(model_code = StanModel, data = list(N = T, x=y),refresh = 0)

print("Parameters for x")
print(x.fit, digits_summary = 3,pars = c('mu','sigma2','phi'),probs = c(0.025,0.975))
cat("\nParameters for y\n")
print(y.fit,digits_summary = 3,pars = c('mu','sigma2','phi'),probs = c(0.025,0.975))

plot(extract(x.fit)$mu,type = 'l',ylab = c("mu"), main = paste("phi = 0.3"))
plot(extract(y.fit)$mu, type = 'l',ylab="mu", main = paste("phi = 0.95"))
plot(extract(x.fit)$mu,extract(x.fit)$phi)
plot(extract(y.fit)$mu,extract(y.fit)$phi)

## c)
data = read.delim("~/Documents/TDDE07/Lab4/campy.dat")

#xt = ar.sim(0.3)

StanModel2 = '
data{
  int<lower=1> N;
  int c[N];
}
parameters{
  real x[N];
  real mu;
  real phi;
  real sigma2;
}
transformed parameters {
  real sigma;
  sigma = sqrt(sigma2);
}
model{
  phi ~ uniform(-1, 1);
  for(n in 2:N){
    x[n] ~ normal(mu + phi*(x[n-1]-mu), sigma);
    c[n-1] ~ poisson(exp(x[n-1]));
  }
  c[N] ~ poisson(exp(x[N]));
}
'

fit.stan = function(stanModel, data){
  c.fit = stan(model_code = stanModel,data = list(N = length(data$c),c=data$c))
}
```

```
plot.stan = function(model, data){
  #Last five elements were not wanted.
  mean = exp(summary(model)$summary[1:140,'mean'])
  cred_97.5 = exp(summary(model)$summary[1:140,'97.5%'])
  cred_2.5 = exp(summary(model)$summary[1:140,'2.5%'])
  plot(data$c)
  lines(mean,col="green")
  lines(cred_2.5,col="red")
  lines(cred_97.5,col="blue")
  legend("topleft", legend = c("Mean", "2.5% cred. interval", "97.5% cred. interval"),
         col = c("green", "red", "blue"), lty = 1, cex = 0.8)
}

c.fit = fit.stan(StanModel2, data)
plot.stan(c.fit, data)

## d)

StanModel3 = '
data{
  int<lower=1> N;
  int c[N];
}
parameters{
  real x[N];
  real mu;
  real phi;
  real<lower=0> sigma2;
}
transformed parameters {
  real sigma;
  sigma = sqrt(sigma2);
}
model{
  phi ~ uniform(-1, 1);
  sigma2 ~ scaled_inv_chi_square(N,0.05);
  for(n in 2:N){
    x[n] ~ normal(mu + phi*(x[n-1]-mu), sigma);
    c[n-1] ~ poisson(exp(x[n-1]));
  }
  c[N] ~ poisson(exp(x[N]));
}
'

c.fit = fit.stan(StanModel3, data)
plot.stan(c.fit, data)
```