Bayesian Leraning

Computer lab 3

Elvin Granat (elvgr805)
Max Fischer (maxfi539)

1. Normal model, mixture of normal model with semi-conjugate prior.

A dataset consisting of 6919 daily records of precipitation was analyzed using two different models:

a) Normal model
b) Mixture normal model

The recordings range from early 1948 to end of 1983 and includes both rain and snow in units of 1/100 inch. Recordings of zero precipitation are excluded.

a) Normal model
The daily precipitation {y1, ..., yn} are assumed to be independent normally distributed:

$$y_1, ..., y_n | \mu, \sigma^2 \sim N(\mu, \sigma^2)$$

$$\mu \sim N(\mu_0, \tau_0^2)$$

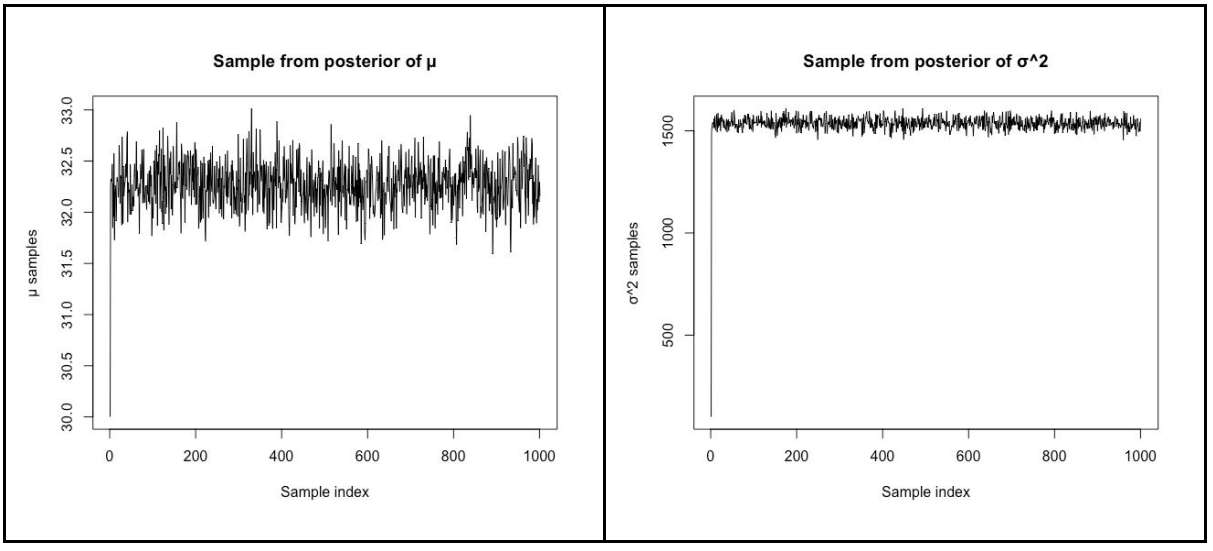
$$\sigma^2 \sim Inv - \chi^2(v_0, \sigma_0^2)$$

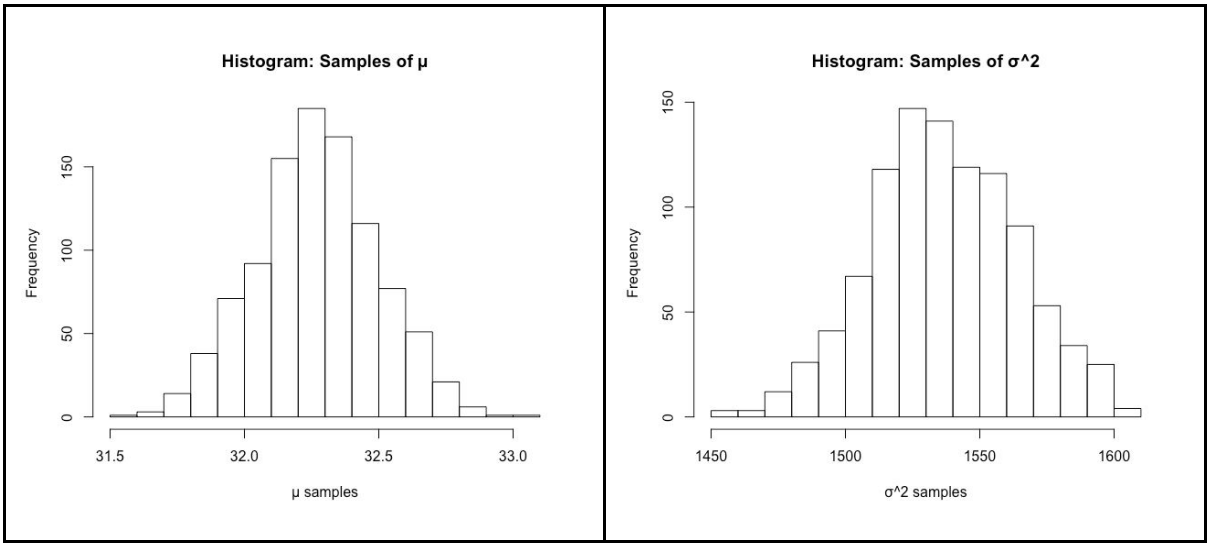
A Gibbs-sampler was implemented that simulated from the joint posterior:

$$p(\mu, \sigma^2 | y_1, ..., y_n)$$

```
# Import data
rainfall <- read.csv("rainfall.dat")

# User input - Priors
my_0 <- 30 # μ prior
tao2_0 <- 1500 # μ prior
v_0 <- 40 # σ^2 prior
sigma2_0 <-100 # σ^2 prior
nDraws <- 1000 # No of Gibbs iterations
n <- dim(rainfall)[1] # No of observations

v_n <- v_0 + n
y_mean <- mean(rainfall$X136) # Mean of data
my_sample <- numeric(nDraws) # Vector for μ samples
my_sample[1] <- my_0
sigma2_sample <- numeric(nDraws) # Vector for σ^2 samples
sigma2_sample[1] <- sigma2_0

for (i in 1:nDraws) {

  w <- (n/sigma2_sample[i])/((n/sigma2_sample[i])+(1/tao2_0)) # Weight to calculate my_n
  tao2_n <- (1/((n/sigma2_sample[i]) + (1/tao2_0))) # Taon to sample μ

  my_n <- w*y_mean + (1 - w)*my_0 # Taon to sample μ

  # Draw posterior of my
  my_sample[i+1] <- rnorm(1, mean = my_n, sd = tao2_n)

  # Draw posterior of sigma2
  X <- rchisq(n=n, df=n-1)
  sigma2_sample[i+1] <- n * ((v_0*sigma2_0 + sum((rainfall$X136 - my_sample[i+1])^2))/v_n)/X
}

plot(my_sample, type="l")
plot(sigma2_sample, type="l")
hist(my_sample[-1], xlab = "μ samples", main = "Histogram: Samples of μ", breaks = 20)
hist(sigma2_sample[-1], xlab = "σ^2 samples", main = "Histogram: Samples of σ^2", breaks = 20)
```

```r
for (i in 1:10000) {
  ## mu sampling
  # mun
  w = (n / sigma2.sample) / ( n / sigma2.sample + 1 / tau0^2 )
  mun = w * mean(x) + (1 - w) / mu0
  # taun2
  taun2 = 1 / (n / sigma2.sample + 1 / tau0^2)
  # sample mu
  mu.sample = rnorm(1, mean = mun, sd = sqrt(taun2))
  #mu.sample = rexp(1, rate = 1 / mun) if you want to try exponential
  mu.samples[i] = mu.sample

  ## sigma sampling
  # vn
  vn = n + v0
  # sigma2n
  sigma2n = (v0 * sigma0^2 + sum((x - mu.sample)^2)) / (n + v0)
  chi2 = rchisq(vn, n=1)
  sigma2.sample = vn * sigma2n / chi2
  sigma2.samples[i] = sigma2.sample
}
# plot of the trajectories
plot(mu.samples, type="l")
plot(sigma2.samples, type="l")
```





According to the trajectory plots above, it seems like the Gibbs samplers of μ and σ^2 converges quite quickly towards:

$$\mu \approx 32.25$$

$$\sigma^2 \approx 1525$$

This is also confirmed by the histogram plots.

b) Mixture normal model

The daily precipitation {y1, ..., yn} are now assumed to follow an iid two-component mixture of normal models:

$$p(y_1,...,y_n|\mu,\ \sigma^2,\pi)\ =\ \pi * N(y_1,...,y_n|\mu_1,\ \sigma_1^2) + (1-\pi) * N(y_1,...,y_n|\mu_2,\ \sigma_2^2)$$

$$\mu = (\mu_1, \mu_2)$$
$$\sigma^2 = (\sigma_1^2, \sigma_2^2)$$

A Gibbs sampling data augmentation algorithm was provided to analyze the daily precipitation data.

The following priors was chosen:

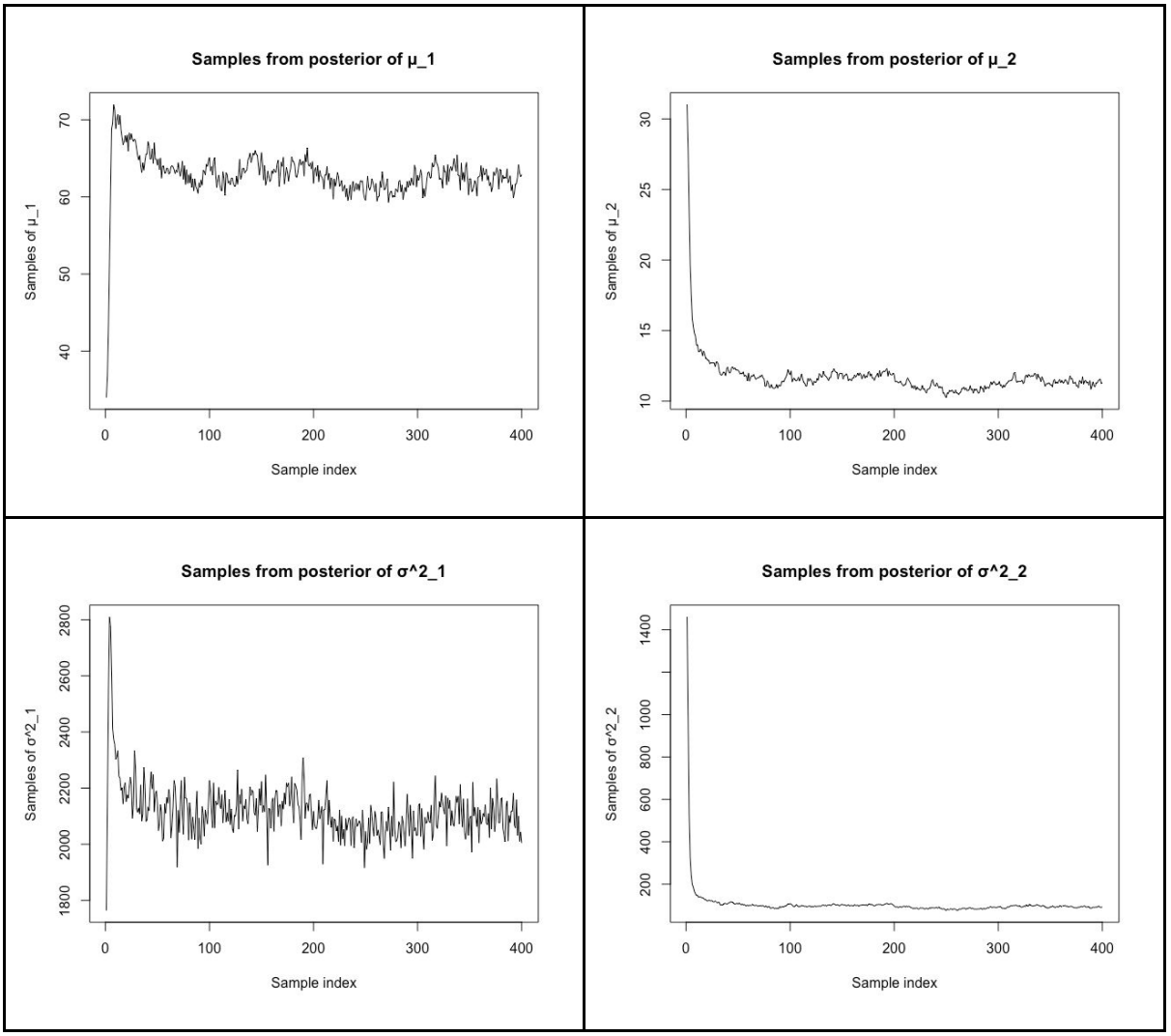
$$\alpha = (\alpha_1, \alpha_2)\ =\ (10,\ 10)$$
$$\mu_0 = (\mu_{0,1}, \mu_{0,2})\ =\ (mean(data),\ mean(data))\ =\ (32.2681,\ 32.2681)$$
$$\tau_0^2 = (\tau_{0,1}^2, \tau_{0,2}^2)\ =\ (100,\ 100)$$
$$\sigma_0^2 = (\sigma_{0,1}^2, \sigma_{0,2}^2)\ =\ (40^2,\ 40^2) = (1600,\ 1600)$$
$$\nu_0 = (\nu_{0,1},\ \nu_{0,2})\ =\ (5,\ 5)$$

Initially, (π_1, π_2) oscillated up to around (0.70, 0.30) at iteration no 8, allocating the observations as following:

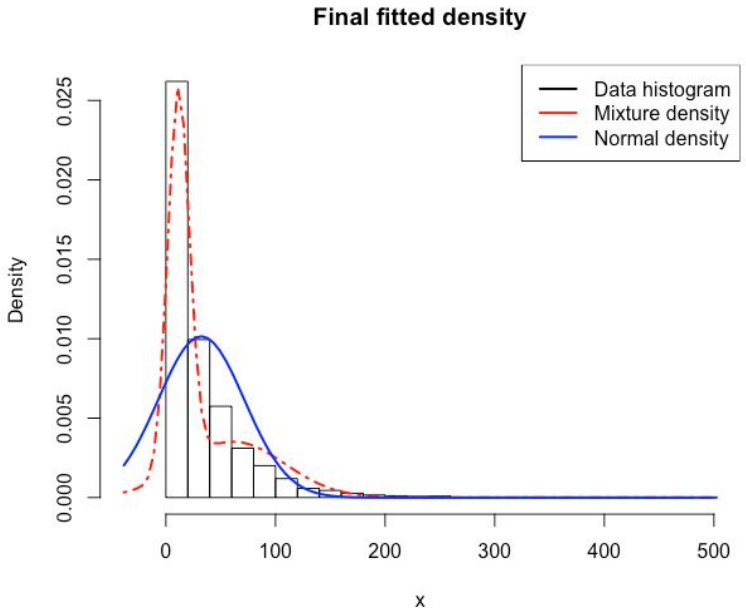
(2155, 4764)

After that, (π_1, π_2) slowly converged towards numbers close to (0.60, 0.30), which happened around iteration no 30, allocating the observations as following:

(2723, 4196)



c) Graphical comparison



2. Time series models in Stan

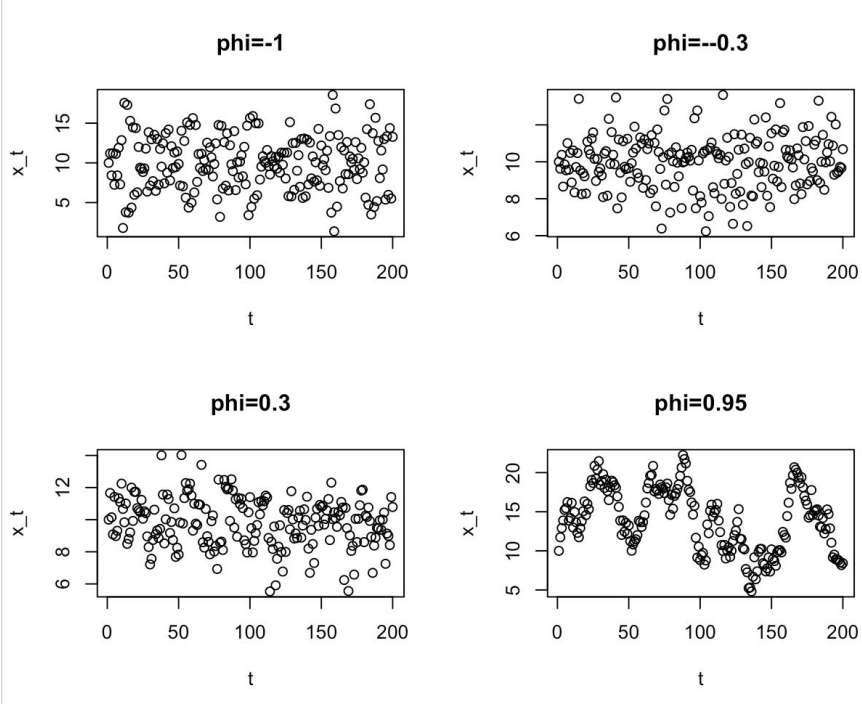
a) Function that simulates from an AR(1) process with given values for μ, σ and values of Φ between -1 and 1 (this is where an AR(1) process is stable)

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

Simulating T values, starting from t = 1 and continuing until t = T = 200.

```
AR.simulation = function(mu, sigma2, T, phi) {
  x.sample = numeric()
  x.sample[1] = mu
  for(i in 2:T) {
    x.sample[i] = mu + phi * (x.sample[i-1] - mu) + rnorm(1, mean = 0, sd = sqrt(sigma2))
  }
  return(x.sample)
}

### Implementation
# a)

phis = seq(-1,1,0.05)
n = length(phis)
x.samples = matrix(nrow=n, ncol=T, 0)
for(i in 1:n) {
  x.samples[i,] = AR.simulation(mu, sigma2, T, phis[i])
}
```

Plotting some results:



Negative Φ resutls in the x_t values alternating a more around the mean value whereas for positive, especially larger, Φ values the curve follows a temporary trend.

b) Now μ, σ and Φ are treated as unknown. Given data sampled using the function above with same values as before with Φ=0.3 and Φ=0.95.

```
# sample AR processes with phi=0.3 and phi=0.95
x.03 = AR.simulation(mu, sigma2, T, phi=0.3)
x.95 = AR.simulation(mu, sigma2, T, phi=0.95)
```

Estimating the values from μ, σ and Φ using MCMC in Rstan:

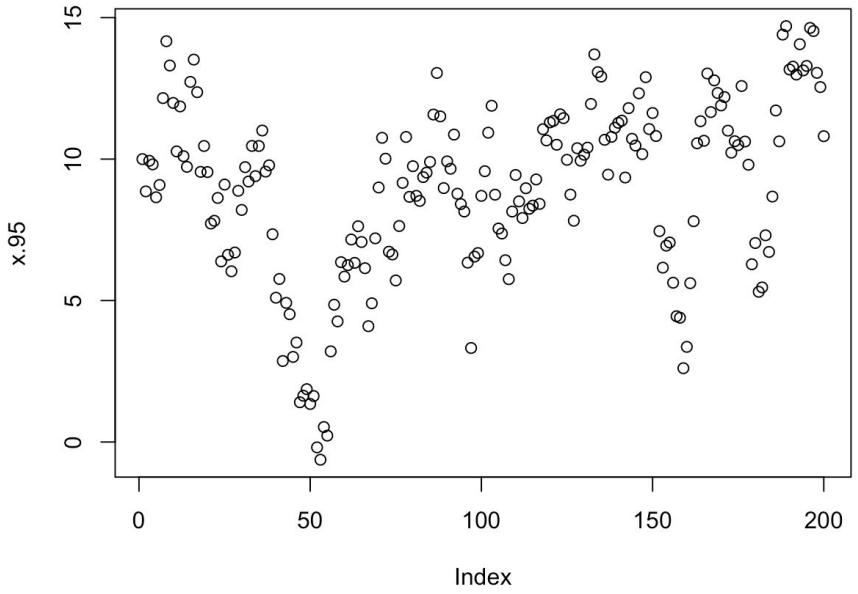
```
ARStanModel = 'data {
  int<lower=0> N;
  vector[N] x;
}
parameters {
  real mu;
  real phi;
  real<lower=0> sigma;
}
model {
  for (n in 2:N)
    x[n] ~ normal(mu + phi * (x[n-1] - mu), sigma);
}'

# perform MCMC
x.fit.03 = stan(model_code=ARStanModel, data=list(x=x.03, N=T))
x.fit.95 = stan(model_code=ARStanModel, data=list(x=x.95, N=T))
```

The number of effective samples, posterior mean and credible interval for each parameter.

Φ = 0.3	#effective samples	Posterior mean	CI
μ	2929	10.123	(9.79, 10.45)
σ	3335	1.438	(1.31, 1.58)
Φ	3151	0.371	(0.241, 0.510)

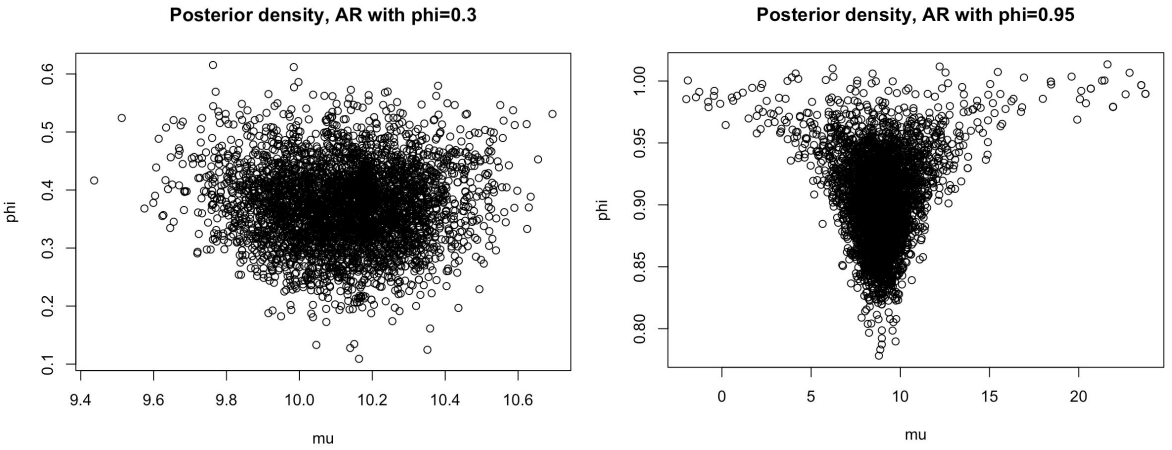
Φ = 0.95	#effective samples	Mean	CI
μ	594	8.960	(5.85, 12.34)
σ	2887	1.479	(1.34, 1.64)
Φ	1517	0.903	(0.838, 0.979)

Both attempts estimate the parameter values well. However, the number of effective samples are lower and the credible intervals are wider for μ with the dataset based on Φ = 0.95. We believe this is because the trendy pattern that can easily to far below and above the μ value may result in a skewed dataset given only 200 observations in the sample.



Plotting the sampled data shows that a estimation of μ below 10 is not unreasonable given how many data points are below 10 given the argument above.

Plotting the joint posterior of Φ and μ:



Comment: The posterior density with Φ = 0.3 looks like the density espected when based on a normal distribution. Φ = 0.95 however looks to be a bit skewed in its density for Φ values close to 1. This may be because:

- the given dataset gives reasonable probabilities when calculating α (acceptance probabiliy) with Φ values close to 1
- that a sampled Φ close to 1 results in high likelihood regardless of sampled μ value.

This is also expressed in the #effective samples for μ and Φ compared to Φ = 0.3 seen in the table above, where both of these parameters have a somewhat low #effective samples, especially μ with 594 out of 4000 effective samples. This calls for either a better proposed distribution when sampling or more samples to witness the “true” shape of the posterior.

c) Model the number of campylobacter infections in four week intervals:

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

Where c_t is contioned on a latent AR(1) process (= we do not have any observations from the AR process). This results in that the AR process x_t should be seen as a parameter.

Estimating the model using Rstan:

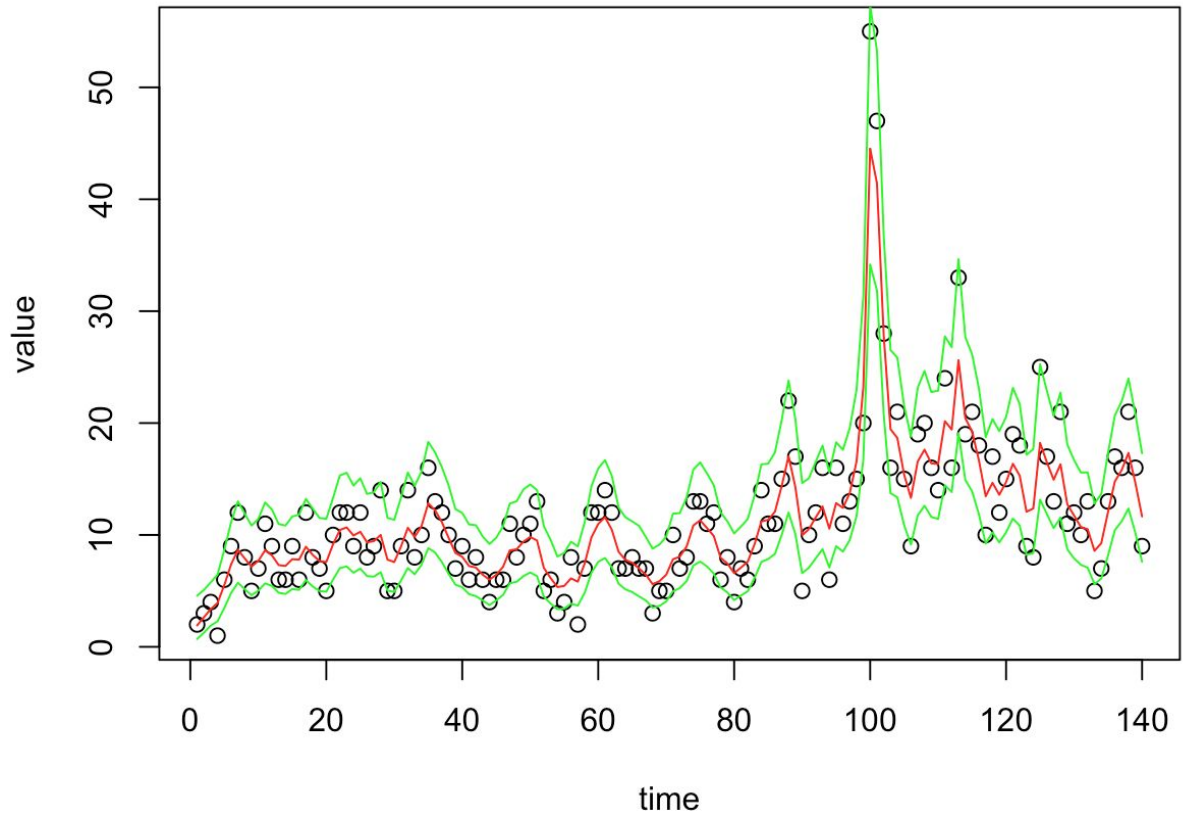
```
data.campy = read.table("campy.dat", header=TRUE)[,1]
N = length(data.campy)

PoissonARStanModel = '
data {
  int<lower=0> N;
  int c[N];
}
parameters {
  real mu;
  real phi;
  real<lower=0> sigma;
  vector[N] x;
}
model {
  // mu ~ normal();
  // sigma2 ~ scaled_inv_chi_square();
  // phi ~ normal(0, 0.6);
  phi ~ uniform(-1,1);
  for (n in 2:N)
    x[n] ~ normal(mu + phi * (x[n-1] - mu), sigma);
  for (n in 1:N)
    c[n] ~ poisson(exp(x[n]));
}'
c.fit = stan(model_code=PoissonARStanModel, data=list(N = N, c = data.campy))
```

We use uninformative priors for sigma and mu since we do not know how the latent variable behaves what so ever. For Φ however we tried using a uniform prior given that we know it should be between -1 and 1, we also tried a normal prior with mean 0 and a reasonable variance, the results were very similar though.

Confidence interval for the latent intensity $\theta_t = \exp(x_t)$:

```
X.summary = summary(c.fit)$summary[-c(1,2,3),]

# extract CI and mean of x
x.upper = X.summary[,"97.5%"]
x.lower = X.summary[,"2.5%"]
x.mean = X.summary[,"mean"]

# plot data, mean and CI of the latent intensity
plot(data.campy, xlab="time", ylab="value")
lines(exp(x.upper), col="green")
lines(exp(x.lower), col="green")
lines(exp(x.mean), col="red")
```

Confidence interval with the data points:

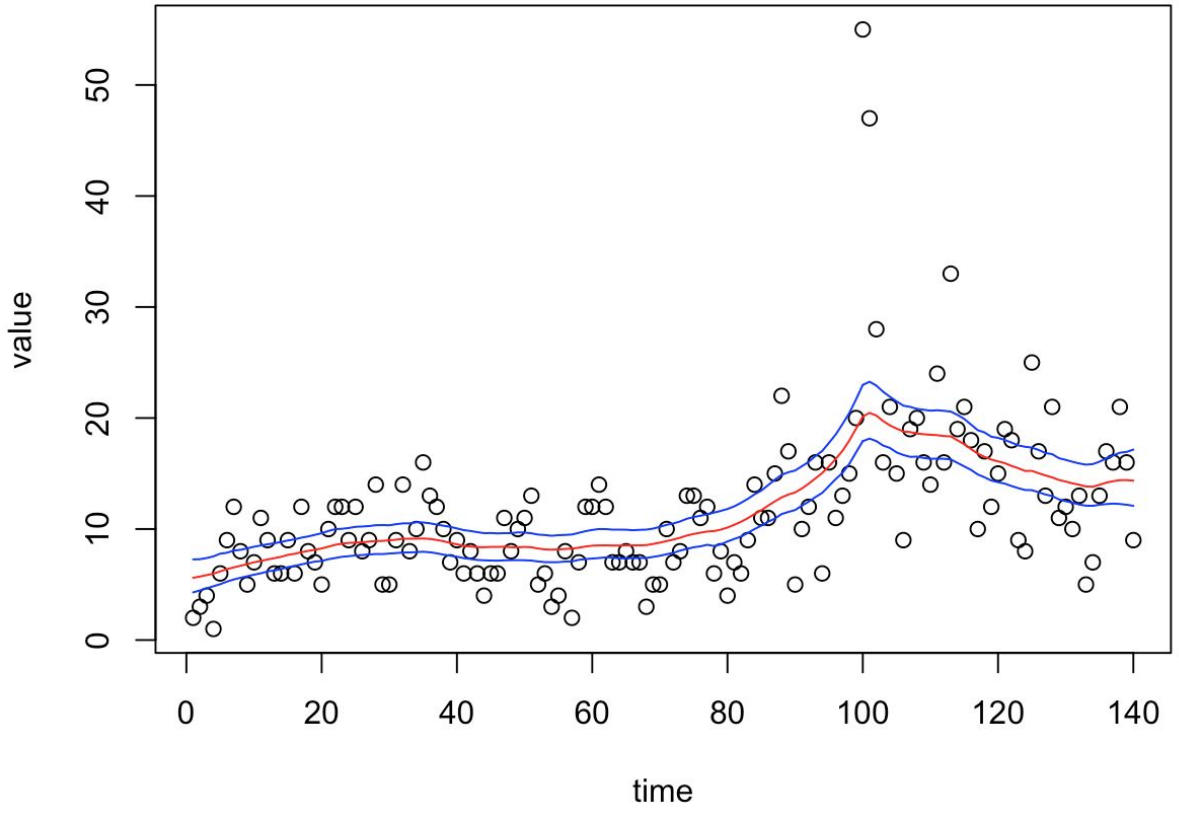


The latent intensity follows the data which is nice, however it seems a bit overfitted given that it follows the data too much. It is very volatile, a smoothness factor seems reasonable.

d) Given the shape of the confidence intervals we added a prior to the variance σ^2 telling the sampler that the value of σ^2 should not be too large. We used a Scaled-Inv χ^2 estimating the σ^2 to be closer to 0:

```
model {
  // mu ~ normal();
  // phi ~ normal(0, 0.6);
  sigma2 ~ scaled_inv_chi_square(N, 0.03);
  phi ~ uniform(-1,1);
  for (n in 2:N)
    x[n] ~ normal(mu + phi * (x[n-1] - mu), sqrt(sigma2));
  for (n in 1:N)
    c[n] ~ poisson(exp(x[n]));
}'
```

Resulting plot of the confidence interval of the latent intensity:



These curves are a lot smoother while still following the pattern of the data, hence it seems like a more reasonable model of the data since it is not as overfitted in case of future predictions.