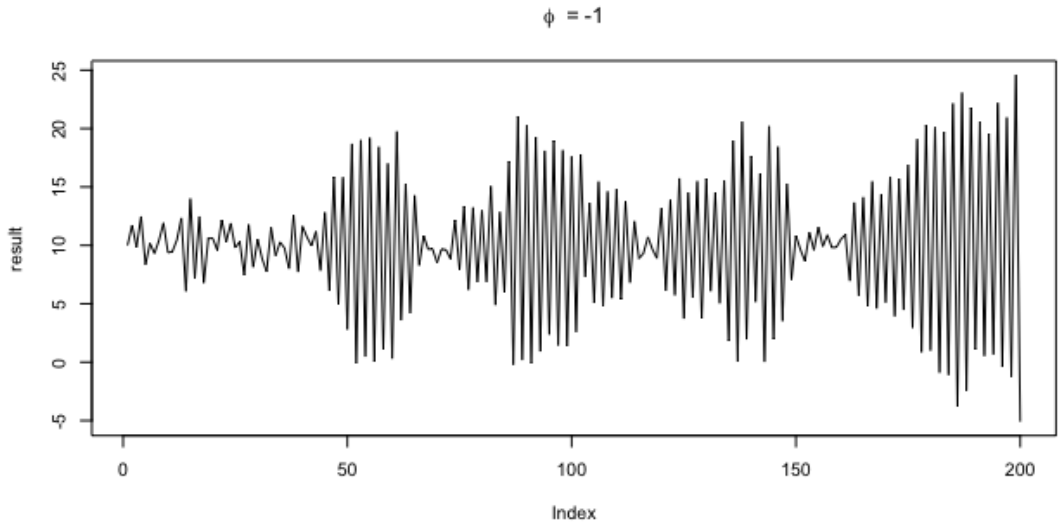
# TDDE07 - Lab 4

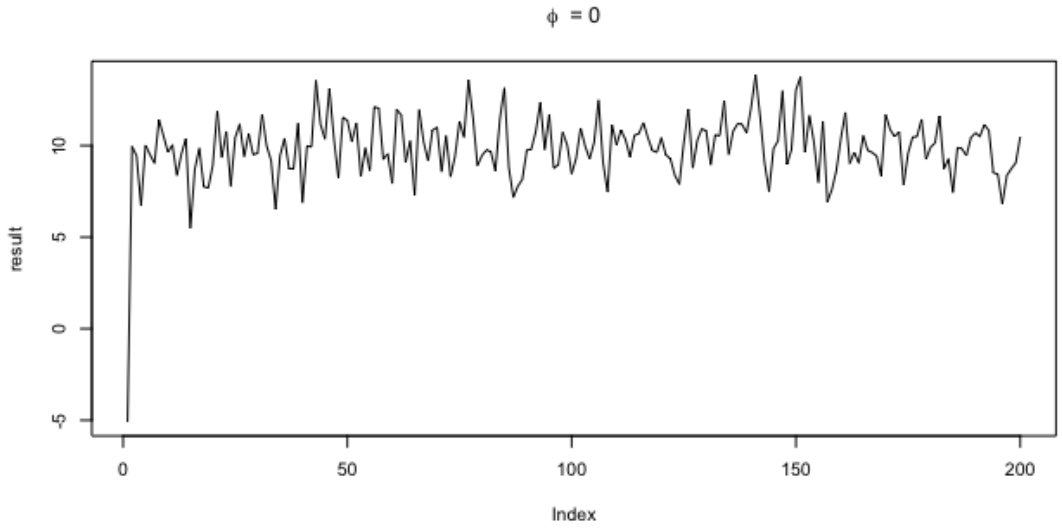
*Sophie Lindberg - sopli268*
*Arvid Edenheim - arved490*

*2019-06-10*

# 1 - Time series models in Stan

## 1a) AR(1)-process




It is clear that $\phi$ has a large effect on $x_{1:T}$. The figure above depicts 3 different scenarios where $\phi$ is equal to -1, 0 and 1. In the first example, we have that

$$\text{x t } = \mu \text{ - (x t-1 } \text{ - } \mu) + \epsilon_t$$

As can be seen in the figure above the values of $x_t$ oscillate between positive and negatiive values when $\phi = -1$. This is because $\phi$ changes the sign of $x_t$ in each iteration.

When $\phi = 0$ we have that

$$\text{x t } = \mu + \epsilon_t$$

In this case the value of $x_t$ only depends on $\mu$ and $\epsilon_t$, meaning the values will be close to $\mu$.

In the last example we have that

$$\text{x t } = \mu + \text{(x t-1 } \text{ - } \mu) + \epsilon_t$$

In this case, $x_t$ value depends on the previous value but won't change sign in each iteration as long as the error is fairly small.

## 1b)

Using our AR-function defined in a) resulted in the following values for the posterior mean and the upper and lower limits of the 95 % credible intreval.

| phi=0.95 | Posterior mean | Lower limit | Upper limit |
|---|---|---|---|
| mu | NA | NA | NA |
| sigma2 | 2.143 | 1.752 | 2.593 |
| phi | 0.968 | 0.944 | 1.000 |

| phi=0.30 | Posterior mean | Lower limit | Upper limit |
|---|---|---|---|
| mu | 10.188 | 9.874 | 10.497 |
| sigma2 | 2.176 | 1.790 | 2.654 |
| phi | 0.328 | 0.194 | 0.464 |

The convergence of $\phi$ and $\mu$ can be seen by plotting the parameters in the same graph. The red dot represent the posterior mean of each parameter. The shape of the joint distribution of $\phi$ and $\mu$ is the same for different initial values of $\phi$.

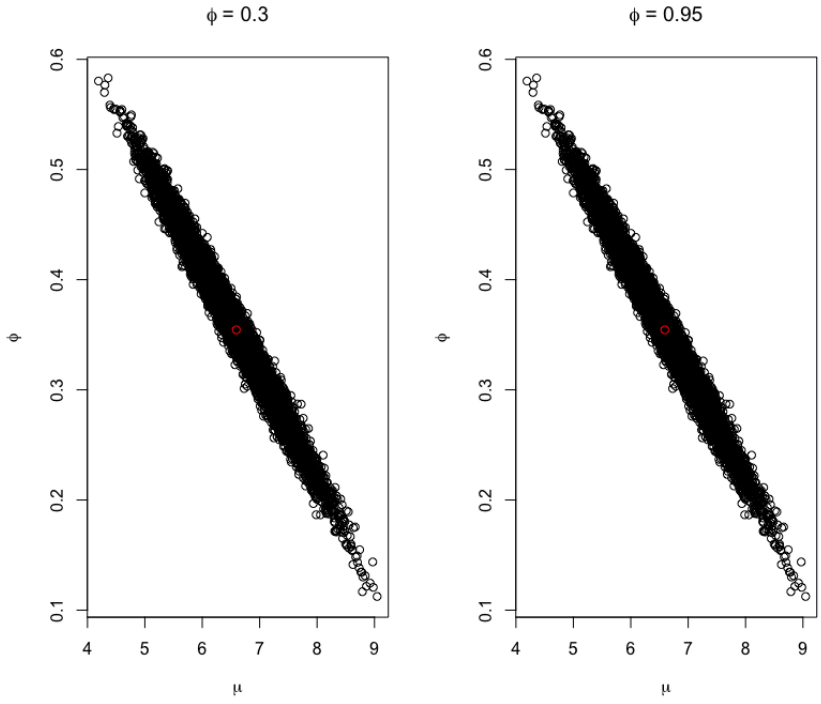



Figure 1: Convergence of parameters

The chains are mixing fairly well for $\phi = 0.3$, with convergence towards the true values ($\mu = 10, \sigma^2 = 2, \phi = 0.3$), while $\phi = 0.95$ results in relatively poor mixing for all but $\sigma^2$, with no actual convergence for $\mu$.

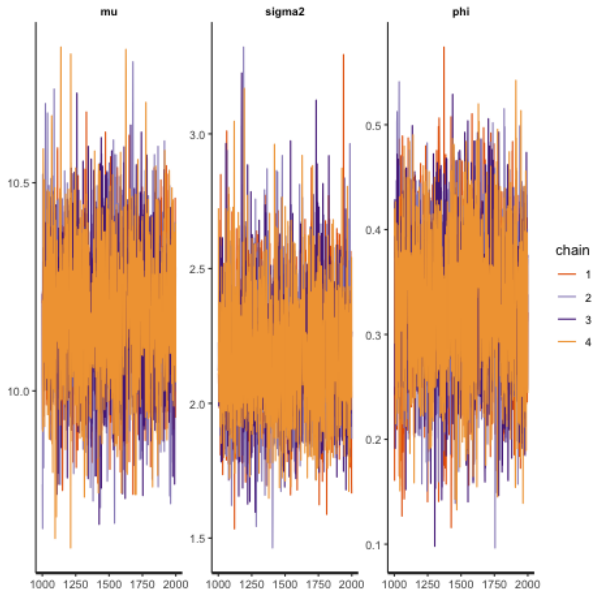



Figure 2: Sample trajectories for $\phi = 0.3$

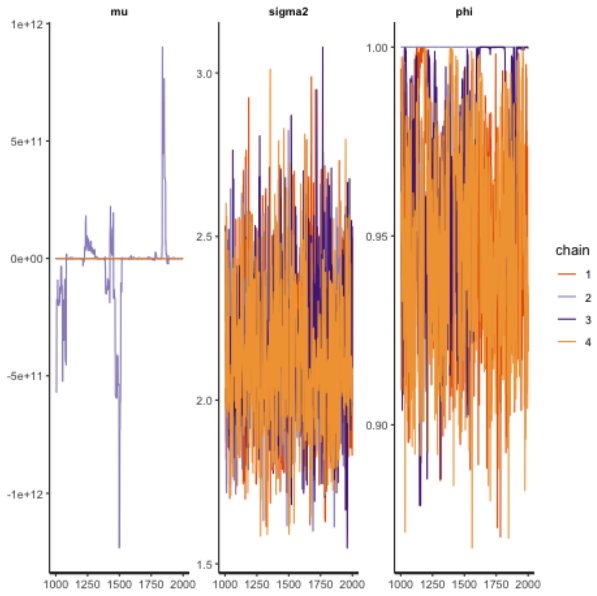



Figure 3: Sample trajectories for $\phi = 0.3$

## 1c)

The figure below shows a plot of the data as well as the posterior mean and a 95 % credible interval for the letaent intensity.

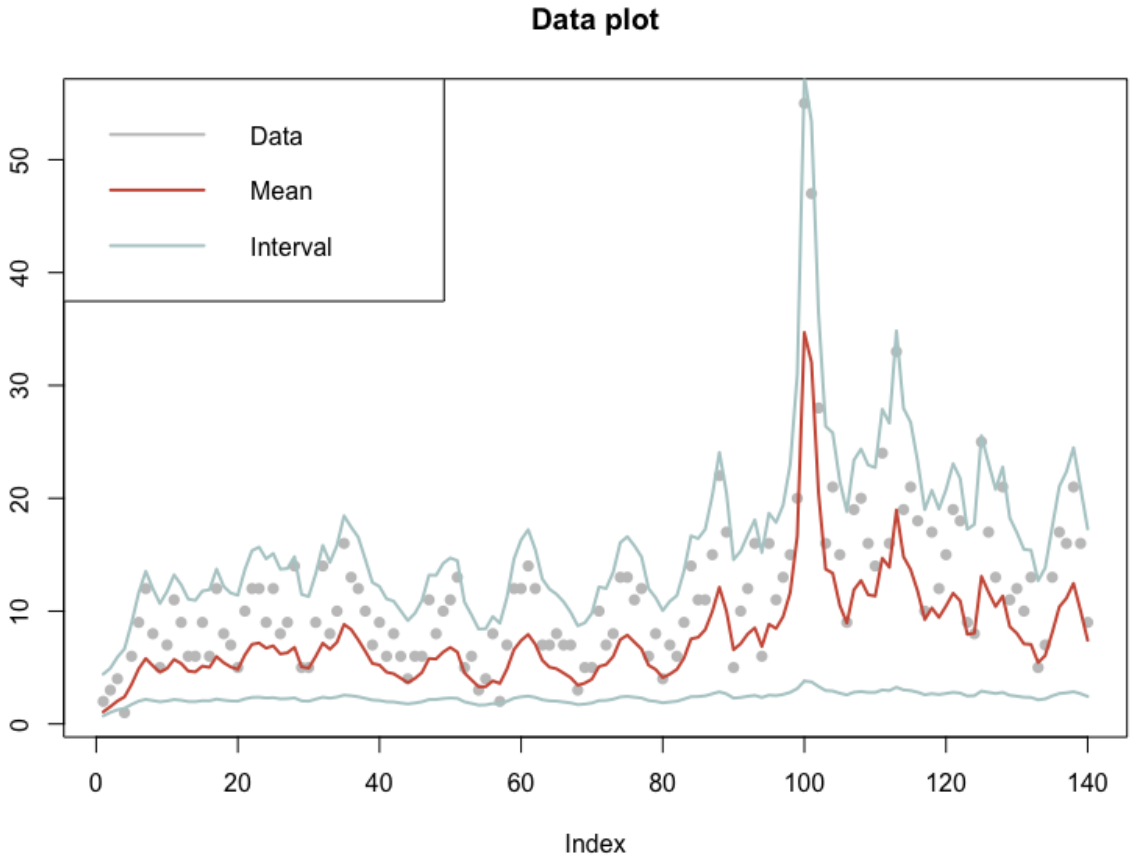



Figure 4: Task C

## 1d)

Here the prior for $\sigma^2$ has been modified to be informative. We set the prior to be

$$\sigma^2 \sim Gamma(1, 1000)$$

Since the prior for sigma has a mean close to 0 and low variance, the effect of the error in the regression model has decreased.

The modified prior had an effect on the posterior mean to be less sensitive to the data. This is an expected outcome since the prior, by being more informative, had a larger impact on the posterior.

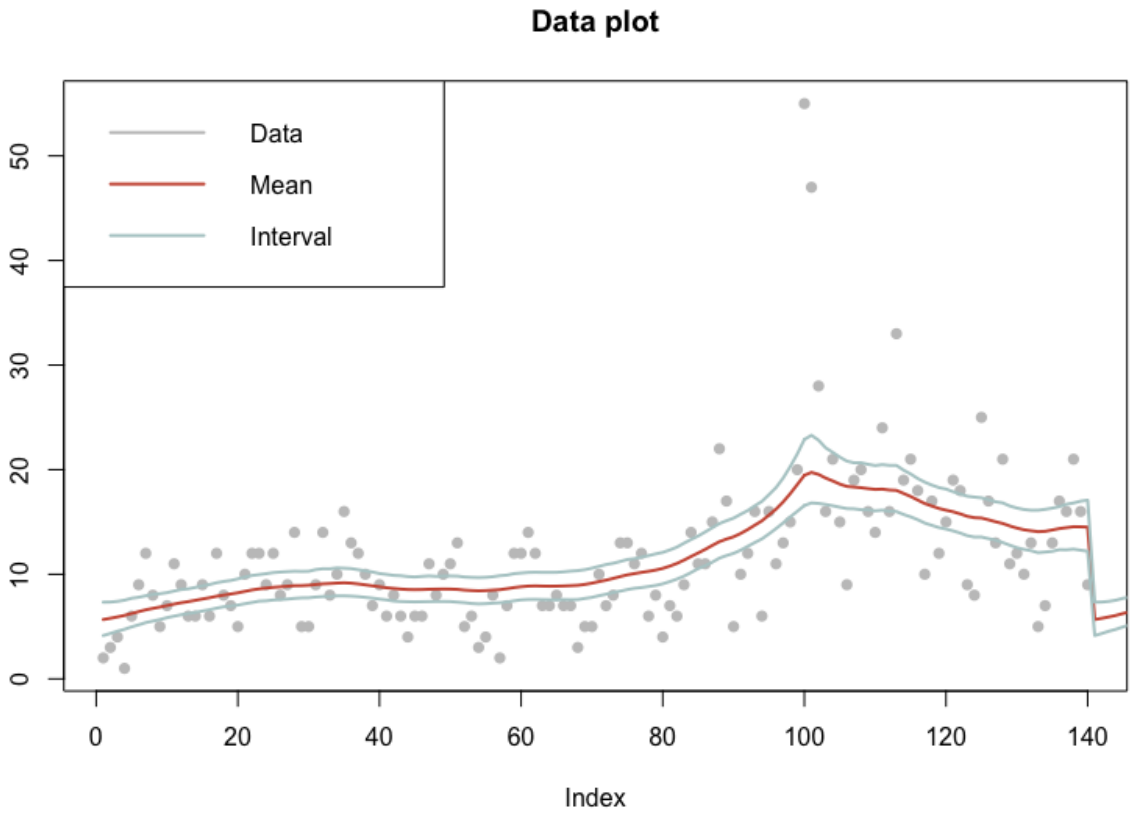



Figure 5: Task D

## Code - 1

```
library('rstan')

########## A ##########
mu <- 10
sigma_sq <- 2
T <- 200
x <- mu

par(mfrow=c(3,1))

AR <- function(mu, prev_x, phi, sigma_sq) {
  error <- rnorm(1, 0, sqrt(sigma_sq))
  return (mu + phi * (prev_x - mu) + error)
}

phis <- c(-1,0.1,0.95)

for(phi in phis) {
  result = c(x)
  for (i in 2:T) {
    x <- AR(mu, x, phi, sigma_sq)
    result <- c(result, x)
  }
  plot(result, type = 'line', main = bquote(phi ~ ' =' ~ .(phi)))
}


########## B ##########

mu <- 10
sigma_sq <- 2
T <- 200
x <- mu

phis <- c(0.3, 0.95)

X = c()
for(phi in phis) {
  result <- c(x)
  for (i in 2:T) {
    x <- AR(mu, x, phi, sigma_sq)
    result <- c(result, x)
  }
  X <- rbind(X, result)
}

model <- stan_model('StanNormalModel.stan')

warmup=1000
iter=2000
niter = 4*(iter-warmup)
```

```
fitX <- sampling(model, data = list(T=200, x=X[1,]), iter = iter, warmup = warmup)
fitY <- sampling(model, data = list(T=200, x=X[2,]), iter = iter, warmup = warmup)

# Print the fitted model
print(fitX,digits_summary=3) # Extract posterior samples
print(fitY,digits_summary=3) # Extract posterior samples

postDrawsX <- extract(fitX)
postDrawsY <- extract(fitX)

par(mfrow=c(1,2))
plot(postDrawsX$mu, postDrawsX$phi, main = expression(phi ~ '= 0.3'), ylab=expression(phi), xlab=expres
lines(mean(postDrawsX$mu), mean(postDrawsX$phi), type='p', col="red")
plot(postDrawsX$mu, type='l')
# legend('topright', legend=expression(mean(postDrawsX$mu) ~ ',' ~ mean(postDrawsX$phi)))
plot(postDrawsY$mu, postDrawsY$phi, main = expression(phi ~ '= 0.95'), ylab=expression(phi), xlab=expre
lines(mean(postDrawsY$mu), mean(postDrawsY$phi), type='p', col="red")
# legend('topright', legend=expression().mean(postDrawsY$mu) ~ ',' ~ ().mean(postDrawsY$phi)))

# plot convergence
png('plots/lab4_b_convergence_x.png')
traceplot(fitX)
dev.off()
png('plots/lab4_b_convergence_y.png')
traceplot(fitY)
dev.off()

########## C ##########
library(ggplot2)
data <- read.table('campy.txt', header=TRUE)

N <- length(data$c)

model <- stan_model('StanPoissonModel.stan')
fit <- sampling(model, data = list(N=N, c=data$c), iter = 2000, warmup = 1000)

# Print the fitted model
print(fit,digits_summary=3) # Extract posterior samples
postDraws <- extract(fit)

x <- postDraws$x

xMean <- c()
thetaUp <- c()
thetaLow <- c()

for (i in 1:length(x[1,])) {
  xMean <- c(xMean, exp(mean(x[,i])))
  thetaUp <- c(thetaUp, exp(quantile(x[,i], probs = 0.975)))
  thetaLow <- c(thetaLow, exp(quantile(x[,i], probs = 0.025)))
}

# draws <- c()
```

```r
# for(i in 1:140) {
#   draws <- c(draws, rpois(1, exp(xMean[i])))
# }

draws <- rpois(140, exp(xMean))

plot(data$c, pch=16, col='gray77', ylab='', main='Data plot')
lines(xMean, col='lightcyan3', lwd=2)
lines(thetaUp, col='lightcyan3', lwd=2)
lines(thetaLow, col='coral3', lwd=2)
legend('topleft',legend = c('Data', 'Mean','Interval'),
       col = c('gray77', 'coral3', 'lightcyan3'), lwd=2)
dev.off()
# geom_ribbon(mapping = aes(seq(1,140,1), ymin = thetaLow, ymax = thetaUp), alpha = 0.25)

# Do traceplots of the first chain
par(mfrow = c(1,1))
# plot(postDraws$mu[1:(1000)],type="l",ylab="mu",main="Traceplot")
# Do automatic traceplots of all chains
# traceplot(fit)
# Bivariate posterior plots
pairs(fit)
# plot(fit)

########## D ##########

data <- read.table('campy.txt', header=TRUE)

N <- length(data$c)

model <- stan_model('StanPoissonModelD.stan')
fit <- sampling(model, data = list(N=N, c=data$c), iter = 2000, warmup = 1000)

# Print the fitted model
print(fit,digits_summary=3) # Extract posterior samples
postDraws <- extract(fit)

x <- postDraws$x

xMean <- c()
thetaUp <- c()
thetaLow <- c()

for (i in 1:length(x[1,])) {
  xMean <- c(xMean, mean(exp(x[,i])))
  thetaUp <- c(thetaUp, quantile(exp(x[,i]), probs = 0.975))
  thetaLow <- c(thetaLow, quantile(exp(x[,i]), probs = 0.025))
}

plot(data$c, pch=16, col='gray77', ylab='', main='Data plot')
lines(xMean, col='coral3', lwd=2)
lines(thetaUp, col='lightcyan3', lwd=2)
lines(thetaLow, col='lightcyan3', lwd=2)
```

```
legend('topleft',legend = c('Data', 'Mean','Interval'),
       col = c('gray77', 'coral3', 'lightcyan3'), lwd=2)


dev.off()
```

## Stan Model for b)

```
data {
  int<lower=0> T; // Number of observations
  real x[T];
}

parameters {
  real mu;
  real<lower=0> sigma2;
  real<lower=-1, upper=1> phi;
}

model {
  //priors
  // mu ~ normal(10,2);
  // phi ~ uniform(-1,1);
  // sigma2 ~ scaled_inv_chi_square(1,2); // Scaled-inv-chi2 with nu
  //
  //likelihood
  for (t in 2:T)
    x[t] ~ normal(mu + phi * x[t-1], sqrt(sigma2));
}
```

## Stan Model for c)

```
data {
  int<lower=0> N; // Number of observations
  int c[N];
}

parameters {
  real mu;
  real<lower=-1,upper=1> phi;
  real<lower=0> sigma;
  real<lower=0> x[N]; // Data points
}

transformed parameters {
  real lambda[N];
  lambda = exp(x);
}

model {
  //likelihood
  for(n in 1:N)
    c[n] ~ poisson(lambda[n]);

  //prior
  for(n in 2:N)
    x[n] ~ normal(mu + phi * x[n-1], sigma);
}
```

## Stan Model for d)

```
data {
  int<lower=0> N; // Number of observations
  int c[N];
}

parameters {
  real mu;
  real phi;
  real<lower=0> sigma;
  real<lower=0> x[N]; // Data points
}

transformed parameters {
  real lambda[N];
  lambda = exp(x);
}

model {
  //likelihood
  for(n in 1:N)
    c[n] ~ poisson(lambda[n]); // Poisson

  //prior
  mu ~ normal(10,2);
  phi ~ normal(0, 1);
  sigma ~ gamma(1,1000);

  for(n in 2:N)
    x[n] ~ normal(mu + phi * x[n-1], sigma);
}
```