TDDE07 - Lab 3

Sebastian Callh, Jacob Lundberg

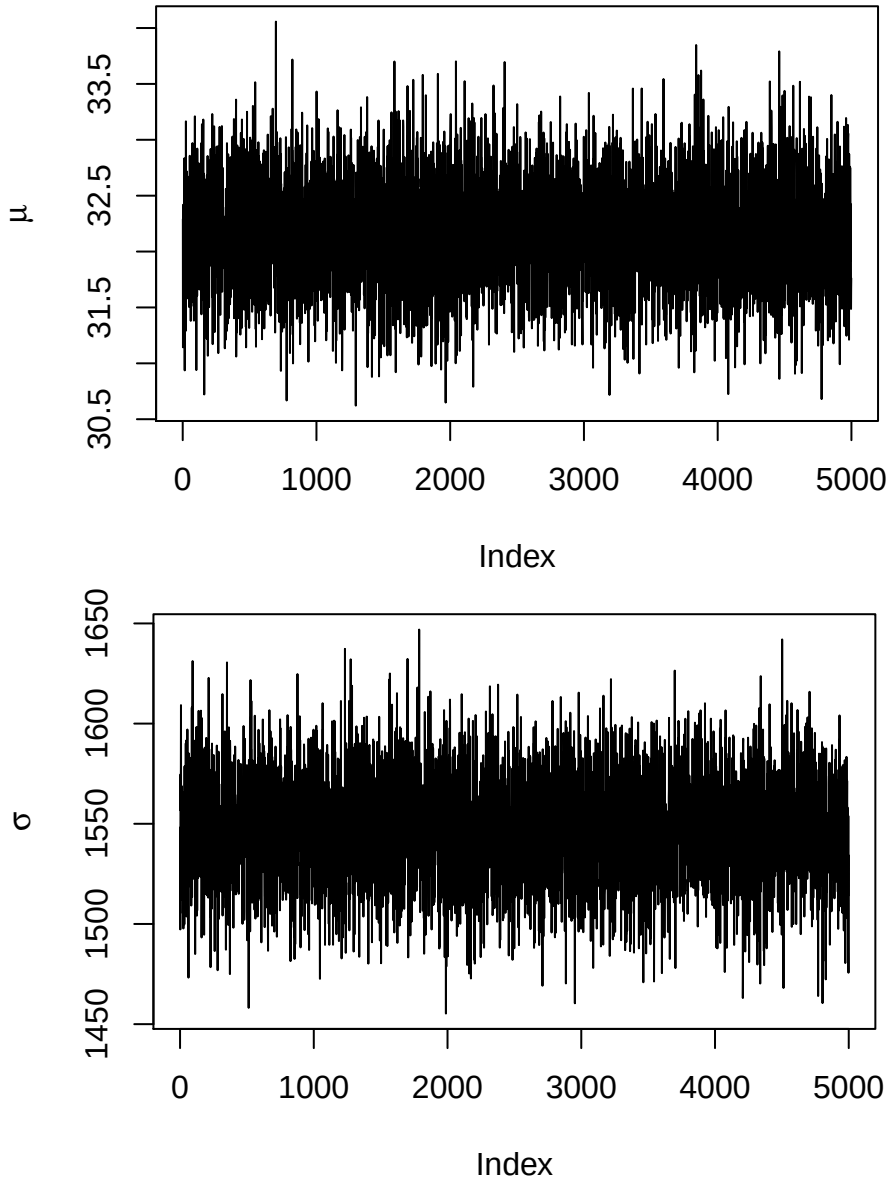
8 maj 2018

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

a) - Normal model

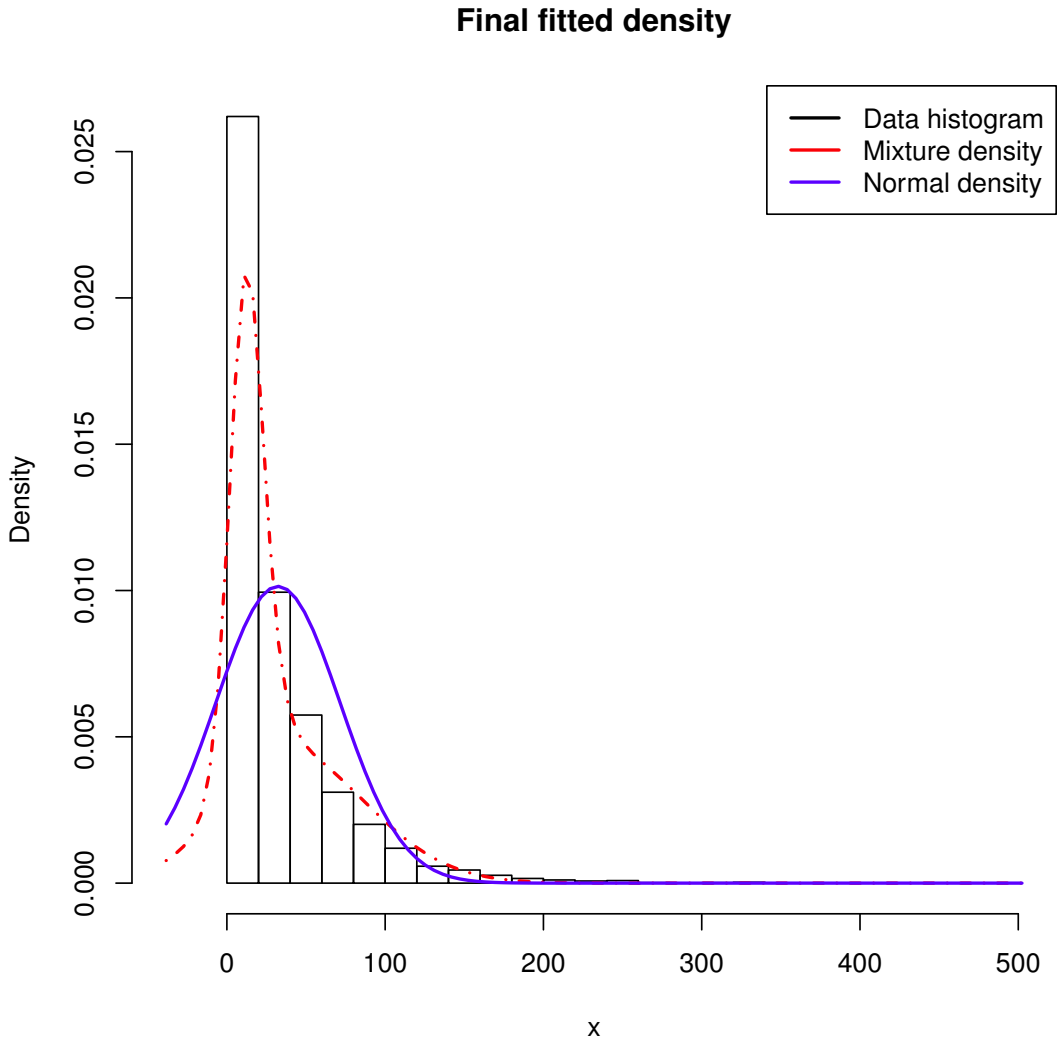
(ii) - Analysis

Looking at the trajectories for μ and σ it can be seen that the values are not very auto correlated, which implies good draws. They also oscillate around a stable value.

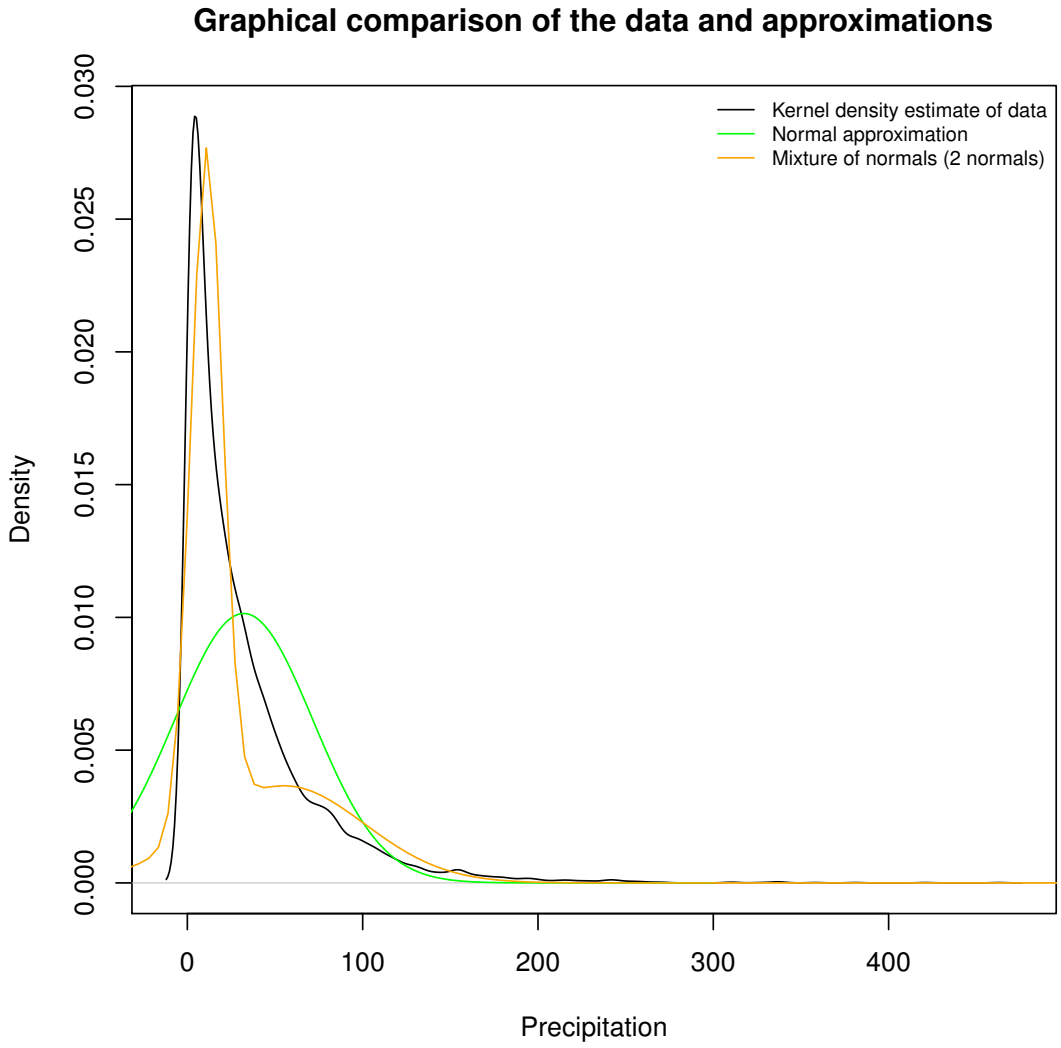


b) Mixture normal model

After running the `NormalMixtureGibbs.R` for 30 iterations the mixture model had converged to the distribution seen in the plot below.



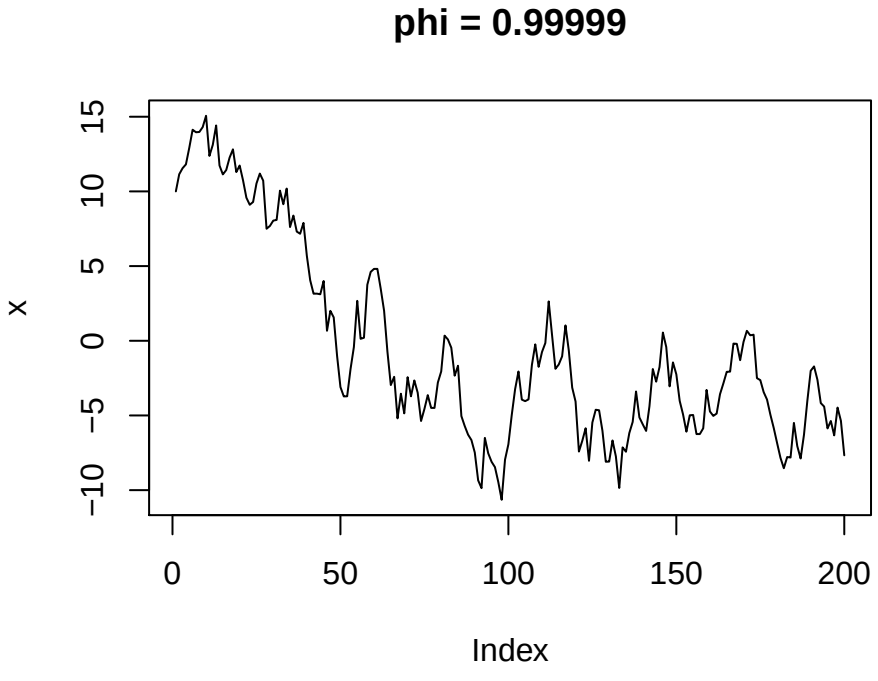
c) Graphical comparison

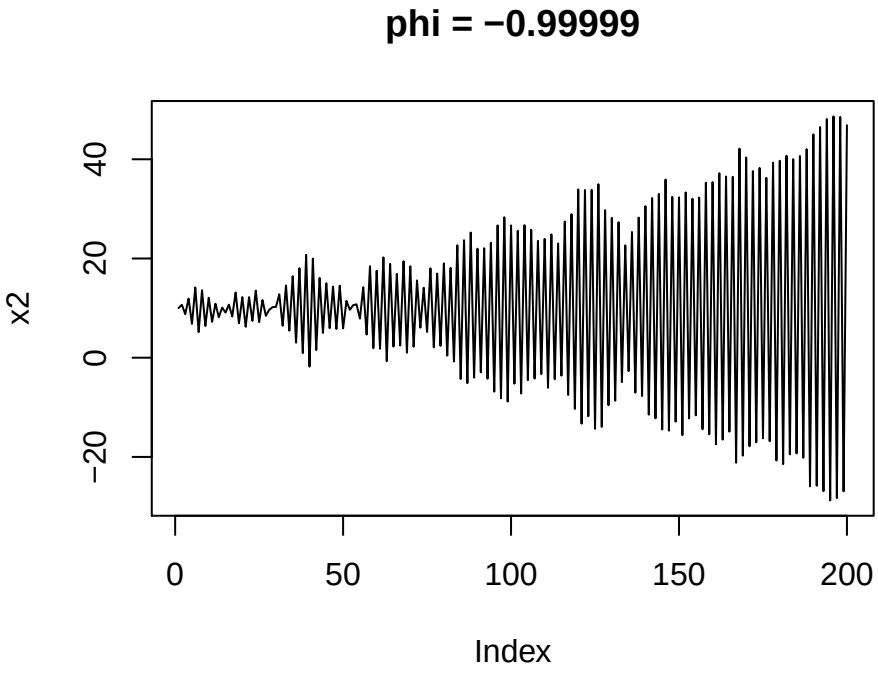


2. Time-series models in Stan

a)

Plotting realizations of the AR(1)-process shows that a high ϕ gives a smoothing effect and a small (into the negatives) gives an oscillating effect. When ϕ is close to zero all contributions of x_{t-1} is supressed and only white noise remains.





b)

i)

Report of the values of the three estimated parameters μ, σ and ϕ can be seen below. For process x the parameters were fairly well approximated (the μ was still quite bad), but for process y the approximated μ was terrible. We were not able to estimate the true value of μ. The estimation was done with 4000 samples.

Table 1: Estimated parameters of AR(1) process X ($\phi = 0.3$)

	μ	σ	ϕ
Mean	7.25	1.38	0.28
2.5%	5.83	1.24	0.14
97.5%	8.62	1.53	0.42
Eff. samples	1108	1450	1098
True value	10	1.41	0.3

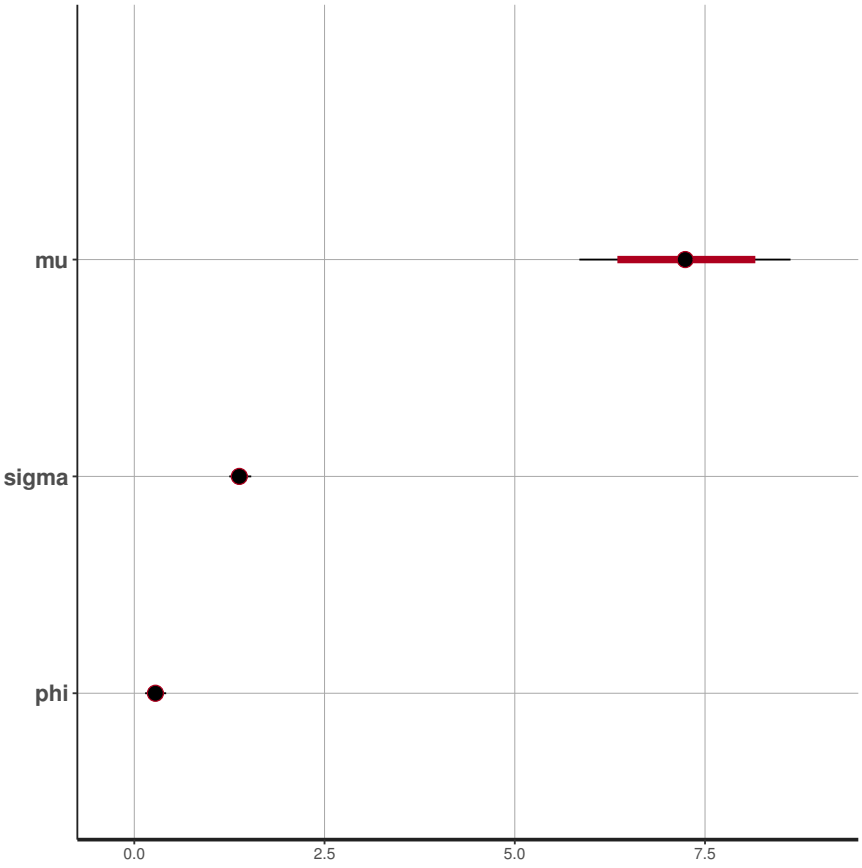
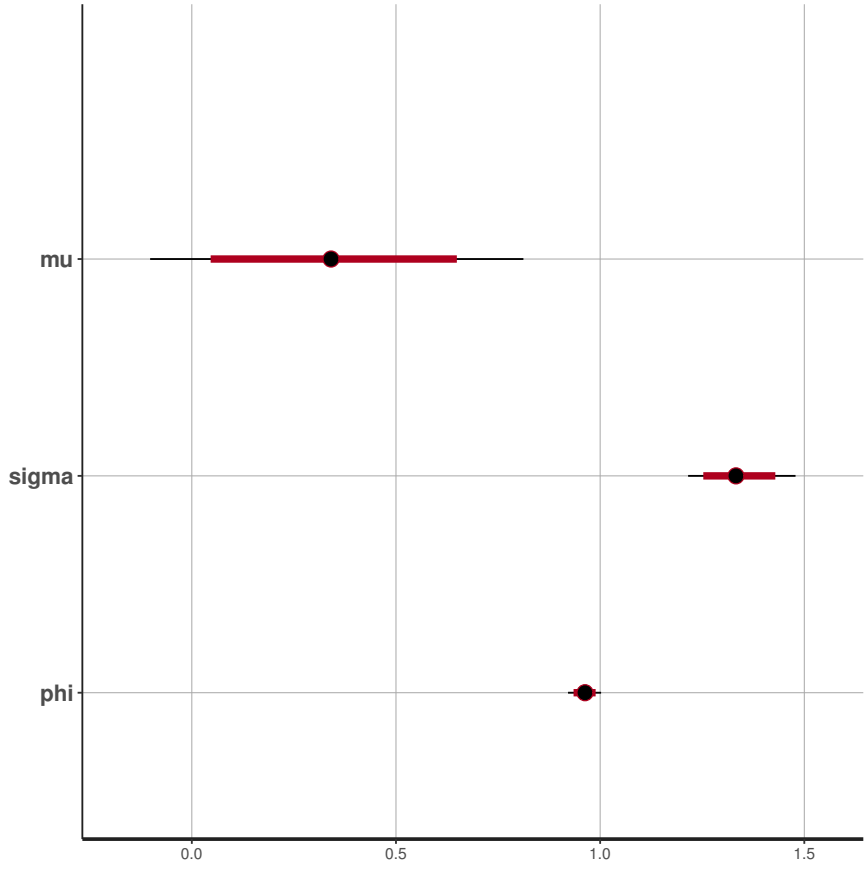


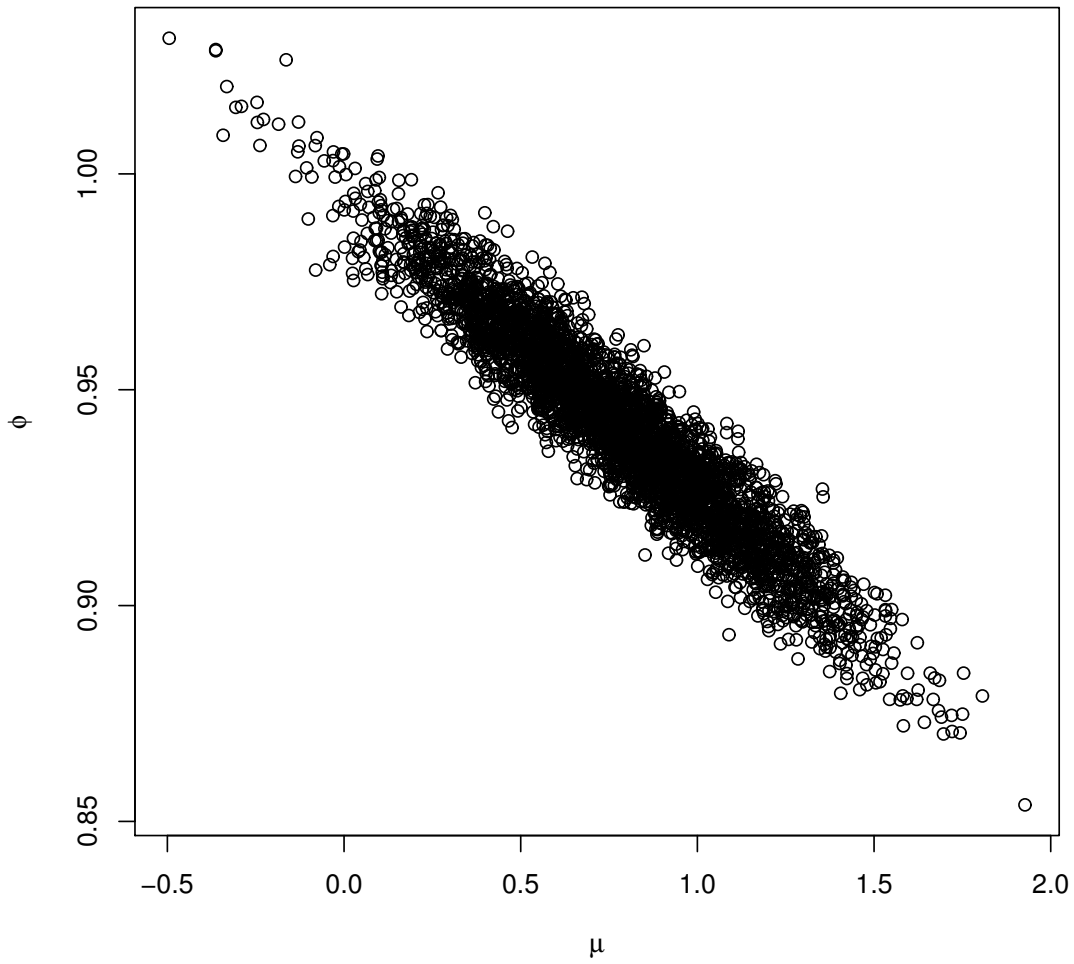
Table 2: Estimated parameters of AR(1) process Y ($\phi = 0.95$)

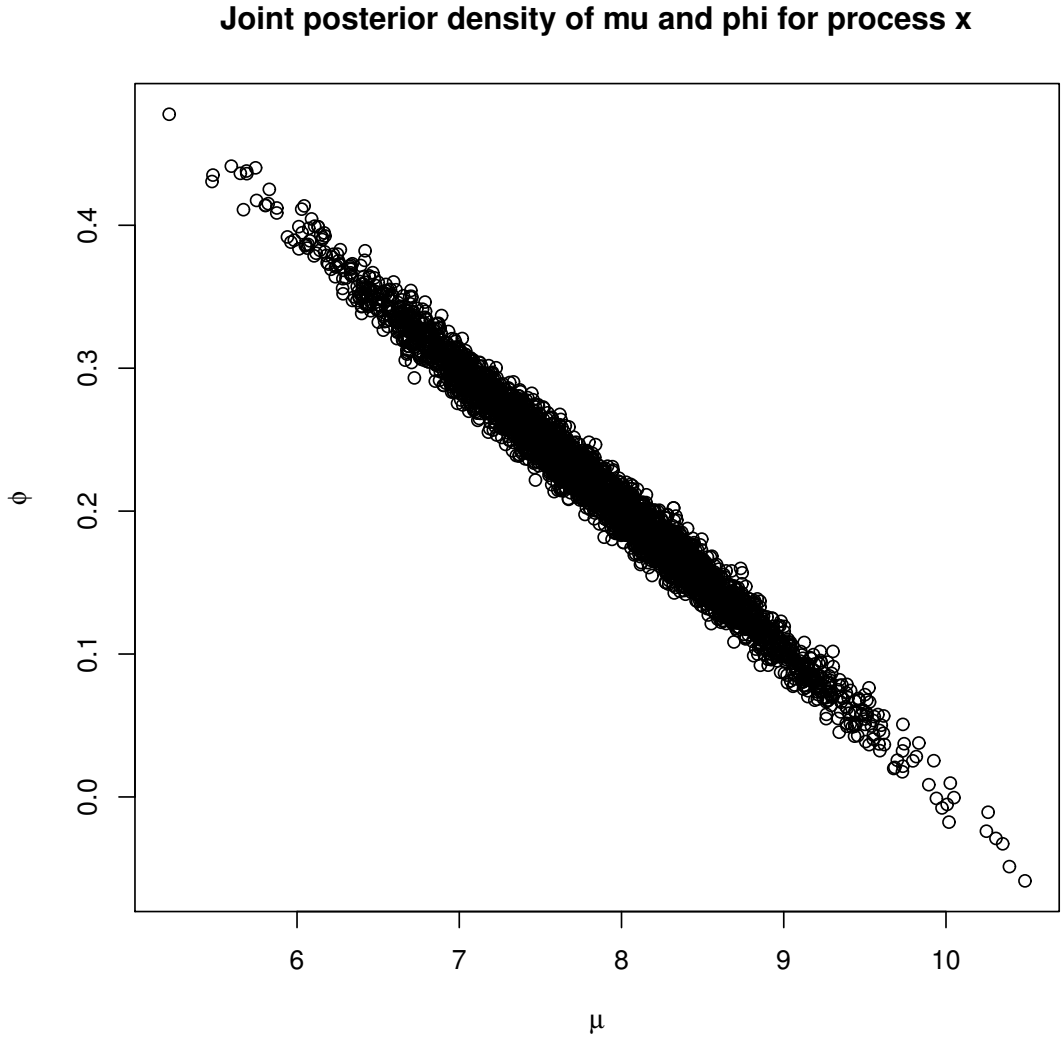
	μ	σ	ϕ
Mean	0.35	1.34	0.96
2.5%	-0.10	1.22	0.92
97.5%	0.81	1.48	1.00
Eff. samples	1726	2166	1729
True value	10	1.41	0.95



ii)

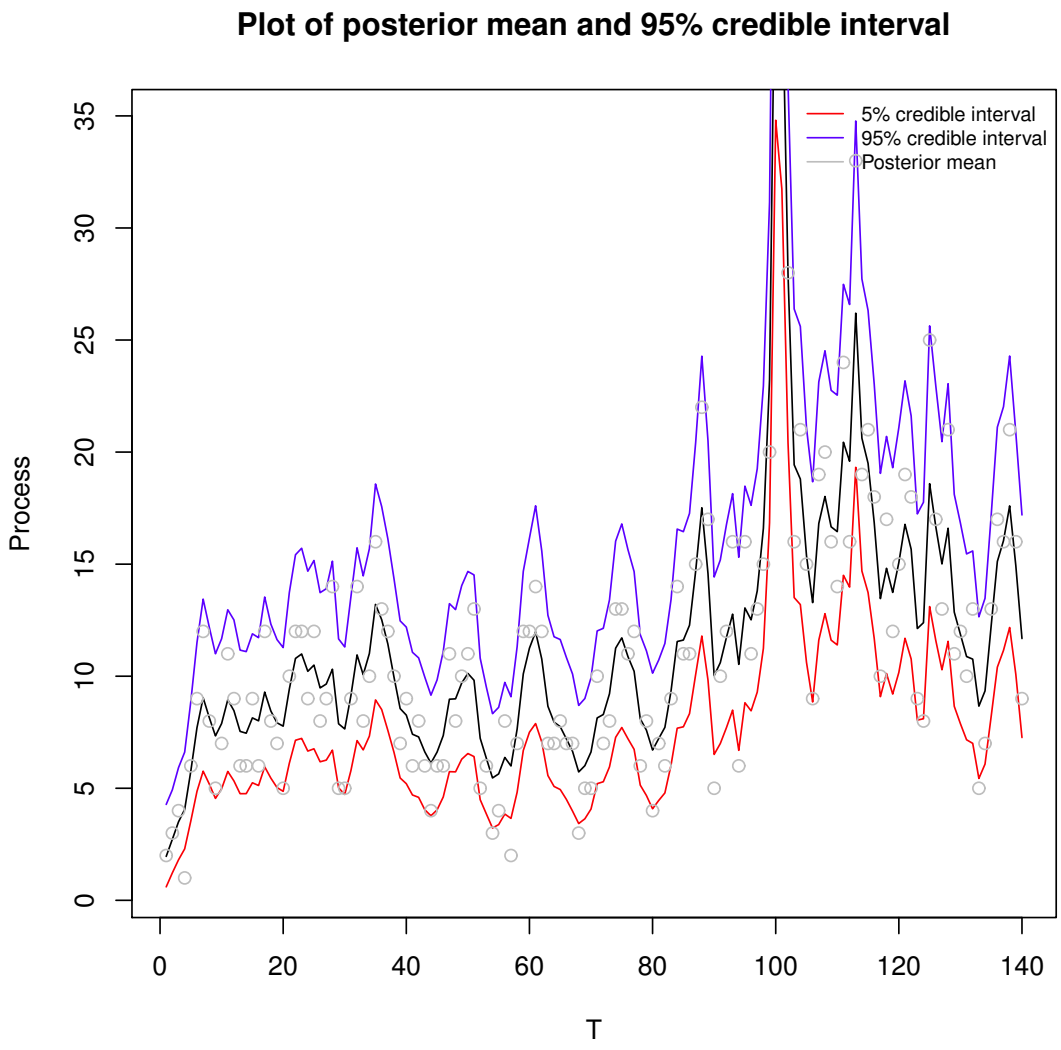
Plotting the joint posterior distributon of μ and ϕ shows that they are highly inversely correlated.


Joint posterior density of mu and phi for process y
ϕ
1.00
0.95
0.90
0.85
−0.5
0.0
0.5
1.0
1.5
2.0
μ



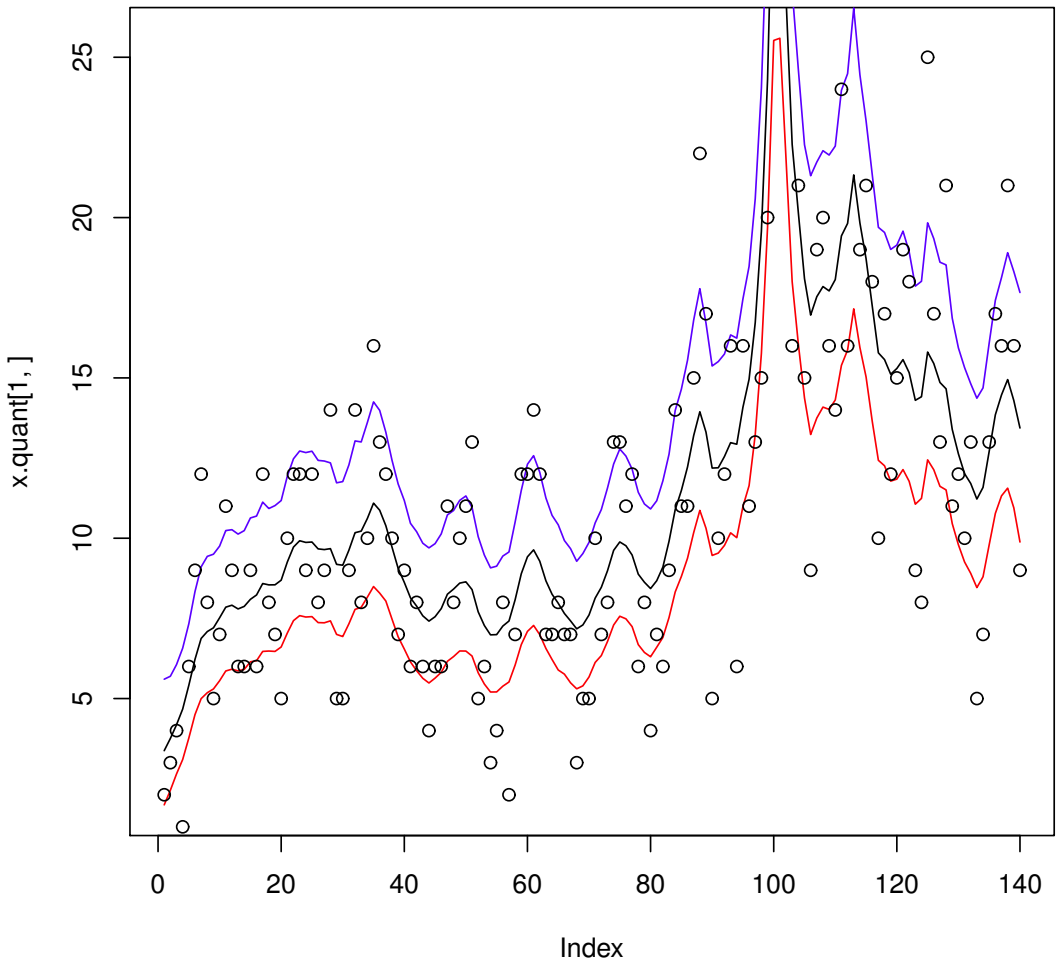
c)

Estimating the model with Stan and plotting the posterior mean with the data resulted in the following plot.



d)

Assuming a normal prior on σ^2 with $\mu = 0$, $\tau^2 = 0.02$ forces the process to vary more slowly. This is reflected in the posterior, which shows that the process is too slow to capture the data.


x.quant[1,]
5
10
15
20
25
0
20
40
60
80
100
120
140
Index

Appendix

```r
# 1 Normal model, mixture of normal model with semi-conjugate prior

# a) Normal model
data <- read.table("rainfall.dat", header=FALSE)
mu0    <- 20 # 0.5 cm
tau0   <- 5  # Variance of expected rainfall
nu0    <- 3  # Variance of expected value of variance
sigma0 <- 5  # Expected value of variance

nIter  <- 5000
theta0 <- c(20, 2)
x      <- data[,1]
n      <- length(x)

rinvchisq <- function(vx, sigmax, draws) {
  vx*sigmax/rchisq(draws, vx)
}

Mu.Conditional.Posterior.Draw <- function(sigma2) {
  taun <- sqrt(1/(n/sigma2 + 1/tau0^2))
  w    <- (n/sigma2) / (n/sigma2 + 1/tau0^2)
  mun  <- w*mean(x) + (1 - w) * mu0
  rnorm(1, mun, taun)
}

Sigma.Conditional.Posterior.Draw <- function(mu) {
  nun    <- n + nu0
  sigman <- (nu0*sigma0^2 + sum((x-mu)^2)) / nun
  rinvchisq(nun, sigman, 1)
}

Gibbs <- function(theta_t) {
  mu     <- Mu.Conditional.Posterior.Draw(theta_t[2])
  sigma2 <- Sigma.Conditional.Posterior.Draw(mu)
  c(mu, sigma2)
}

thetas     <- matrix(rep(0, nIter*2), nrow = nIter)
thetas[1,] <- theta0

for(i in 2:nIter) {
  thetas[i,] <- Gibbs(thetas[i-1,])
}

plot.normal.approximation <- function () {
  plot(thetas[-1,2],
       type = 'l',
       ylab = 'Value')
}

# b) Mixture normal model
```

```r
# Model options
nComp <- 2                  # Number of mixture components

# MCMC options
nIter <- 30 # Number of Gibbs sampling draws

# Prior options
alpha     <- 10*rep(1,nComp)   # Dirichlet(alpha)
muPrior   <- rep(mu0,nComp)    # Prior mean of mu
tau2Prior <- rep(tau0,nComp)   # Prior std of mu
sigma2_0  <- rep(sigma0,nComp) # s20 (best guess of sigma2)
nu0       <- rep(nu0,nComp)    # degrees of freedom for prior on sigma2

#setEPS()
#postscript("mixture-of-normals.eps")
#source("NormalMixtureGibbs.R")
#points(campy.data)
#dev.off()


# c) Graphical comparison

gibbs.mu    <- mean(thetas[,1])
gibbs.sigma <- sqrt(mean(thetas[,2]))
delta <- 0.05
grid  <- seq(-100, 300, delta)

plot.graphical.comparison <- function () {
    plot(density(x), col = 'black',
         main ="Graphical comparison of the data and approximations",
         ylab = "Density",
         xlab  = "Precipitation")
  lines(grid, dnorm(grid, gibbs.mu, gibbs.sigma), type = 'l', col = 'green')
  lines(xGrid, mixDens, type = 'l', col = "orange")
  legend("topright",
       c("Kernel density estimate of data", "Normal approximation", "Mixture of normals (2 normals)"),
       lty=1,
       col=c("black", "green", "orange"),
       bty='n',
       cex=.75)
}

# 2 Time series models in Stan
#library(rstan)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)

# a)

ar.process <- function (phi, init, sigma, T) {
    y     <- rep (0, T)
    y[1]  <- init
    for (t in 2:T) {
```

```
        y[t] <- mu + phi*(y[t-1] - mu) + rnorm(1, 0, sigma)
    }
    y
}

T     <- 200
mu    <- 10
phi1  <- 0.3
phi2  <- 0.95
s <- sqrt(2)
x <- ar.process(phi1, mu, s, T)
y <- ar.process(phi2, mu, s, T)

plot.ar.process <- function () {
    plot(x, type ='l')
}

# b)

x.fit <- stan(file = "time-series.stan",
            data = list(
                x = x,
                N = T
            ))
y.fit <- stan(file = "time-series.stan",
            data = list(
                x = y,
                N = T
            ))

posterior.mean.x <- get_posterior_mean(x.fit)
posterior.mean.y <- get_posterior_mean(y.fit)

head(posterior.mean.y)
head(posterior.mean.x)
mu.x.post    <- posterior.mean.x[1, 5]
sigma.x.post <- posterior.mean.x[2, 5]
phi.x.post   <- posterior.mean.x[3, 5]
x.params <- extract(x.fit, pars = c("mu", "phi"))
y.params <- extract(y.fit, pars = c("mu", "phi"))
plot(x    = x.params$mu,
     y    = x.params$phi,
     main = 'Joint posterior density of mu and phi for process x',
     xlab = expression(mu),
     ylab = expression(phi))

plot(x    = y.params$mu,
     y    = y.params$phi,
     main = 'Joint posterior density of mu and phi for process y',
     xlab = expression(mu),
     ylab = expression(phi))

mu.y.post    <- posterior.mean.y[1, 5]
```

```
sigma.y.post <- posterior.mean.y[2, 5]
phi.y.post   <- posterior.mean.y[3, 5]

z.x <- ar.process(phi.x.post, mu.x.post, sigma.x.post, T)
z.y <- ar.process(phi.y.post, mu.y.post, sigma.y.post, T)

setEPS()
postscript("estimated-parameters-of-from-y.eps")
plot(y.fit)
dev.off()

setEPS()
postscript("estimated-parameters-of-from-x.eps")
plot(x.fit)
dev.off()

plot.sigma.prior <- function () {
    d <- 0.01
    grid <- seq(0, 2, d)
    plot(grid, dnorm(grid, 0, 0.02),
         type = 'l',
         main = '(Unnormalized) Prior density for sigma',
         xlab = expression(sigma),
         ylab = 'Density'
         )
}

# c)

campy.data <- as.vector(read.table("campy.dat", header = TRUE)[,1])
c.fit <- stan(file ="campy.stan",
              data = list (
                  c = campy.data,
                  N = length(campy.data)
              ))

params <- extract(c.fit, pars = c("mu", "sigma"))
x <- extract(c.fit, pars = "x")

mean(params$sigma)
theta.t <- exp(x$x)
x.mean <- apply(theta.t, 2, mean)
x.quant <- apply(theta.t, 2, quantile, probs=c(0.025,0.975))

#setEPS()
#postscript("posterior-with-sigma-0.02-prior.eps")
plot.ar.posterior.mean <- function () {
plot(x.quant[1,], type = 'l', col='red',
     main = "Plot of posterior mean and 95% credible interval",
     xlab = "T",
     ylab = "Process")
lines(x.quant[2,], type = 'l', col='blue')
lines(x.mean, type = 'l')
```

```
points(campy.data, col = 'grey')
legend("topright",
       c("5% credible interval", "95% credible interval", "Posterior mean"),
       lty=1,
       col=c("red", "blue", "grey"),
       bty='n',
       cex=.75)
}

c.df <- as.data.frame(fit)
head(as.matrix(c.fit)[,1])

# d)

# posterior sigma = 0.2603307 with uniform
# posterior sigma = 0.2481805 with 0.15
# posterior sigma = 0.1074629 with 0.02
```

Stan file

```
data {
  int<lower=0> N;
  vector[N] x;
}

parameters {
  real mu;y <- ar.process(phi2, mu, s, T)
  real<lower=0> sigma;
  real phi;
}

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

Stan file for `campy.dat`

```
data {
  int<lower=0> N;
  int c[N];
}

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

model {
  // sigma  ~ normal(0, 0.02);
```

```
  phi    ~ normal(0, 0.6);
  x[2:N] ~ normal(mu + phi * x[1:(N - 1)], sigma);
  for (n in 1:N)
    c[n] ~ poisson(exp(x[n]));
}
```