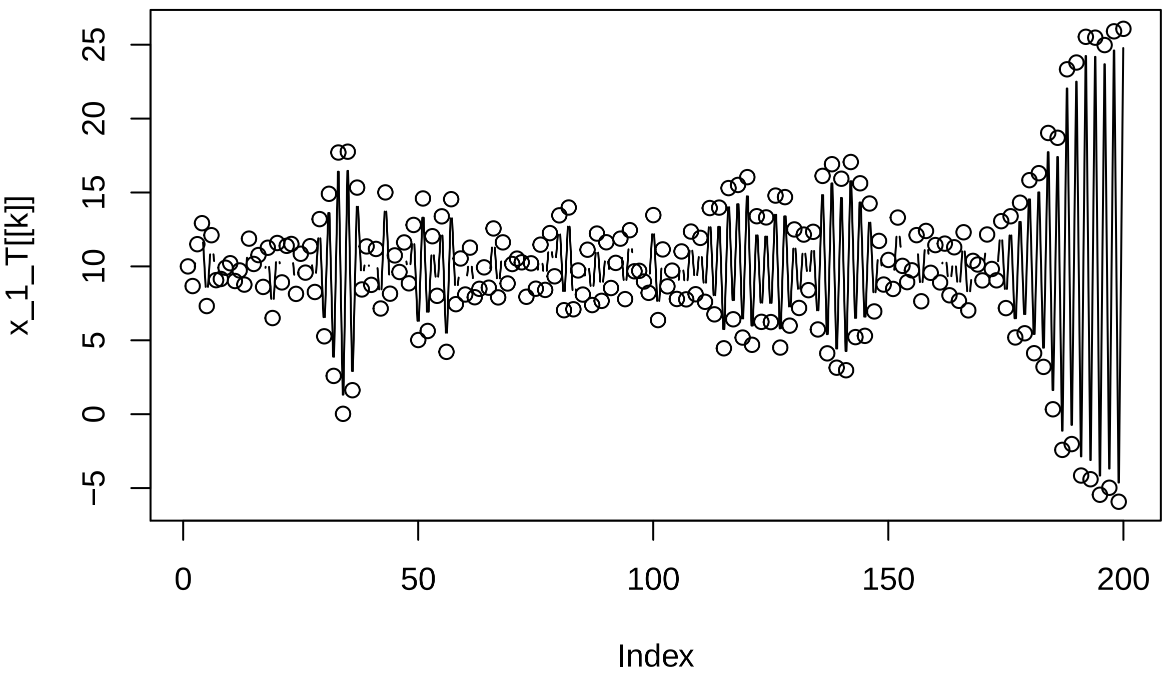
Bayesian Learning Lab 4

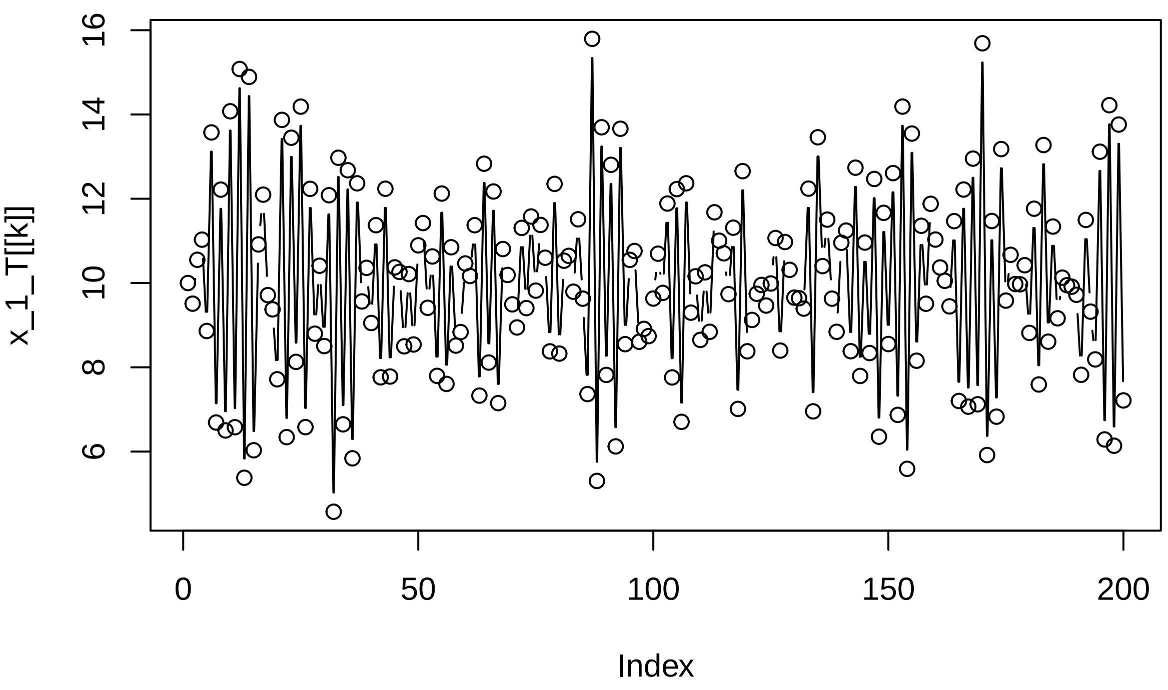
vinbe289, tejma768

1

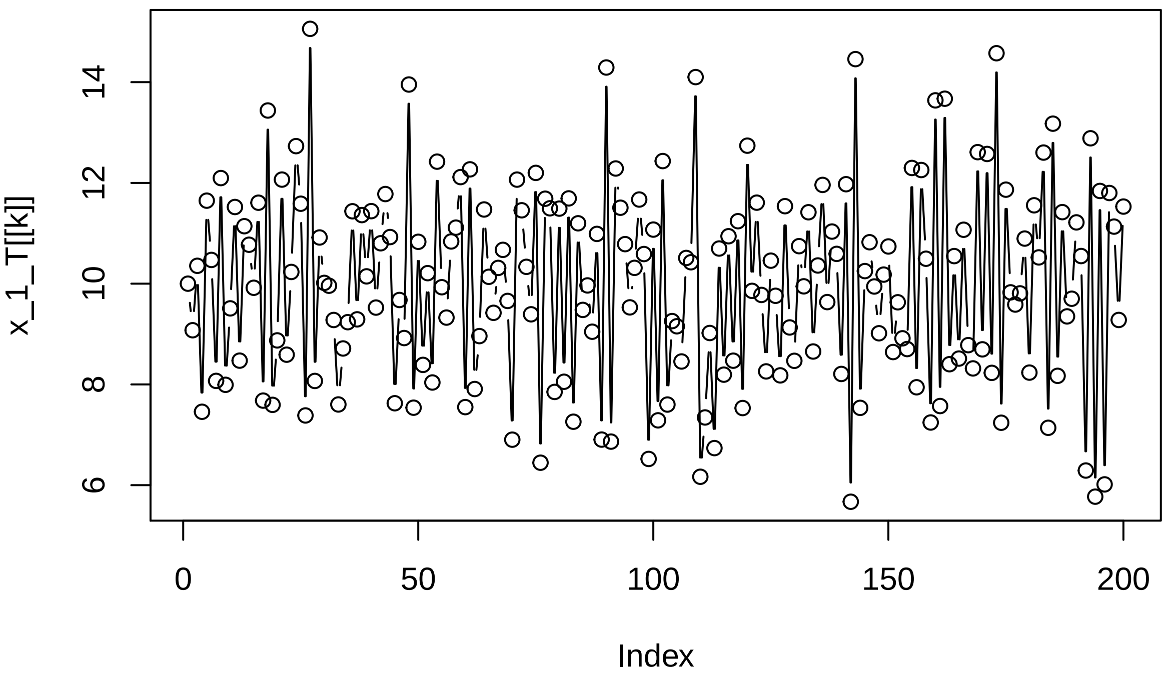
a

```r
AR <- function(phi){
  mu <- 10
  phi <- phi
  sigmasq <- 2
  sigma <- sqrt(sigmasq)
  T <- 200
  x_T <-c(mu,rep(0,length(2:T)))
  df <- rep(NULL)

  for(j in 1:length(phi)){
    for(i in 2:T){
      ep_t <- rnorm(n = 1,mean = 0,sd = sigma)
      x_T[i] <- mu + phi[j]*(x_T[i-1]-mu) + ep_t
    }
    df <- cbind(df,x_T)
  }
  data.frame(df)
  colnames(df) <- phi
  rownames(df) <- 1:T
  return(df)
}
x_1_T <- AR(phi=c(seq(-1,1,0.2)))
x_1_T <- as.data.frame(x_1_T)

phi=c(seq(-1,1,0.2))
for (k in 1:length(phi)){
  plot(x = x_1_T[[k]],type = "b")
}
```

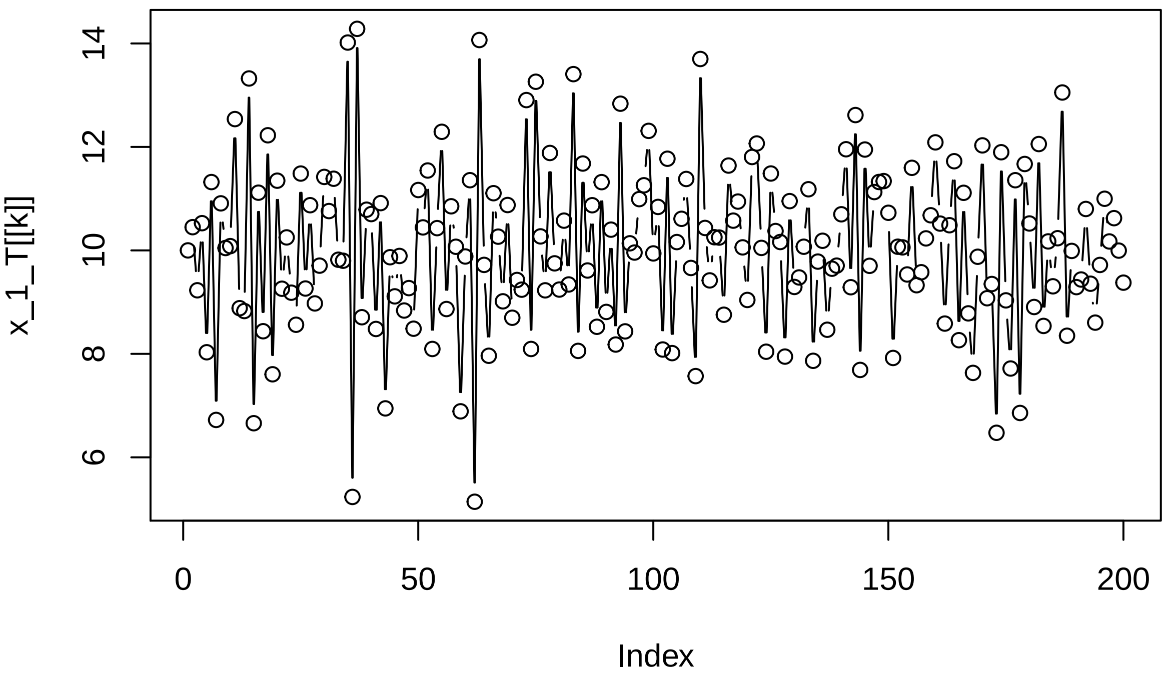

x_1_T[[k]]
Index
25
20
15
10
5
0
−5
0
50
100
150
200



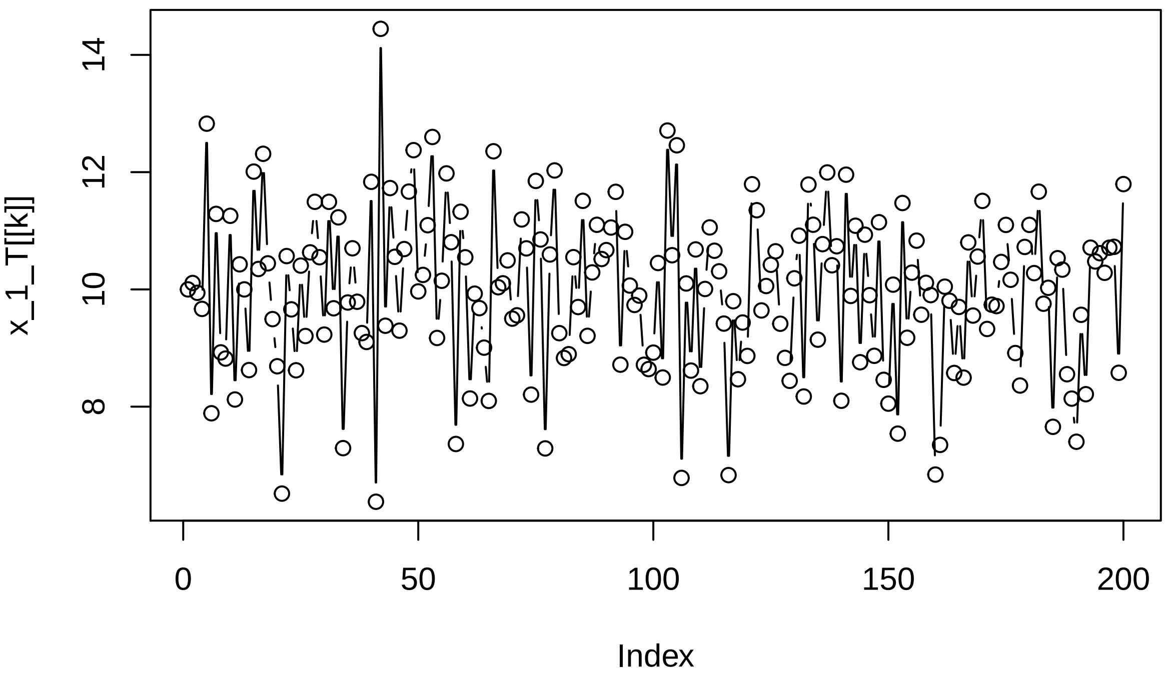
x_1_T[[k]]
16
14
12
10
8
6
0
50
100
150
200
Index



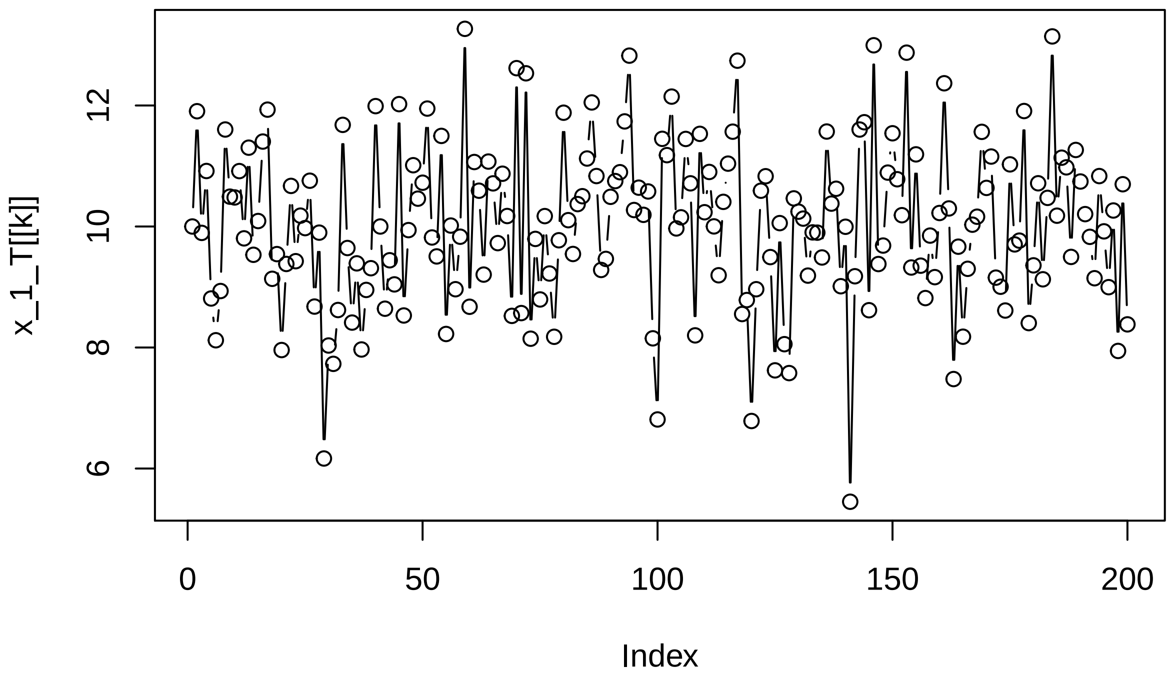
x_1_T[[k]]
6
8
10
12
14
0
50
100
150
200
Index

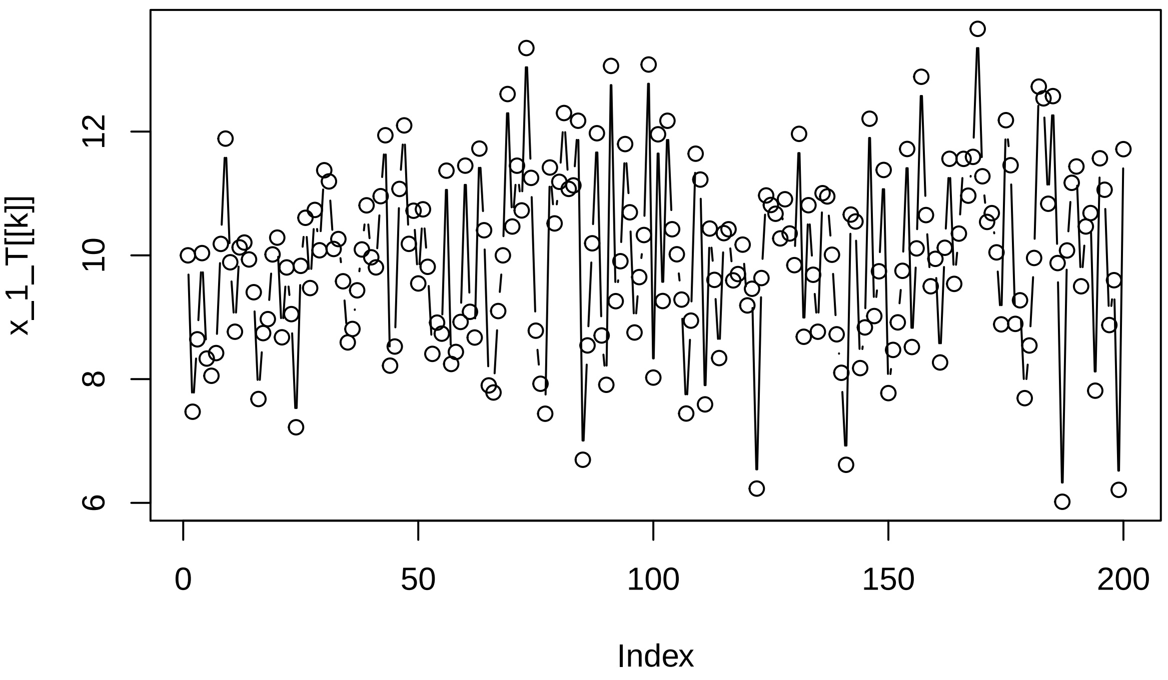


x_1_T[[k]]
14
12
10
8
6
0
50
100
150
200
Index

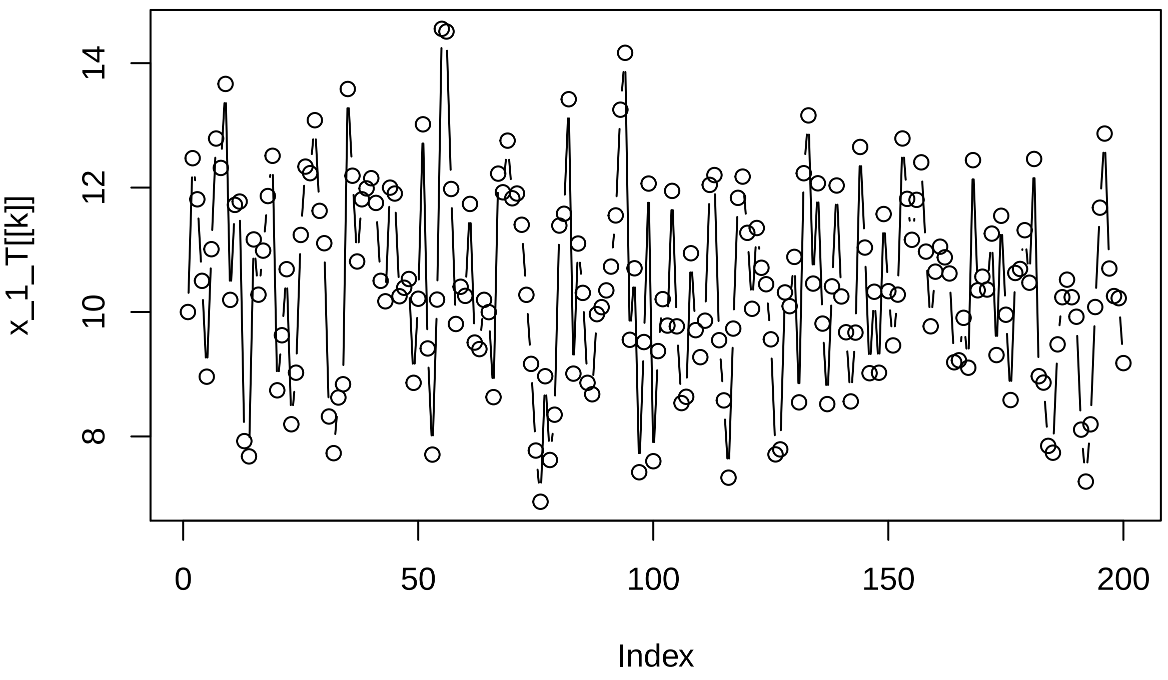


x_1_T[[k]]
14
12
10
8
0
50
100
150
200
Index

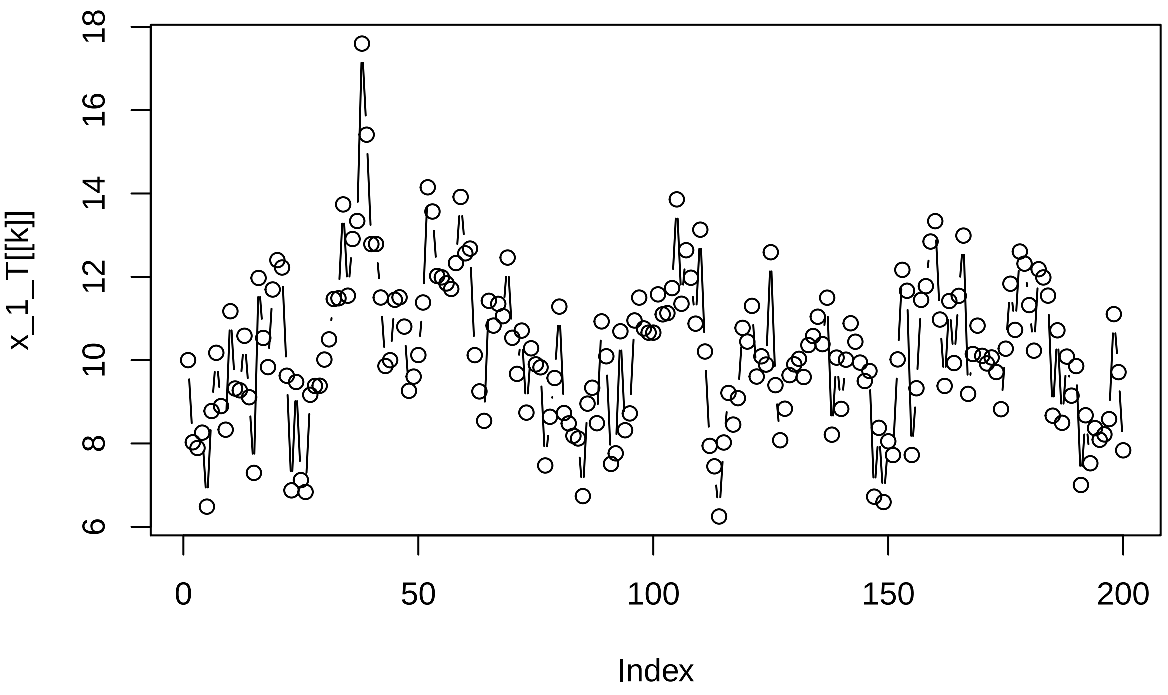

x_1_T[[k]]
Index
0
50
100
150
200
6
8
10
12



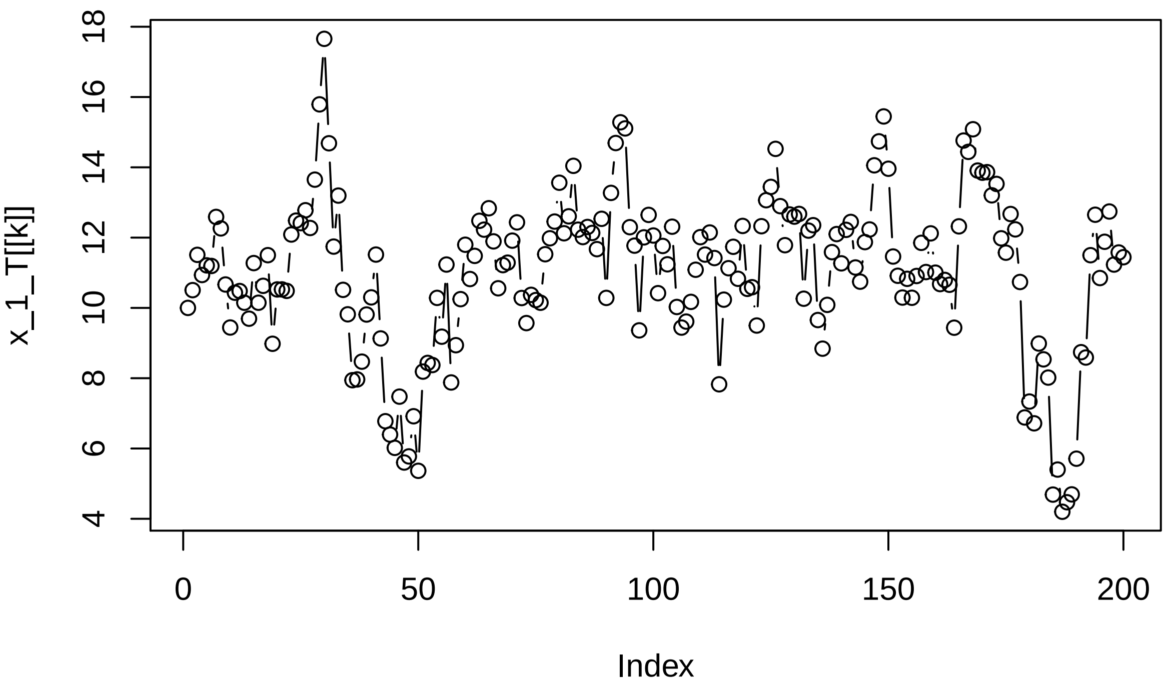
x_1_T[[k]]
12
10
8
6
0
50
100
150
200
Index

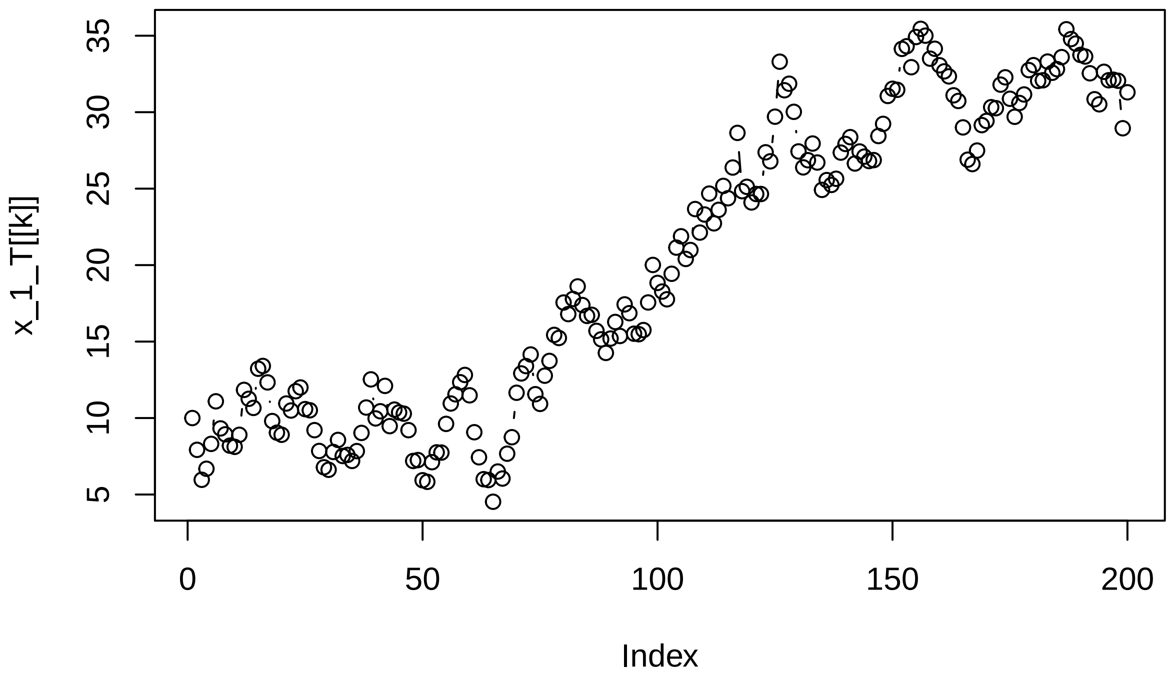


14
12
10
8
x_1_T[[k]]
0
50
100
150
200
Index



x_1_T[[k]]
6
8
10
12
14
16
18
0
50
100
150
200
Index

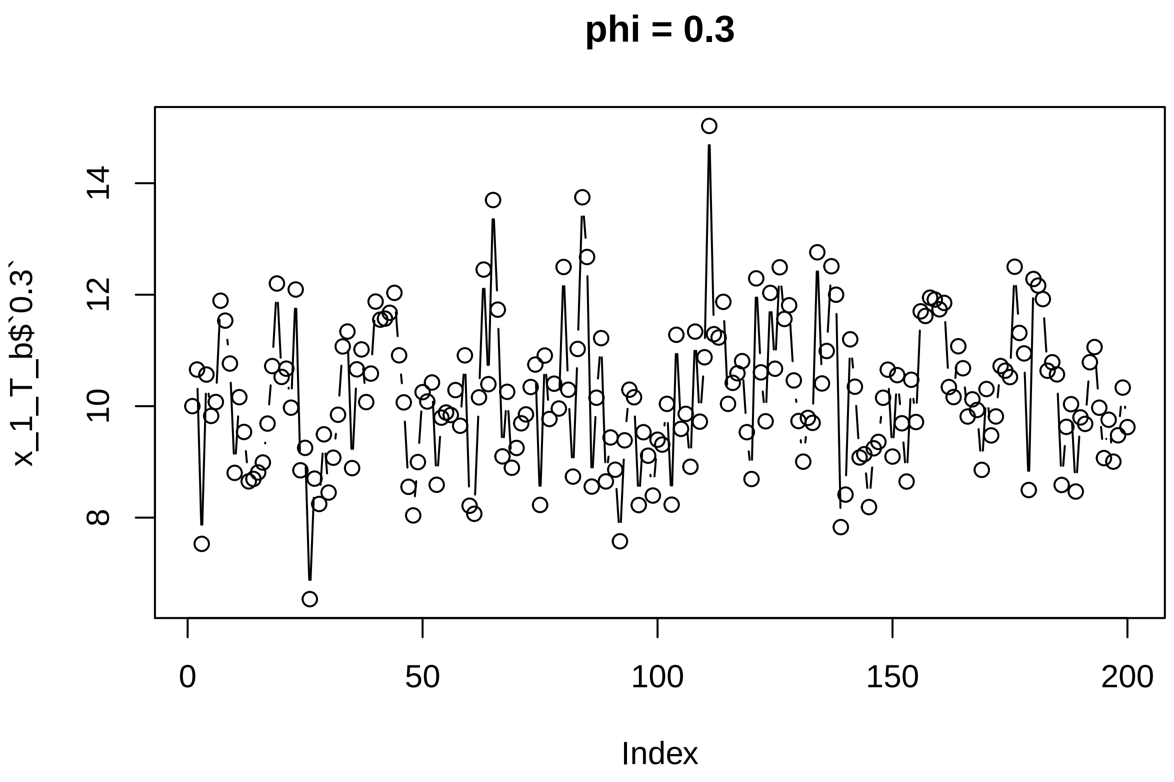

x_1_T[[k]]
Index
4
6
8
10
12
14
16
18
0
50
100
150
200



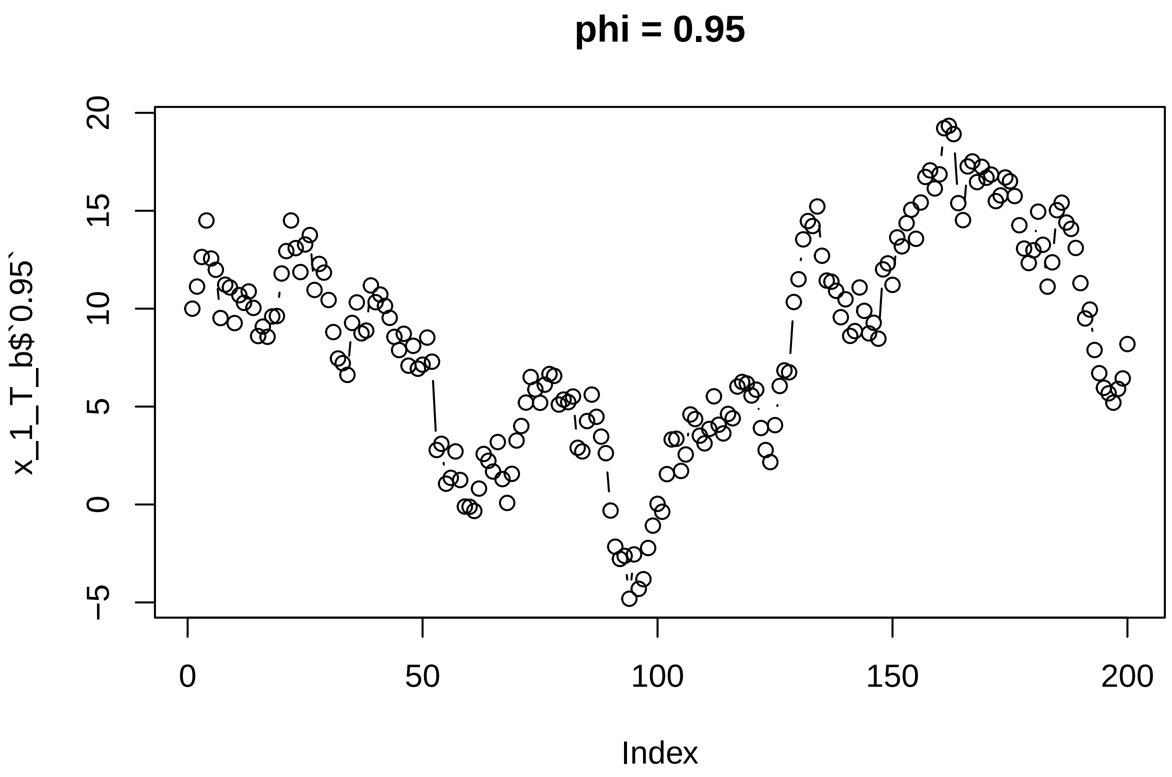
The increase in the ϕ values increase the smoothness of the curve. This is edivent by the plots above. The data points in $x_{1:T}$ are more scattered due to larger ϕ. Larger ϕ showed stronger autocorrelation.

b

```
library(rstan)

x_1_T_b <- AR(phi=c(0.3,0.95))
x_1_T_b <- as.data.frame(x_1_T_b)

plot(x = x_1_T_b$`0.3`,type = "b",main = "phi = 0.3")
```



```r
plot(x = x_1_T_b$`0.95`,type = "b",main = "phi = 0.95")
```



```
ARprocess_0.3 <- x_1_T_b$`0.3`
ARprocess_0.95 <- x_1_T_b$`0.95`

mu <- 10
T = 200
sigmasq <- 2
phi <- 0.5

ARmodel =
  'data  {
        int<lower=0> T;
        vector[T] x;
      }
      parameters {
        real mu;
        real sigmasq;
        real phi;
      }
      model {
        mu ~ normal(0,10);
        phi ~ uniform(-1, 1);
        sigmasq ~ scaled_inv_chi_square(1, 3);

        for(t in 2:T) {
          x[t] ~ normal(mu + phi * (x[t-1] - mu), sqrt(sigmasq));
      }
```

```
    }'

Fit_0.3 = stan(model_code = ARmodel, data = list(x = ARprocess_0.3, T = T)
               ,warmup = 1000, iter = 2000,chains = 4)
Fit_0.95 = stan(model_code = ARmodel, data = list(x = ARprocess_0.95, T = T)
                ,warmup = 1000, iter = 2000,chains = 4)

posteriorDraws_0.3 = extract(Fit_0.3)
posteriorDraws_0.95 = extract(Fit_0.95)

posterior_mean_0.3 <- get_posterior_mean(Fit_0.3)
posterior_mean_0.95 <- get_posterior_mean(Fit_0.95)
```

Posterior Mean and Number of Effective Samples for $\phi = 0.3$

```
fit3 <- extract(Fit_0.3,permuted = FALSE, inc_warmup = TRUE)
monitor(fit3)
```

```
## Inference for the input samples (4 chains: each with iter=2000; warmup=1000):
##
##             mean se_mean  sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## mu          10.2       0 0.1    9.9   10.1   10.2   10.3   10.5  4049    1
## sigmasq      1.5       0 0.2    1.3    1.4    1.5    1.6    1.9  3378    1
## phi          0.3       0 0.1    0.2    0.3    0.3    0.4    0.5  3591    1
## lp__      -143.9       0 1.3 -147.2 -144.4 -143.5 -143.0 -142.4  1833    1
##
## 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).
```

Posterior Mean and Number of Effective Samples for $\phi = 0.95$

```
fit95 <- extract(Fit_0.95,permuted = FALSE, inc_warmup = TRUE)
monitor(fit95)
```

```
## Inference for the input samples (4 chains: each with iter=2000; warmup=1000):
##
##             mean se_mean  sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## mu           5.6     0.3 6.0  -10.5    3.1    6.5    9.1   15.8   321    1
## sigmasq      1.9     0.0 0.2    1.6    1.8    1.9    2.1    2.3  1580    1
## phi          1.0     0.0 0.0    0.9    1.0    1.0    1.0    1.0   949    1
## lp__      -166.9     0.0 1.1 -169.7 -167.4 -166.7 -166.2 -165.6   935    1
##
## 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).
```

```
muCI0.3 <- quantile(posteriorDraws_0.3$mu, probs = c(0.025,.975))
phiCI0.3 <- quantile(posteriorDraws_0.3$phi, probs = c(0.025,.975))
sigmaCI0.3 <- quantile(posteriorDraws_0.3$sigmasq, probs = c(0.025,.975))
```

```
muCI0.95 <- quantile(posteriorDraws_0.95$mu, probs = c(0.025,.975))
phiCI0.95 <- quantile(posteriorDraws_0.95$phi, probs = c(0.025,.975))
sigmaCI0.95 <- quantile(posteriorDraws_0.95$sigmasq, probs = c(0.025,.975))
```

95% C.I. for phi = 0.3

```
data.frame(variable=c("mu","sigmasq","phi"),
              lower=c(muCI0.3[1],sigmaCI0.3[1],
              phiCI0.3[1]),upper=c(muCI0.3[2],
              sigmaCI0.3[2],phiCI0.3[2]))
```

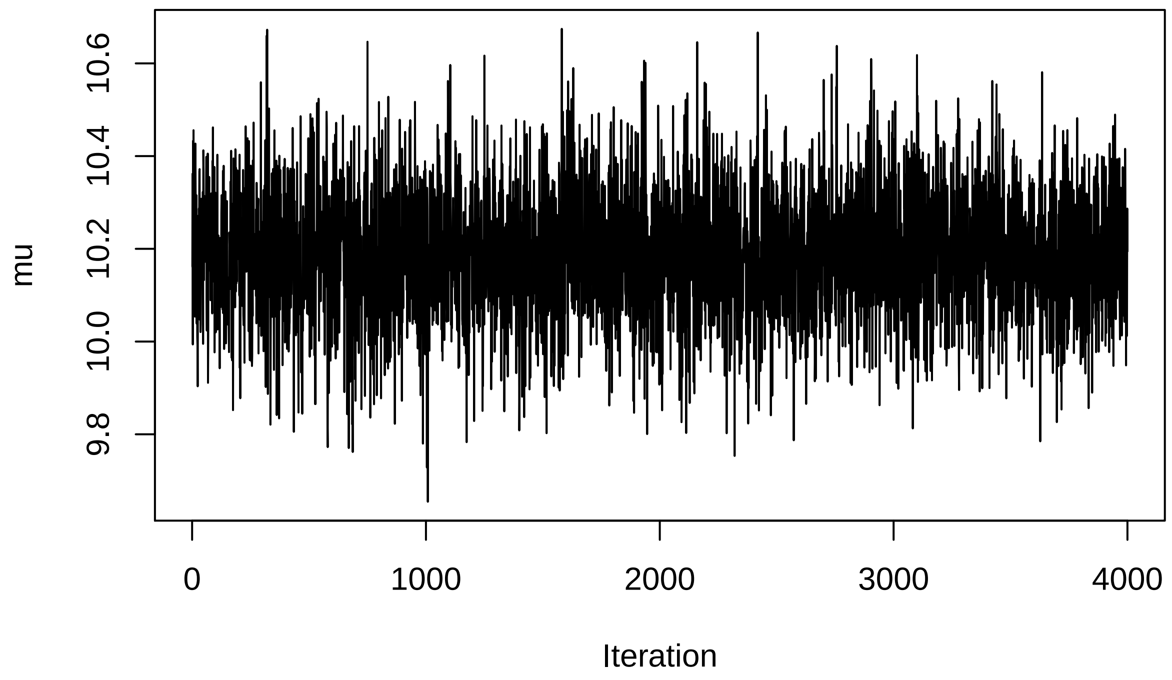
```
##   variable     lower      upper
## 1       mu 9.9143931 10.4625126
## 2  sigmasq 1.2653209  1.8846356
## 3      phi 0.2151836  0.4860134
```

95% C.I. for phi = 0.95

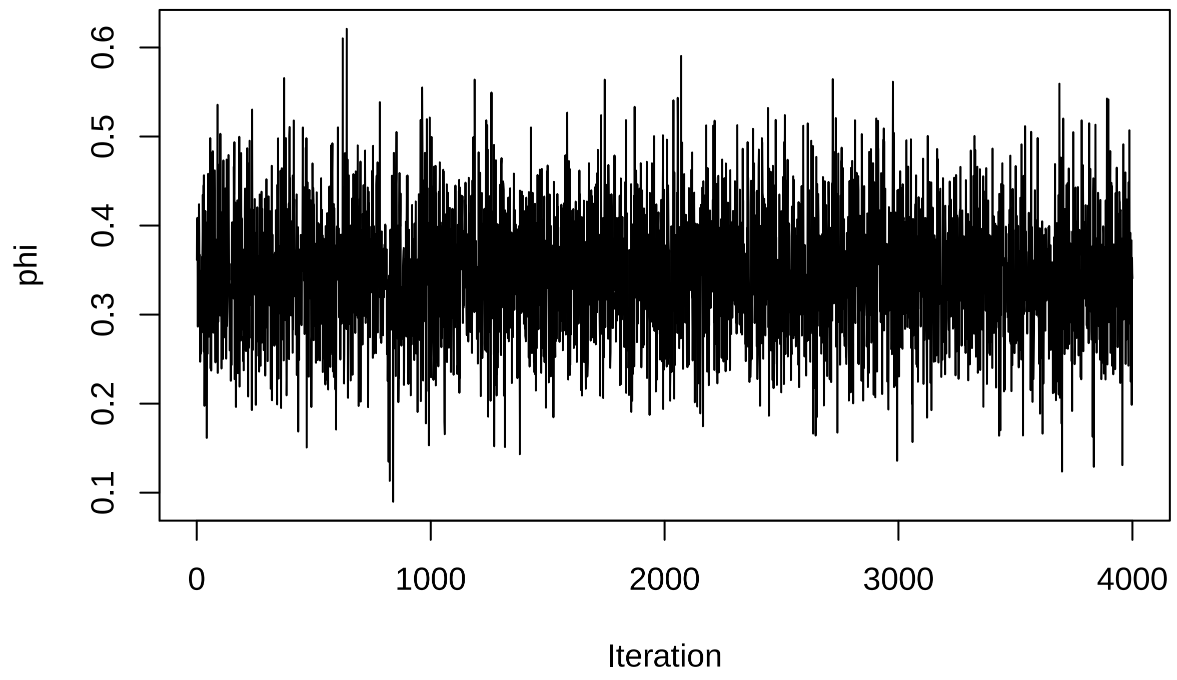
```
data.frame(variable=c("mu","sigmasq","phi"),
            lower=c(muCI0.95[1],sigmaCI0.95[1],
            phiCI0.95[1]),upper=c(muCI0.95[2],
            sigmaCI0.95[2],phiCI0.95[2]))
```

```
##   variable       lower      upper
## 1       mu -10.4967836 15.8300985
## 2  sigmasq   1.6078806  2.3492135
## 3      phi   0.9417923  0.9987418
```

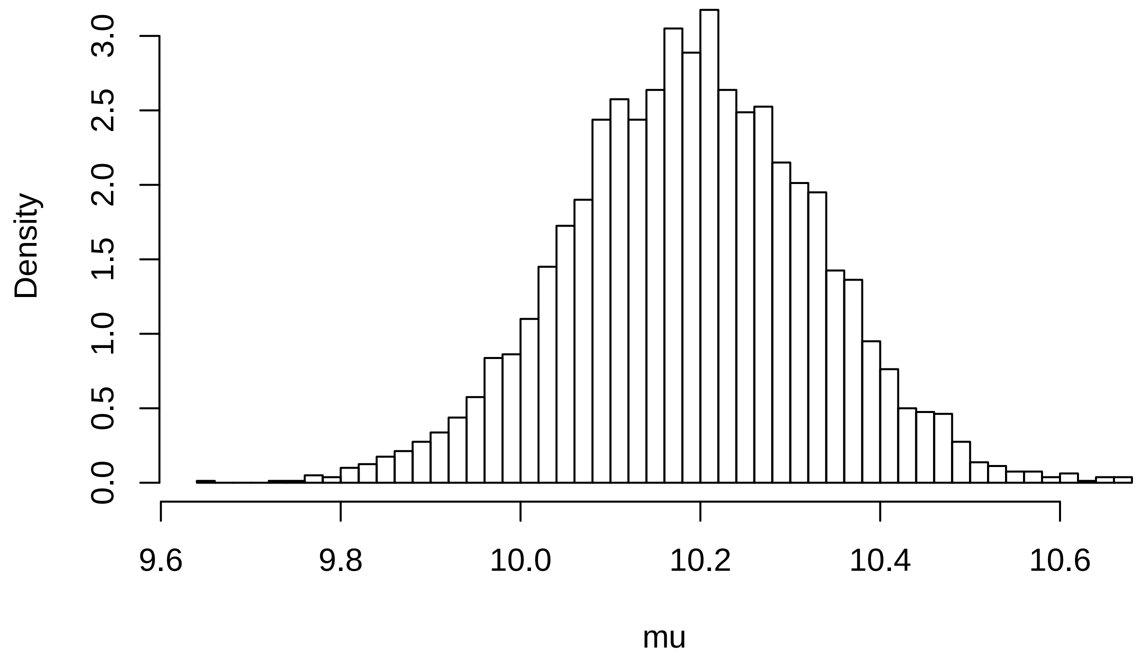
```
plot(posteriorDraws_0.3$mu, type='l', xlab="Iteration", ylab="mu")
```



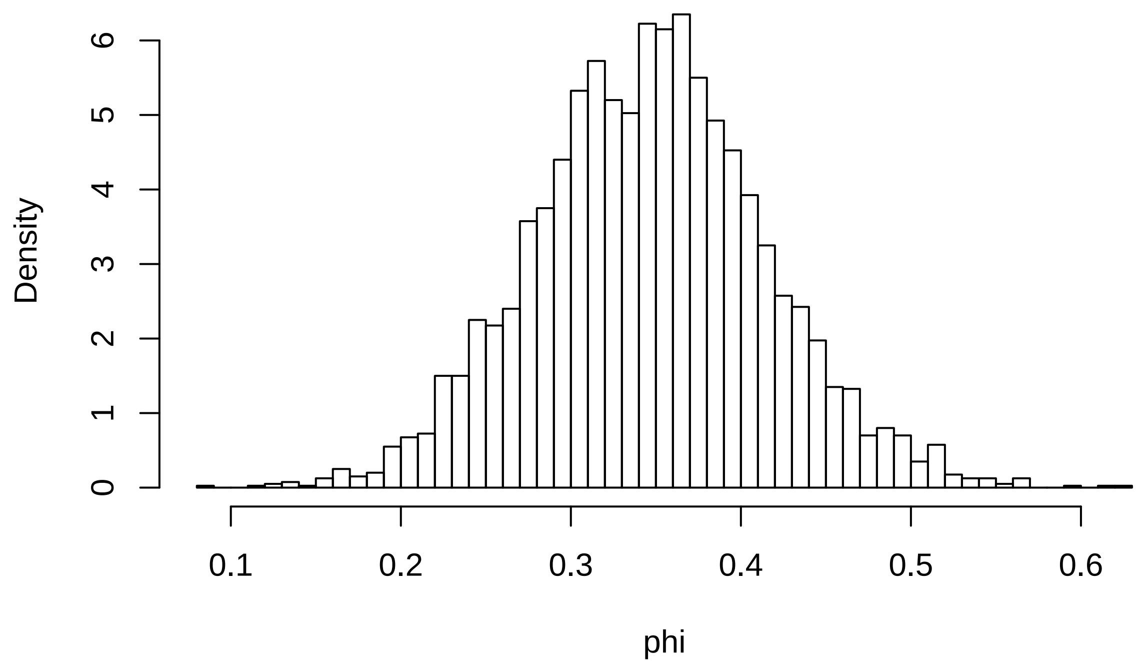
```r
plot(posteriorDraws_0.3$phi, type='l', xlab="Iteration", ylab="phi")
```



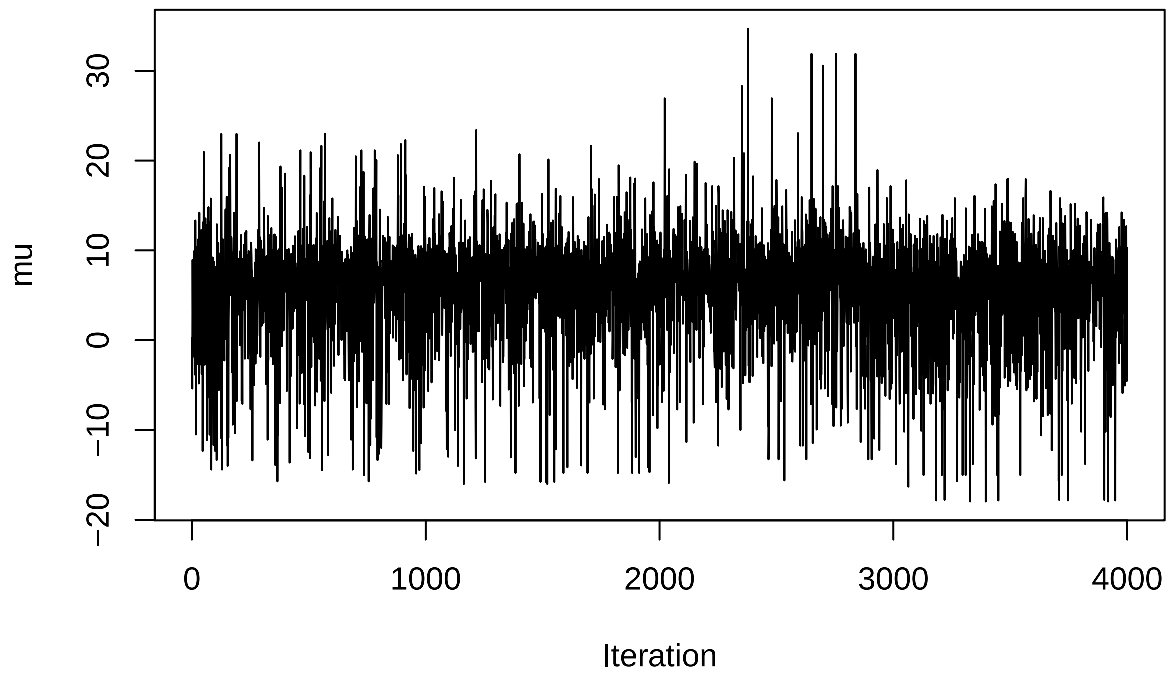
```
hist(posteriorDraws_0.3$mu,freq = FALSE, breaks=50, main="", xlab="mu")
```



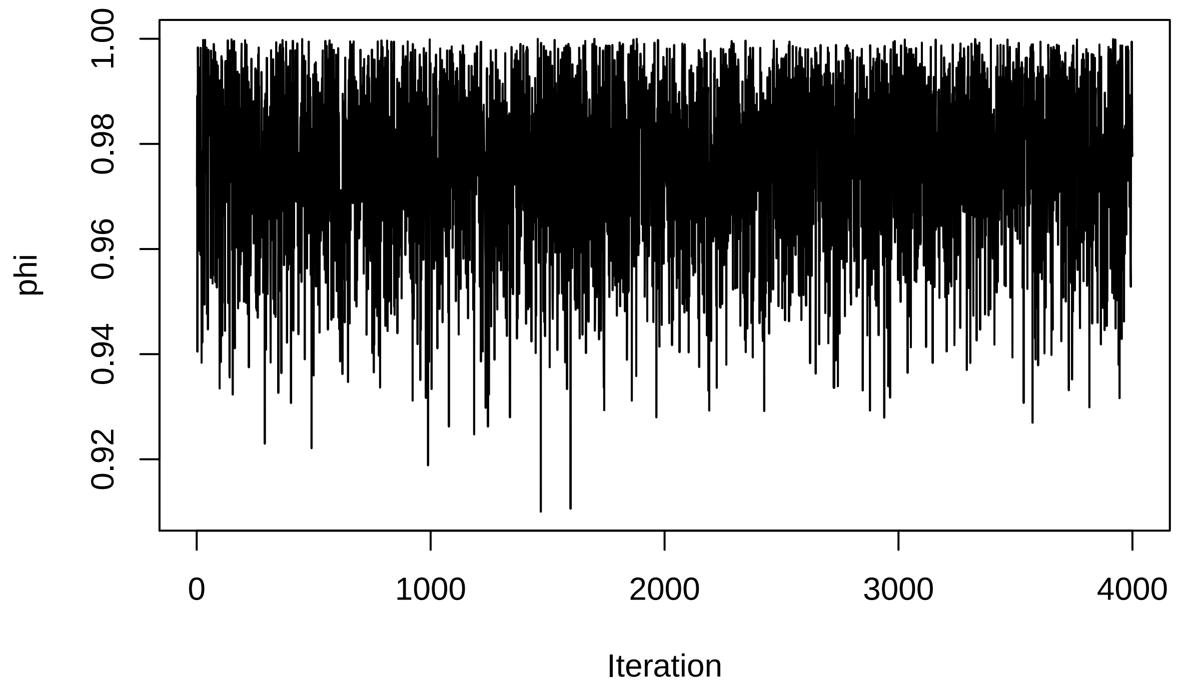
```r
hist(posteriorDraws_0.3$phi,freq = FALSE, breaks=50, main="", xlab="phi")
```



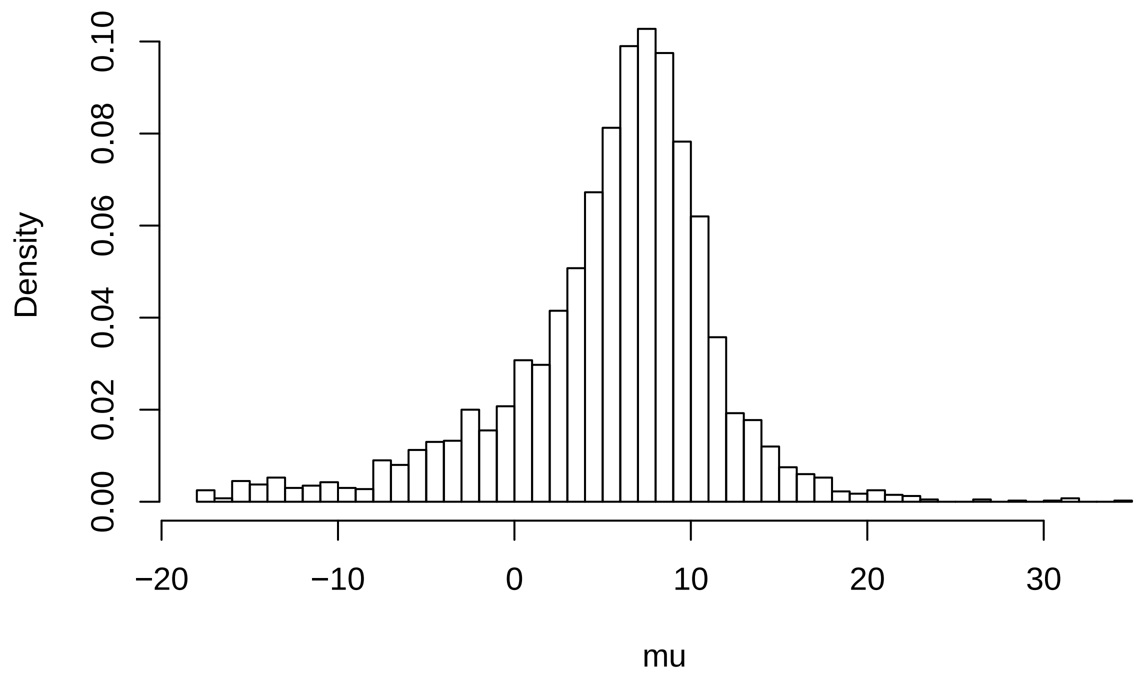
```r
plot(posteriorDraws_0.95$mu, type='l', xlab="Iteration", ylab="mu")
```



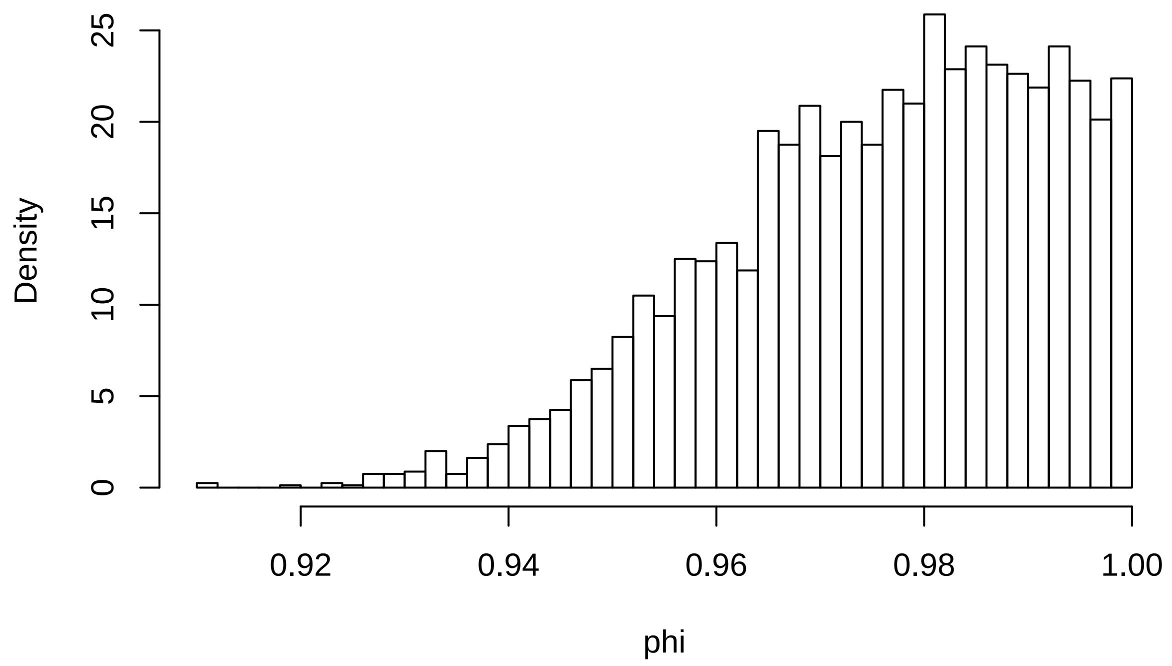
```r
plot(posteriorDraws_0.95$phi, type='l', xlab="Iteration", ylab="phi")
```



```r
hist(posteriorDraws_0.95$mu,freq = FALSE, breaks=50, main="", xlab="mu")
```

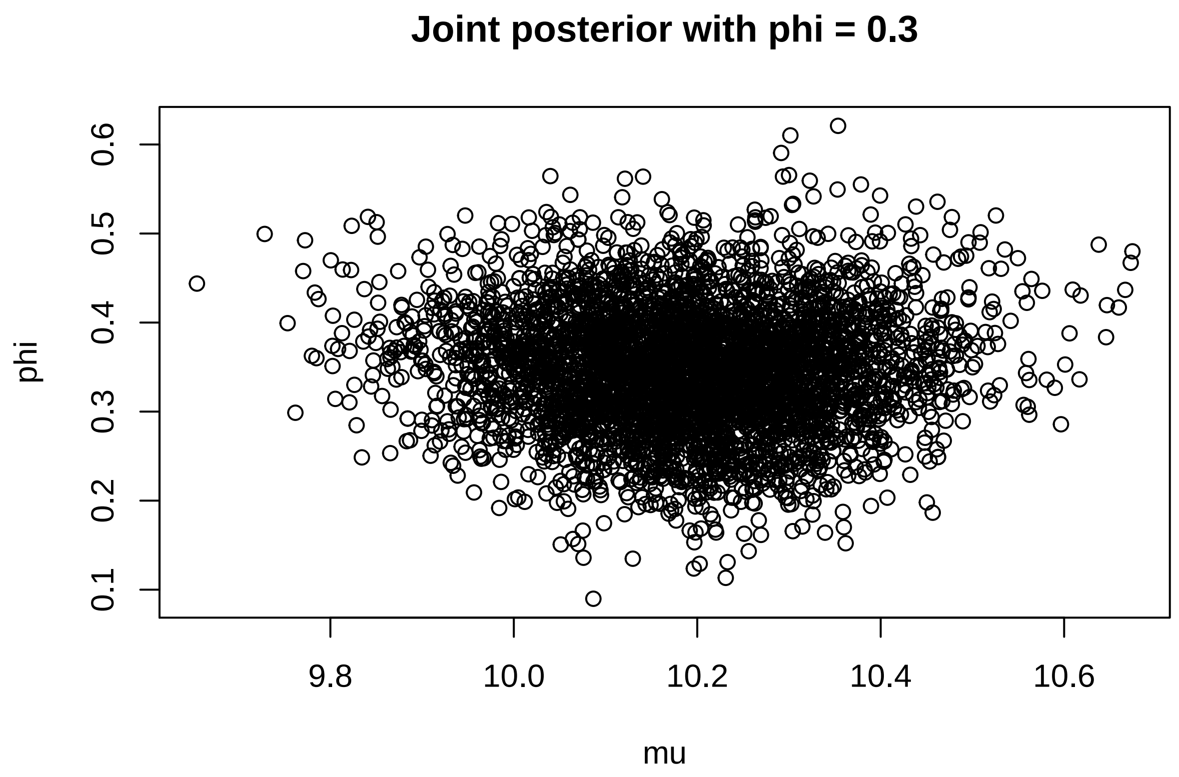


```
hist(posteriorDraws_0.95$phi,freq = FALSE, breaks=50, main="", xlab="phi")
```

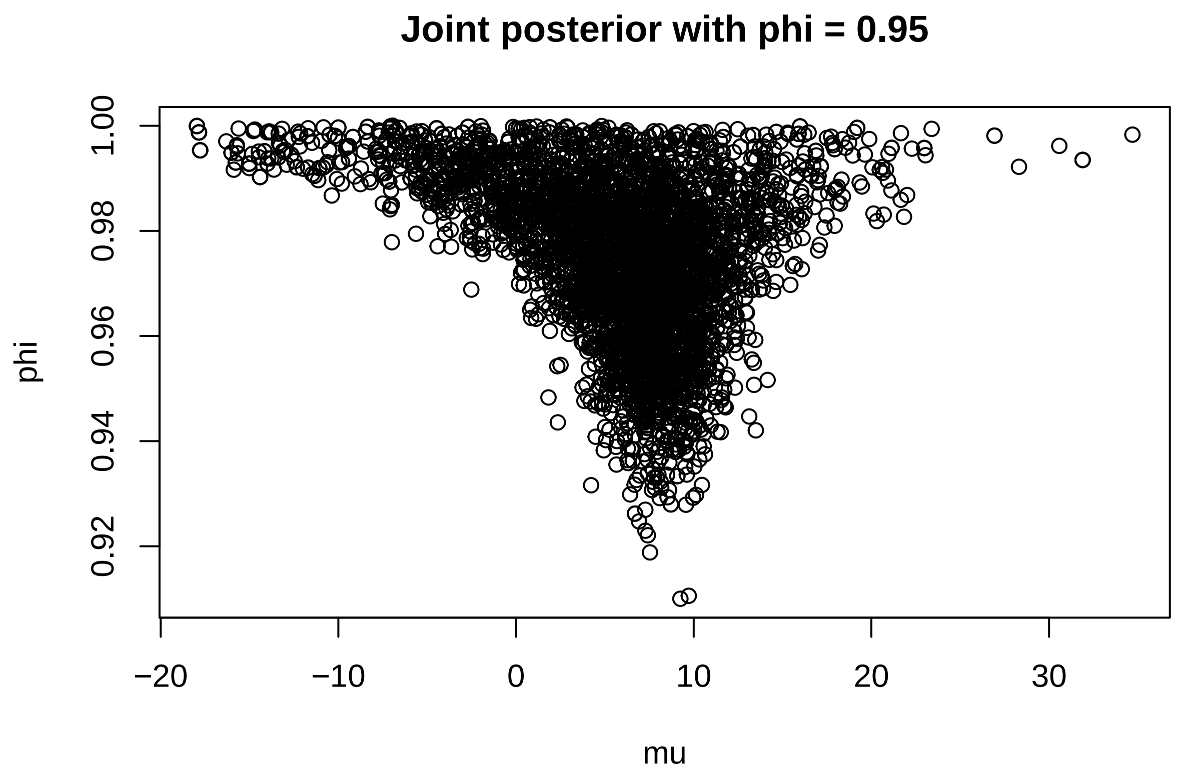


It can be seen that the mu is not closer to the true values when phi value is 0.95.

```
plot(x = posteriorDraws_0.3$mu, y = posteriorDraws_0.3$phi,  xlab = "mu",
ylab = "phi",main = "Joint posterior with phi = 0.3")
```

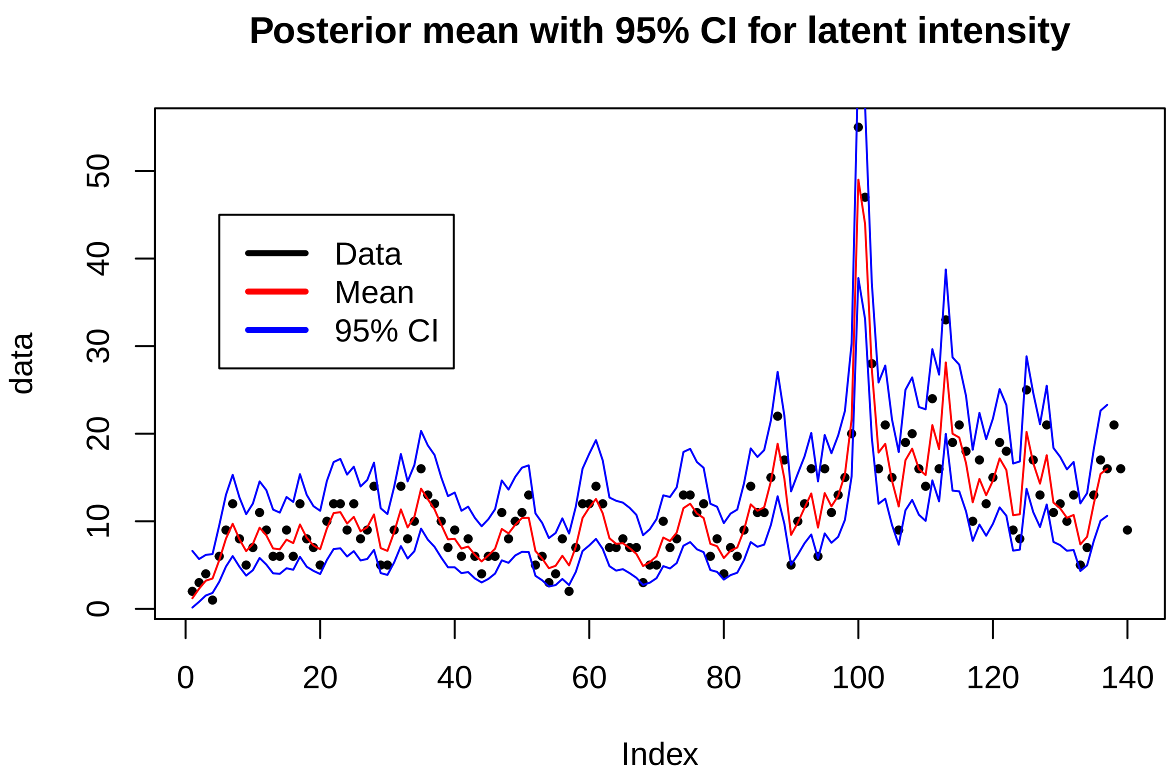


```
plot(x = posteriorDraws_0.95$mu, y = posteriorDraws_0.95$phi, xlab = "mu",
ylab = "phi",main = "Joint posterior with phi = 0.95")
```



c

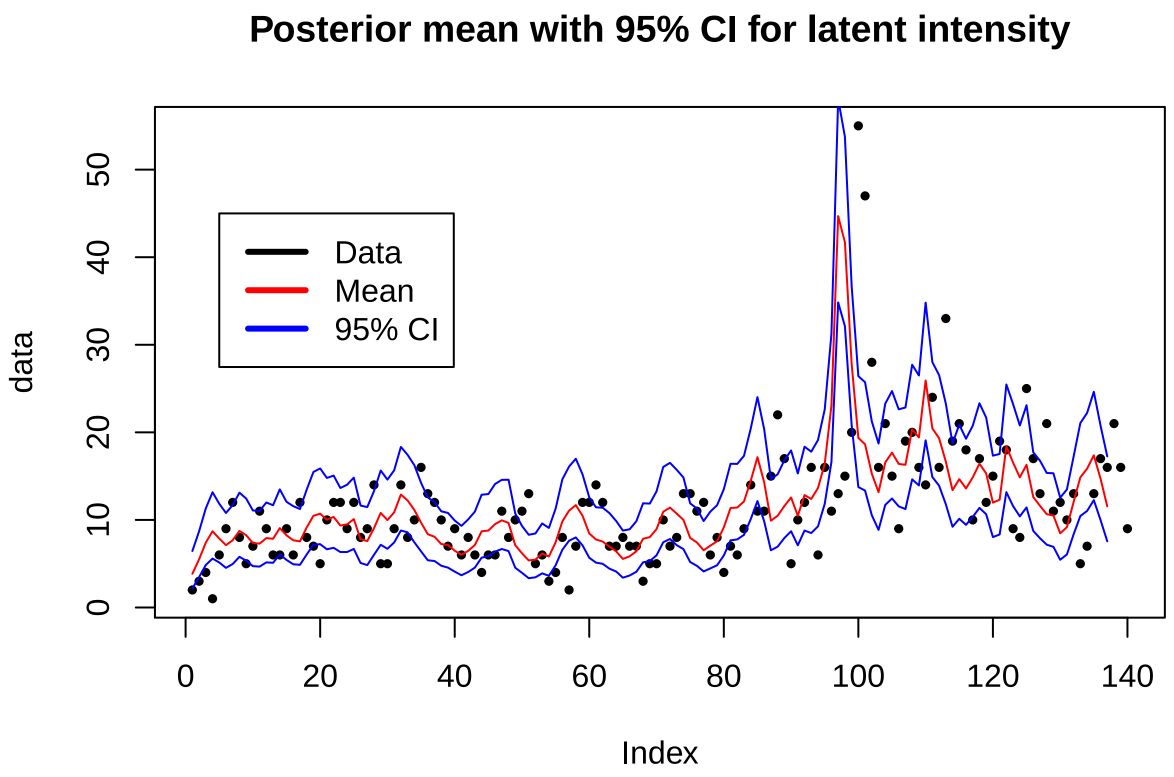
```
data = read.table("campy.dat", header=TRUE)[,1]
n = length(data)

PoissonModel =
  'data {
        int<lower=0> N;
        int c[N];
      }
      parameters {
        real mu;
        real sigmasq;
        real phi;
        vector[N] x;
      }
      model {
          mu ~ normal(0,100);
          phi ~ uniform(-1, 1);
          sigmasq ~ scaled_inv_chi_square(1, 2);

          for(t in 2:N) {
            x[t] ~ normal(mu + phi * (x[t-1] - mu), sqrt(sigmasq));
            c[t] ~ poisson(exp(x[t]));
          }
      }'

ARpoisson = stan(model_code=PoissonModel, data=list(N=n, c=data))

PoissonCI = summary(ARpoisson)$summary

x.upper = PoissonCI[4:n, "97.5%"]
x.lower = PoissonCI[4:n, "2.5%"]
x.mean = PoissonCI[4:n, "mean"]

plot(data ,main="Posterior mean with 95% CI for latent intensity",
     type = "p", pch= 19, cex = 0.5, col = 1)
lines(exp(x.upper), col="blue")
lines(exp(x.lower), col="blue")
lines(exp(x.mean), col="red")
legend(x = 5, y=45, c("Data", "Mean", "95% CI"), col=c("black", "red", "blue"), lwd = 3)
```



The posterior mean seems to be correct in following the general trend of the data. Most of the data points seems to fall in the 95% CI.

d

```r
PoissonModelAR =
  'data {
      int<lower=0> N;
      int<lower=0> c[N];
    }
    parameters {
      real x[N];
      real mu;
      real<lower=-1, upper=1> phi;
      real<lower=0> sigmasq;
    }
    model {
      mu ~ normal(0,100);
      phi ~ uniform(-1, 1);
      sigmasq ~ gamma(1, 3);   // this is the change we did in this part
      for(t in 2:N) {
        x[t] ~ normal(mu + phi * (x[t-1] - mu), sqrt(sigmasq));
        c[t] ~ poisson(exp(x[t]));
      }
    }'

ARpoisson = stan(model_code=PoissonModelAR, data=list(N=n, c=data))

PoissonCI = summary(ARpoisson)$summary

x.upper = PoissonCI[4:n, "97.5%"]
x.lower = PoissonCI[4:n, "2.5%"]
x.mean = PoissonCI[4:n, "mean"]

plot(data ,main="Posterior mean with 95% CI for latent intensity",
     type = "p", pch= 19, cex = 0.5, col = 1)
lines(exp(x.upper), col="blue")
lines(exp(x.lower), col="blue")
lines(exp(x.mean), col="red")
legend(x = 5, y=45, c("Data", "Mean", "95% CI"), col=c("black", "red", "blue"), lwd = 3)
```



The plot does not seem to have changed much. More data points fall outside the 95% CI compared to the previous plot though.