"WHAT LIES BEHIND YOU AND WHAT LIES IN FRONT OF YOU, PALES IN COMPARISON TO WHAT LIES INSIDE OF YOU."
— **RALPH WALDO EMERSON**

BAYESIAN STATISTICS

CLASS 2

WHAT DID WE LEARN FROM CLASS 1?

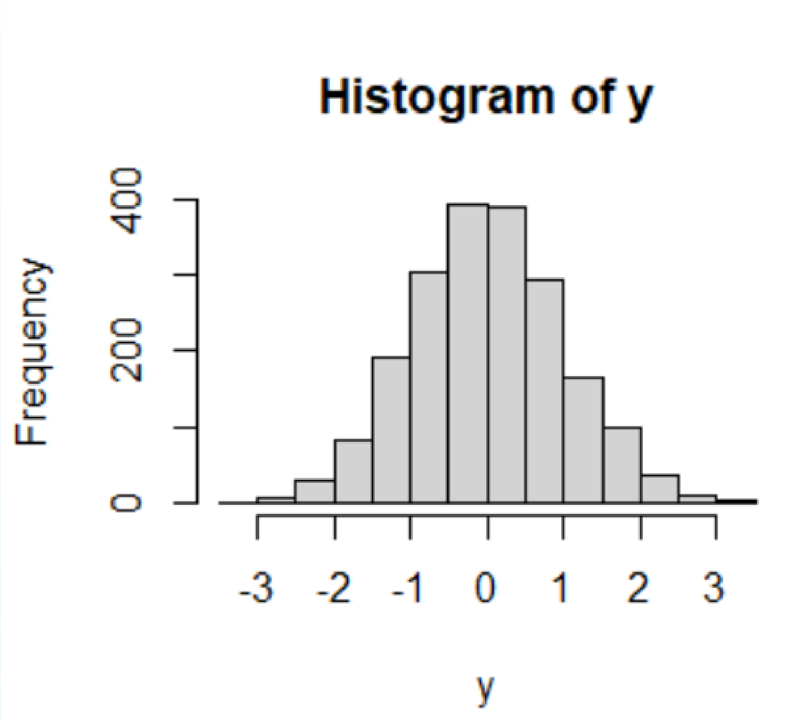
- Terminology: Prior, Sampling distribution, Posterior
- How to define problem (decide sampling distribution of data, define priors for parameters, use Stan to generate posterior distribution of parameters)
- How to use posterior to answer questions about the parameter
- How data (sample size) and prior contribute to the posterior
- Why prior is VERY important when sample size is small

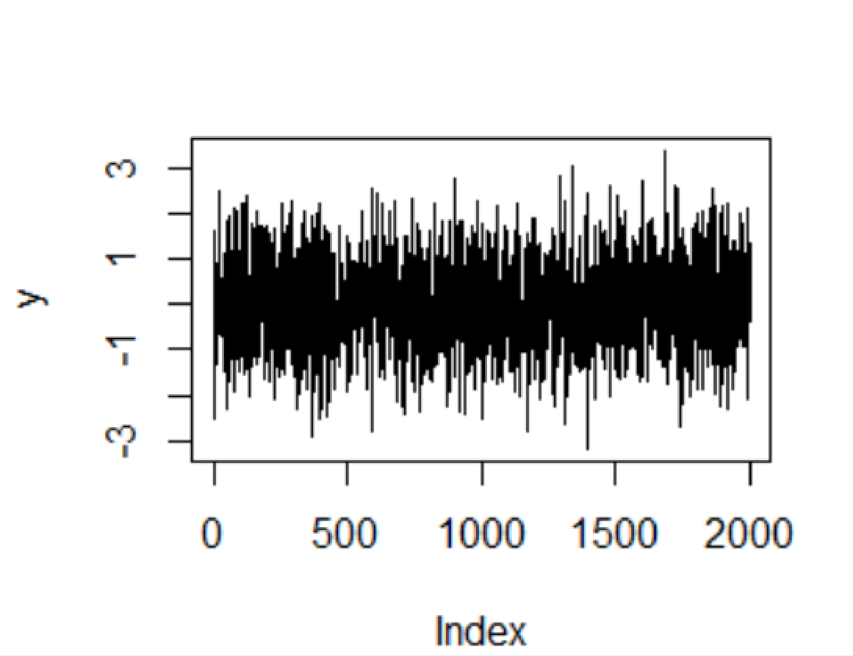
GOALS FOR TODAY

- MCMC - Markov Chain Monte Carlo
  - What it is
  - Has it converged
  - Options to help convergence
- Options in running MCMC to get posterior distribution
- Another in-class example

SIMULATING A DISTRIBUTION

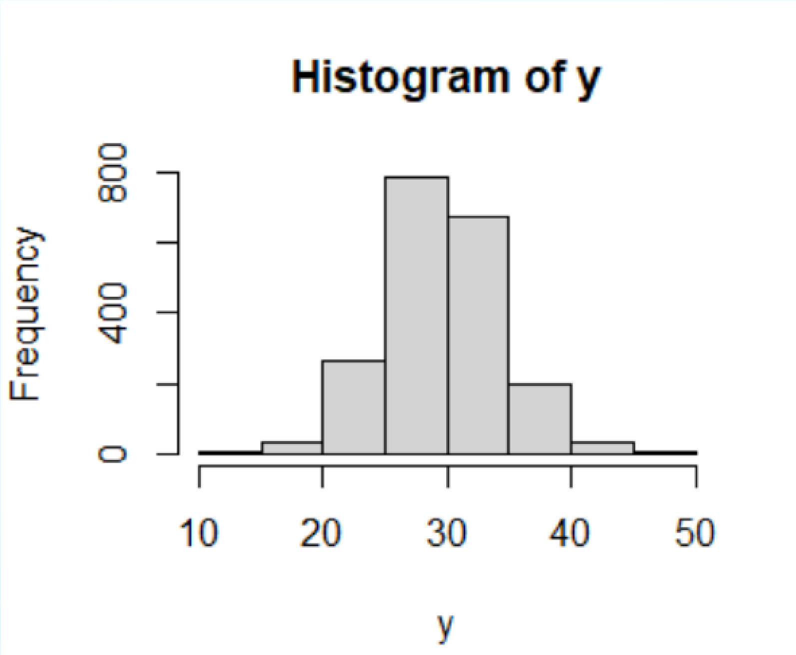
```
y=rnorm(2000)
> hist(y)
> plot(y,type='l')
```

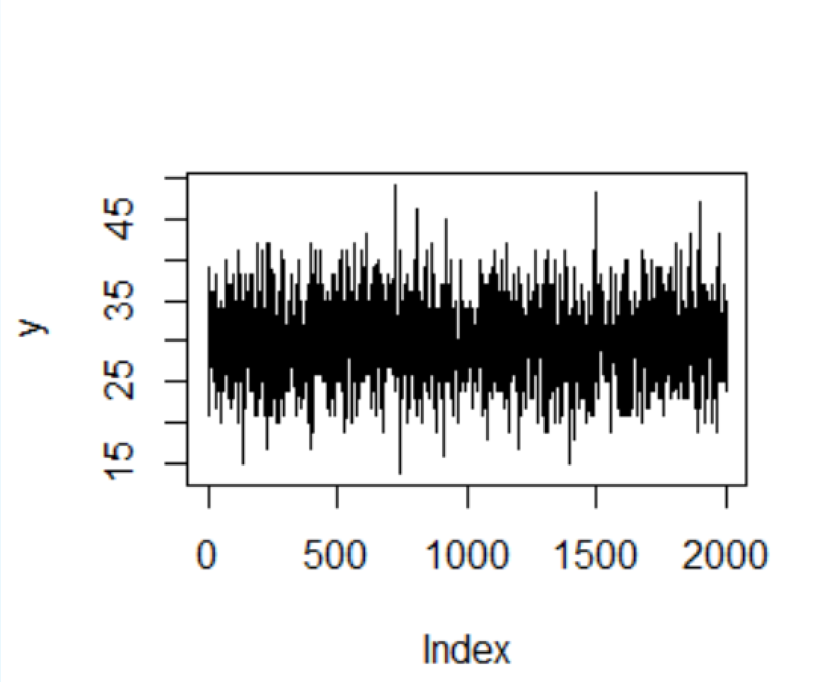




ANOTHER DISTRIBUTION

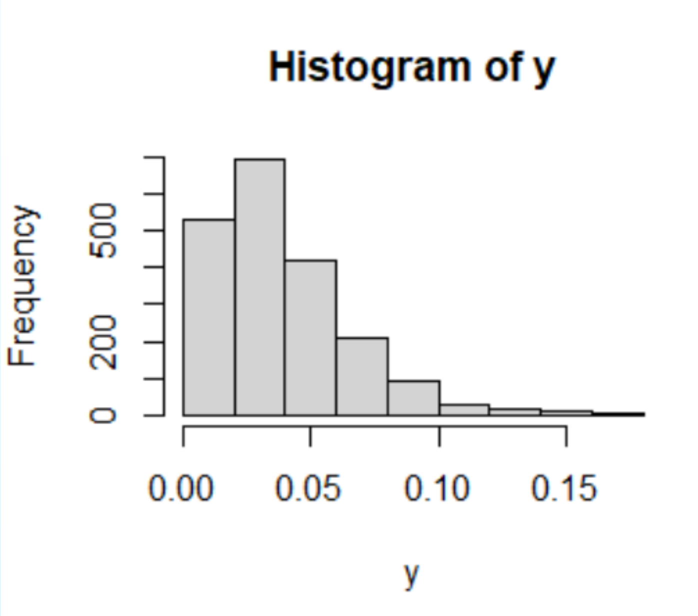
```
> y=rbinom(2000,100,0.3)
> hist(y)
> plot(y,type='l')
```

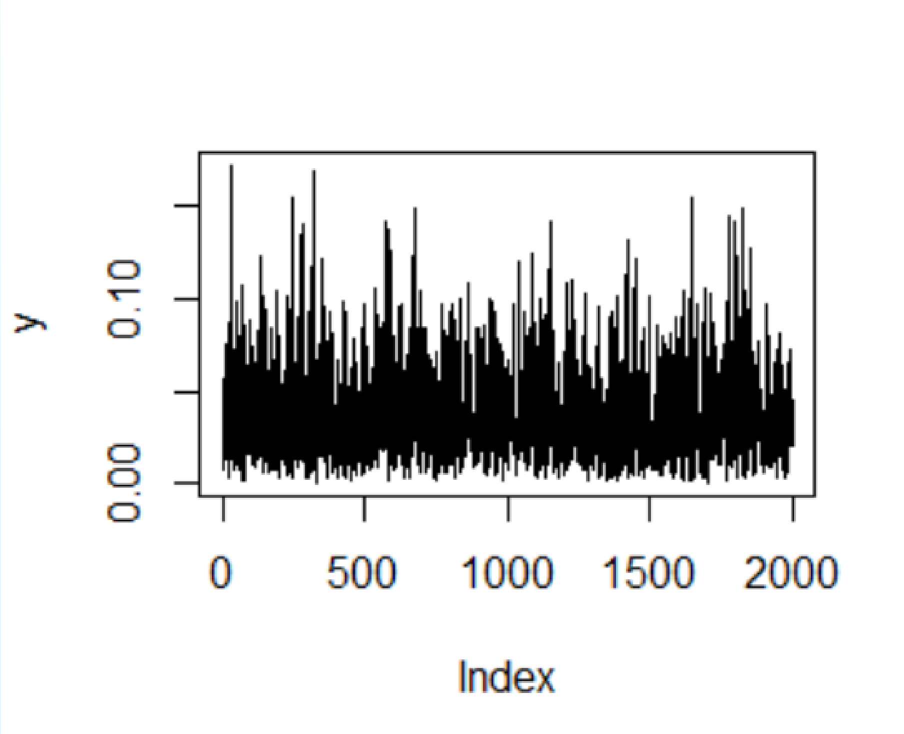




SKEWED DISTRIBUTION

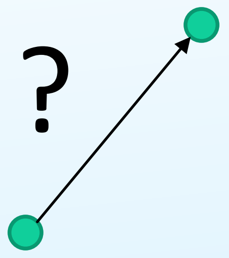
```
> y=rbeta(2000,2,50)
> hist(y)
> plot(y,type='l')
```



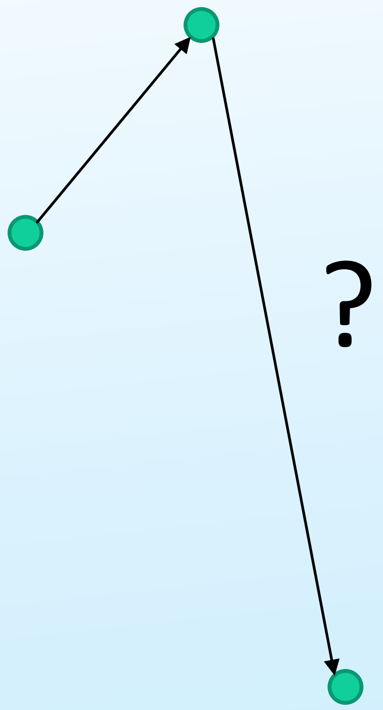


MARKOV CHAIN MONTE CARLO

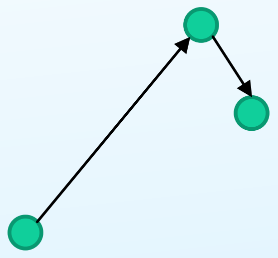
MARKOV CHAIN MONTE CARLO



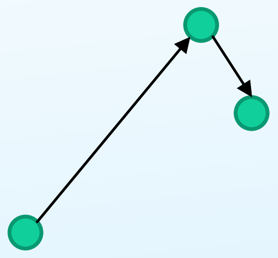
MARKOV CHAIN MONTE CARLO



MARKOV CHAIN MONTE CARLO



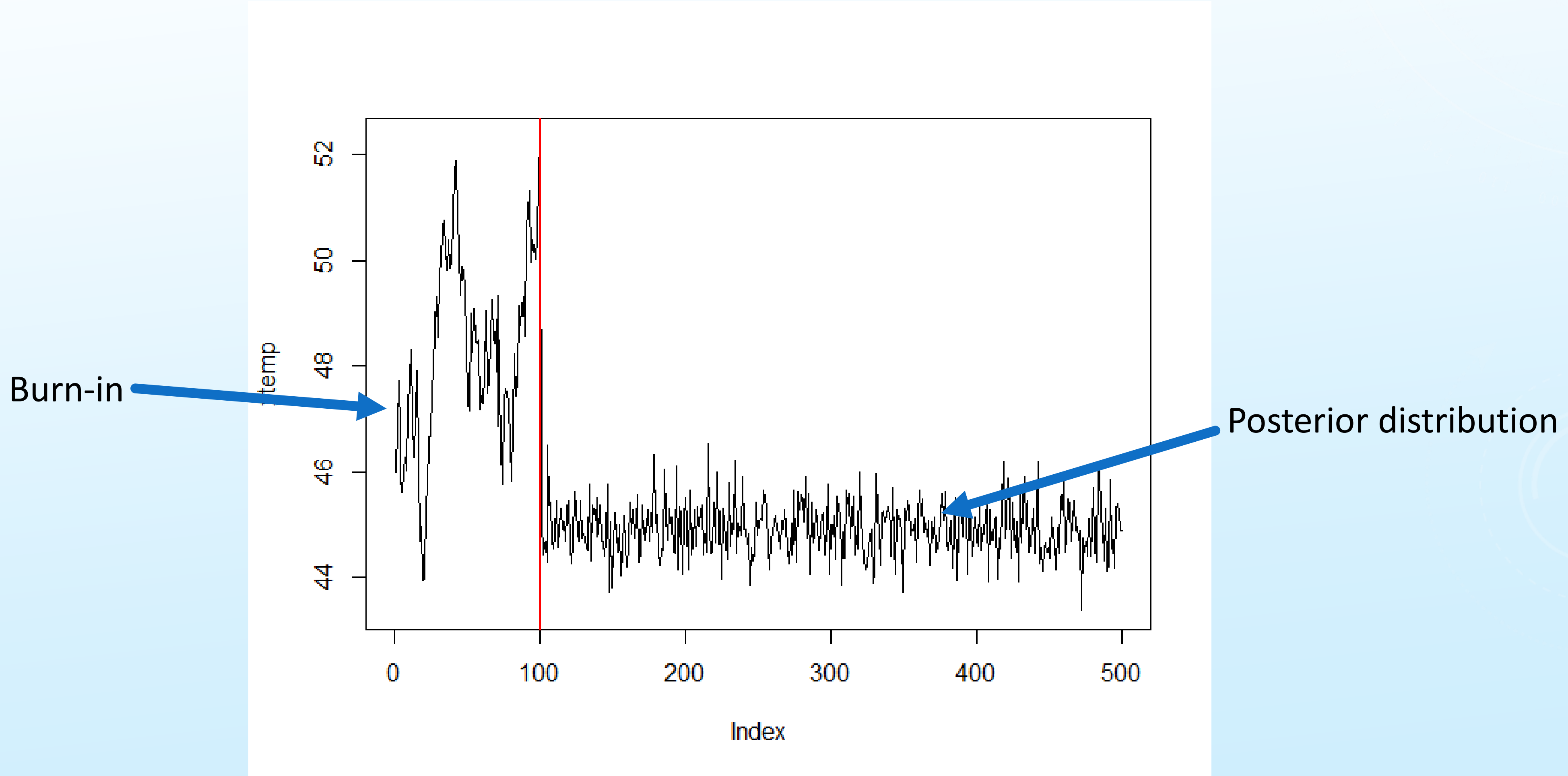
MARKOV CHAIN MONTE CARLO



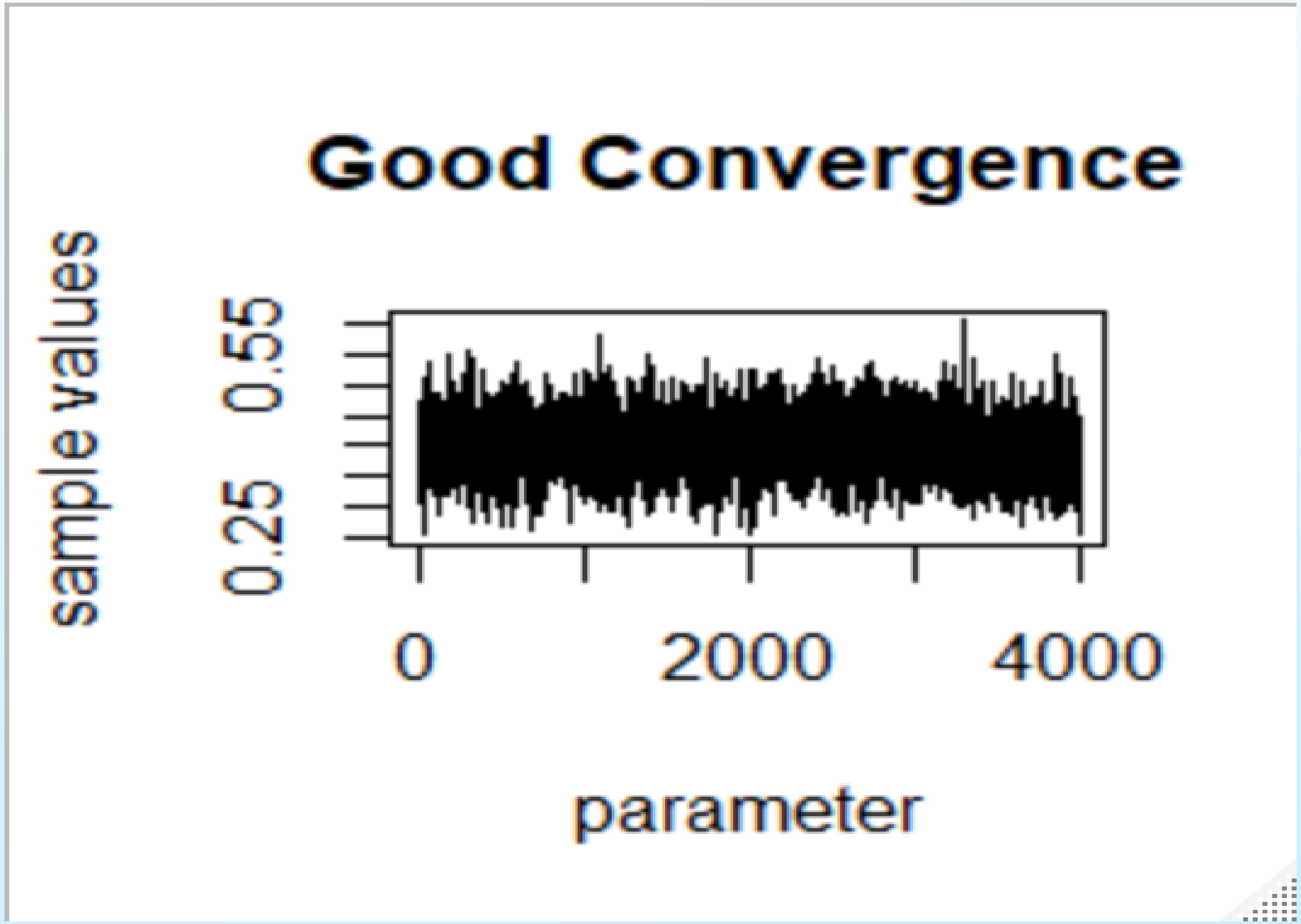
Stan uses the Hamiltonian Monte Carlo method for its Markov Chain and its adaptive variant the no U-turn sampler (NUTS). For more details, see https://mc-stan.org/docs/2_19/reference-manual/hamiltonian-monte-carlo.html

In Python: Stan, PyMC3

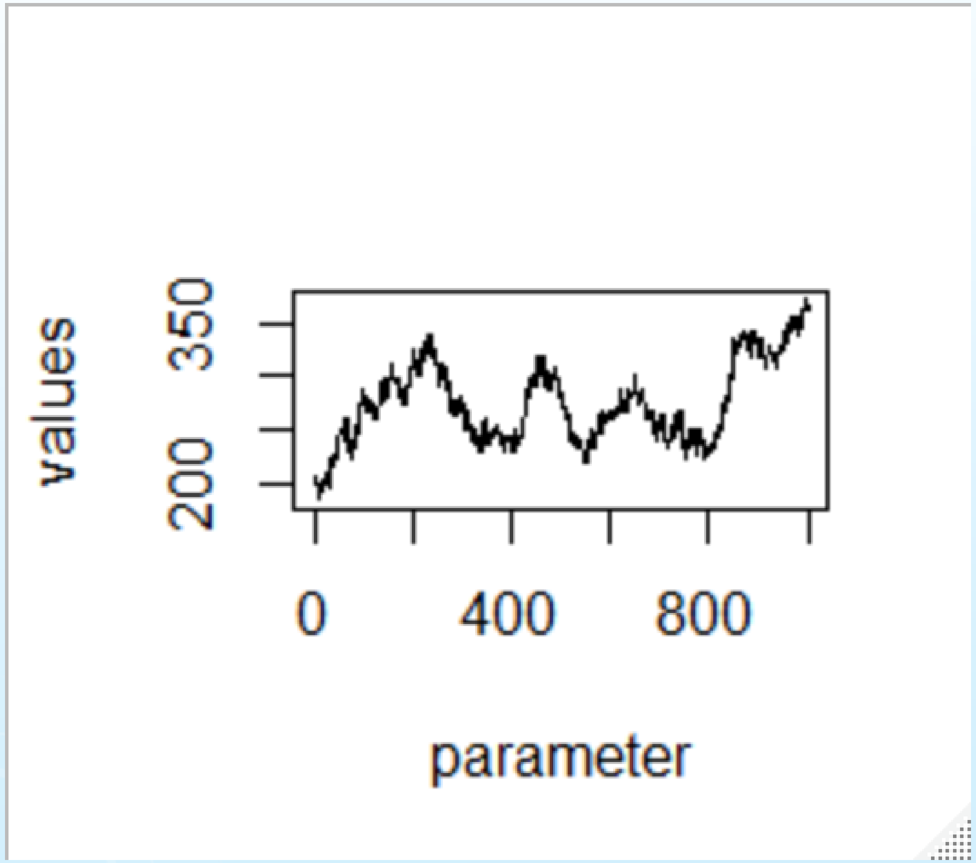
MCMC

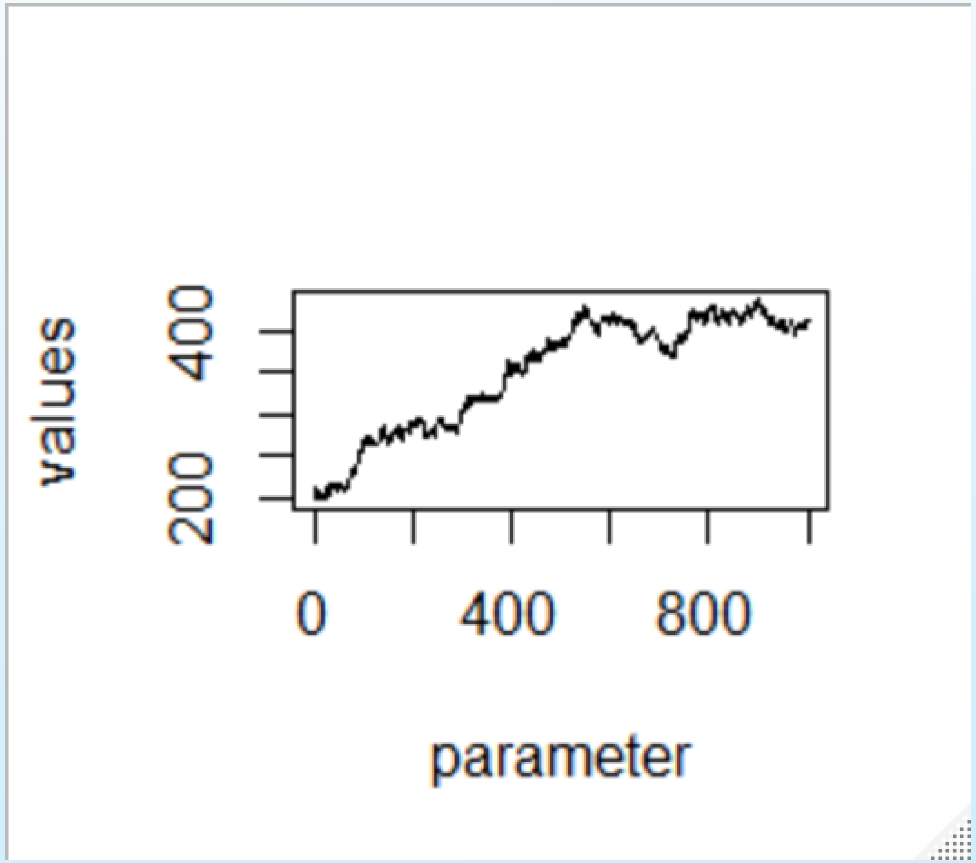

Burn-in
Posterior distribution
temp
52
50
48
46
44
0
100
200
300
400
500
Index

CONVERGENCE



NONCONVERGENCE





FIXES

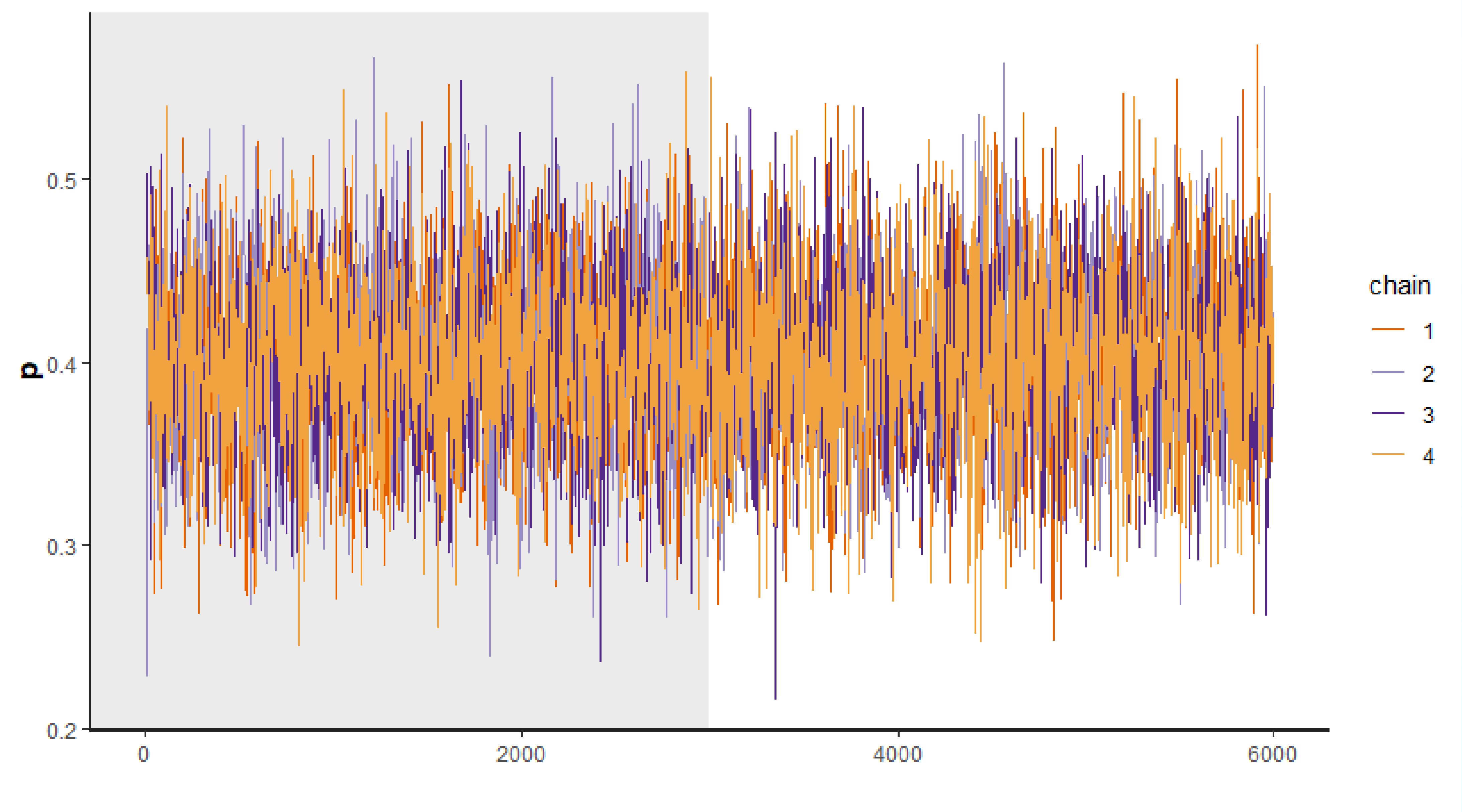
- Improper posterior or bad prior
    - Fix: New prior distribution
- Hasn't converged yet
    - Let the chain run longer
- Chain continues to increase
    - Potentially a bad starting point…provide a new starting point (or change prior)
- Too much autocorrelation in chain
    - Thin the chain

OPTIONS FOR STAN CODE

```
binom.stan=stan(model_code = ex1,data=binom.data,seed=98763,
chains = 4,      # number of Markov chains
warmup = 3000,   # number of warmup iterations per chain
iter = 6000,     # total number of iterations per chain
refresh = 0,     # no progress shown
thin=3,                    # will 'thin' the chains.help with autocorrelated posterior samples
init=0.3                   #specify initial values)
```

Creates four chains; each chain has 6000 values, however, only every 3rd value is taken (now down to 2000 per chain that is useful); first 3000 (well, actually only 1000 since we are thinning) is burn-in meaning it is not used

End result will have a total of 4000 posterior values (1000 from each chain) traceplot(binom.stan, inc_warmup = TRUE)



DIAGNOSTICS

```
print(binom.stan, pars=c("p", "lp__"), probs=c(.1,.5,.9))
```

	mean	se_mean	sd	10%	50%	90%	n_eff	Rhat
p	0.40	0.00	0.05	0.34	0.40	0.46	3268	1
lp__	-69.23	0.01	0.72	-70.10	-68.96	-68.74	3290	1

Log posterior value (unnormalized)

Want these to be close to 1 (this means convergence); if greater than 1.1, then there could potentially be a problem (Rhat or potential scale reduction)

DIAGNOSTICS

```
print(binom.stan, pars=c("p", "lp__"), probs=c(.1,.5,.9))
```

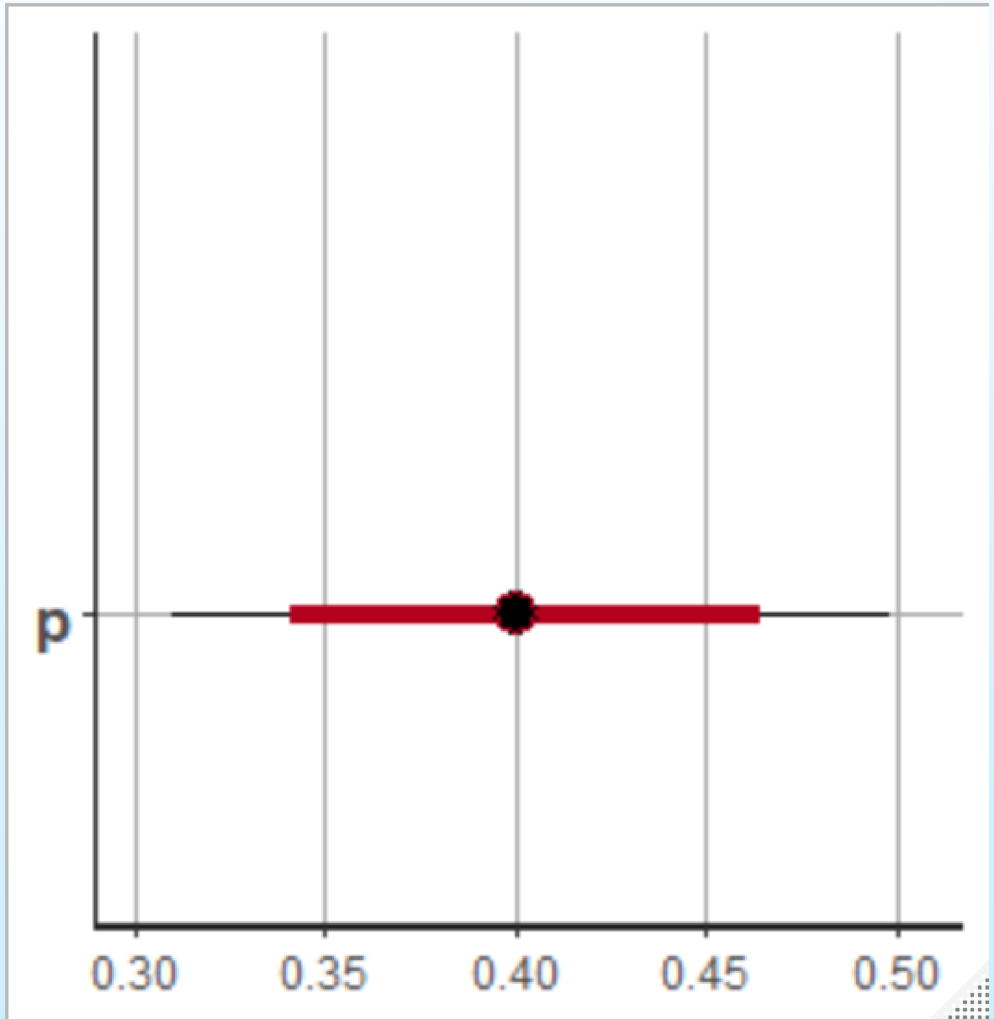
	mean	se_mean	sd	10%	50%	90%	n_eff	Rhat
p	0.40	0.00	0.05	0.34	0.40	0.46	3268	1
lp__	-69.23	0.01	0.72	-70.10	-68.96	-68.74	3290	1

Log posterior value (unnormalized)

Effective sample size…bigger is better (more than 1000 is fine). If too small, would recommend trying to thin chains.

PROBABILITY INTERVAL

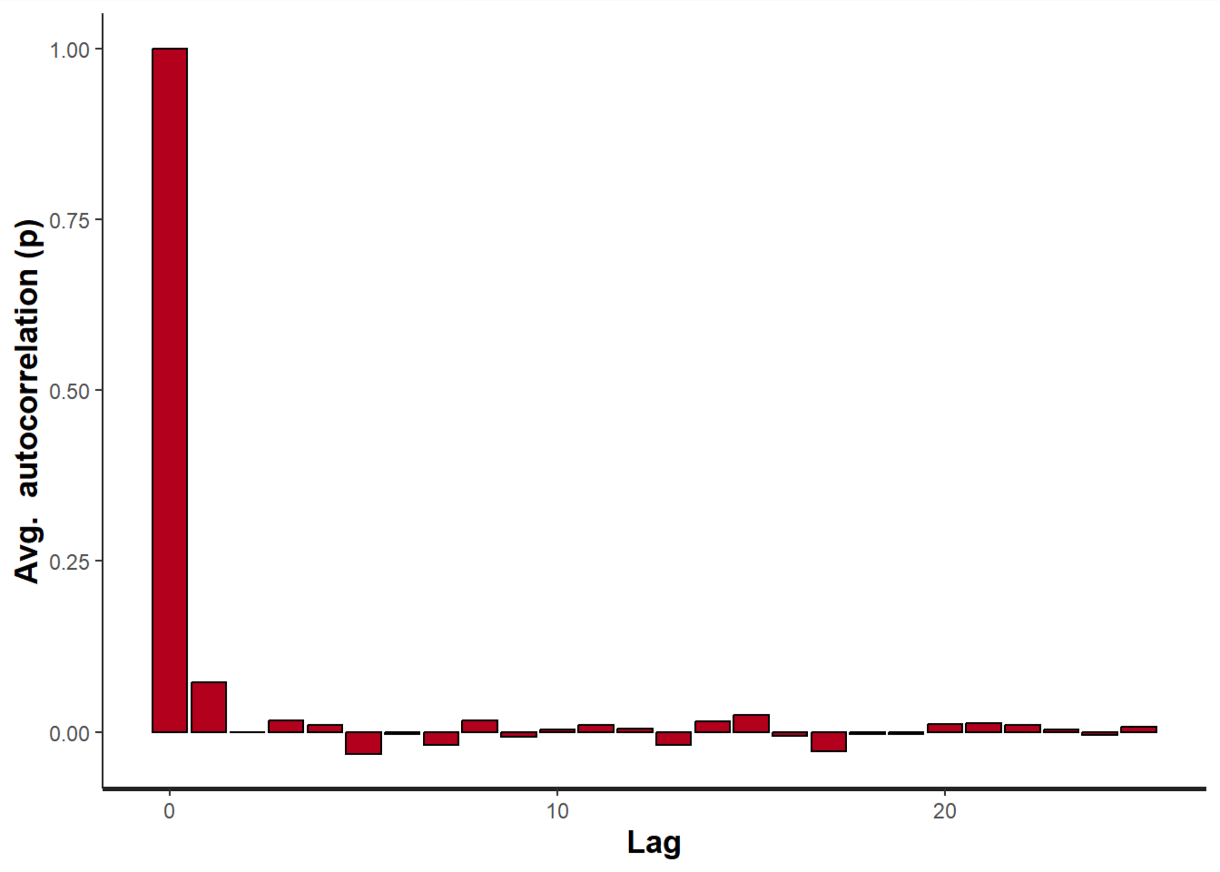
plot(binom.stan)



Inner is 80% probability interval
Outer is 95% probability interval

CHECK AUTOCORRELATION

stan_ac(binom.stan,pars=c("p"))



ANOTHER POTENTIAL WARNING

Warning: There were 2 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help. See http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

Fix: add this line to your options…

```
control=list(adapt_delta=0.9)
```

ESTIMATE VALUE AT RISK

- Recall from Simulation and Risk calculating Value at Risk (VaR)
- Say we are interested in estimating VaR for Apple stock (AAPL) in Rate of Change (ROC) for one day
- If we assume Rate of Change (R_t) follows a Normal distribution with mean μ and standard deviation σ
- We then need to assign a distribution to μ and σ^2
  - Assume μ is distributed as Normal(0,100)
  - Assume σ^2 is distributed as Inv-Gamma(0.001,0.001)
- Once we get posterior for μ and σ^2, we can use this to get the 1st quantile

DATA

```
library(quantmod)
tickers = c("AAPL")
getSymbols(tickers)
AAPL$aapl_r <- ROC(AAPL$AAPL.Close)
AAPL <- cbind(tail(AAPL$AAPL.Close, 500), tail(AAPL$aapl_r, 500))
```

STAN CODE

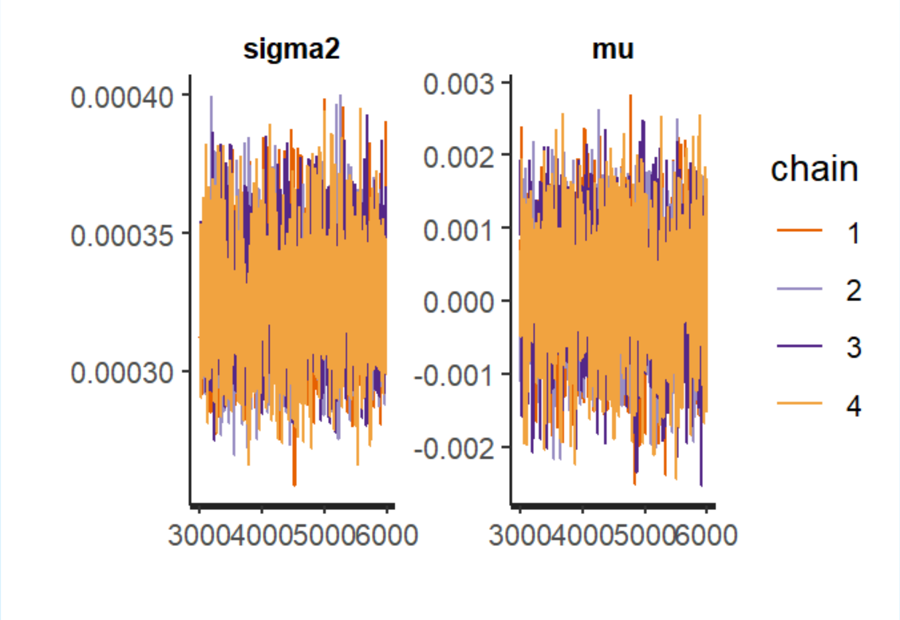
```
ex2 <- “
data {
	int <lower=0> n;
	real y[n];
}
parameters {
	real <lower=0> sigma2;
	real mu;
}
model {
	mu ~ normal(0,100);
	y ~ normal(mu,sigma);
sigma2 ~ inv_gamma(0.001,0.001);
}
"
```

RUN STAN CODE

```
stock.data=list(n=nrow(stocks), y=as.vector(stocks$aapl_r))

stock.stan=stan(model_code = ex2,data=stock.data,seed=15893, chains = 4, warmup = 3000, iter = 6000,
refresh = 0, thin=3)
```

CHECK CONVERGENCE



traceplot(stock.stan,inc_warmup=F)

print(stock.stan, pars=c("mu", "sigma2"), probs=c(.1,.5,.9))

	mean	n_eff	Rhat
mu	0	3995	1
sigma2	0	2986	1

*** Values are VERY small!

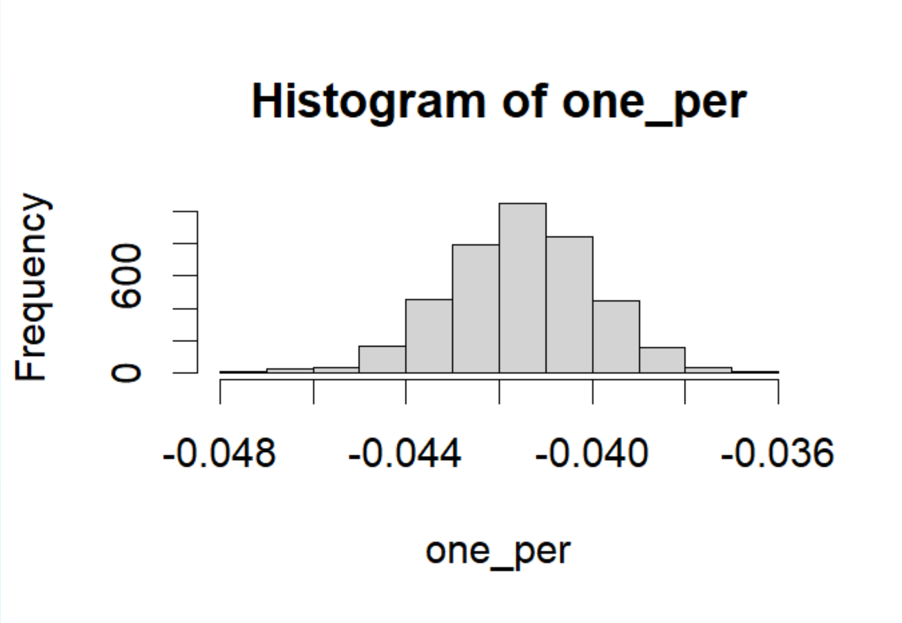
USING POSTERIOR INFORMATION:

Posterior samples

```
post.params=extract(stock.stan)
new.mu=post.params$mu
new.sigma2=post.params$sigma2
head(cbind(new.mu,new.sigma2))
          new.mu   new.sigma2
[1,] -0.0003626559 0.0003317394
[2,] -0.0004120245 0.0003196366
[3,] -0.0008917359 0.0003210392
[4,]  0.0003888528 0.0003505453
[5,]  0.0003630797 0.0003195843
[6,] -0.0001102442 0.0003167621
```

VaR ROC at 99% confidence

- What we need is the 1st percentile for ROC:
- one_per=qnorm(0.01,new.mu,sqrt(new.sigma2))



USE THIS INFO FOR IN-CLASS ASSIGNMENT