"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?

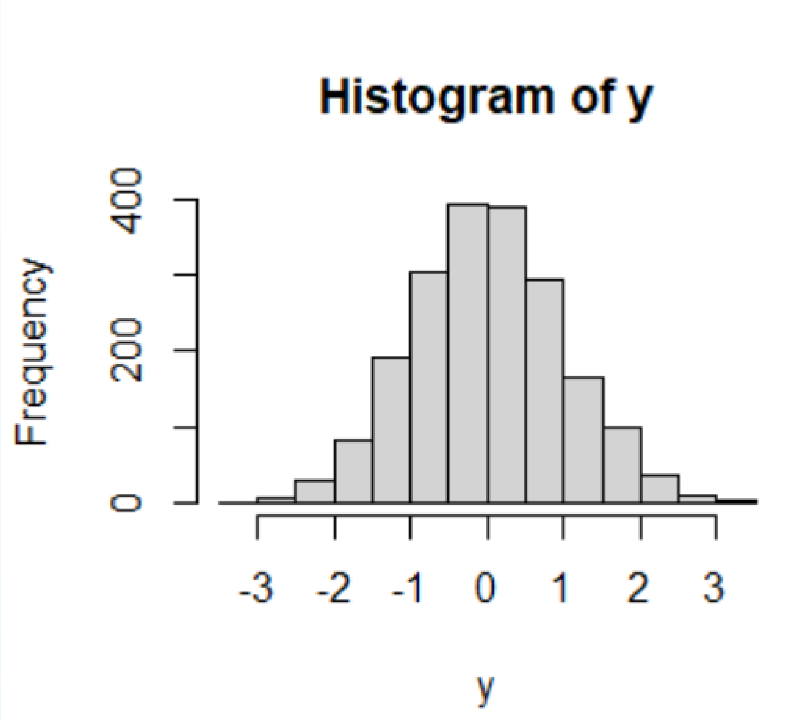
- Terminology: Prior, Sampling distribution, Posterior
- How to define problem (decide sampling distribution of data, define priors for parameters, use pymc to generate posterior distribution of parameters)
- How to use posterior to answer questions about the parameter
- How data (sample size) and prior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

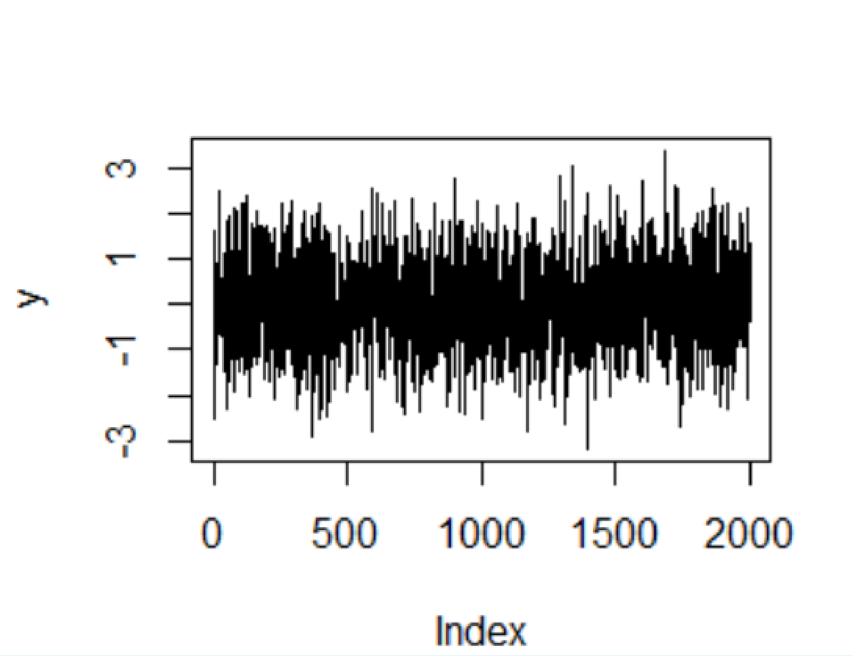
GOALS FOR TODAY

- 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

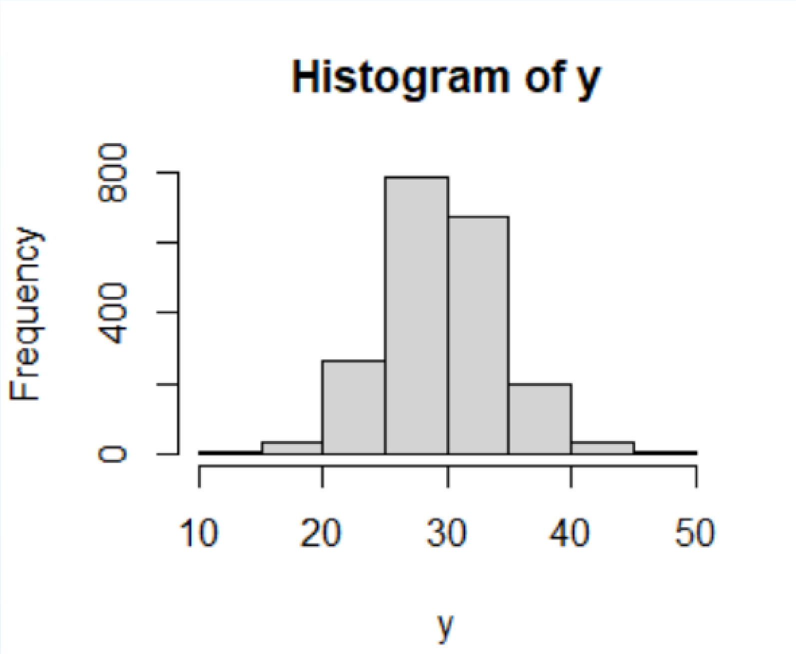
```
y=rnorm(2000)
> hist(y)
> plot(y,type='l')
```

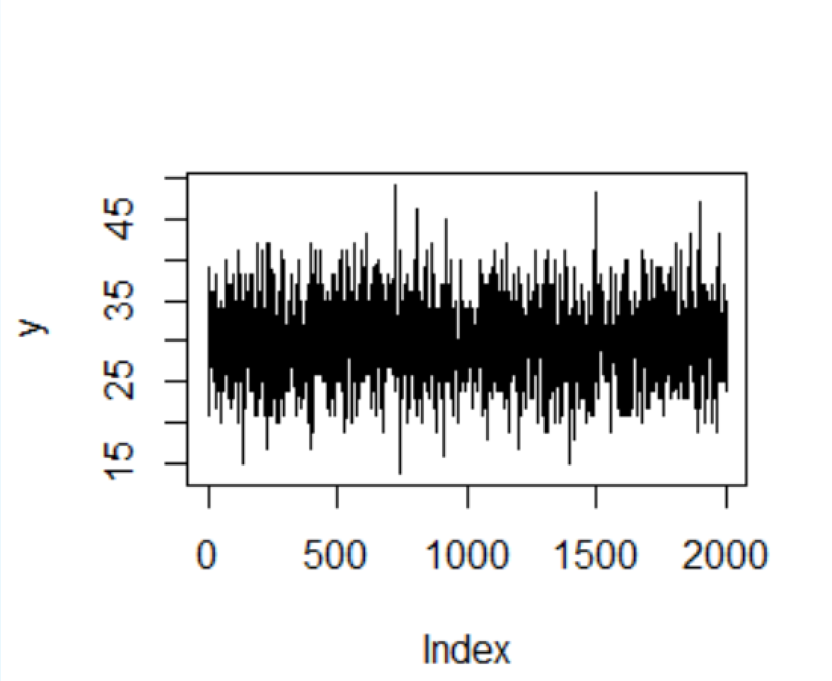




ANOTHER DISTRIBUTION

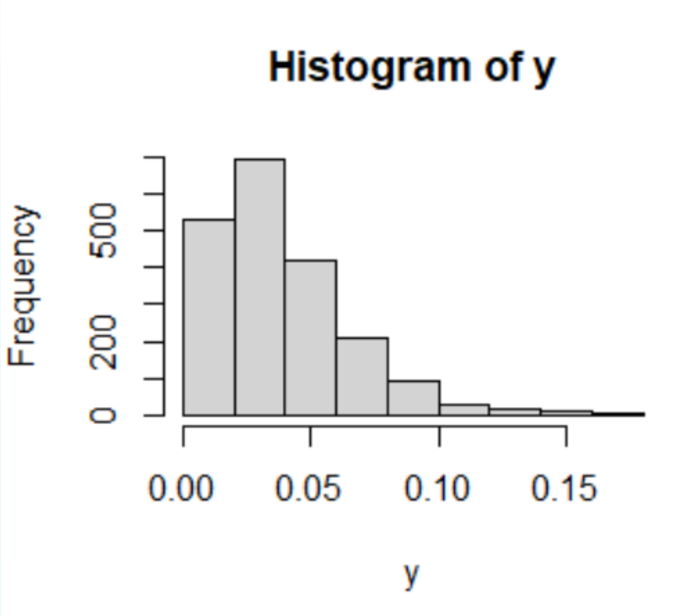
```
> y=rbinom(2000,100,0.3)
> hist(y)
> plot(y,type='l')
```

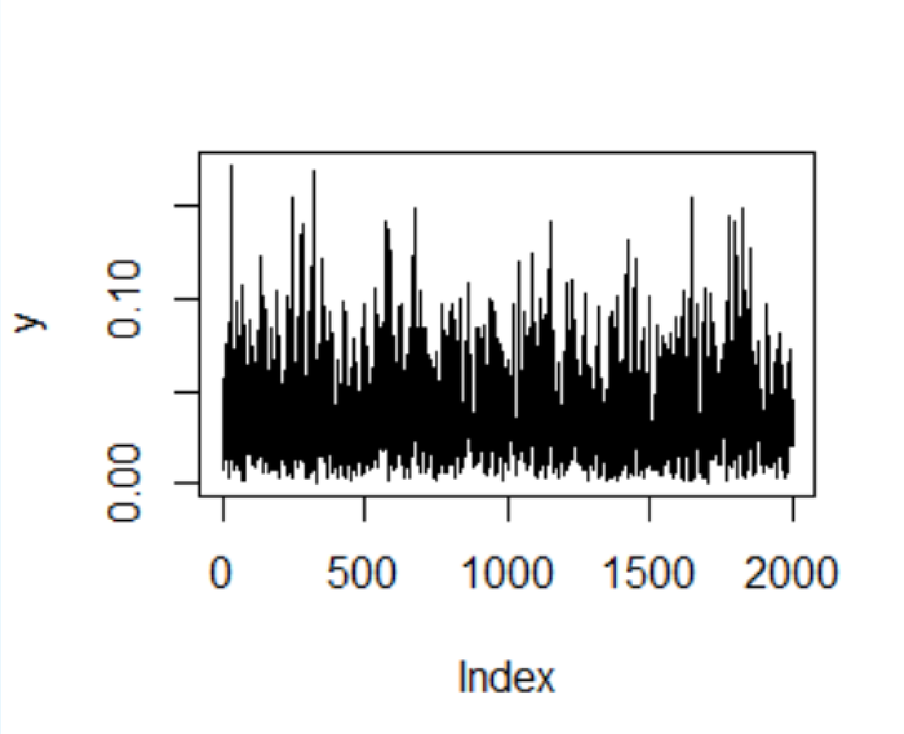




SKEWED DISTRIBUTION

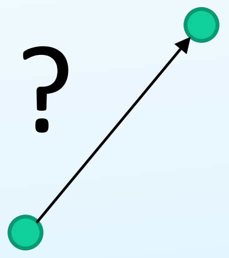
```
> y=rbeta(2000,2,50)
> hist(y)
> plot(y,type='l')
```



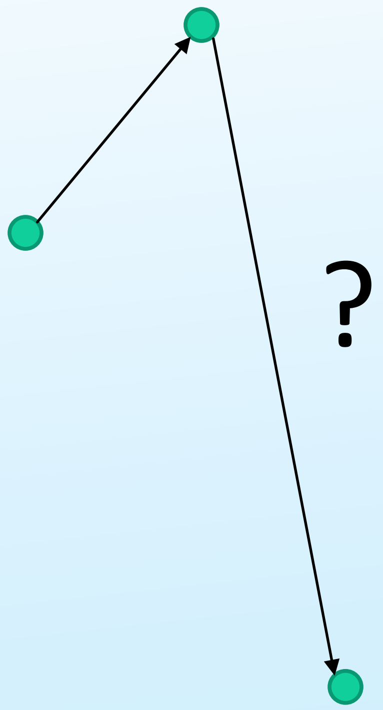


MARKOV CHAIN MONTE CARLO

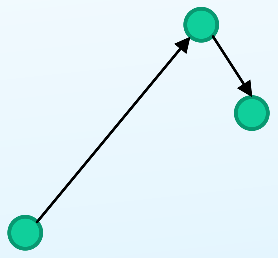
MARKOV CHAIN MONTE CARLO



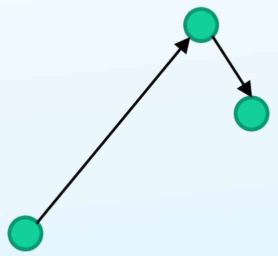
MARKOV CHAIN MONTE CARLO



MARKOV CHAIN MONTE CARLO

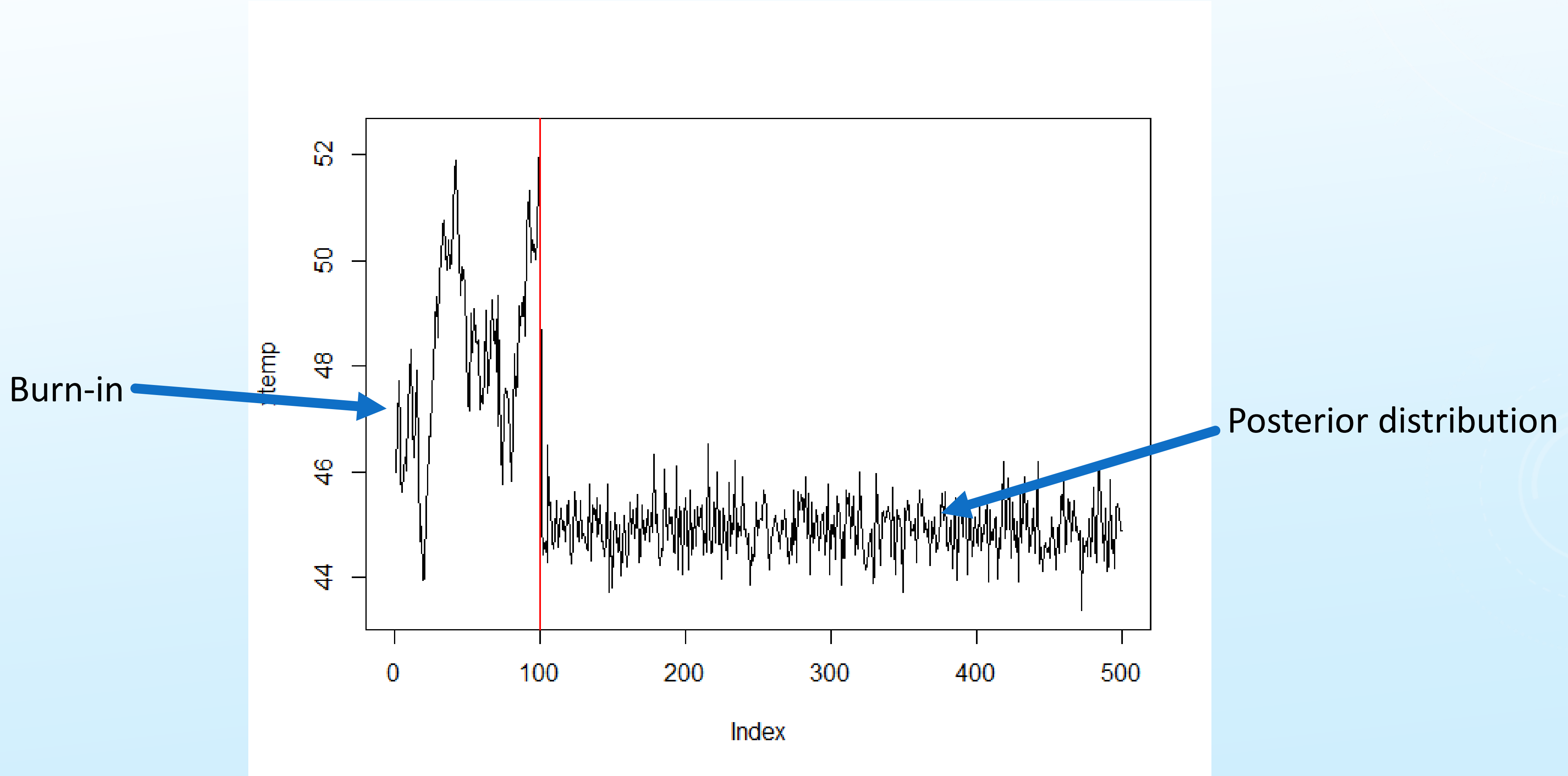


MARKOV CHAIN MONTE CARLO

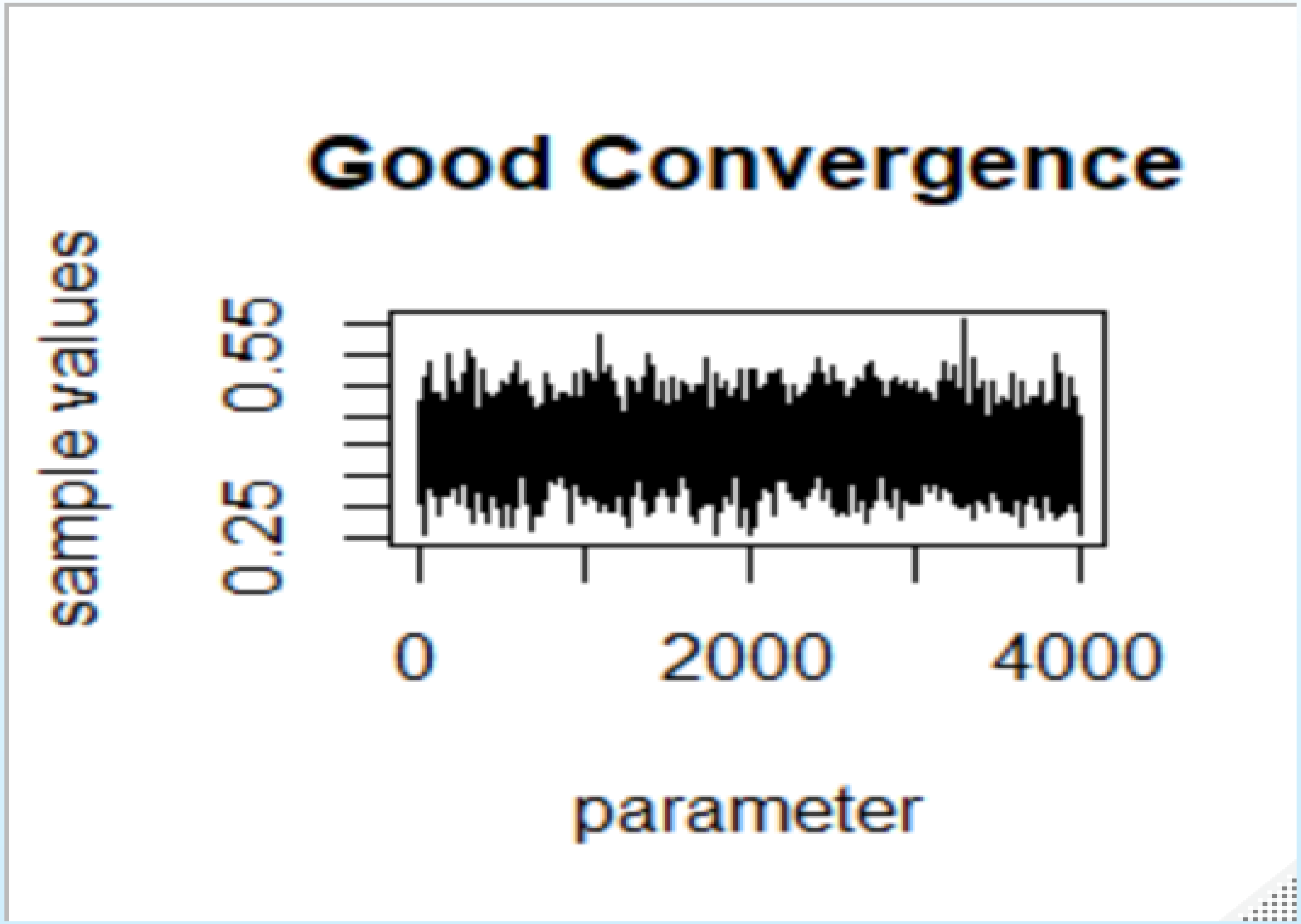


pymc uses the Hamiltonian Monte Carlo method for its Markov Chain and its adaptive variant called the no U-turn sampler (NUTS). For an interesting read, see https://towardsdatascience.com/python-hamiltonian-monte-carlo-from-scratch-955dba96a42d/
Stan also uses NUTS.

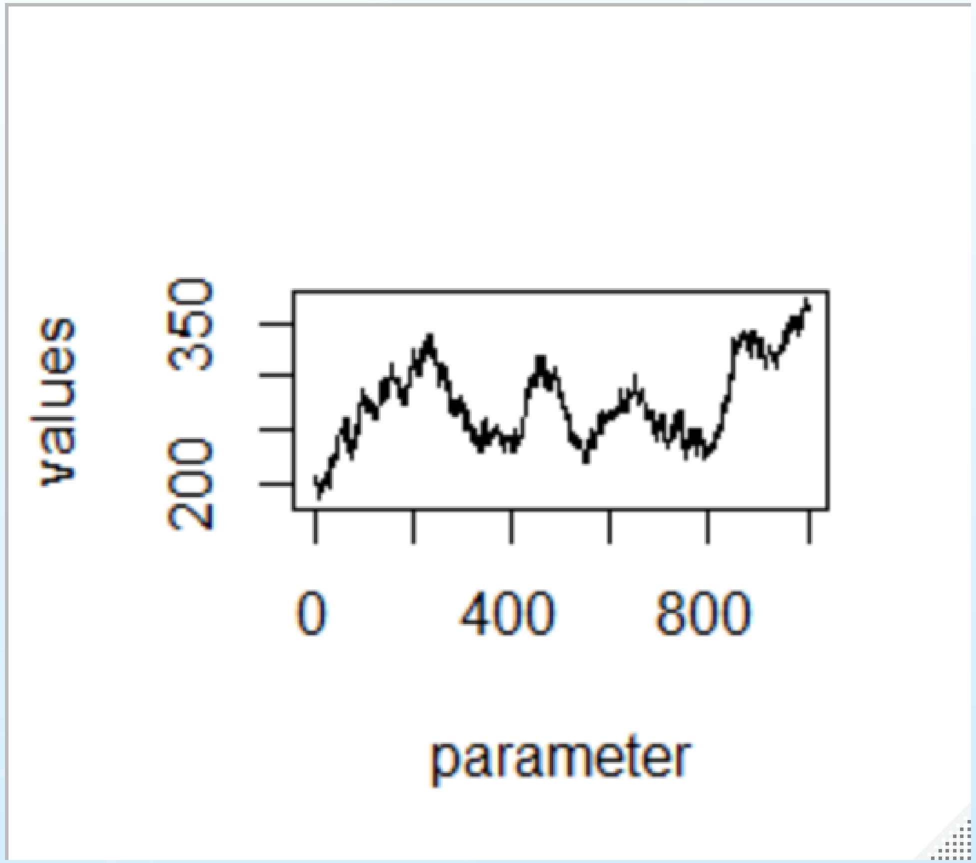
MCMC

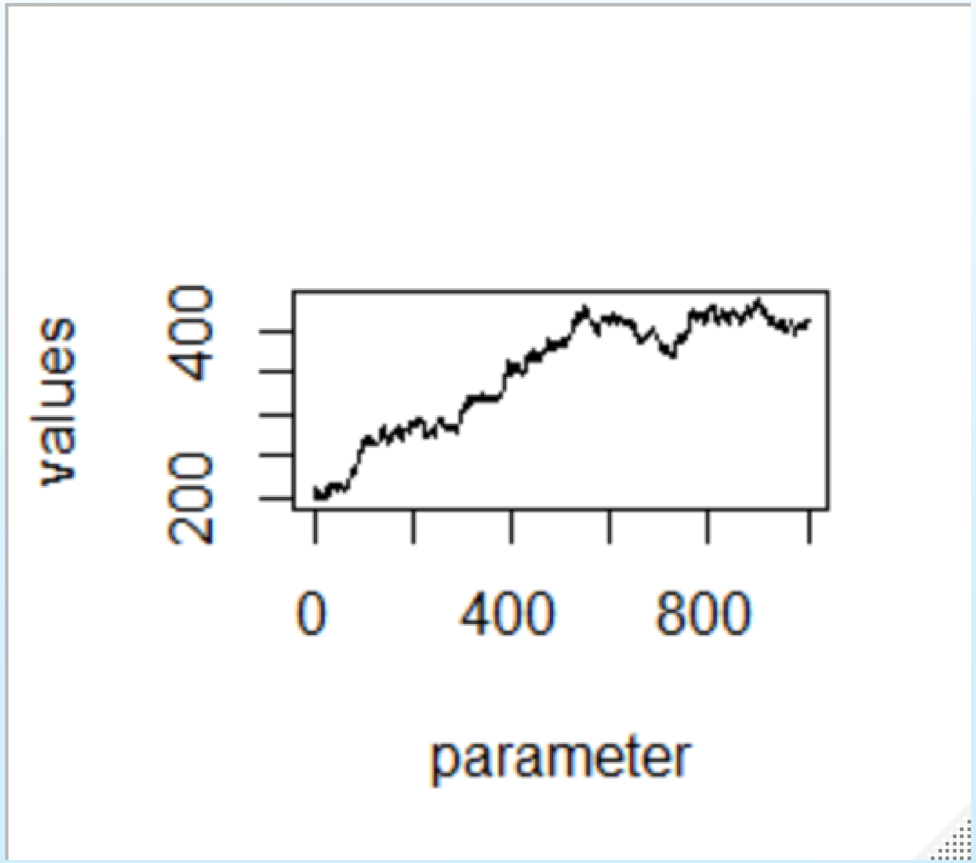

Burn-in
Posterior distribution
temp
52
50
48
46
44
0
100
200
300
400
500
Index

CONVERGENCE



NONCONVERGENCE





FIXES

- Improper posterior or bad prior
  - Fix: New prior distribution
- Hasn’t converged yet
  - Let the chain run longer
- Too much autocorrelation in chain
  - Thin the chain

OPTIONS FOR PYMC CODE

```
n = 100
y = 40
# Model
with pm.Model () as binom_model :
        p = pm.Beta ("p", alpha=1, beta=1) ### Prior distribution
        y_obs = pm.Binomial("y_obs", n=n, p=p, observed=y)  ### Sampling distribution
        trace = pm.sample (draws=3000,  # samples after burn-in
        tune=3000,   # Warmup iterations
        chains=4,    # Number of Markov chains
        thin=3,      # Thinning factor (really doesn't work)
        random_seed=98763,
        return_inferencedata=True ) ### specify information
az.summary(trace) ### Summary
p1 = trace.posterior["p"].values.flatten()
thin_val=3
p_post = p1[::thin_val] ### would use this for inference
```

Creates four chains; each chain has 3000 values (it automatically throws away burn-in), thin does not work…. Need to thin after you pull off values (if autocorrelation is an issue)

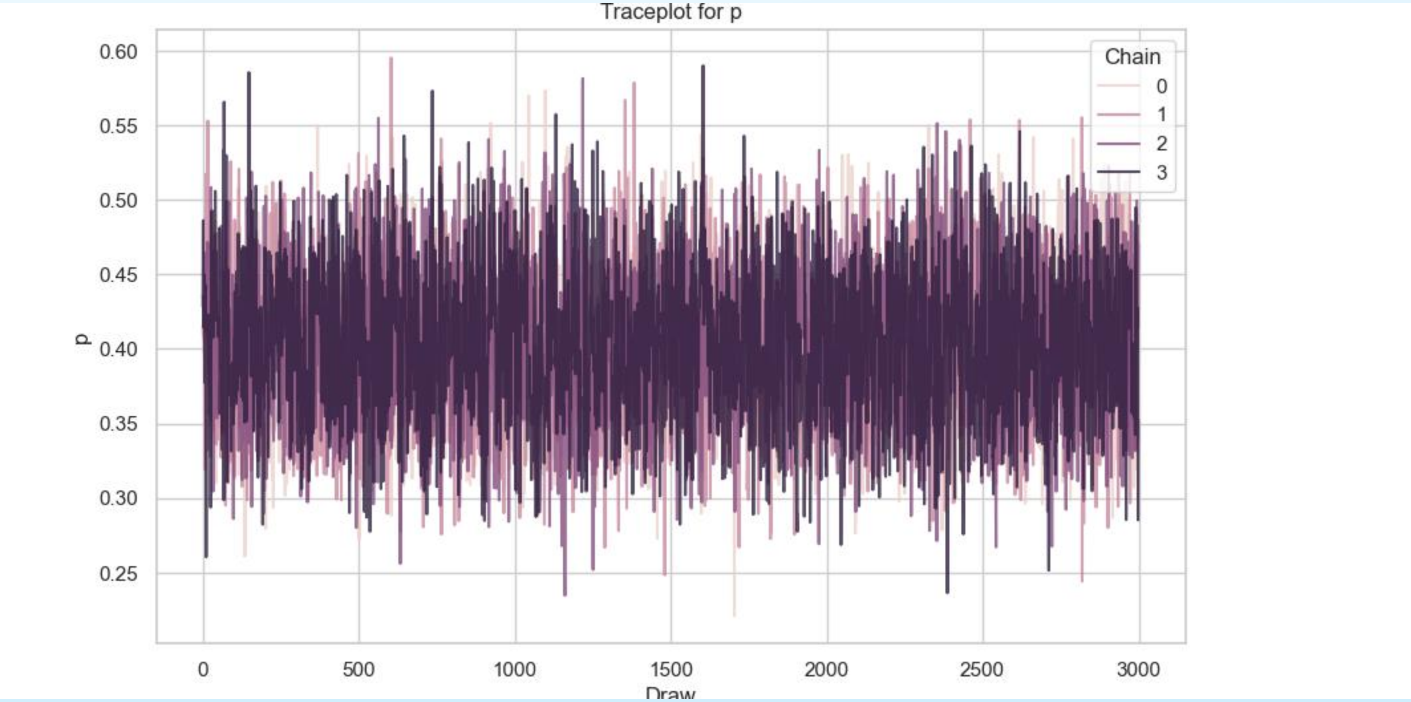
p1 has 12000 observations (3000x4) and p_post has 4000 observations (after thinning)

TRACEPLOT OF FOUR CHAINS

Code

```
import seaborn as sns
df_p = az.extract(trace, var_names=["p"]).to_dataframe()
sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_p, x="draw", y="p", hue="chain", alpha=0.8)
plt.xlabel("Draw")
plt.ylabel("p")
plt.title(“Traceplot for p")
plt.legend(title="Chain")
plt.show()
```

Traceplot



DIAGNOSTICS

az.summary(trace)

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
p	0.403	0.048	0.313	0.491	0.001	0.0	5323.0	8853.0	1.0

Want this to be close to 1 (this means convergence); if greater than 1.1, then there could potentially be a problem

DIAGNOSTICS

az.summary(trace)

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
p	0.403	0.048	0.313	0.491	0.001	0.0	5323.0	8853.0	1.0

Effective sample size…bigger is better. If either of these are small, then there is a potential issue with independence of samples (ess_bulk looks at the middle range of your posterior values while ess_tail looks at the tails of the distribution). FIX if these are small: consider thinning.

ANOTHER POTENTIAL WARNING

Divergences after tuning: if your chains are diverging after tuning, you can try to increase target_accept, increase burn-in period or try changing prior distributions.

VALUE AT RISK (VAR)

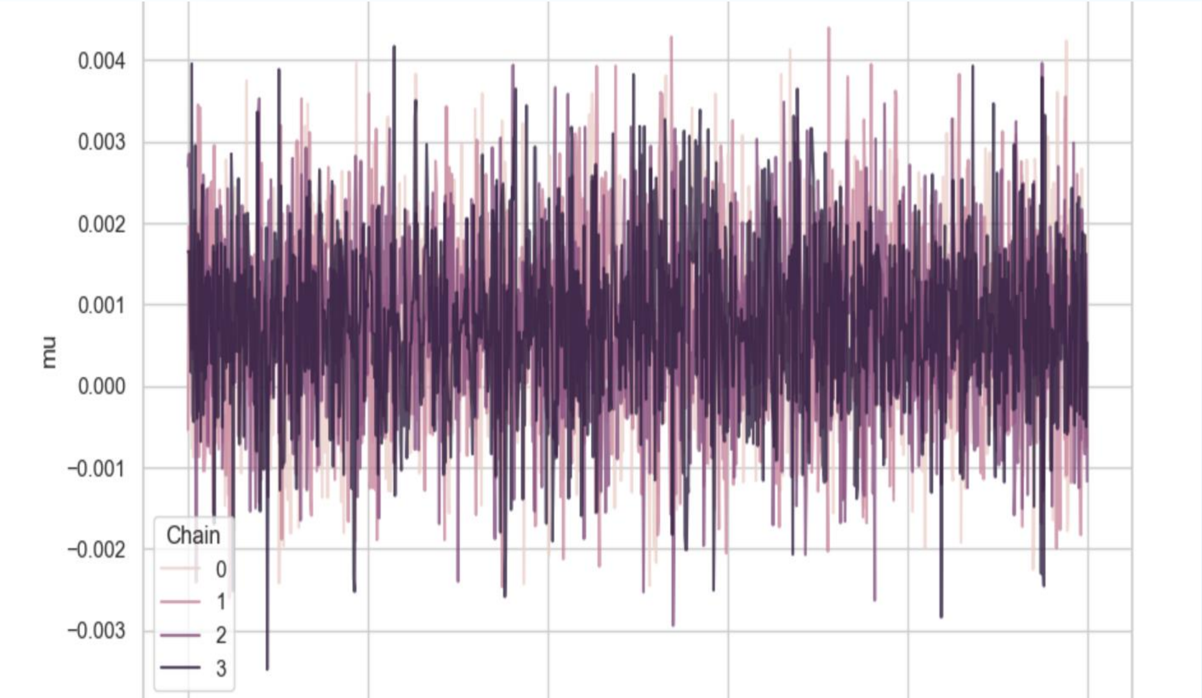
ESTIMATE VALUE AT RISK

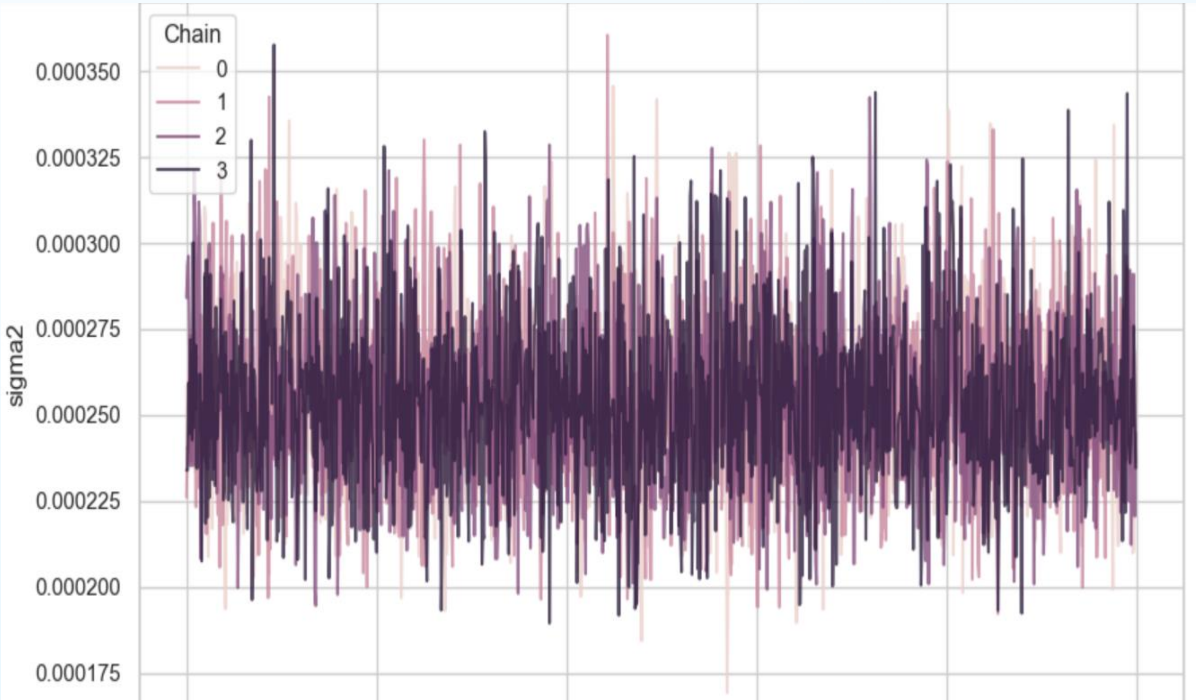
- Recall from Simulation and Risk calculating Value at Risk (VaR)
- Say we are interested in estimating $VaR_{0.01}$ for Apple stock (AAPL) in Rate of Change (ROC) for one day
- If we assume Rate of Change (R_t) follows a Normal distribution with mean μ and standard deviation σ
- We then need to assign 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

```
import pandas as pd
import yfinance as yf
import math

# Define the ticker symbol
ticker = 'AAPL‘

# Download stock data
data = yf.download(ticker, start='2024-01-01')

# Calculate returns
data['aapl_r'] = data['Close'].pct_change()

# Select last 200 rows of Close and aapl_r columns
stocks = data[['Close', 'aapl_r']].tail(200)
```

```
import scipy.stats as stats
# Define the data
n = len(stocks)
y = stocks['aapl_r'].values
```

```
# Define the PyMC model
with pm.Model() as model:
# Priors
        mu = pm.Normal('mu', mu=0, sigma=100)
        sigma2 = pm.InverseGamma('sigma2', alpha=0.001, beta=0.001)
# Likelihood
        y_lik = pm.Normal('y_lik', mu=mu, sigma=pm.math.sqrt(sigma2),
observed=y)
# Sample posterior
trace = pm.sample(1000, tune=500)
```


0.004
0.003
0.002
0.001
0.000
−0.001
−0.002
−0.003
mu
Chain
0
1
2
3


Chain
0
1
2
3
0.000350
0.000325
0.000300
0.000275
0.000250
0.000225
0.000200
0.000175
sigma2

Looks like we have convergence

```
az.summary(trace)
```

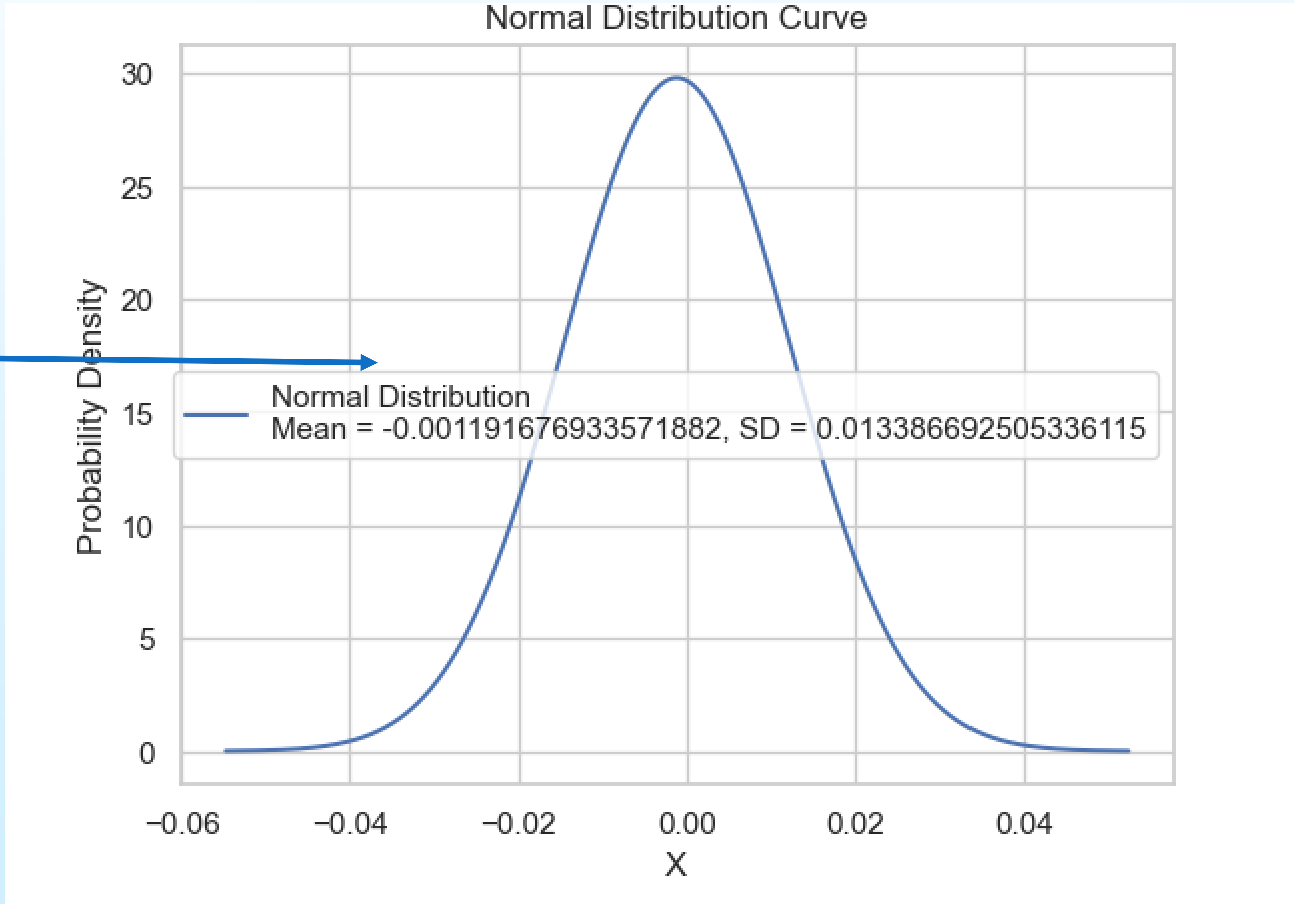
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
mu	0.001	0.001	-0.001	0.003	0.0	0.0	4486.0	2875.0	1.0
sigma2	0.000	0.000	0.000	0.000	0.0	0.0	4080.0	3166.0	1.0

USING POSTERIOR INFORMATION:

Posterior samples

[31]:

	mu	sigma2
0	-0.001192	0.000179
1	-0.000943	0.000188
2	-0.001685	0.000180
3	-0.001862	0.000179
4	-0.002084	0.000178



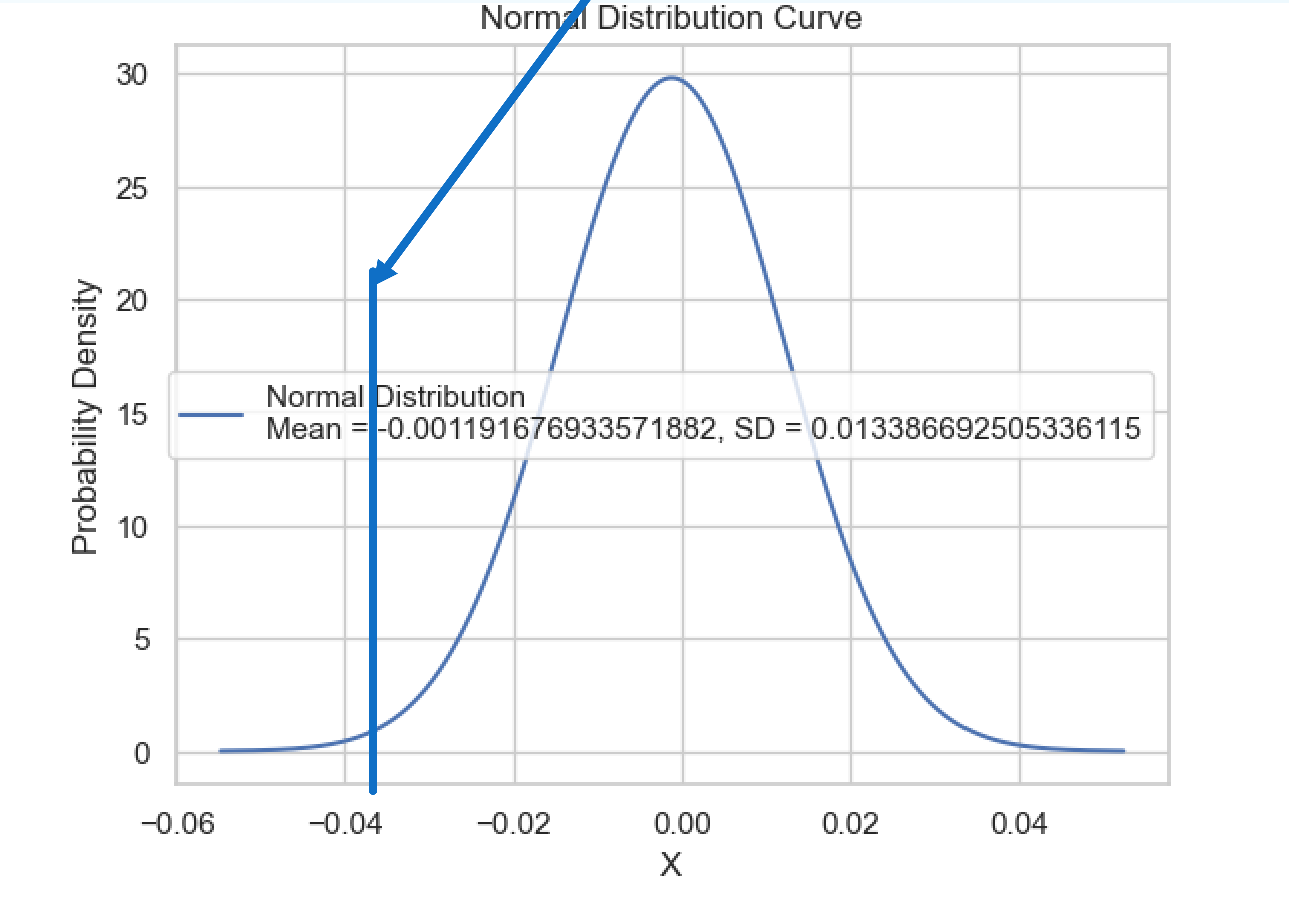
First quantile (VaR0.01)

USING POSTERIOR INFORMATION:

Posterior samples

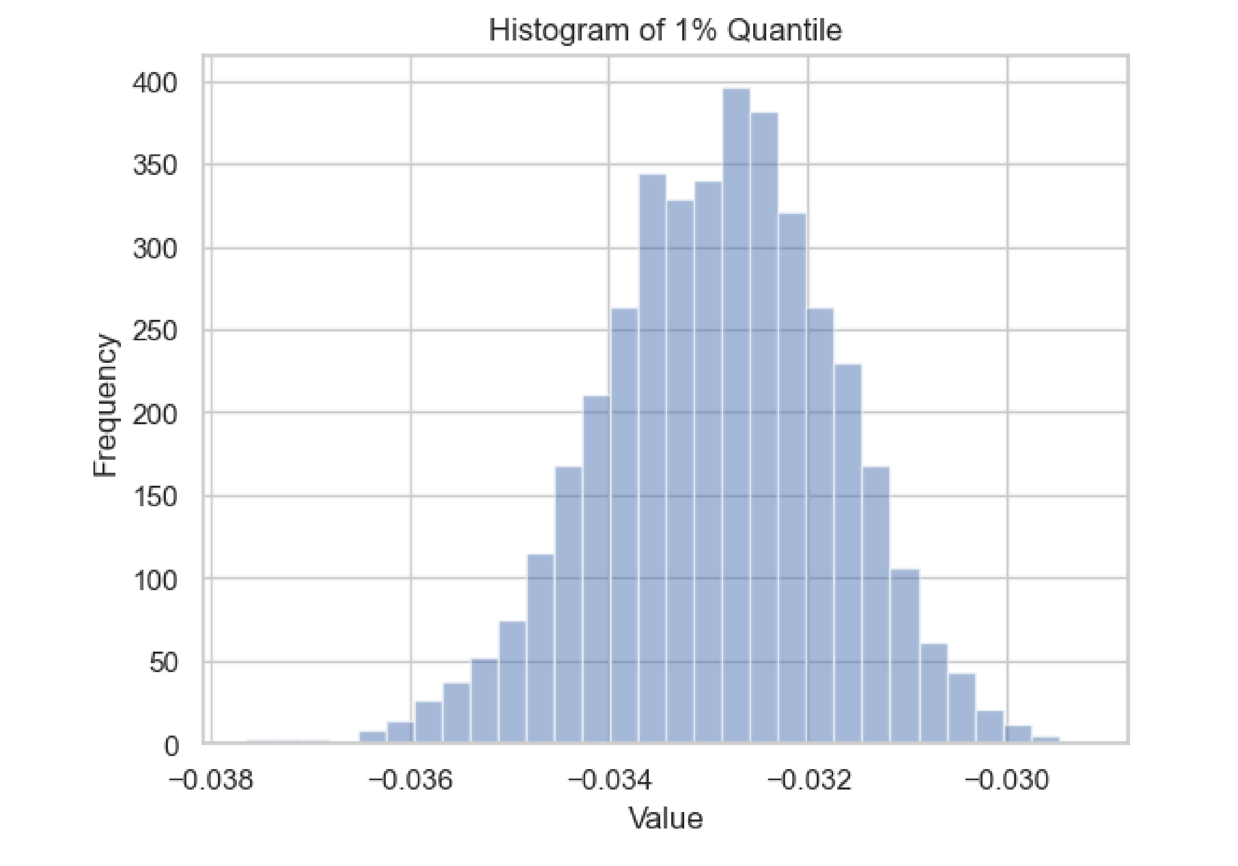
[31]:

	mu	sigma2
0	-0.001192	0.000179
1	-0.000943	0.000188
2	-0.001685	0.000180
3	-0.001862	0.000179
4	-0.002084	0.000178



Posterior distribution of VaR0.01

```
posterior_mu = trace.posterior["mu"].values.flatten()
posterior_sigma2= trace.posterior["sigma2"].values.flatten()
# Calculate 1% quantile
one_per = stats.norm.ppf(0.01, loc=posterior_mu,
scale=np.sqrt(posterior_sigma2))
# Plot histogram of 1% quantile
plt.hist(one_per, bins=30, density=False, alpha=0.5)
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram of 1% Quantile')
plt.grid(True)
plt.show()
```



USE THIS INFO FOR IN-CLASS ASSIGNMENT