Untitled

Metropolis-Hastings

- evaluate

$$X = x_i$$

  to get an initial value
- generate new value from a proposed distribution

$q(x_i + 1 \mid x_i)$

- compute the proability of accepting the new value

  $p_{a(x_1+1 \mid x_i)} \mid = \min\left(1, \frac{p(x_1+1)q(x_i \mid x_i+1)}{p(x_i)q(x_i+1 \mid x_i)}\right)$
- if $p_a > R$ where R~$U(0,1)$ save proposal else save old value
- iterate until you have n samples

Beta binomial

θ~$\mathrm{B}(\sigma, \beta)$

Y~ $\mathrm{Bin}(n = 1, p = \theta)$

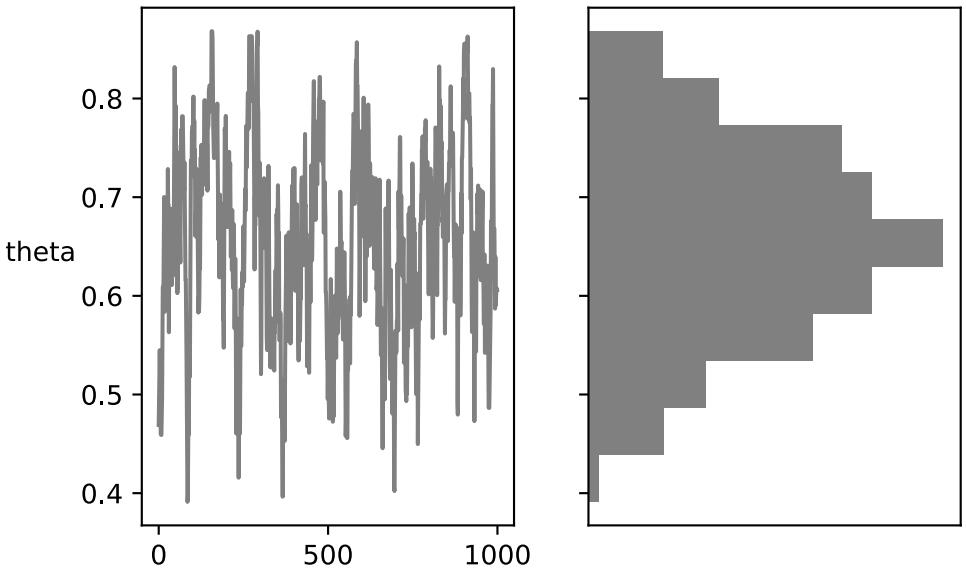
```
import scipy.stats as stats
import numpy as np


def post(theta, y , sigma =  1, beta = 1):
    if 0 <= theta <= 1:
        prior = stats.beta(sigma, beta).pdf(theta)
        like = stats.bernoulli(theta).pmf(Y).prod()
        prob = like * prior
    else:
        prob = -np.inf
    return prob

Y = stats.bernoulli(0.7).rvs(20)
```

```
n_iters = 1000
can_sd  = 0.05
sigma  = beta =  1
theta = 0.5
trace = {"theta": np.zeros(n_iters)}
```

```
p2 = post(theta, Y, sigma, beta)

for iter in range(n_iters):
    theta_can = stats.norm(theta, can_sd).rvs(1)
    p1 = post(theta_can, Y, sigma, beta)
    pa = p1/ p2

    if pa > stats.uniform(0,1).rvs(1):
        theta = theta_can
        p2 = p1

    trace["theta"][iter] = theta
```

```
/tmp/ipykernel_7848/274699115.py:17: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure
you extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  trace["theta"][iter] = theta
```

```
import matplotlib.pyplot as plt

_, axes = plt.subplots(1,2, sharey=True)
axes[0].plot(trace['theta'], '0.5')
axes[0].set_ylabel('theta', rotation=0, labelpad=15)
axes[1].hist(trace['theta'],       color='0.5',       orientation="horizontal",
density=True)
axes[1].set_xticks([])
```



```
import arviz as az

az.summary(trace, kind = "stats", round_to = 2)
```

	mean	sd	hdi_3%	hdi_97%
theta	0.65	0.1	0.46	0.81

```
# Basic
import numpy as np
from scipy import stats
import pandas as pd
from patsy import bs, dmatrix
import matplotlib.pyplot as plt

# Exploratory Analysis of Bayesian Models
import arviz as az

# Probabilistic programming languages
import bambi as bmb
import tensorflow_probability as tfp
import pymc as pm

tfd = tfp.distributions

# Computational Backend
import tensorflow as tf
```

```
2025-02-24 14:02:04.995260: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-02-24 14:02:05.397471: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1740423725.569094 7848 cuda_dnn.cc:8310] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered
E0000 00:00:1740423725.613705 7848 cuda_blas.cc:1418] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
2025-02-24 14:02:05.894947: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
```

```
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations,
rebuild TensorFlow with the appropriate compiler flags.
```

```
with pm.Model() as model:
    theta = pm.Beta('theta', alpha=1, beta=1)
    y_obs = pm.Binomial("y_obs", n=1, p=theta, observed=Y)
    idata= pm.sample(1000, return_inferencedata=True)
```

```
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [theta]
```

```
/home/kirvanlewis/projects/bmcp/.venv/lib/python3.11/site-packages/rich/
live.py:231: UserWarning: install
"ipywidgets" for Jupyter support
  warnings.warn('install "ipywidgets" for Jupyter support')
```

```

```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws
total) took 1 seconds.
```

Plot HDI for posterior samples

```{python

```