In [20]:

```
import numpy as np
import pymc as pm
import matplotlib.pyplot as plt
```

In [22]:

```
# Generate synthetic data
np.random.seed(42)
X = np.linspace(0, 10, 50)
true_intercept = 1
true_slope = 2
Y = true_intercept + true_slope * X + np.random.normal(scale=1, size=X.size)
```

In [24]:

```
# Bayesian linear regression model
with pm.Model() as model:
    # Priors for unknown model parameters
    intercept = pm.Normal("Intercept", mu=0, sigma=10)
    slope = pm.Normal("Slope", mu=0, sigma=10)
    sigma = pm.HalfNormal("Sigma", sigma=1)

    # Expected value of outcome
    Y_pred = intercept + slope * X

    # Likelihood of observations
    Y_obs = pm.Normal("Y_obs", mu=Y_pred, sigma=sigma, observed=Y)

    # Sampling posterior
    trace = pm.sample(2000, return_inferencedata=True, progressbar=True)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [Intercept, Slope, Sigma]
```

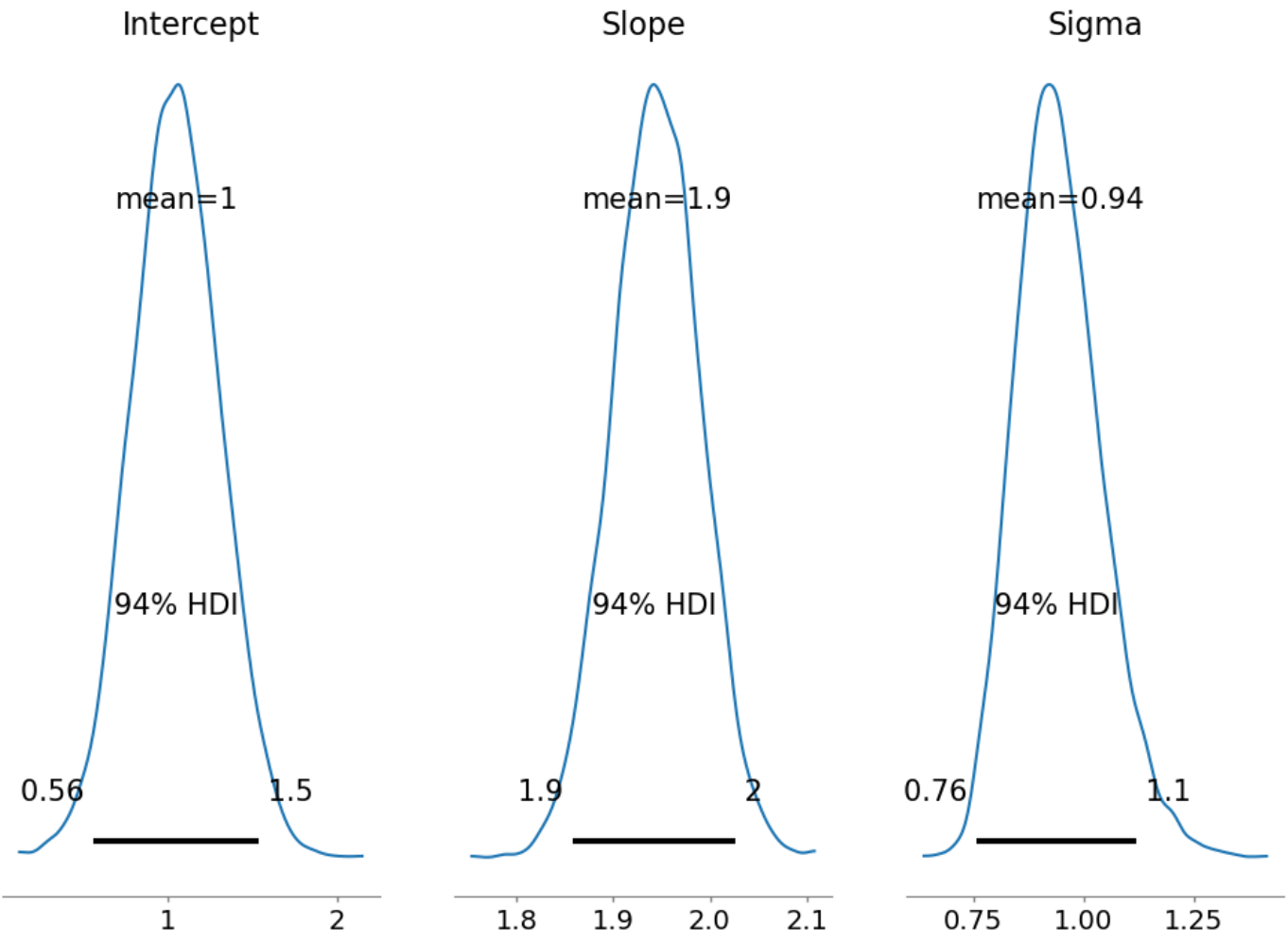
Output()

```
Sampling 4 chains for 1_000 tune and 2_000 draw iterations (4_000 + 8_000 draws total) t
ook 31 seconds.
```

In [25]:

```
# Plot posterior distributions
pm.plot_posterior(trace, figsize=(12, 8))
plt.show()

# Summary of the posterior
print(pm.summary(trace, hdi_prob=0.95))
```



	mean	sd	hdi_2.5%	hdi_97.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
Intercept	1.049	0.261	0.558	1.575	0.004	0.003	3622.0	3361.0	1.0
Slope	1.944	0.045	1.852	2.029	0.001	0.001	3662.0	3375.0	1.0
Sigma	0.945	0.098	0.760	1.136	0.001	0.001	4884.0	4498.0	1.0