PyMC

Adapted from PyMC tutorial notebooks

Example: Linear Regression

Model:

$$Y \sim \mathcal{N}(\mu, \sigma^2)$$
$$\mu = \alpha + \beta_1 X_1 + \beta_2 X_2$$

Priors:

$$\alpha \sim \mathcal{N}(0, 100)$$
$$\beta_i \sim \mathcal{N}(0, 100)$$
$$\sigma \sim |\mathcal{N}(0, 1)|$$

Example: Linear Regression

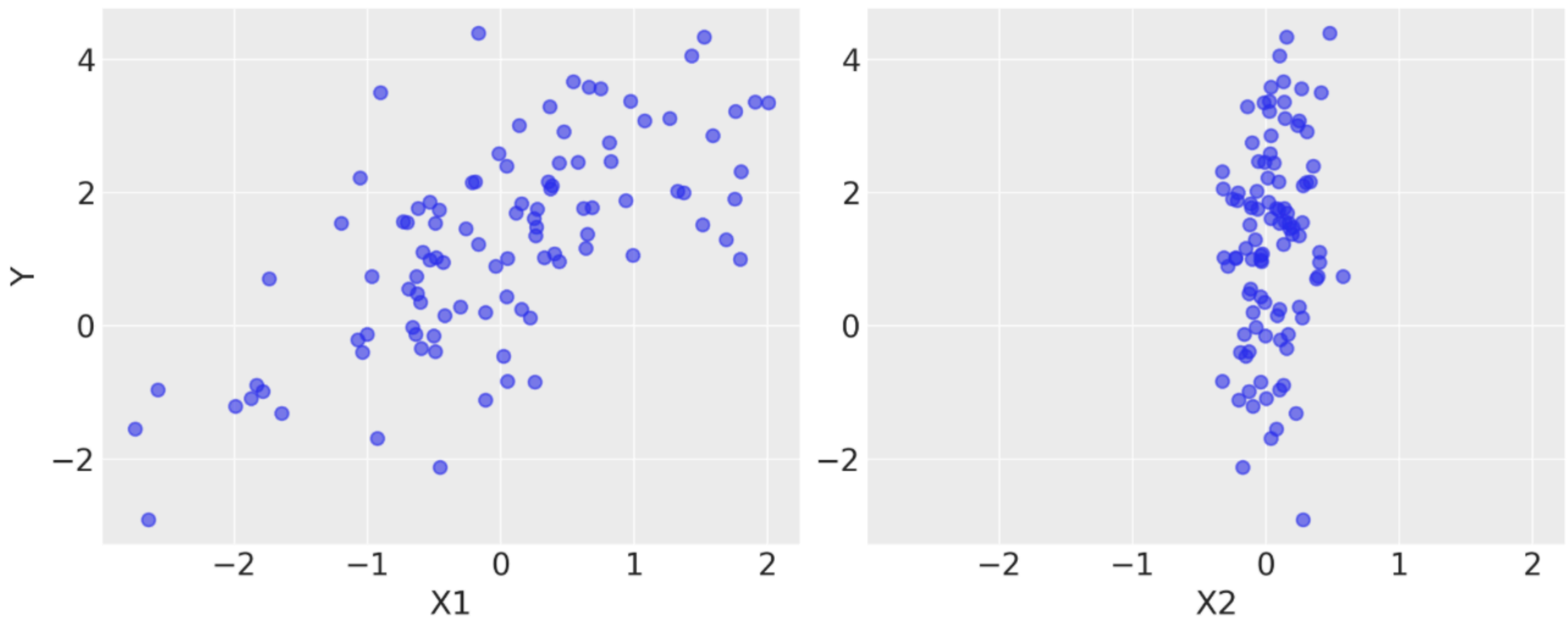
```python
import arviz as az
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```python
%config InlineBackend.figure_format = 'retina'
# Initialize random number generator
RANDOM_SEED = 8927
rng = np.random.default_rng(RANDOM_SEED)
az.style.use("arviz-darkgrid")
```

```python
# True parameter values
alpha, sigma = 1, 1
beta = [1, 2.5]

# Size of dataset
size = 100

# Predictor variable
X1 = np.random.randn(size)
X2 = np.random.randn(size) * 0.2

# Simulate outcome variable
Y = alpha + beta[0] * X1 + beta[1] * X2 + rng.normal(size=size) * sigma
```

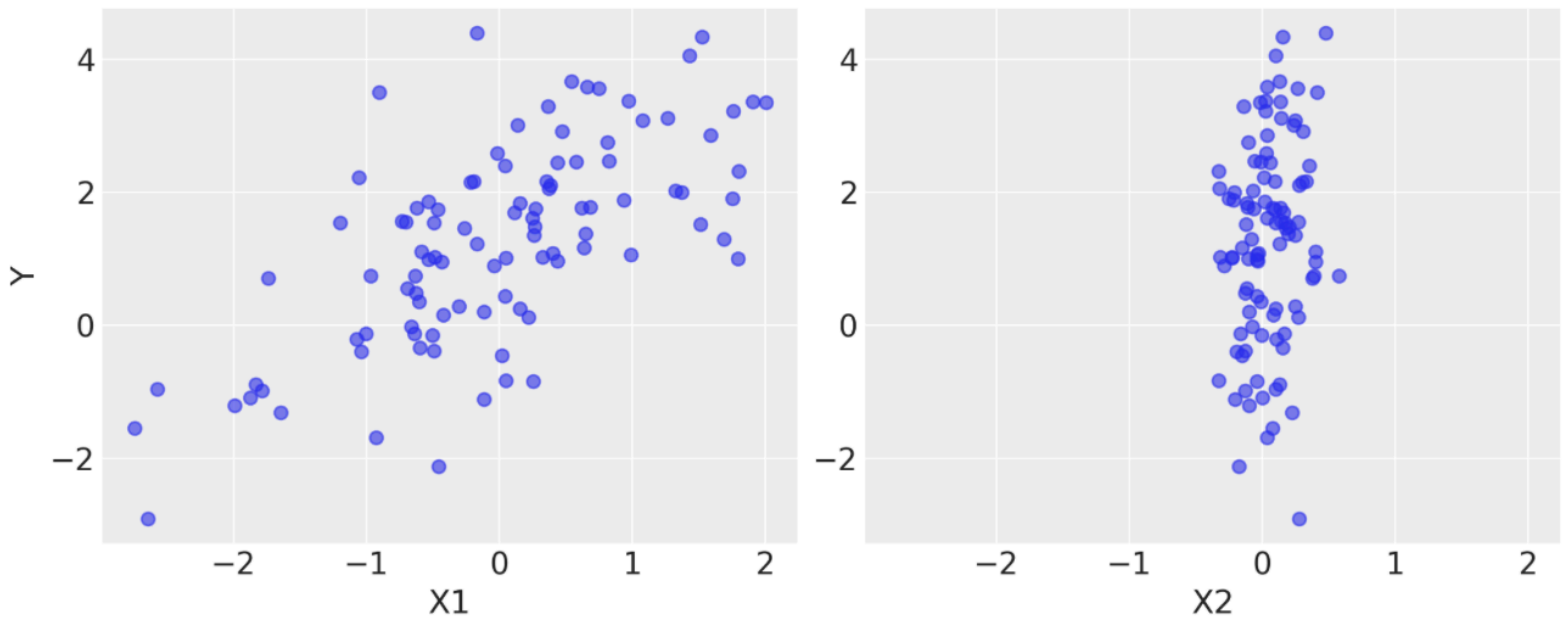
Example: Linear Regression

```
fig, axes = plt.subplots(1, 2, sharex=True, figsize=(10, 4))
axes[0].scatter(X1, Y, alpha=0.6)
axes[1].scatter(X2, Y, alpha=0.6)
axes[0].set_ylabel("Y")
axes[0].set_xlabel("X1")
axes[1].set_xlabel("X2");
```



Example: Linear Regression

```python
fig, axes = plt.subplots(1, 2, sharex=True, figsize=(10, 4))
axes[0].scatter(X1, Y, alpha=0.6)
axes[1].scatter(X2, Y, alpha=0.6)
axes[0].set_ylabel("Y")
axes[0].set_xlabel("X1")
axes[1].set_xlabel("X2");
```



Example: Linear Regression

```python
import pymc as pm

print(f"Running on PyMC v{pm.__version__}")
```

```python
basic_model = pm.Model()

with basic_model:
    # Priors for unknown model parameters
    alpha = pm.Normal("alpha", mu=0, sigma=10)
    beta = pm.Normal("beta", mu=0, sigma=10, shape=2)
    sigma = pm.HalfNormal("sigma", sigma=1)

    # Expected value of outcome
    mu = alpha + beta[0] * X1 + beta[1] * X2

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

Example: Linear Regression

```python
with basic_model:
    # draw 1000 posterior samples
    idata = pm.sample()
```

```
Auto-assigning NUTS sampler...
```

```
Initializing NUTS using jitter+adapt_diag...
```

```
Sequential sampling (2 chains in 1 job)
```

```
NUTS: [alpha, beta, sigma]
```

100.00% [2000/2000 00:01<00:00 Sampling chain 0, 0 divergences]

100.00% [2000/2000 00:01<00:00 Sampling chain 1, 0 divergences]

```
Sampling 2 chains for 1_000 tune and 1_000 draw iterations (2_000 + 2_000 d
```

```
We recommend running at least 4 chains for robust computation of convergenc
```

Example: Linear Regression

▾ posterior

xarray.Dataset

▸ Dimensions: (**chain**: 2, **draw**: 1000, **beta_dim_0**: 2)

▾ Coordinates:

chain	(chain)	int64	0 1
draw	(draw)	int64	0 1 2 3 4 5 ... 995 996 997 9
beta_dim_0	(beta_dim_0)	int64	0 1

▾ Data variables:

alpha	(chain, draw)	float64	1.203 1.072 1.157 ... 1.216 1.2
beta	(chain, draw, beta_dim_0)	float64	1.09 3.601 1.236 ... 0.9842 2.
sigma	(chain, draw)	float64	0.9528 1.297 ... 0.9366 0.93

▸ Indexes: (3)

▾ Attributes:

created_at :	2024-03-13T09:41:48.146524
arviz_version :	0.17.0
inference_libr...	pymc
inference_libr...	5.10.4+28.ga06081e.dirty
sampling_tim...	3.585268259048462
tuning_steps :	1000

Example: Linear Regression

▾ sample_stats

xarray.Dataset

▸ Dimensions: (**chain**: 2, **draw**: 1000)

▾ Coordinates:

chain	(chain)	int64	0 1
draw	(draw)	int64	0 1 2 3 4 5 ... 995 996 997 998 999

▸ Data variables: (17)

▸ Indexes: (2)

▾ Attributes:

created_at :	2024-03-13T09:41:48.159750
arviz_version :	0.17.0
inference_libr...	pymc
inference_libr...	5.10.4+28.ga06081e.dirty
sampling_tim...	3.585268259048462
tuning_steps :	1000

Example: Linear Regression

▾ Data variables:

process_time...	(chain, draw)	float64	0.0006175 0.0005986 ... 0.0005643
step_size_bar	(chain, draw)	float64	0.9347 0.9347 ... 0.9736 0.9736
n_steps	(chain, draw)	float64	3.0 3.0 3.0 1.0 ... 3.0 7.0 3.0 3.0
tree_depth	(chain, draw)	int64	2 2 2 1 2 2 2 2 ... 2 2 2 3 2 3 2 2
index_in_traje...	(chain, draw)	int64	-1 -2 2 1 -2 -3 -1 ... 1 1 3 3 2 0
largest_eigval	(chain, draw)	float64	nan nan nan nan ... nan nan nan nan
reached_max...	(chain, draw)	bool	False False False ... False False
step_size	(chain, draw)	float64	1.091 1.091 1.091 ... 0.8056 0.8056
lp	(chain, draw)	float64	-152.5 -158.6 ... -152.0 -152.0
energy	(chain, draw)	float64	153.8 160.3 159.8 ... 154.8 157.6
perf_counter...	(chain, draw)	float64	0.0006172 0.0005982 ... 0.0005639
acceptance_r...	(chain, draw)	float64	0.7693 0.552 ... 0.9812 0.4659
smallest_eigval	(chain, draw)	float64	nan nan nan nan ... nan nan nan nan
diverging	(chain, draw)	bool	False False False ... False False
energy_error	(chain, draw)	float64	0.3507 1.022 -1.162 ... -0.2658 0.0
max_energy_...	(chain, draw)	float64	0.4248 1.022 ... -0.2658 1.188
perf_counter...	(chain, draw)	float64	2.801e+04 2.801e+04 ... 2.801e+04

Example: Linear Regression

▾ observed_data

xarray.Dataset

▸ Dimensions: (**Y_obs_dim_0**: 100)

▾ Coordinates:

Y_obs_dim_0	(Y_obs_dim_0)	int64	0 1 2 3 4 5 6 ... 94 95 96 97 98 99

▾ Data variables:

Y_obs	(Y_obs_dim_0)	float64	-1.091 1.548 ... 1.615 2.402

▸ Indexes: (1)

▾ Attributes:

created_at :	2024-03-13T09:41:48.165581
arviz_version :	0.17.0
inference_libr...	pymc
inference_libr...	5.10.4+28.ga06081e.dirty

Example: Linear Regression

```
idata.posterior["alpha"].sel(draw=slice(0, 4))
```

xarray.DataArray 'alpha' (**chain**: 2, **draw**: 5)

```
array([[1.2032439 , 1.07161394, 1.15694926, 1.10665511, 1.18414293],
       [1.00368682, 1.13578702, 1.18218925, 1.28672868, 1.0686125 ]])
```

▼ Coordinates:

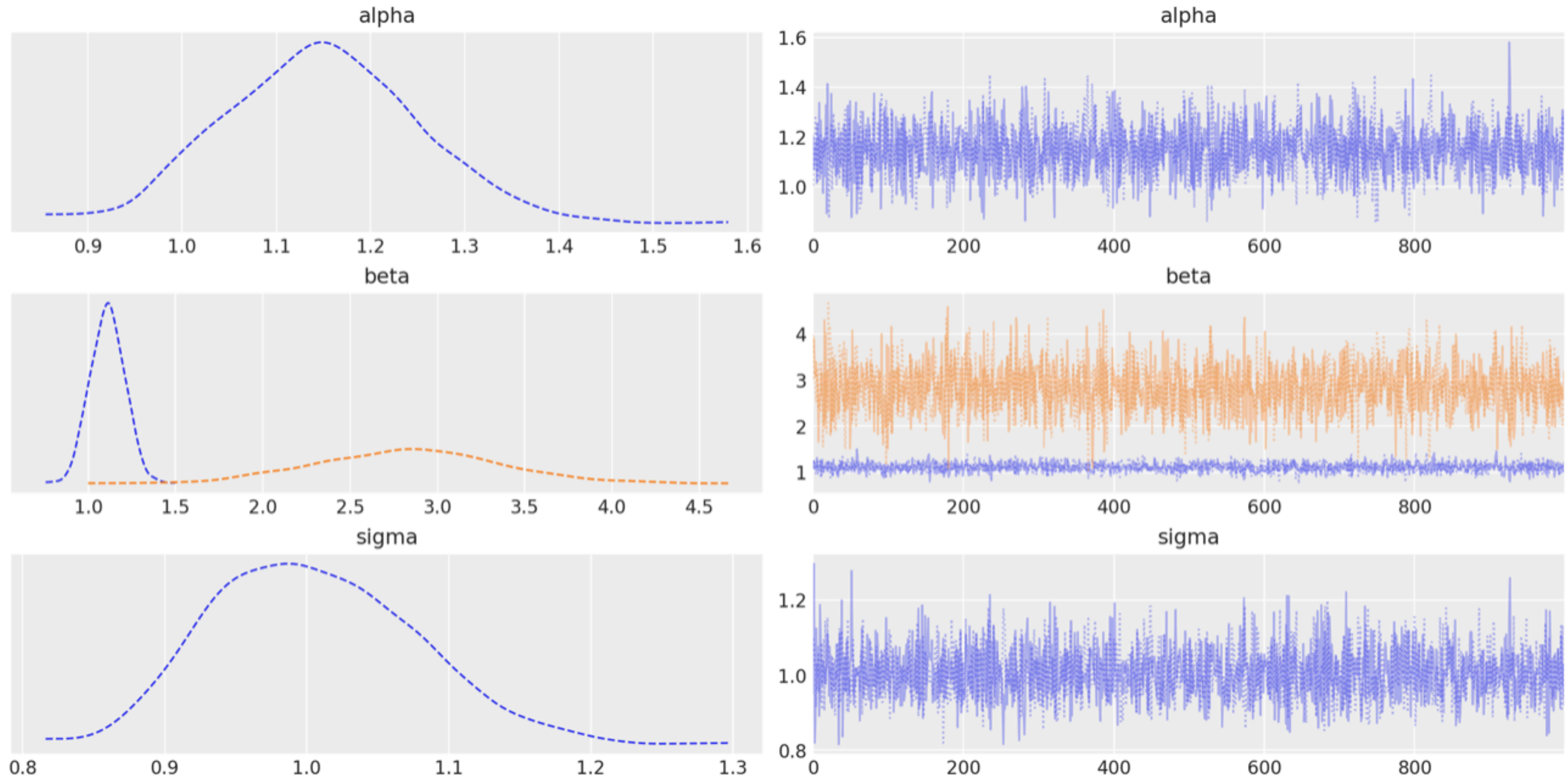
chain	(chain)	int64	0 1
draw	(draw)	int64	0 1 2 3 4

▶ Indexes: (2)

▶ Attributes: (0)

Example: Linear Regression

```
az.plot_trace(idata, combined=True);
```

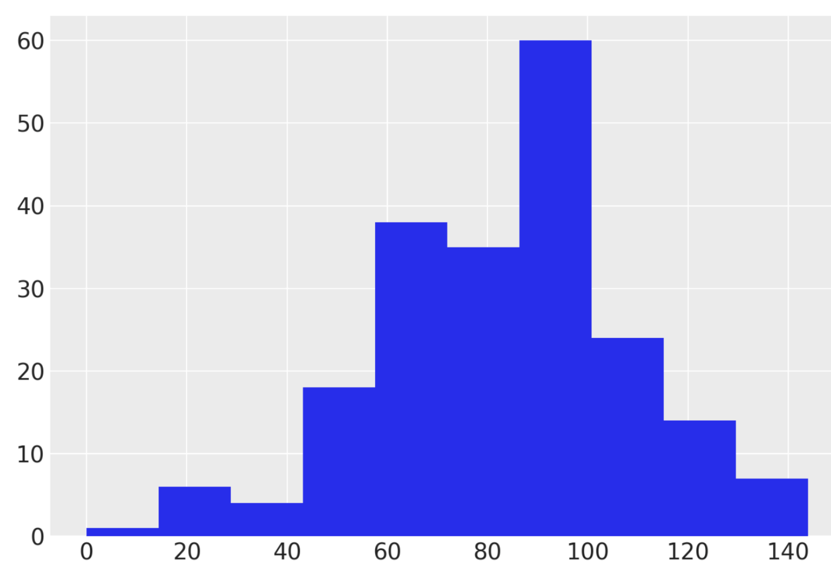


Educational Outcomes for Hearing-impaired Children

```
test_scores = pd.read_csv(pm.get_data("test_scores.csv"), index_col=0)
test_scores.head()
```

	score	male	siblings	family_inv	non_english	prev_disab	age_test	non_severe_hl	mother_hs	early_ident	non_white
0	40	0	2.0	2.0	False	NaN	55	1.0	NaN	False	False
1	31	1	0.0	NaN	False	0.0	53	0.0	0.0	False	False
2	83	1	1.0	1.0	True	0.0	52	1.0	NaN	False	True
3	75	0	3.0	NaN	False	0.0	55	0.0	1.0	False	False
5	62	0	0.0	4.0	False	1.0	50	0.0	NaN	False	False

```
test_scores["score"].hist();
```



Educational Outcomes for Hearing-impaired Children

```python
# Dropping missing values is a very bad idea in general, but we do so here
X = test_scores.dropna().astype(float)
y = X.pop("score")

# Standardize the features
X -= X.mean()
X /= X.std()

N, D = X.shape
```