Markov Chain Monte Carlo and model fitting

BAYESIAN DATA ANALYSIS IN PYTHON

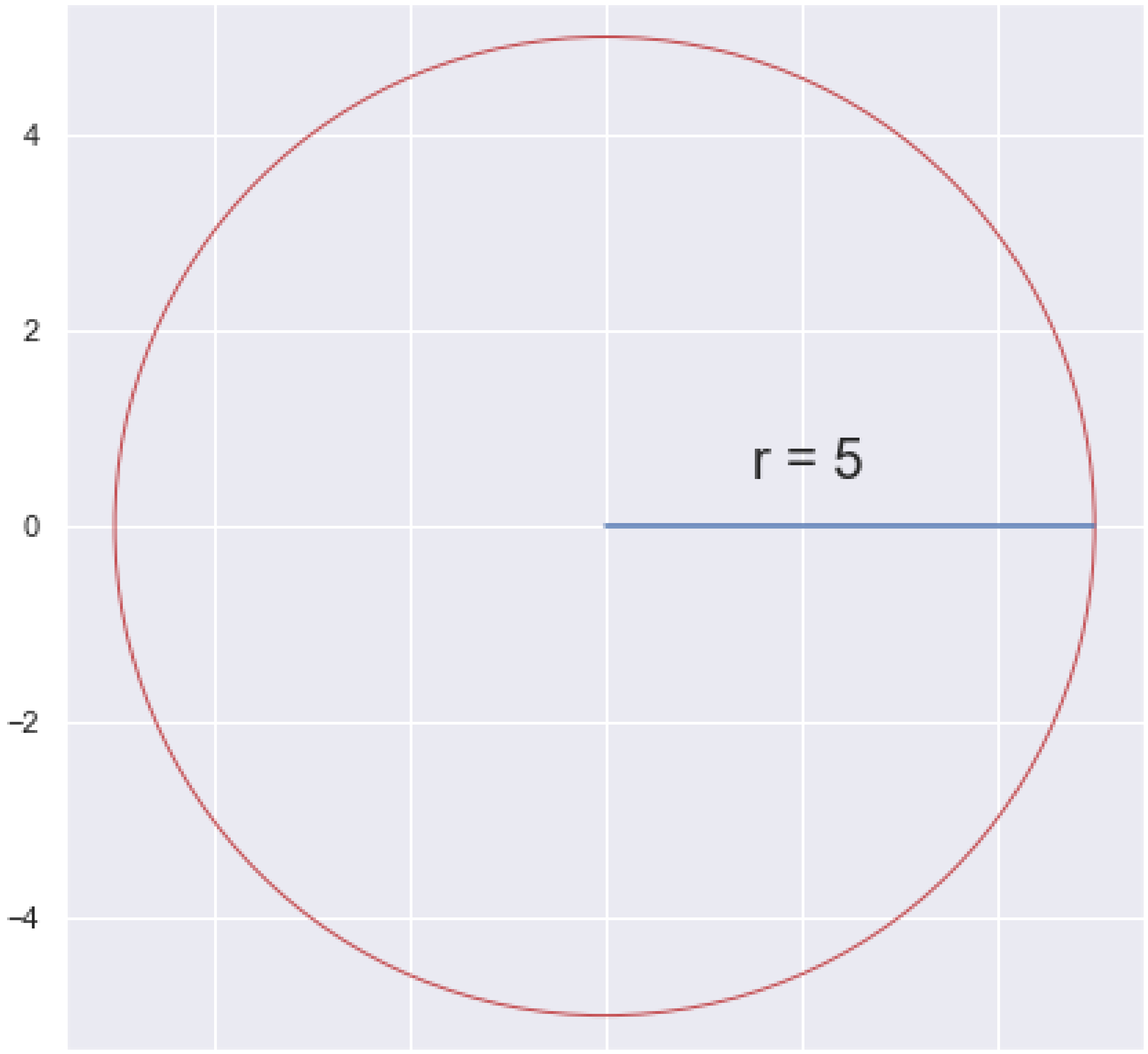
Michal Oleszak
Machine Learning Engineer

Bayesian data analysis in production

- Grid approximation: inconvenient with many parameters
- Sampling from known posterior: requires conjugate priors
- Markov Chain Monte Carlo (MCMC): sampling from unknown posterior!

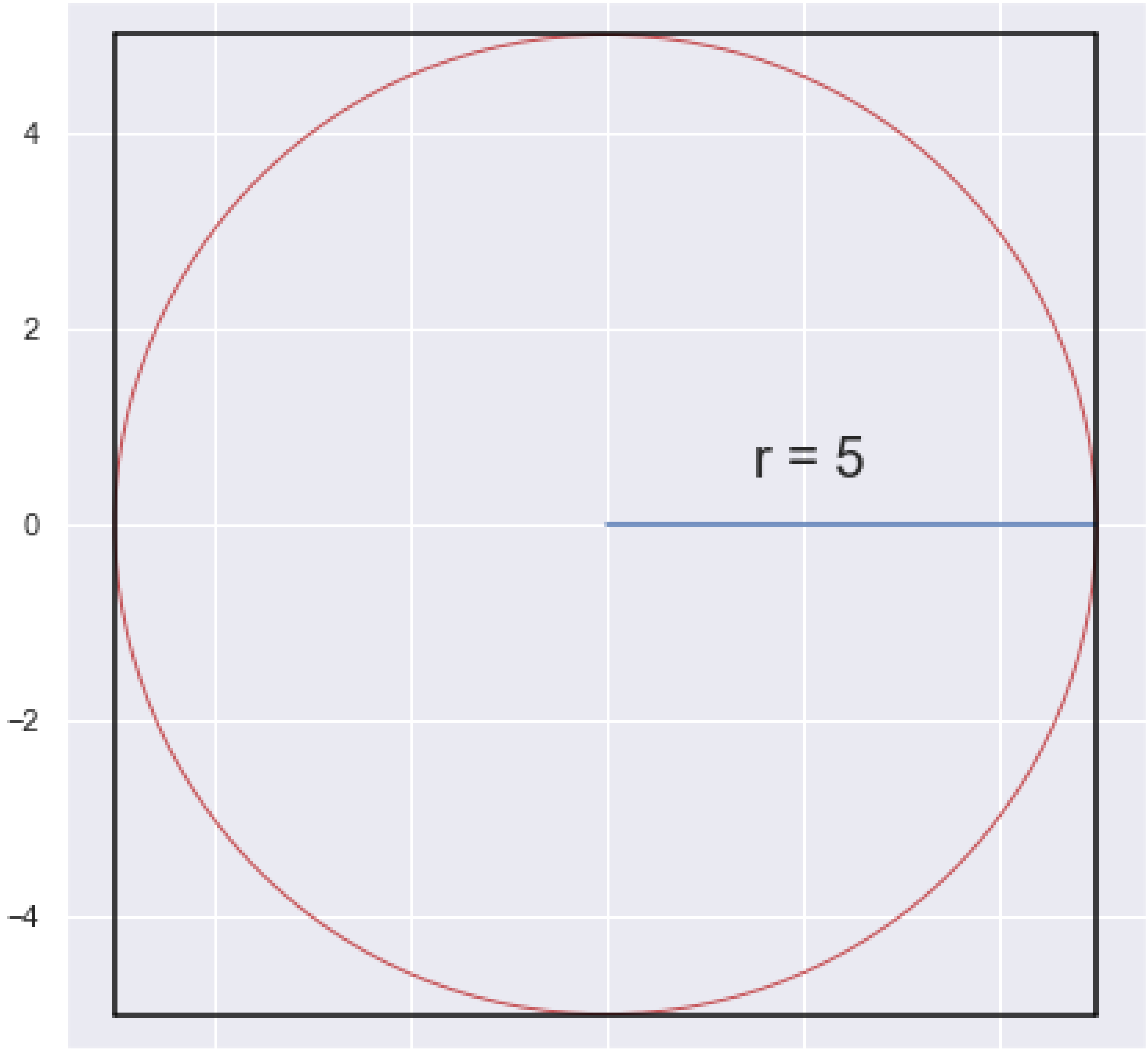
Monte Carlo

- Approximating some quantity by generating random numbers
- From the formula, $\pi r^2 \simeq 78.5$



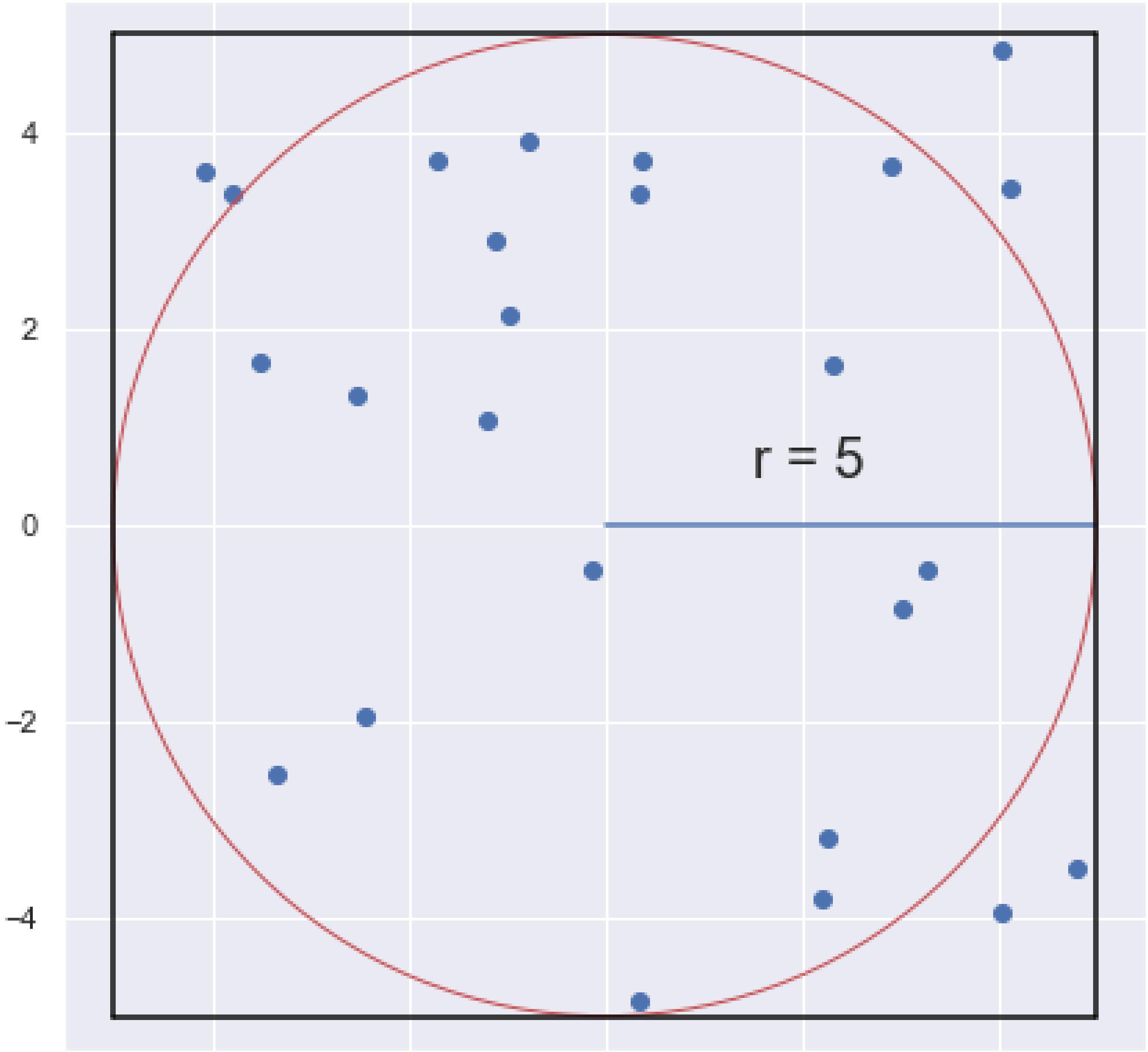
Monte Carlo

- Approximating some quantity by generating random numbers
- From the formula, $\pi r^2 \simeq 78.5$
- Draw a 10x10 square around the circle.



Monte Carlo

- Approximating some quantity by generating random numbers
- From the formula, $\pi r^2 \simeq 78.5$
- Draw a 10x10 square around the circle.
- Sample 25 random points in the square.
- How many are within the circle?
  $19/25 = 76\%$
- Circle's area approximation: 76% * 100 = 76



Markov Chains

- Models a sequence of states, between which one transitions with given probabilities.

Markov Chains

- Models a sequence of states, between which one transitions with given probabilities.
- After many time periods, transition probabilities become the same no matter where we started.

What will the bear do next:

	hunt	**eat**	**sleep**
hunt	0.1	0.8	0.1
eat	0.05	0.4	0.55
sleep	0.8	0.15	0.05

Markov Chains

- Models a sequence of states, between which one transitions with given probabilities.
- After many time periods, transition probabilities become the same no matter where we started.

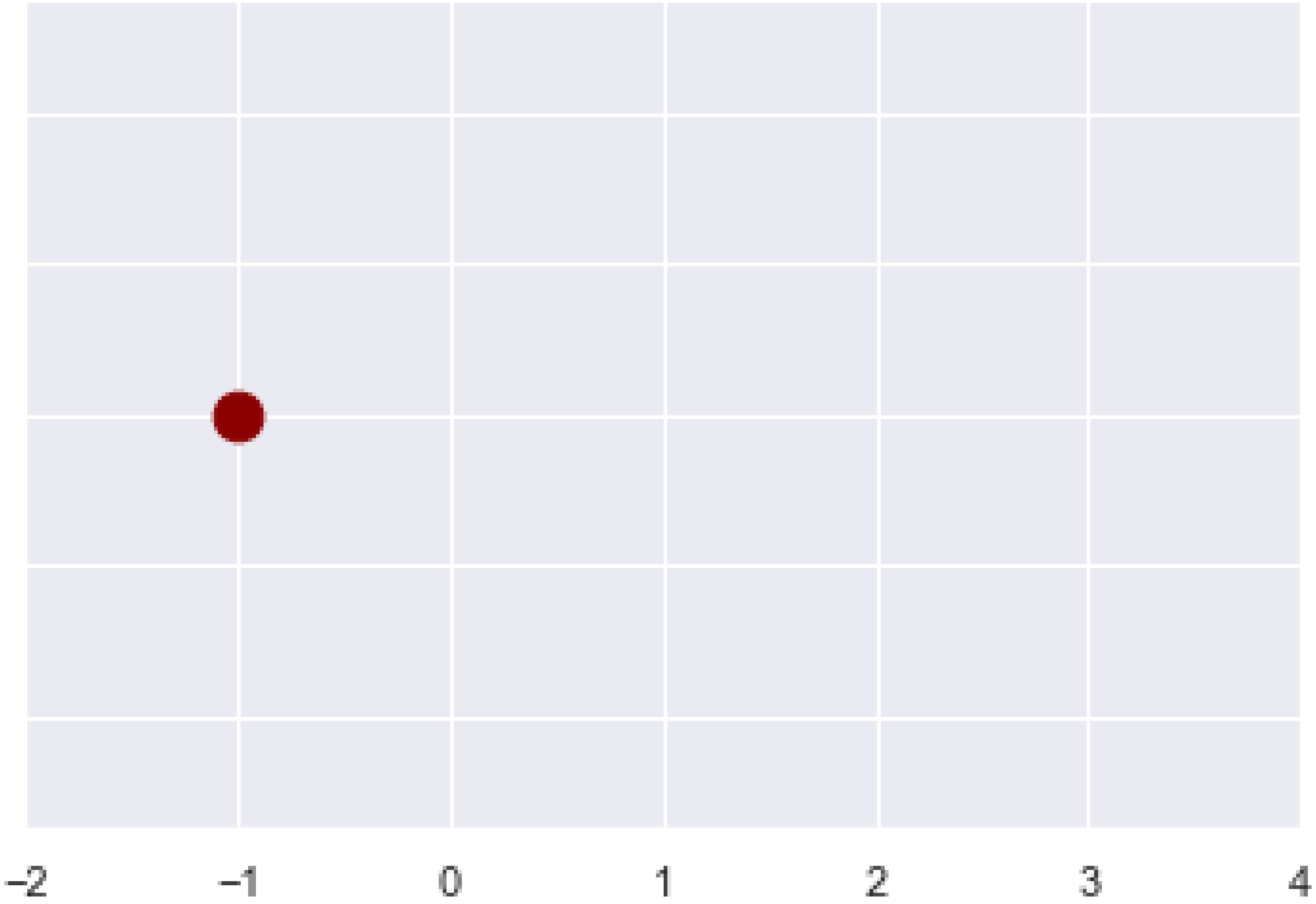
What will the bear do next:

	hunt	eat	sleep
hunt	0.1	0.8	0.1
eat	0.05	0.4	0.55
sleep	0.8	0.15	0.05

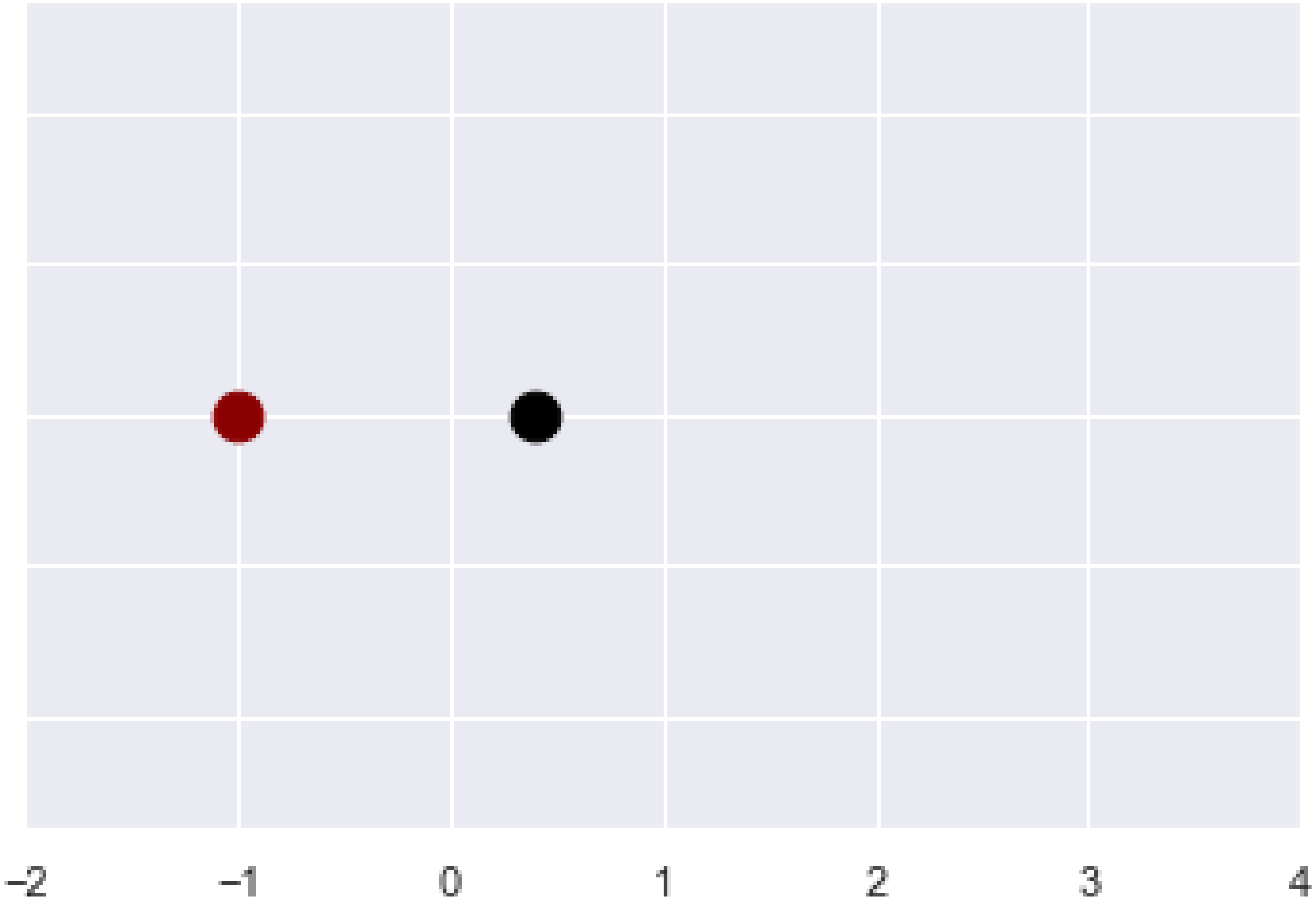
What will the bear do in a distant future:

	hunt	eat	sleep
hunt	0.28	0.44	0.28
eat	0.28	0.44	0.28
sleep	0.28	0.44	0.28

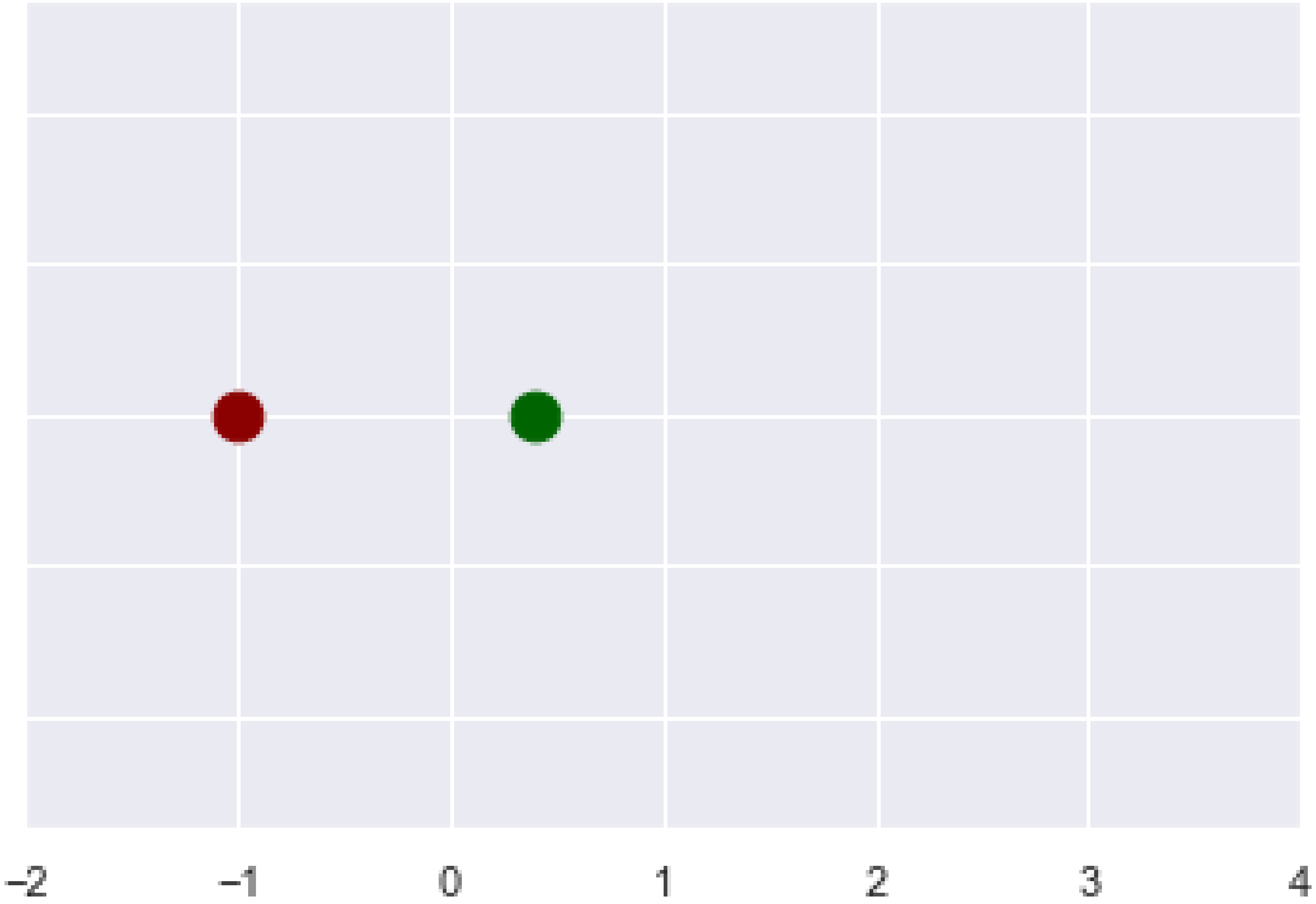
Markov Chain Monte Carlo



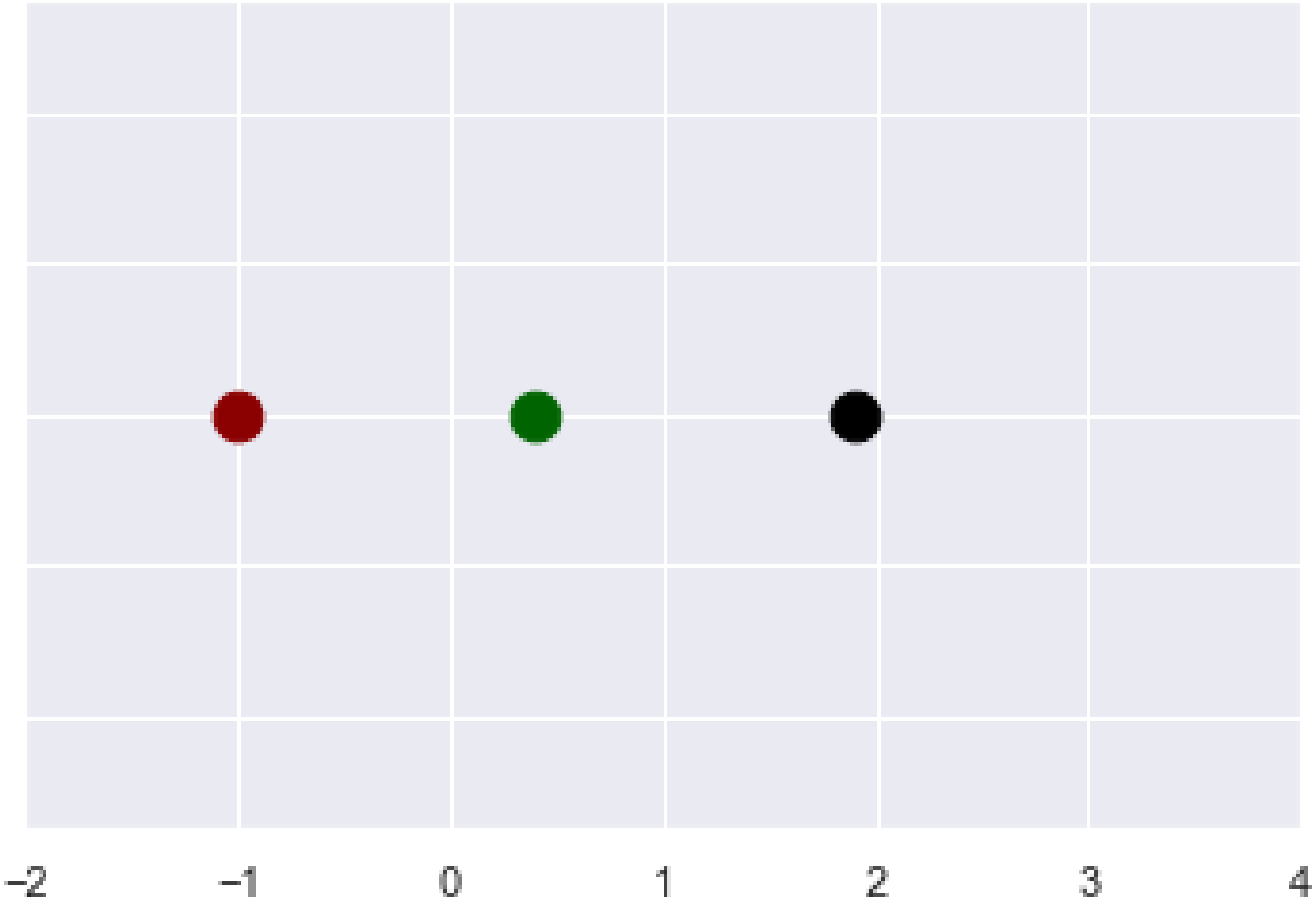
Markov Chain Monte Carlo



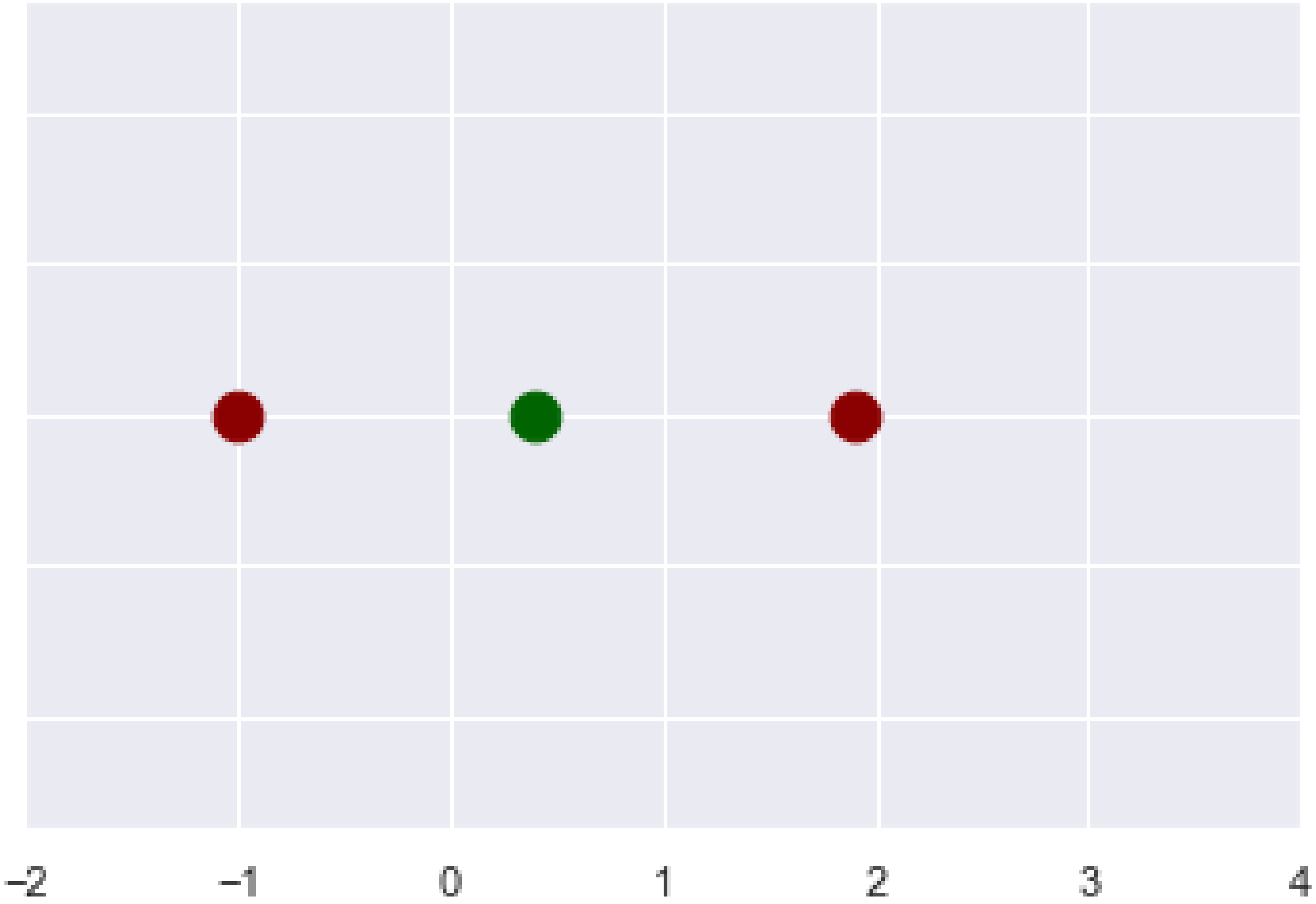
Markov Chain Monte Carlo



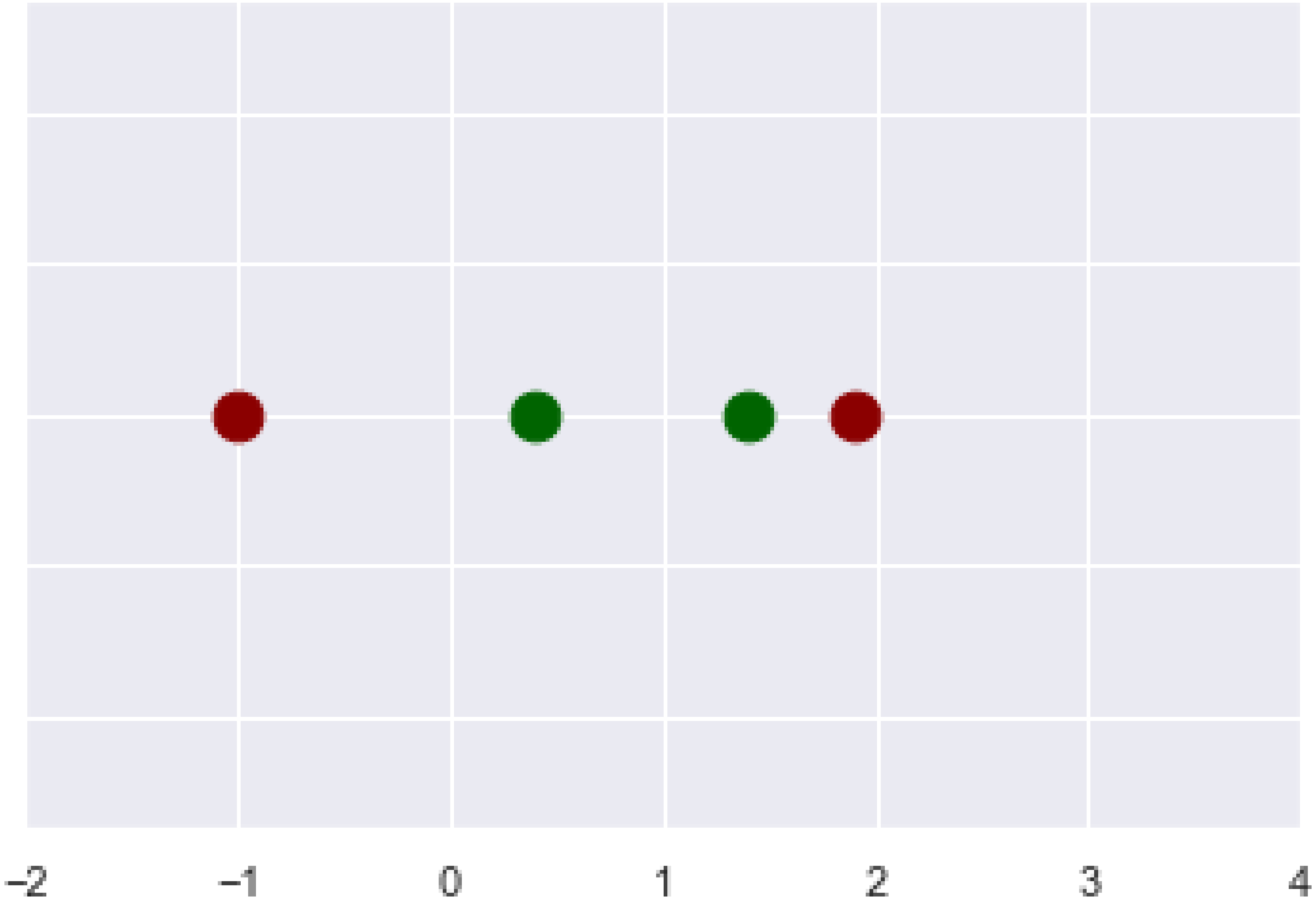
Markov Chain Monte Carlo



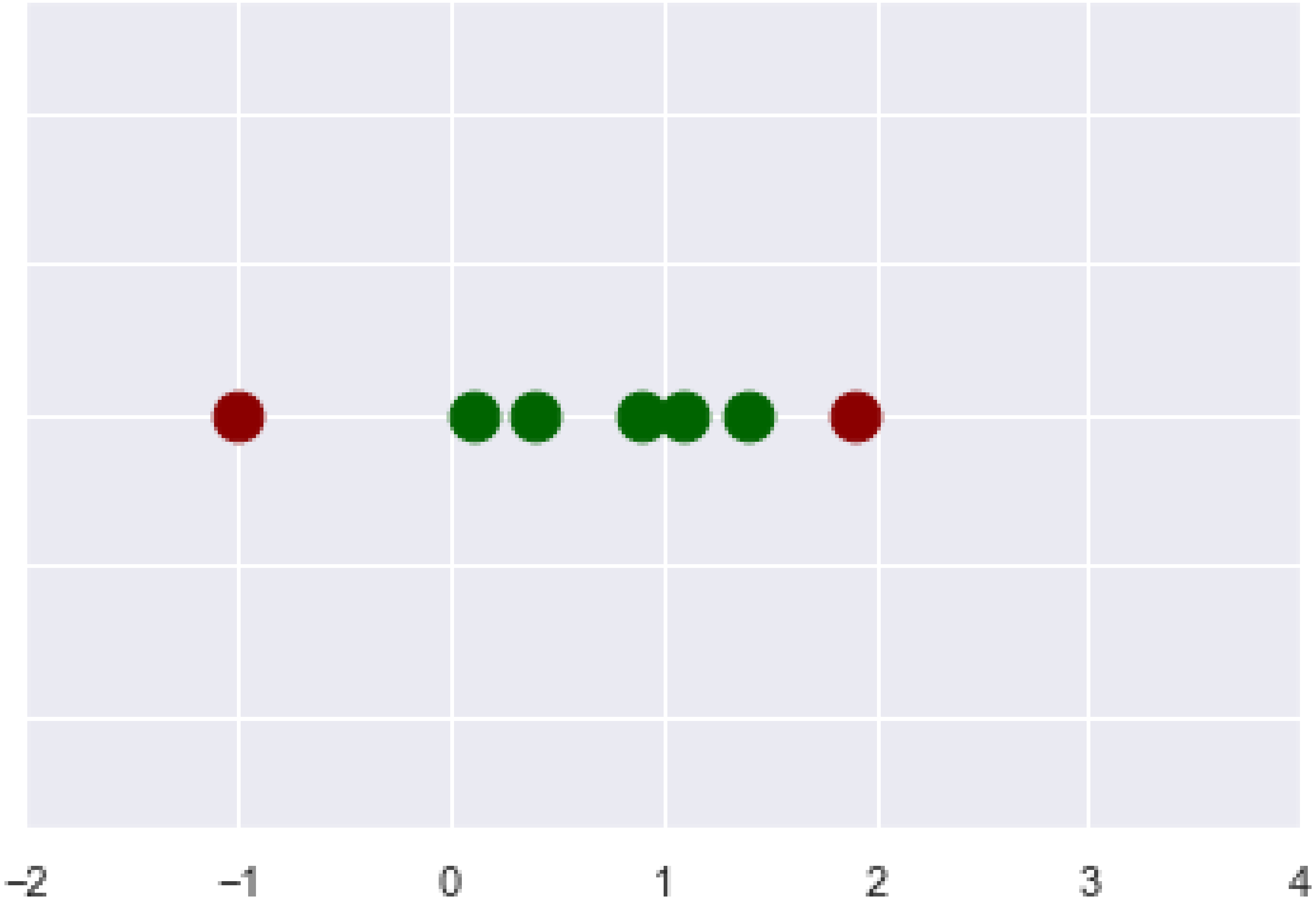
Markov Chain Monte Carlo



Markov Chain Monte Carlo



Markov Chain Monte Carlo



Markov Chain Monte Carlo

Aggregated ads data

```
print(ads_aggregated)
```

```
           date  clothes_banners_shown  sneakers_banners_shown  num_clicks
0    2019-01-01                     20                      18           2
1    2019-01-02                     24                      19           8
2    2019-01-03                     20                      20           5
..          ...                    ...                     ...         ...
148  2019-05-29                     24                      25           8
149  2019-05-30                     26                      27          11
150  2019-05-31                     26                      24           8

[151 rows x 4 columns]
```

Linear regression with pyMC3

```
formula = "num_clicks ~ clothes_banners_shown + sneakers_banners_shown"

with pm.Model() as model:
    pm.GLM.from_formula(formula, data=ads_aggregated)
    # Print model specification
    print(model)
    # Sample posterior draws
    trace = pm.sample(draws=1000, tune=500)
```

```
             Intercept ~ Flat
 clothes_banners_shown ~ Normal
sneakers_banners_shown ~ Normal
               sd_log__ ~ TransformedDistribution
                     sd ~ HalfCauchy
                      y ~ Normal
```

Let's practice MCMC!

BAYESIAN DATA ANALYSIS IN PYTHON

Interpreting results and comparing models

BAYESIAN DATA ANALYSIS IN PYTHON

Michal Oleszak
Machine Learning Engineer

Running the model revisited

```
formula = "num_clicks ~ clothes_banners_shown + sneakers_banners_shown"

with pm.Model() as model_1:
    pm.GLM.from_formula(formula, data=ads_aggregated)
    trace_1 = pm.sample(draws=1000, tune=500)
```

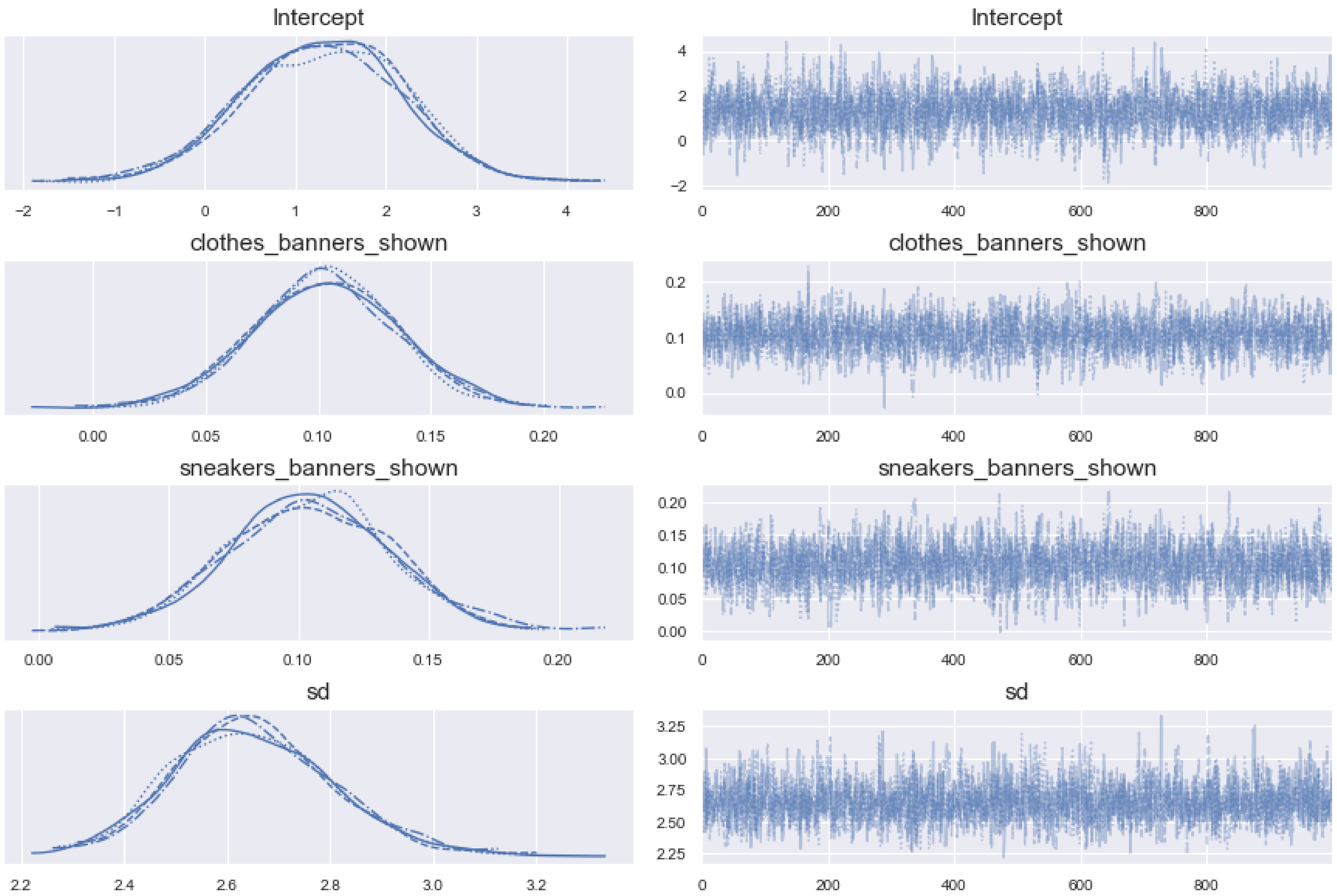
Running the model revisited

```
formula = "num_clicks ~ clothes_banners_shown + sneakers_banners_shown"

with pm.Model() as model_1:
    pm.GLM.from_formula(formula, data=ads_aggregated)
    trace_1 = pm.sample(draws=1000, tune=500, chains=4)
```

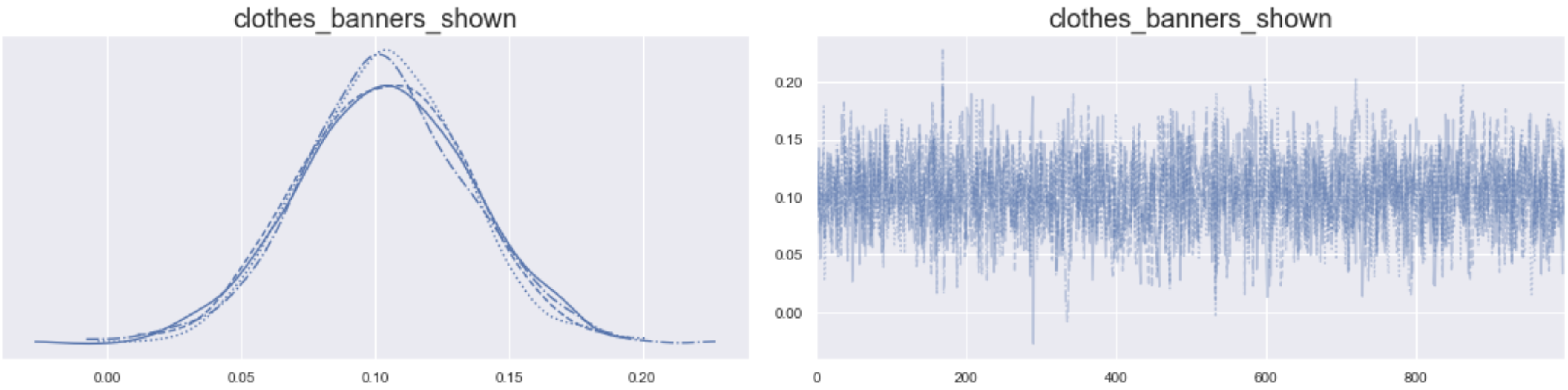
- Number of parameters: 4
- Number of draws for each parameter: $1000 \times 4 = 4000$

Trace plot

```
pm.traceplot(trace_1)
```

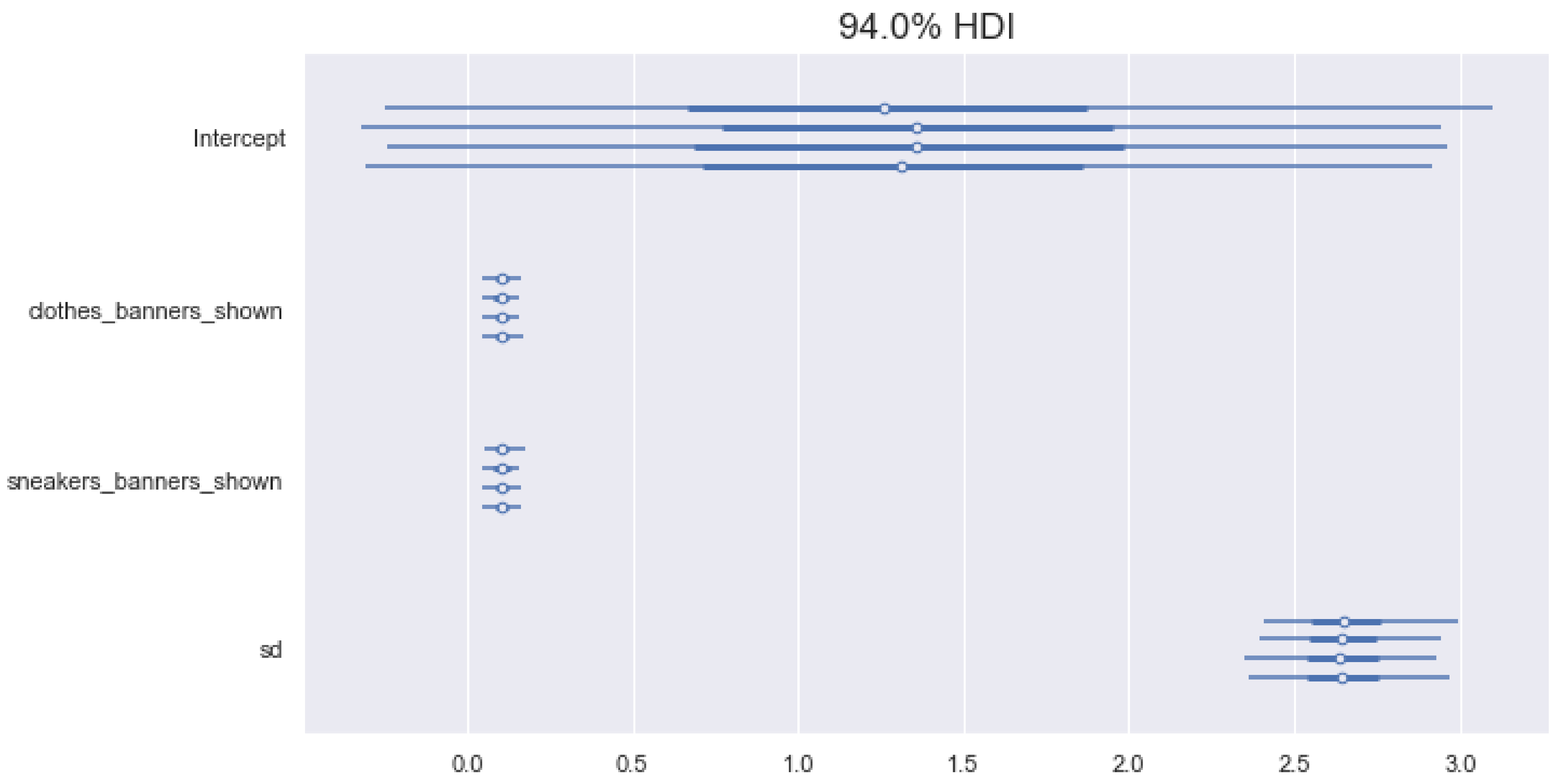


Trace plot: zoom in on one parameter



Forest plot

```
pm.forestplot(trace_1)
```



Trace summary

```
pm.summary(trace_1)
```

```
                        mean     sd  hdi_3%  hdi_97%  mcse_mean  mcse_sd  \
Intercept              1.307  0.886  -0.305    2.962      0.018    0.013
clothes_banners_shown  0.103  0.031   0.043    0.160      0.001    0.000
sneakers_banners_shown 0.104  0.032   0.045    0.163      0.001    0.001
sd                     2.654  0.157   2.382    2.970      0.003    0.002

                        ess_mean  ess_sd  ess_bulk  ess_tail  r_hat
Intercept                 2346.0  2318.0    2351.0    2083.0    1.0
clothes_banners_shown     2085.0  2085.0    2089.0    1868.0    1.0
sneakers_banners_shown    2105.0  1953.0    2122.0    1869.0    1.0
sd                        2615.0  2590.0    2646.0    1834.0    1.0
```

Fitting another model

```
formula = "num_clicks ~ clothes_banners_shown + sneakers_banners_shown + weekend"

with pm.Model() as model_2:
    pm.GLM.from_formula(formula, data=ads_aggregated)
    trace_2 = pm.sample(draws=1000, tune=500)
```

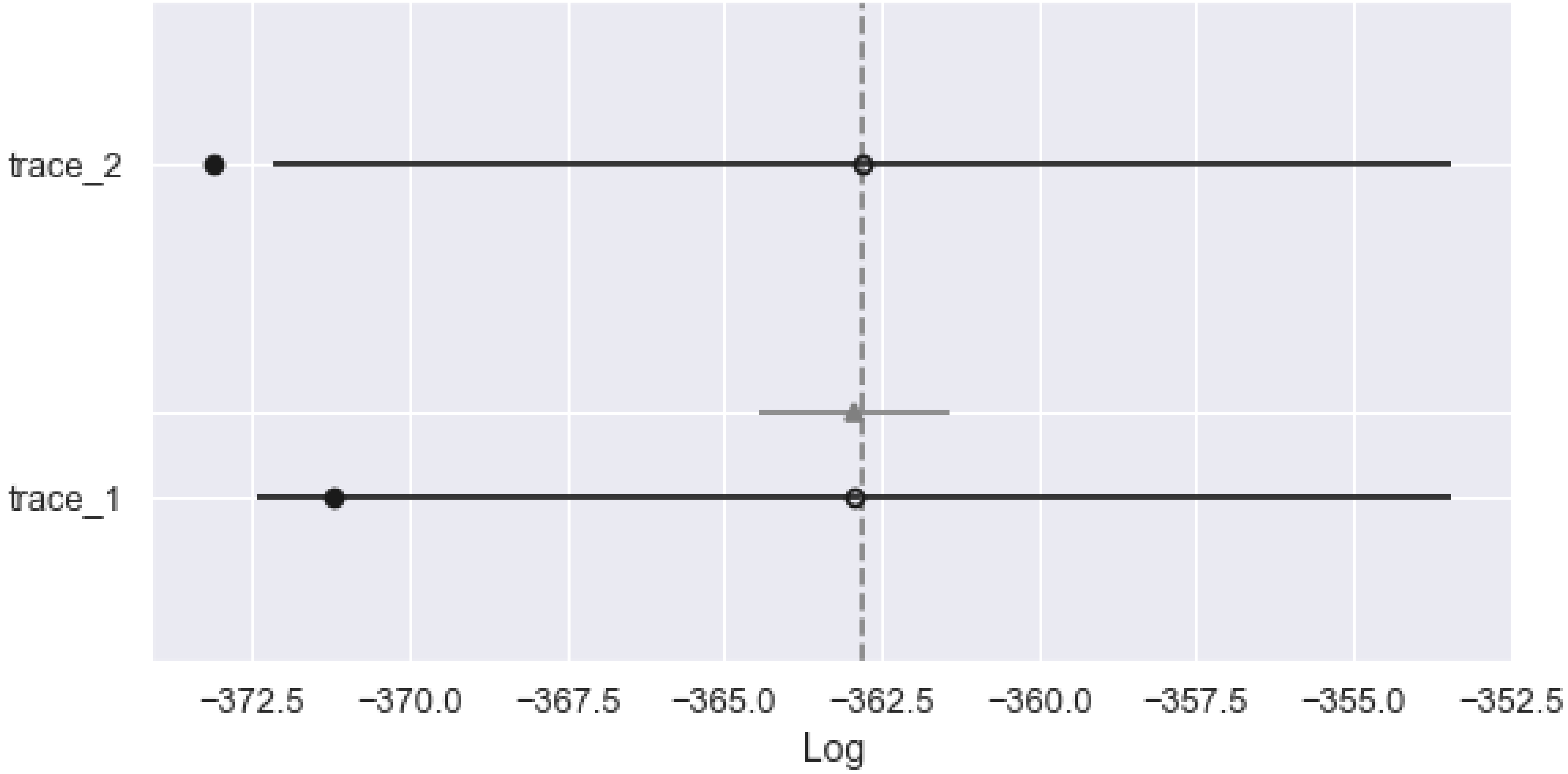
Widely Applicable Information Criterion (WAIC)

```
comparison = pm.compare({"trace_1": trace_1, "trace_2": trace_2},
                        ic="waic", scale="deviance")
print(comparison)
```

```
        rank     waic   p_waic    d_waic    weight       se      dse warning  \
trace_2    0   -362.8   5.1576         0  0.513792  9.37269        0    True
trace_1    1 -362.926  4.13318  0.126236  0.486208  9.48352  1.50682    True

        waic_scale
trace_2        log
trace_1        log
```

Compare plot

```
pm.compareplot(comparison)
```



Let's practice comparing models!

BAYESIAN DATA ANALYSIS IN PYTHON

Making predictions

BAYESIAN DATA ANALYSIS IN PYTHON

Michal Oleszak
Machine Learning Engineer

Number-of-clicks model again

```
formula = "num_clicks ~ clothes_banners_shown + sneakers_banners_shown + weekend"

with pm.Model() as model_2:
    pm.GLM.from_formula(formula, data=ads_aggregated)
    trace_2 = pm.sample(draws=1000, tune=500)
```

Ads test data

```
print(ads_test)
```

```
   clothes_banners_shown  sneakers_banners_shown  num_clicks  weekend
0                     40                      36           7     True
1                     42                      47           8    False
2                     45                      37          11    False
3                     22                      15           4    False
4                     20                      18           2    False
```

Sampling predictive draws

```
with pm.Model() as model:
    pm.GLM.from_formula(formula, data=ads_test)
    posterior_predictive = pm.fast_sample_posterior_predictive(trace_2)
```

Predictive draws

```
posterior_predictive["y"].shape
```

```
(4000, 5)
```

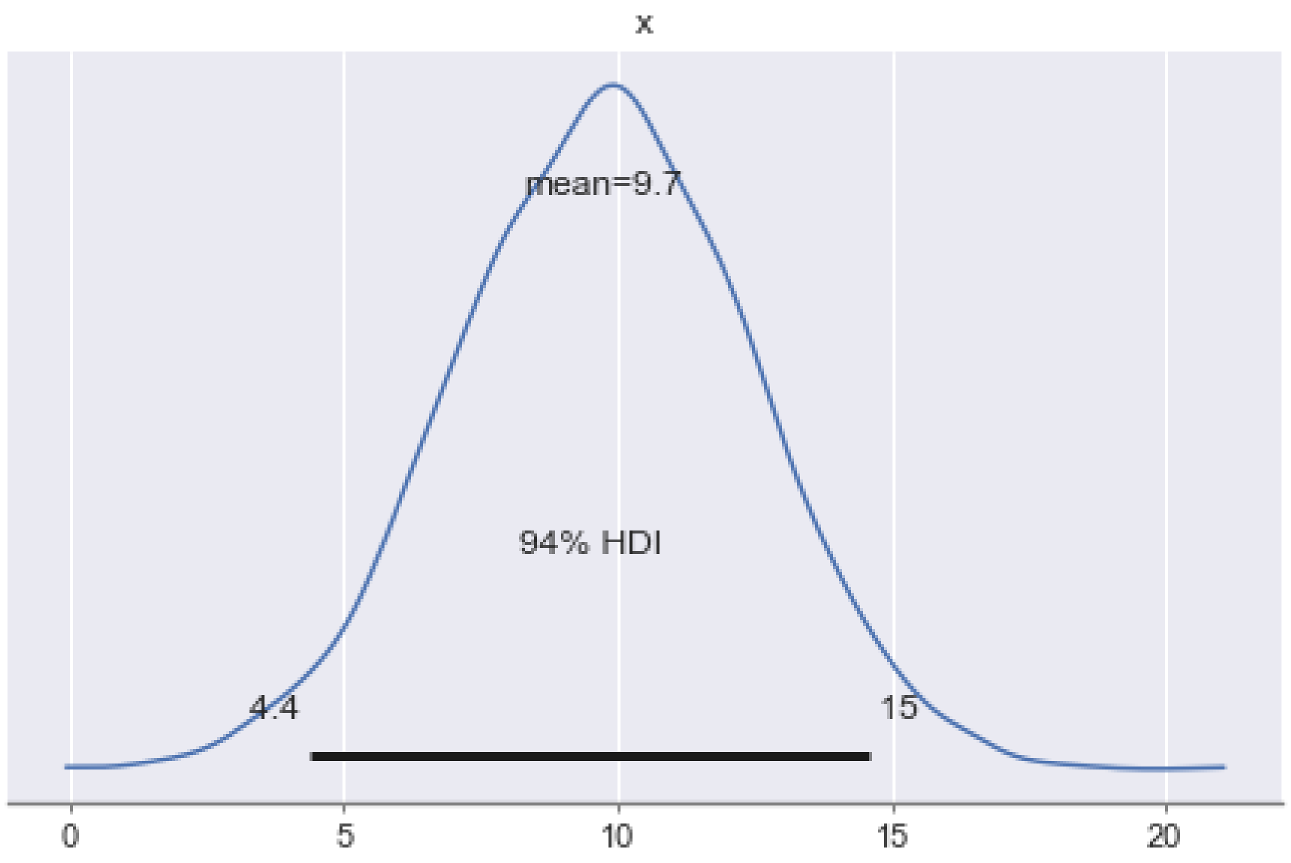
```
print(posterior_predictive["y"])
```

```
array([[12.83527253, 10.22454815, 11.20386868,  7.50227286,  6.85458594],
       [ 3.1015655 ,  6.1253004 , 11.38324931,  2.1844722 ,  4.21451756],
       [ 3.40141276,  9.10157964,  6.57689421,  8.26669814,  4.23812161],
       ...,
       [10.97303606,  9.0772305 , 10.6877039 ,  1.78448969,  6.75663075],
       [ 8.53734584, 12.14079593, 11.00969881,  4.69875055,  8.317338  ],
       [16.44713387, 17.35163824, 19.59359831,  2.84058536,  4.21108186]])
```

How good is the prediction?

```
   clothes_banners_shown  sneakers_banners_shown  num_clicks  weekend
0                     40                      36           7     True
```

```
pm.plot_posterior(posterior_predictive["y"][:, 0])
```



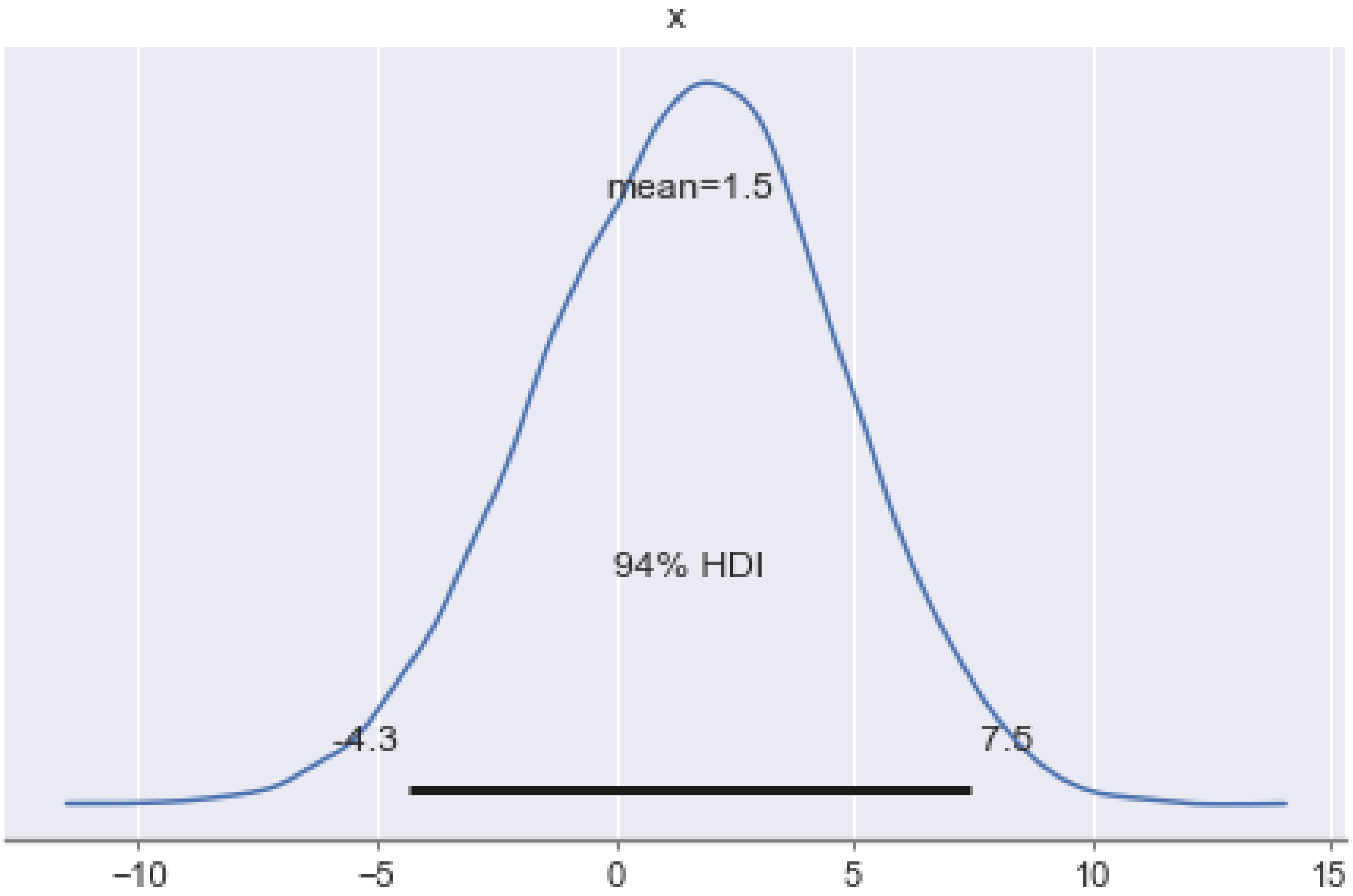
Test error distribution

```
errors = []

for index, test_example in ads_test.iterrows():
    error = posterior_predictive["y"][:, index] - test_example["num_clicks"]
    errors.append(error)

error_distribution = np.array(errors).reshape(-1)
error_distribution.shape
```

```
(20000,)
```

```
pm.plot_posterior(error_distribution)
```

Test error distribution



Let's make predictions!

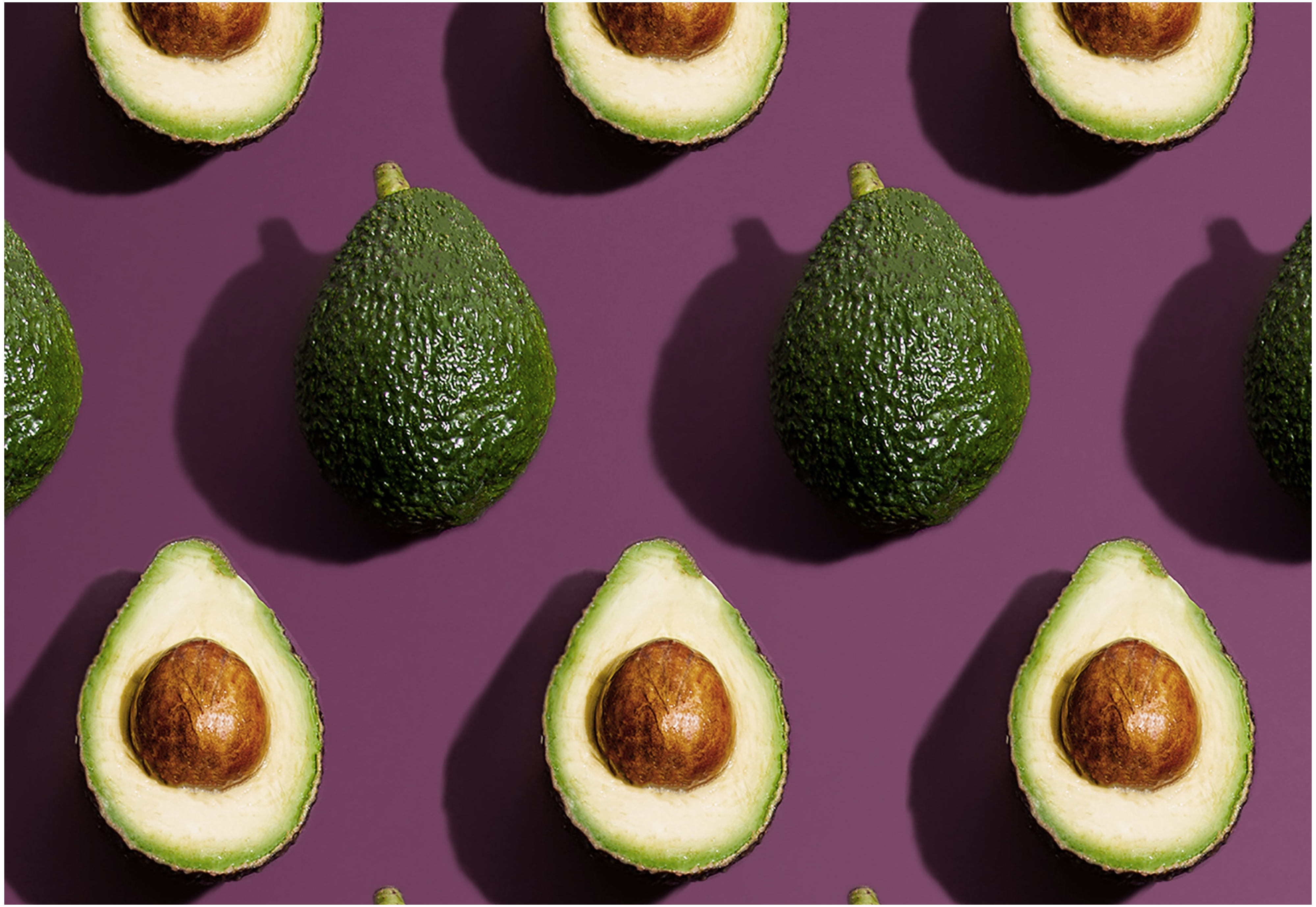
BAYESIAN DATA ANALYSIS IN PYTHON

How much is an avocado?

BAYESIAN DATA ANALYSIS IN PYTHON

Michal Oleszak
Machine Learning Engineer

The Avocado, Inc.



Case study: estimating price elasticity

Goal: estimate price elasticity of avocados and optimize the price

(price elasticity = impact of the change in price on the sales volume)

1. Fit a Bayesian regression model.
2. Inspect the model to verify its correctness.
3. Predict sales volume for different prices.
4. Propose the profit-maximizing price and the associated uncertainty.

Avocado data

```
print(avocado)
```

```
           date  price      volume  type_organic
0    2015-01-04   0.95  313.242777             0
1    2015-01-11   1.01  290.635427             0
2    2015-01-18   1.03  290.434588             0
3    2015-01-25   1.04  284.703108             0
..          ...    ...         ...           ...
334  2018-03-04   1.52   16.344308             1
335  2018-03-11   1.52   16.642349             1
336  2018-03-18   1.54   16.758042             1
337  2018-03-25   1.55   15.599672             1
```

[1] Data source: https://www.kaggle.com/neuromusic/avocado-prices

Priors in pymc3

```
formula = "num_bikes ~ temp + work_day + wind_speed"

with pm.Model() as model:

    pm.GLM.from_formula(formula, data=bikes)
    trace = pm.sample(draws=1000, tune=500)
```

Priors in pymc3

```
formula = "num_bikes ~ temp + work_day + wind_speed"

with pm.Model() as model:
    priors = {"wind_speed": pm.Normal.dist(mu=-5)}
    pm.GLM.from_formula(formula, data=bikes, priors=priors)
    trace = pm.sample(draws=1000, tune=500)
```

Extracting draws from trace

```
temp_draws = trace.get_values("temp")
print(temp_draws)
```

```
array([6.8705346, 6.7421152, 6.7393061, ..., 5.966574 , 6.1274128, 6.7149277])
```

What you will need

Model fitting:

- `pm.Model()`
- `pm.GLM.from_formula()`
- `pm.sample()`
- `pm.Normal()`

Visualization:

- `pm.forestplot()`
- `pm.traceplot()`

Making predictions:

- `pm.fast_sample_posterior_predictive()`

Inference:

- `pm.hpd()`

Let's put what you've learned to practice!

BAYESIAN DATA ANALYSIS IN PYTHON

Final remarks

BAYESIAN DATA ANALYSIS IN PYTHON

Michal Oleszak
Machine Learning Engineer

What you know

Chapter 1: The Bayesian Way

- Bayesian vs. frequentist approach
- Probability theory & distributions
- Updating beliefs with more data

Chapter 2: Bayesian Estimation

- Grid approximation
- Prior distributions
- Reporting Bayesian results

Chapter 3: Bayesian Inference

- A/B testing
- Decision analysis
- Forecasting & regression

Chapter 4: Bayesian Linear Regression

- Markov Chain Monte Carlo (MCMC)
- Fitting and interpreting models with `pymc3`
- Bayesian data analysis: a case study

More Bayes

- Hierarchical models:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\beta_2 = \beta_{20} + \beta_{21} x_3$$

- More regression (logistic, Poisson, ...)
- Bayesian machine learning

- PyMC3 docs:
  https://pymc3.readthedocs.io/en/latest
- Think Bayes by Allen Downey
  http://allendowney.github.io/ThinkBayes2

Congratulations and good luck!

BAYESIAN DATA ANALYSIS IN PYTHON