ECE 20875

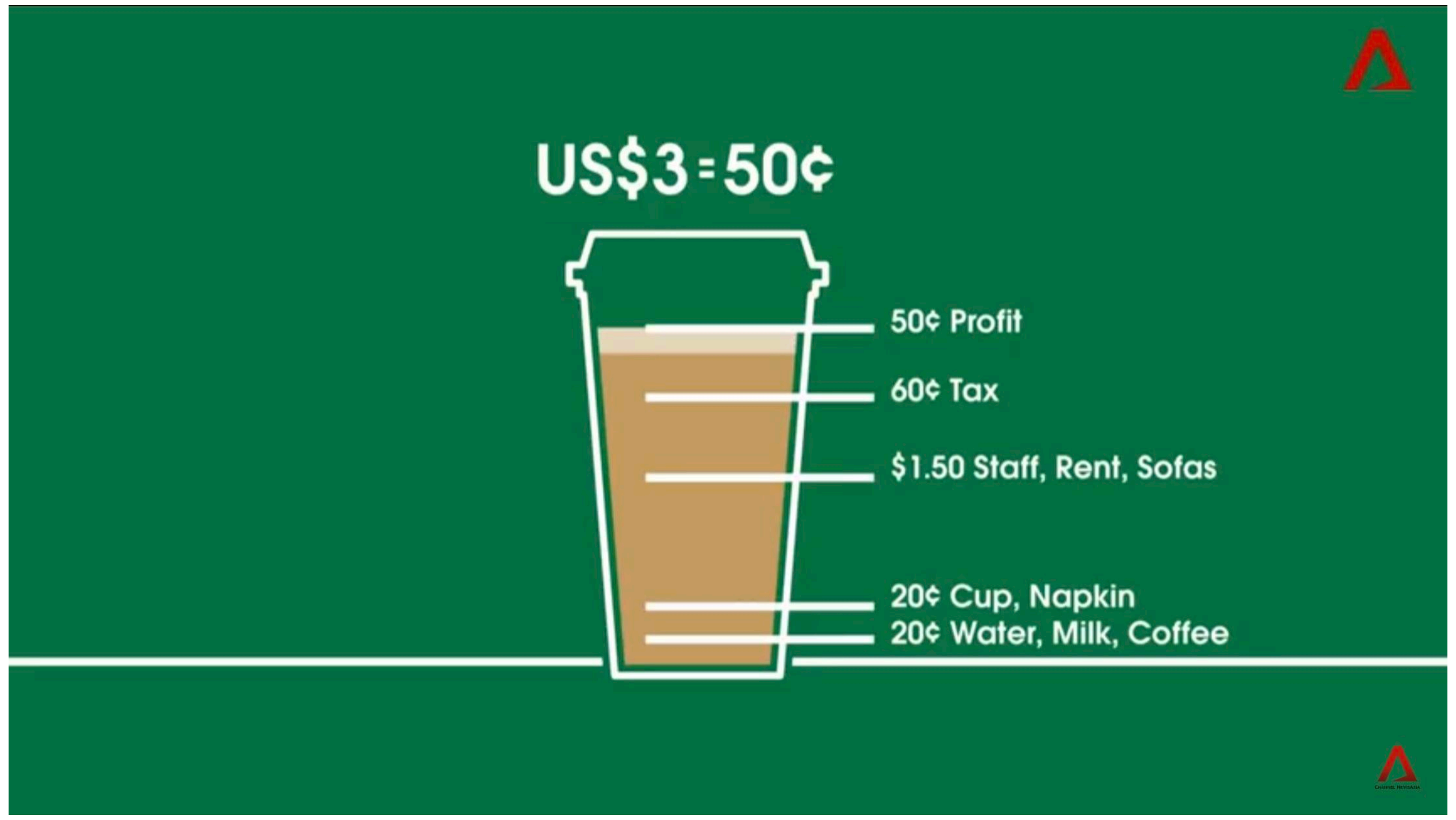
Python for Data Science

Milind Kulkarni and Chris Brinton

Histograms

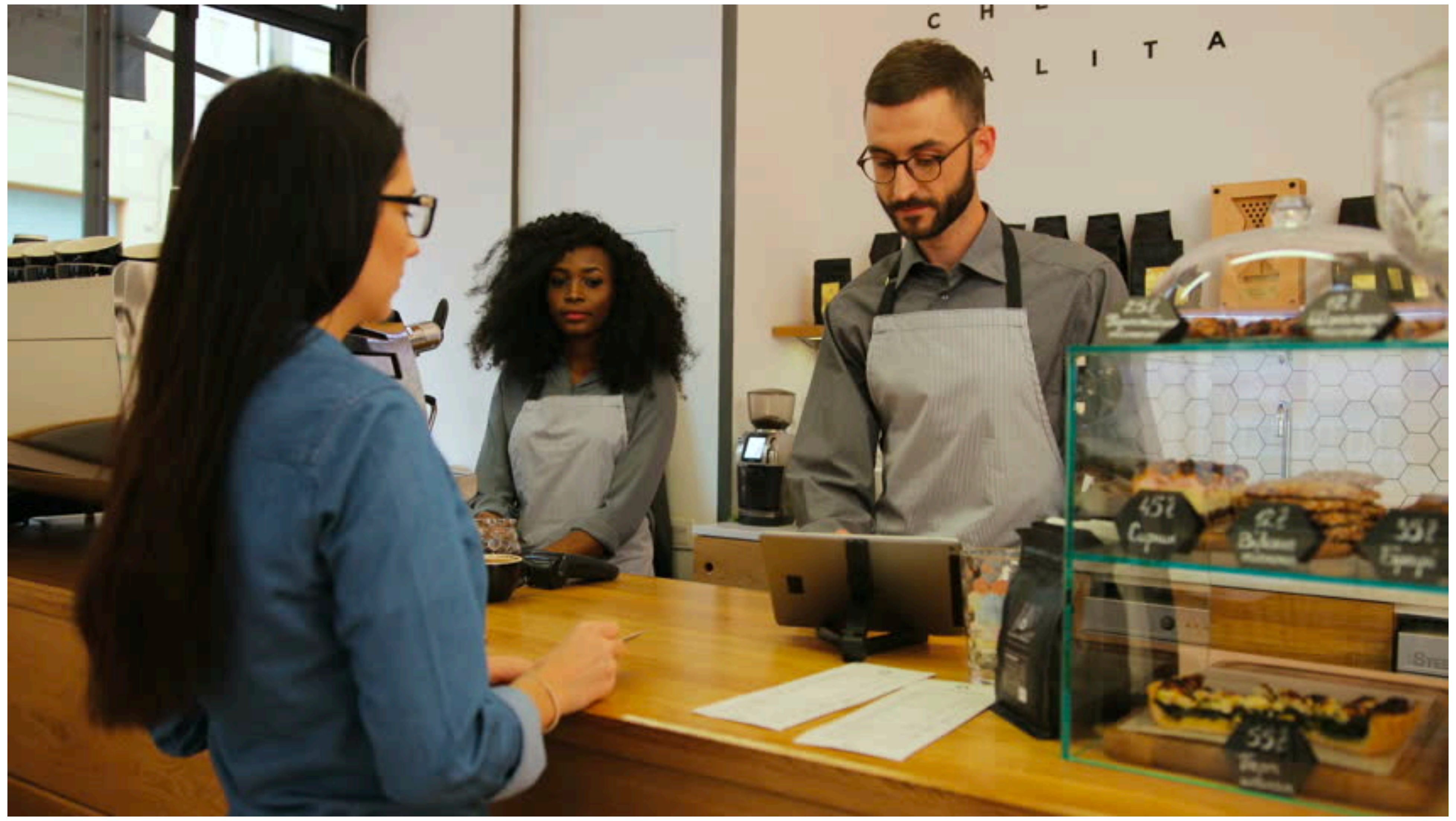
a problem

- You're managing a coffee shop
- Assuming you want to maximize profit, how much coffee should you buy for each day?
  - Too much → Surplus, waste money :(
  - Too little → Unsatisfied demand, under-caffeinated customers :(
- What should you do?



collect data

- Count how many people get coffee in a day
  - Day 1: 37 people
  - Likely different each day of the week, and the type of coffee (cold brew, late, etc.) also has an impact
  - Assume such factors do not matter (problem is still interesting!)
- Should we just get enough coffee for 37 people?



(keep) collect(ing) data

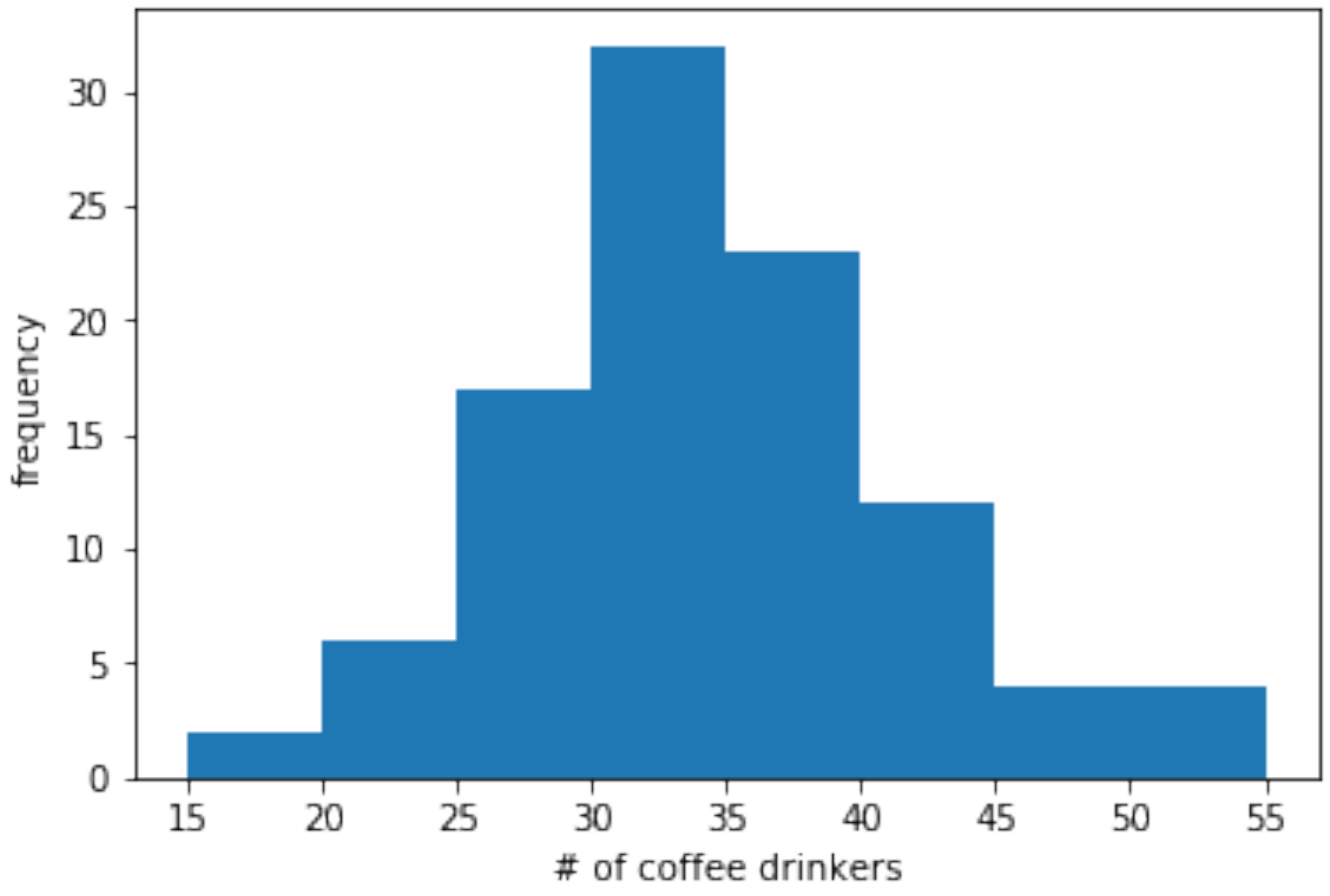
- Day 2: 43
- Day 3: 48
- Day 4: 41
- Day 5: 46
- Day 6: 19 (!)
- Day 7: 38
- …

100 days later ...

```
[37, 43, 48, 41, 46, 19, 28, 35, 34, 38,
31, 32, 32, 23, 23, 33, 35, 39, 34, 28,
39, 28, 29, 38, 28, 30, 25, 35, 39, 35,
31, 28, 25, 26, 15, 31, 28, 32, 40, 21,
34, 38, 30, 47, 34, 31, 51, 30, 41, 36,
33, 51, 22, 25, 29, 50, 32, 39, 25, 37,
54, 33, 36, 25, 30, 22, 41, 35, 31, 40,
30, 33, 27, 36, 27, 34, 24, 41, 37, 29,
48, 40, 31, 32, 33, 32, 40, 31, 32, 40,
31, 33, 32, 38, 37, 41, 37, 39, 38, 42]
```

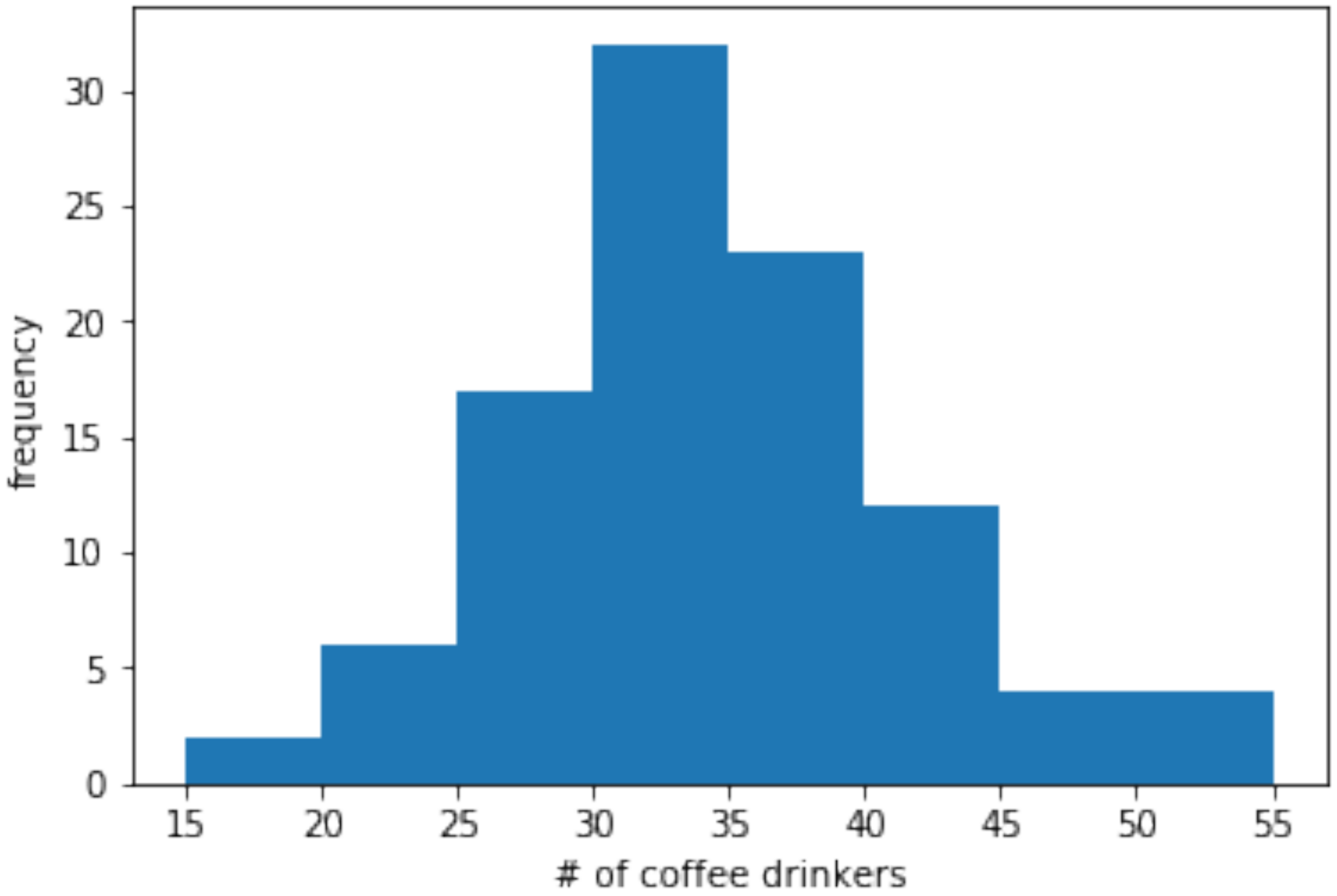
visualize the data

- Staring at a list of numbers is not very illuminating
- Visualizing the data in a useful way can help reveal patterns
  - **Data visualization** is an important subset of data science
- Since the data consists of a single, numeric variable, we can try a **histogram**



building a histogram

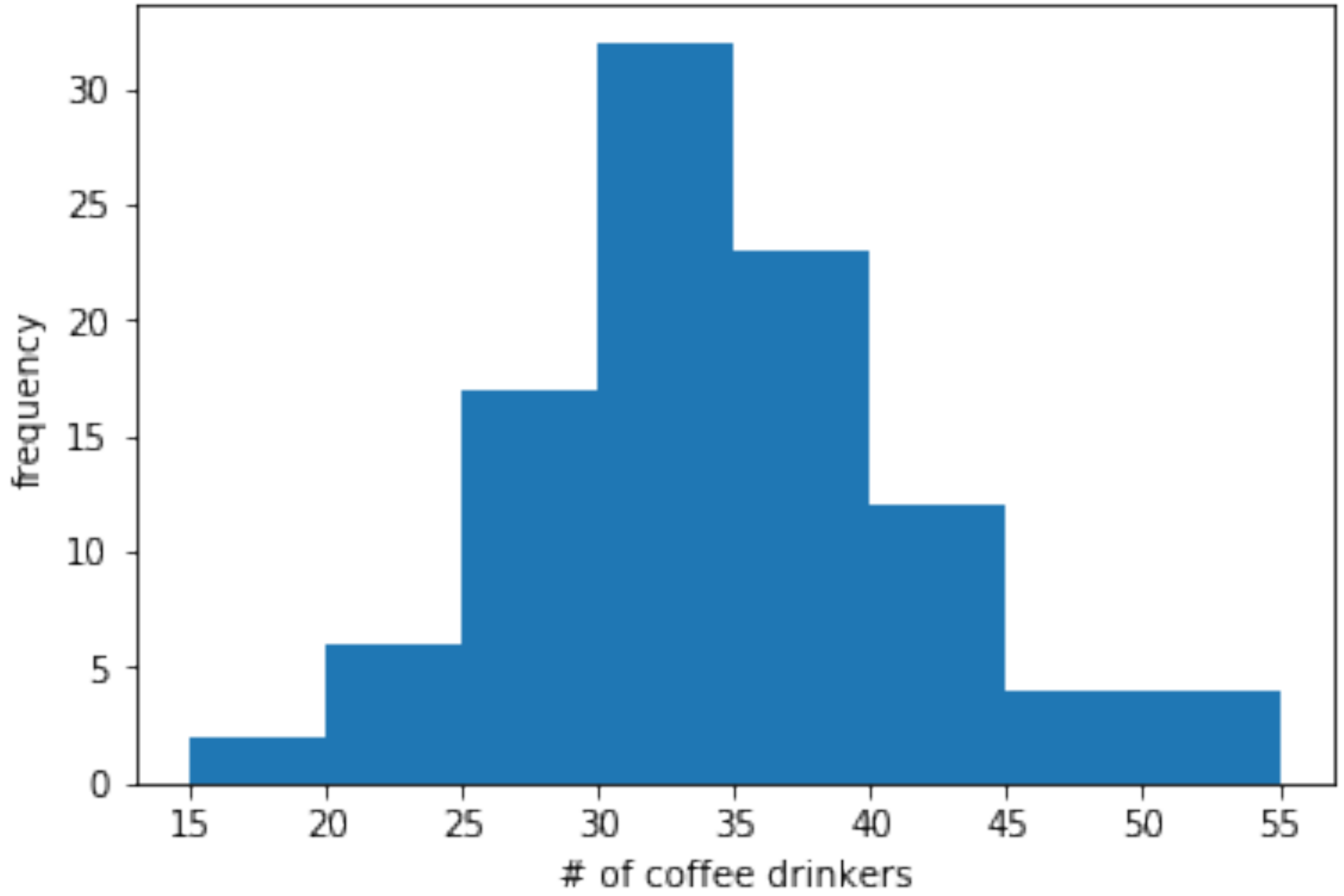
- A histogram visualizes observations (samples) of a random variable d
- Each bar in a histogram is a **bin**
  - $x_1, x_2, \ldots$
- Each observation is placed into one bin
  - $x_1 : 15 \leq d < 20,\ x_2 : 20 \leq d < 25, \ldots$
- The **count** (size/height) of each bin is the number of observations in that bin
  - $x_1 : 2,\ x_2 : 6, \ldots$
- The empirical (measured) **frequency** of each bin is the fraction of data in that bin
  - $\hat{p}_1 = 0.02,\ \hat{p}_2 = 0.06, \ldots$ $\sum_k \hat{p}_k = 1$



```
_ = plt.hist(data, bins=8, range=(15,55))
plt.xlabel('# of coffee drinkers')
plt.ylabel('frequency')
```

repeating the experiment

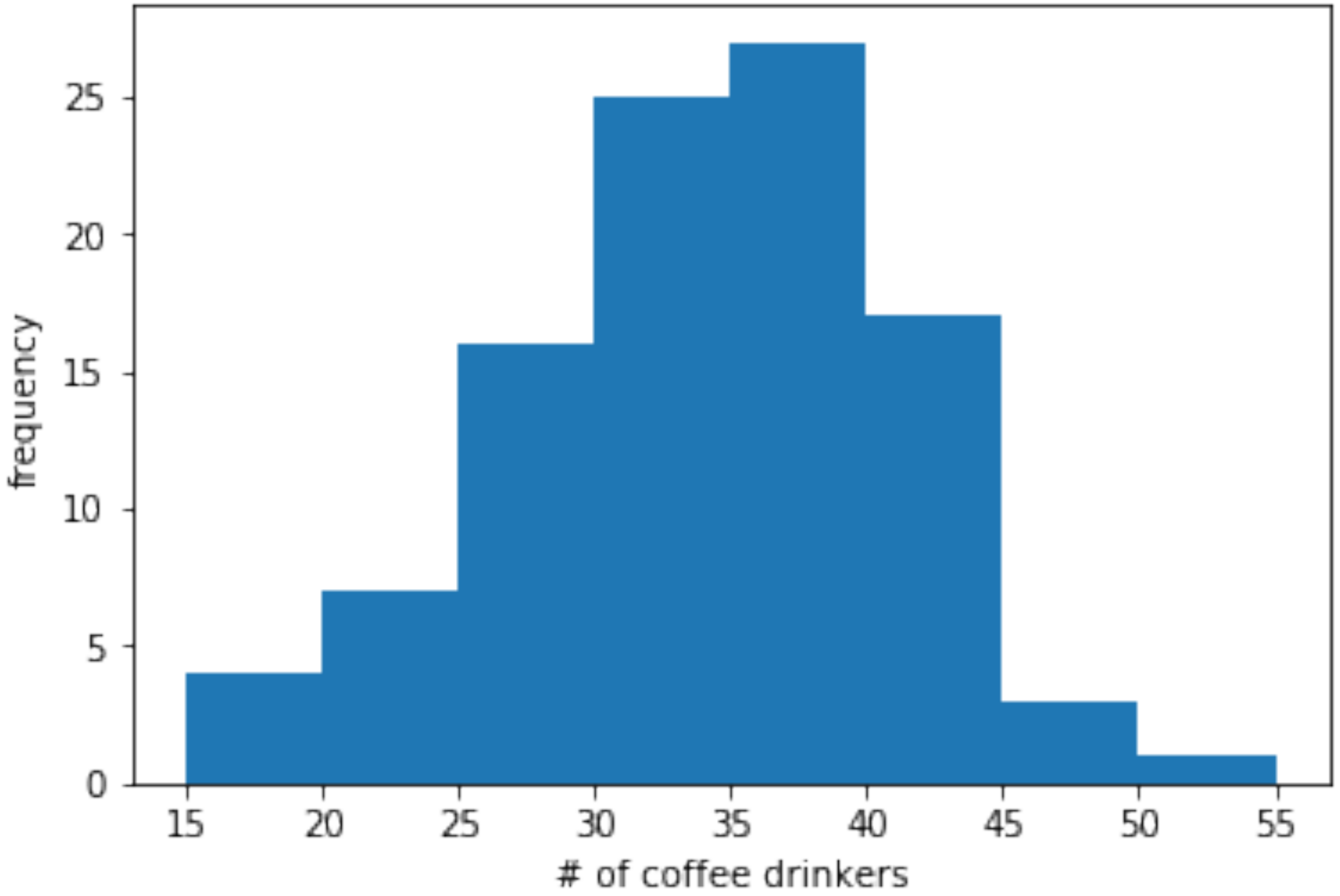
- Remember: This histogram comes from observed data
- If we repeat the experiment, we might not get the same histogram!
    - In fact, there will almost surely be some difference at this sample size
- This is because what we have is a **sample** of the true distribution



```
_ = plt.hist(data, bins=8, range=(15,55))
plt.xlabel('# of coffee drinkers')
plt.ylabel('frequency')
```

repeating the experiment

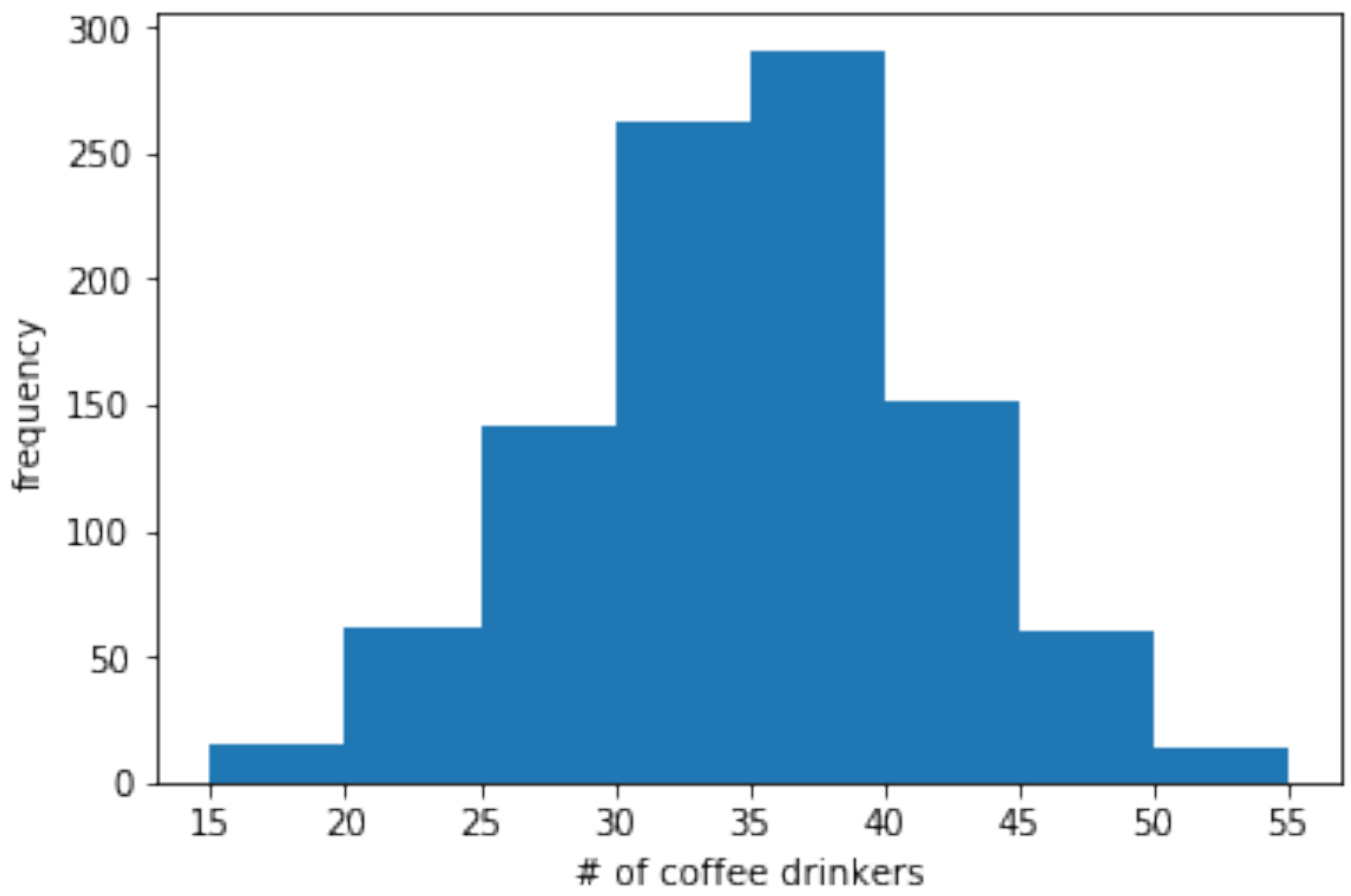
- Remember: This histogram comes from observed data
- If we repeat the experiment, we might not get the same histogram!
  - In fact, there will almost surely be some difference at this sample size
- This is because what we have is a **sample** of the true distribution



```
_ = plt.hist(data, bins=8, range=(15,55))
plt.xlabel('# of coffee drinkers')
plt.ylabel('frequency')
```

collecting a larger sample

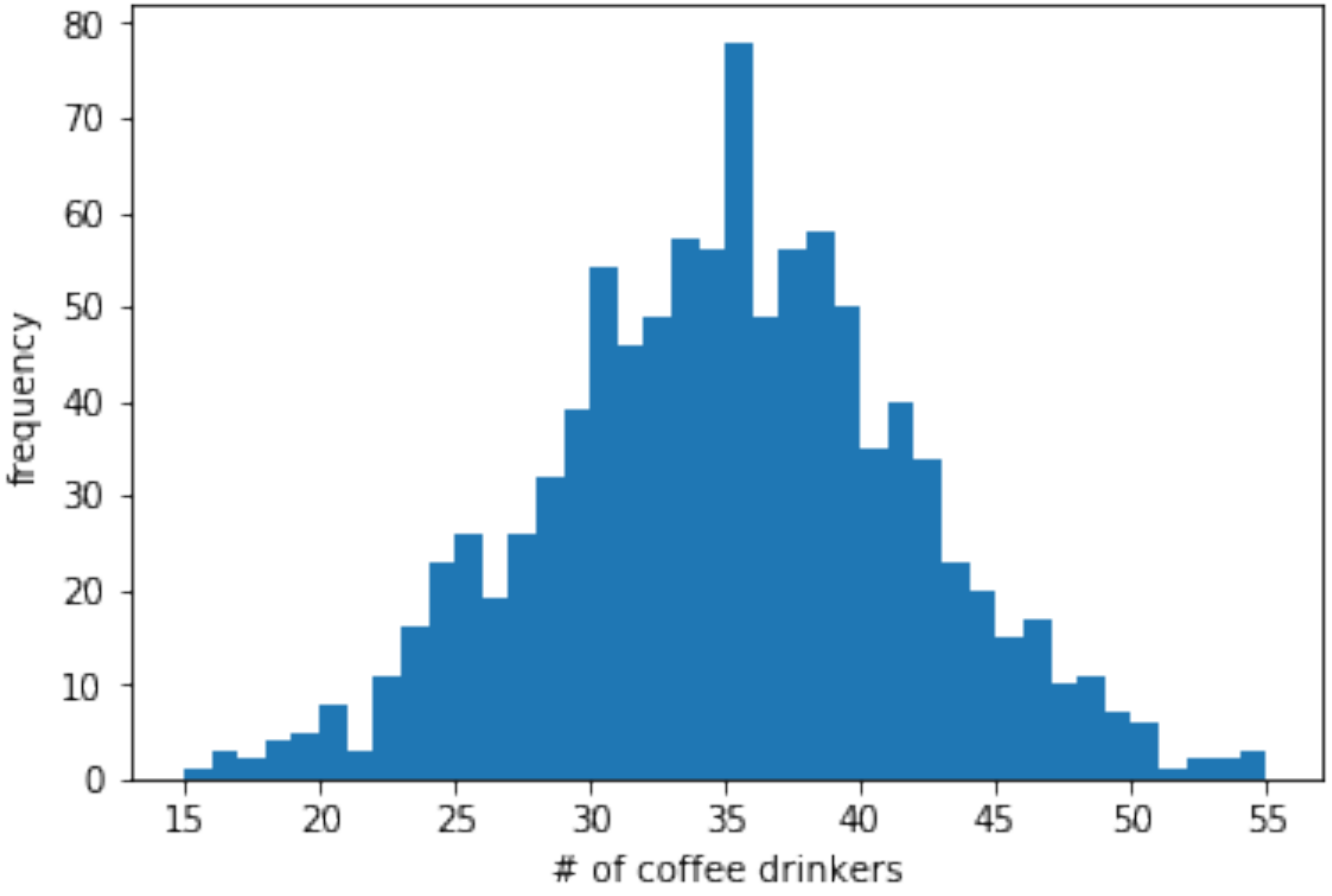
- Suppose we collect 1000 samples instead of 100
- The result on the right looks basically the same!
- Using the same number of bins
  - Each bin has more observations in it, but the relative frequencies are not changing much
- But now that we have a larger sample, we can add more bins to see a finer granularity of the distribution



```
_ = plt.hist(data, bins=8, range=(15,55))
plt.xlabel('# of coffee drinkers')
plt.ylabel('frequency')
```

adding more bins

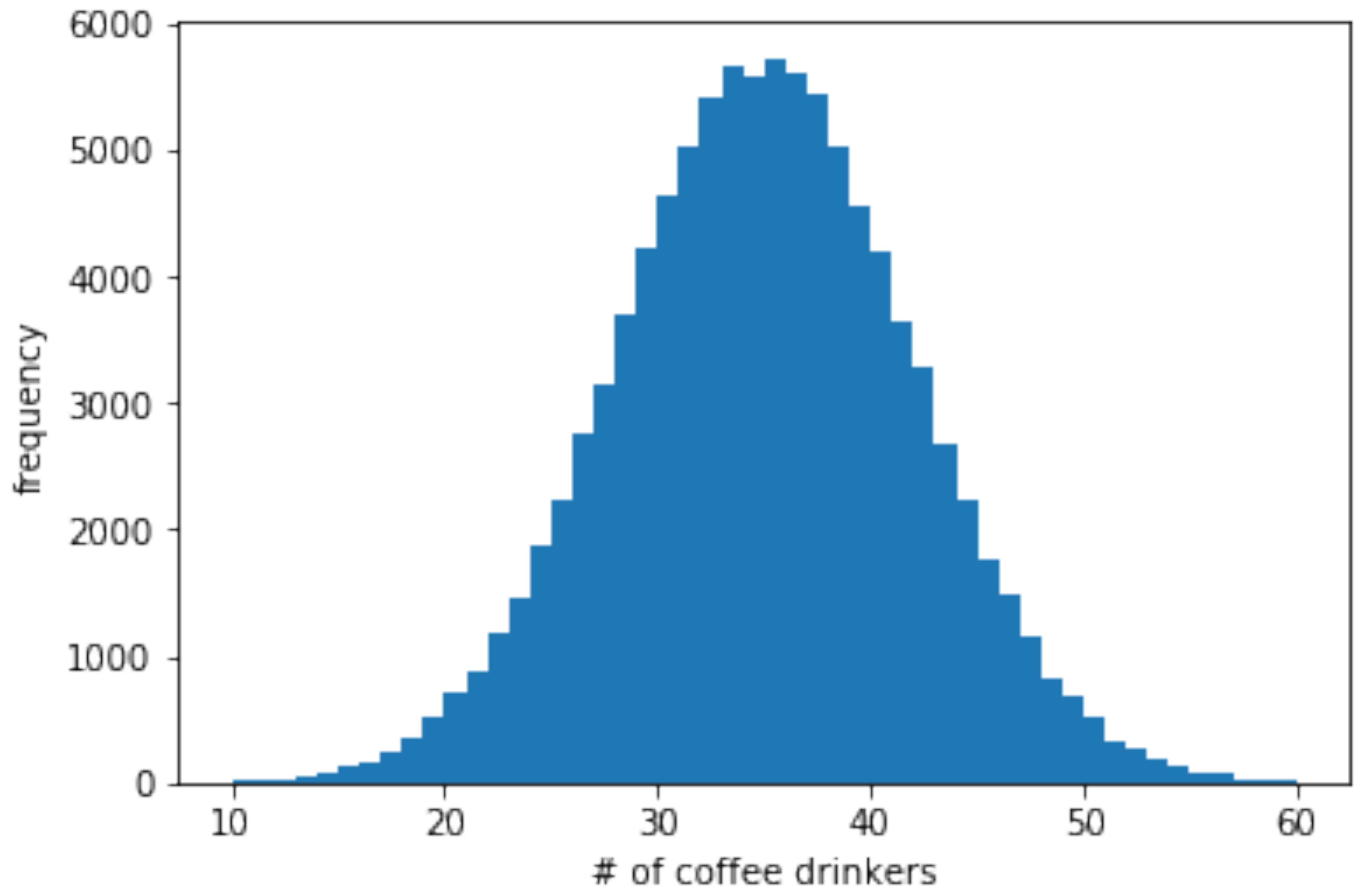
- This looks better!
- Gives us a good sense of what the data looks like, and what the underlying distribution is
- What would happen if we used more than 40 bins here?



```
_ = plt.hist(data, bins=40, range=(15,55))
plt.xlabel('# of coffee drinkers')
plt.ylabel('frequency')
```

adding even more data

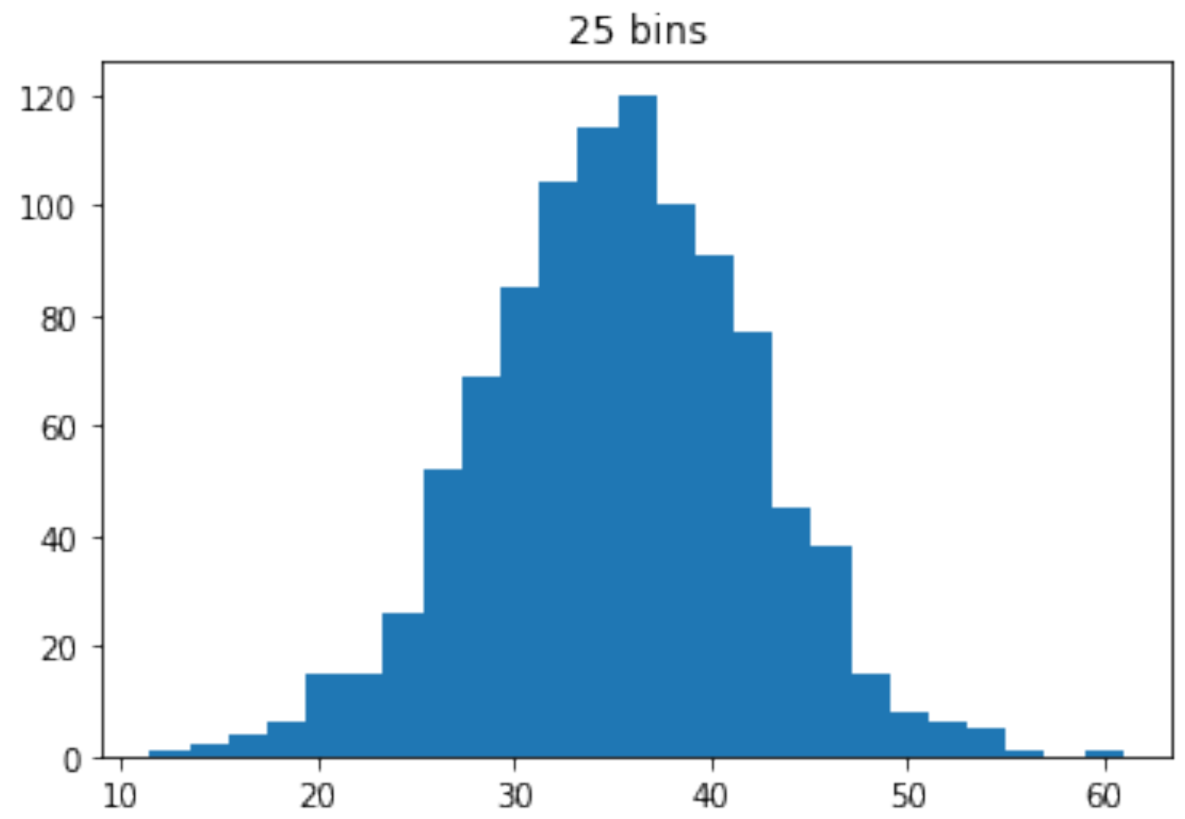
- This looks even better!
- As we add more data points, our histogram begins to look more and more like the "true" shape of the data
  - We'll get in to what this means when we talk about distributions and sampling

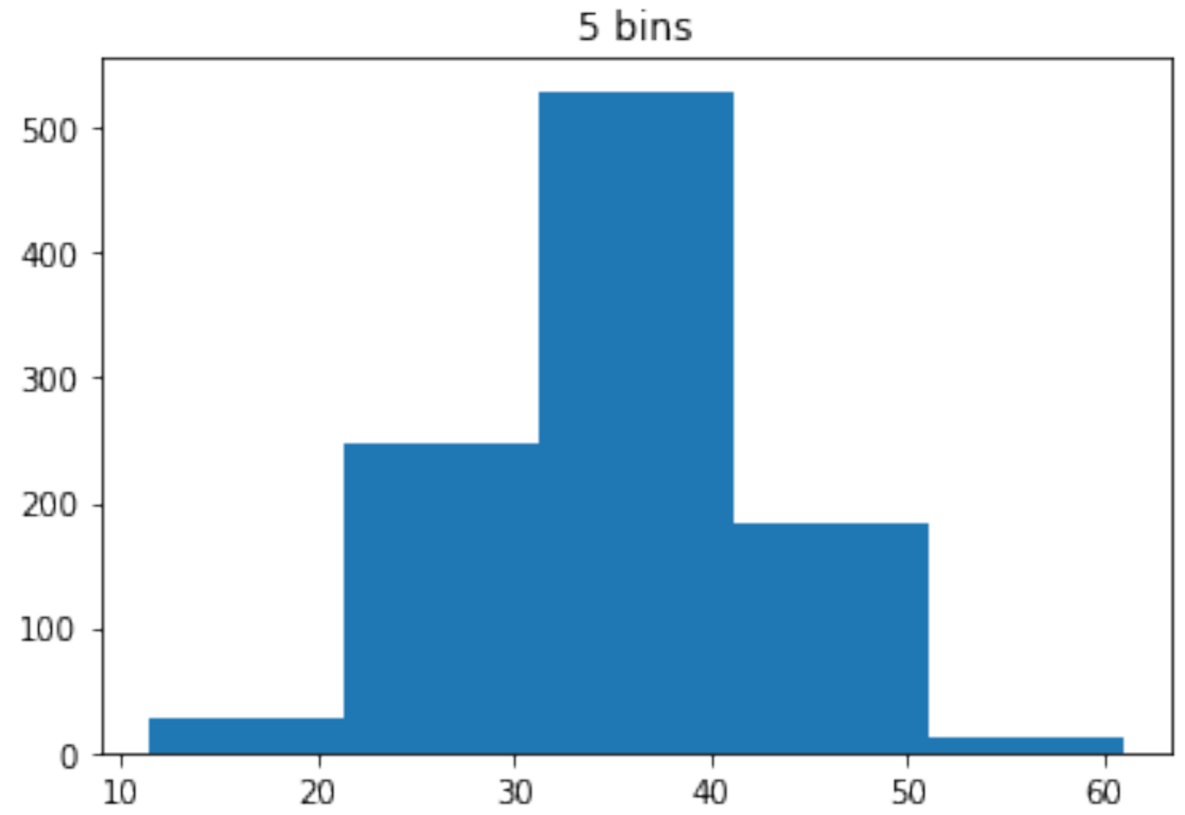


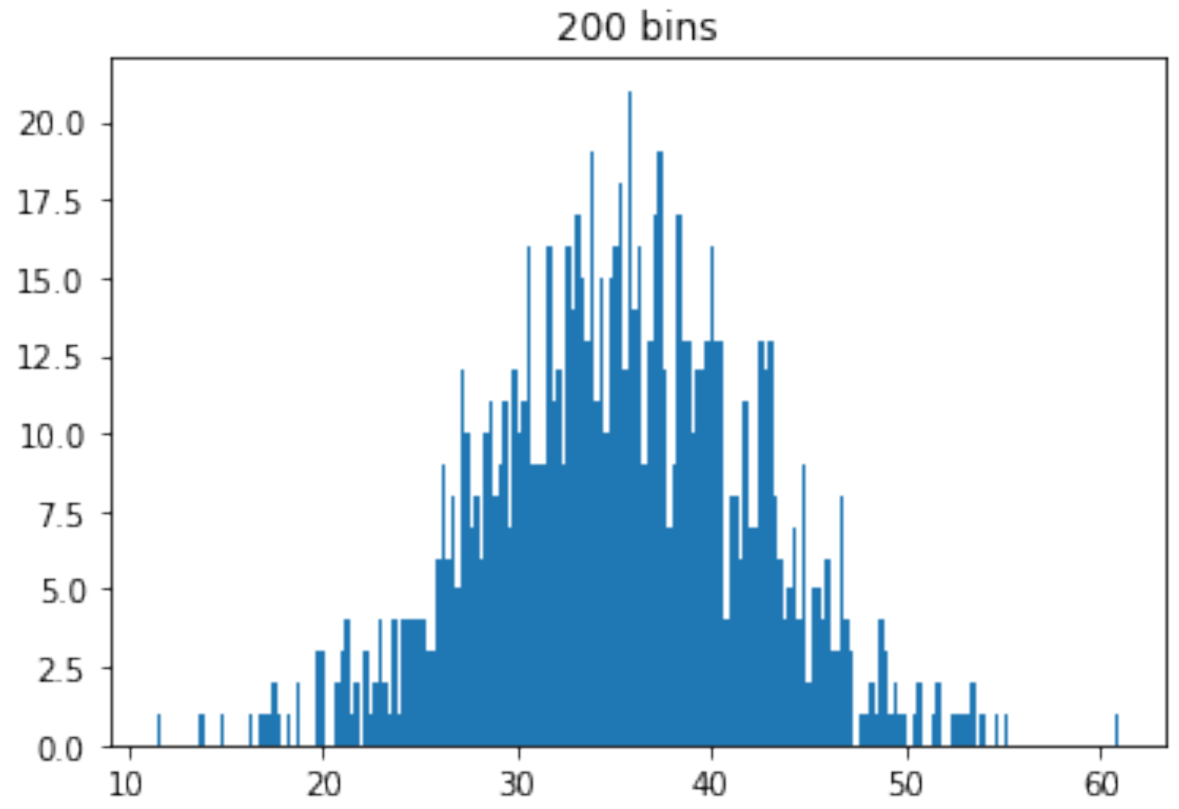
```
_ = plt.hist(data, bins=40, range=(15,55))
plt.xlabel('# of coffee drinkers')
plt.ylabel('frequency')
```

choice of bins

- The histogram has a few parameters
  - Number of bins k, width of bins h, and even number of samples n can be viewed as one
  - Bins don't even have to be homogeneous
- Several formulas have been proposed for choosing k and/or h based on the sample
  - Square root: $k = \lceil \sqrt{n} \rceil$
  - Sturges' formula: $k = \lceil \log_2 n \rceil + 1$
  - Rice rule: $k = \lceil 2n^{1/3} \rceil$
  - Scott's normal reference rule: $h = 3.5\hat{\sigma}/n^{1/3}$
- How do we reason about the "optimal" choice?

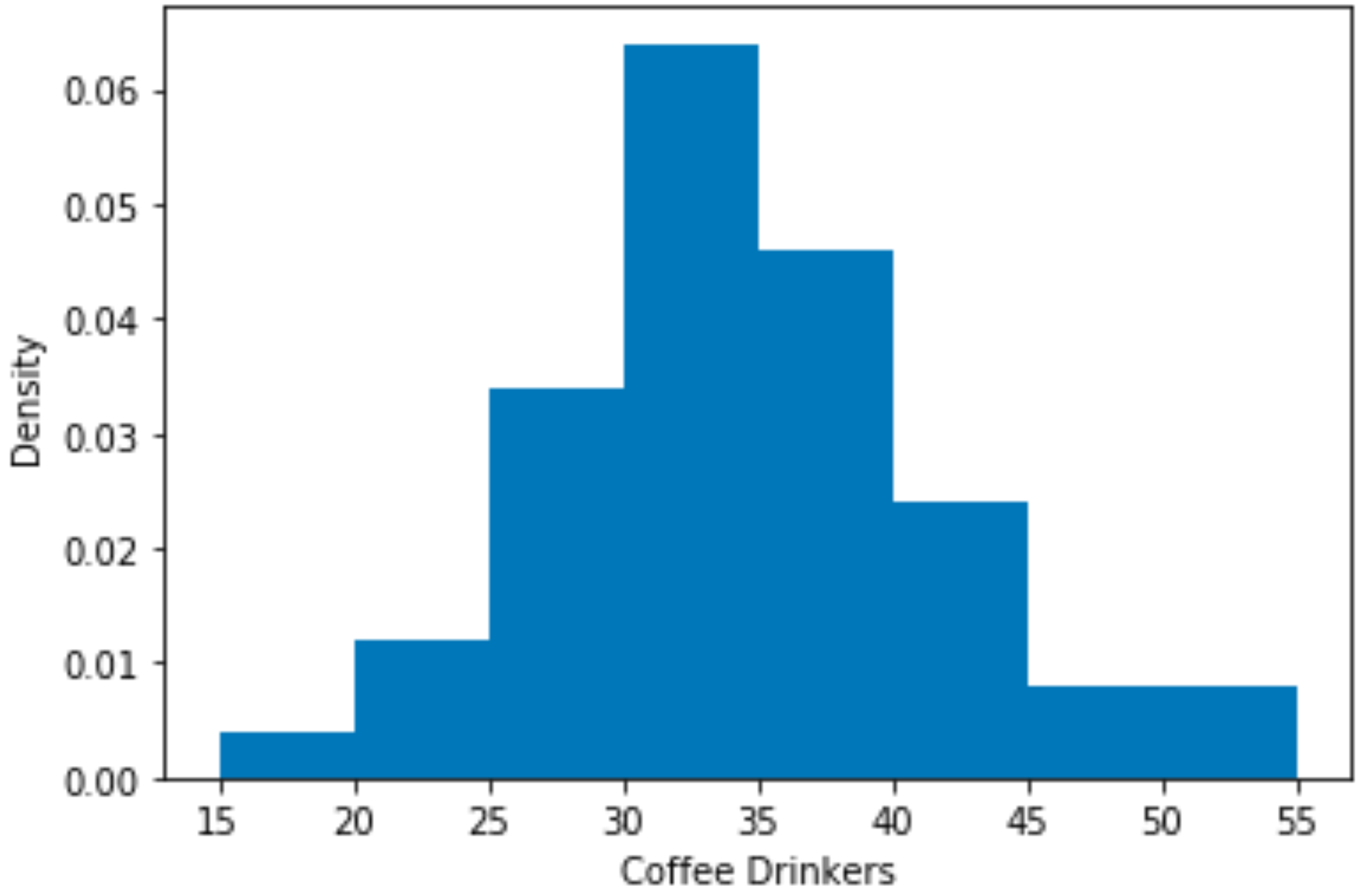






histogram as an estimator

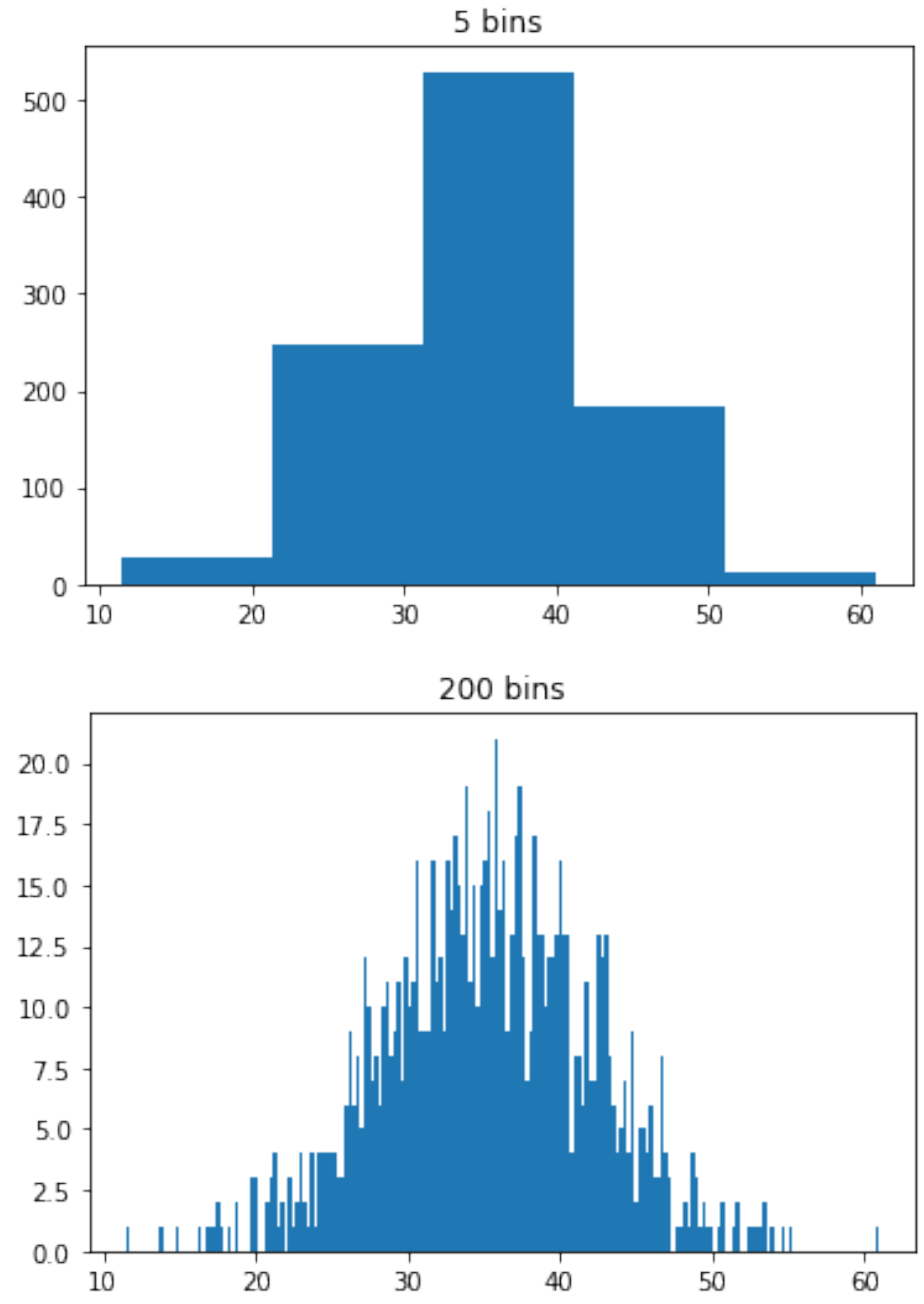
- Intuition: The histogram estimates the "true" distribution of data using the sample observed
- Said another way, the histogram frequencies estimate how "likely" a new datapoint is to fall into a given bin
  - $\hat{p}_4 = 0.32$: Estimate a 32% chance that $d \in [30, 35)$
  - All datapoints in the same bucket get the same estimate
- On the right, what is the difference between "frequency" and **density**?



```
_ = plt.hist(data, bins=8, range=(15,55),
density='True')
plt.xlabel('# of coffee drinkers')
plt.ylabel('density')
```

bucket size intuition

- Formulas typically assume normal distribution
- Choosing large bin size h
  - Decreasing **precision**
  - Broad range of points (some rare, some common) put into the same bin and given the same estimate
- Choosing small bin size h
  - Decreasing **accuracy**
  - Each bin is based on fewer samples, so harder to estimate how likely the bin is
  - Worst case: Buckets of size 0 (is it practical?)
- So how do we choose the bin size in general?



minimize error of estimator

- We can pick the bucket size h that minimizes the error of estimating a point
- The Mean Square Error (**MSE**) of a histogram can be written as a function of the smoothing parameter:

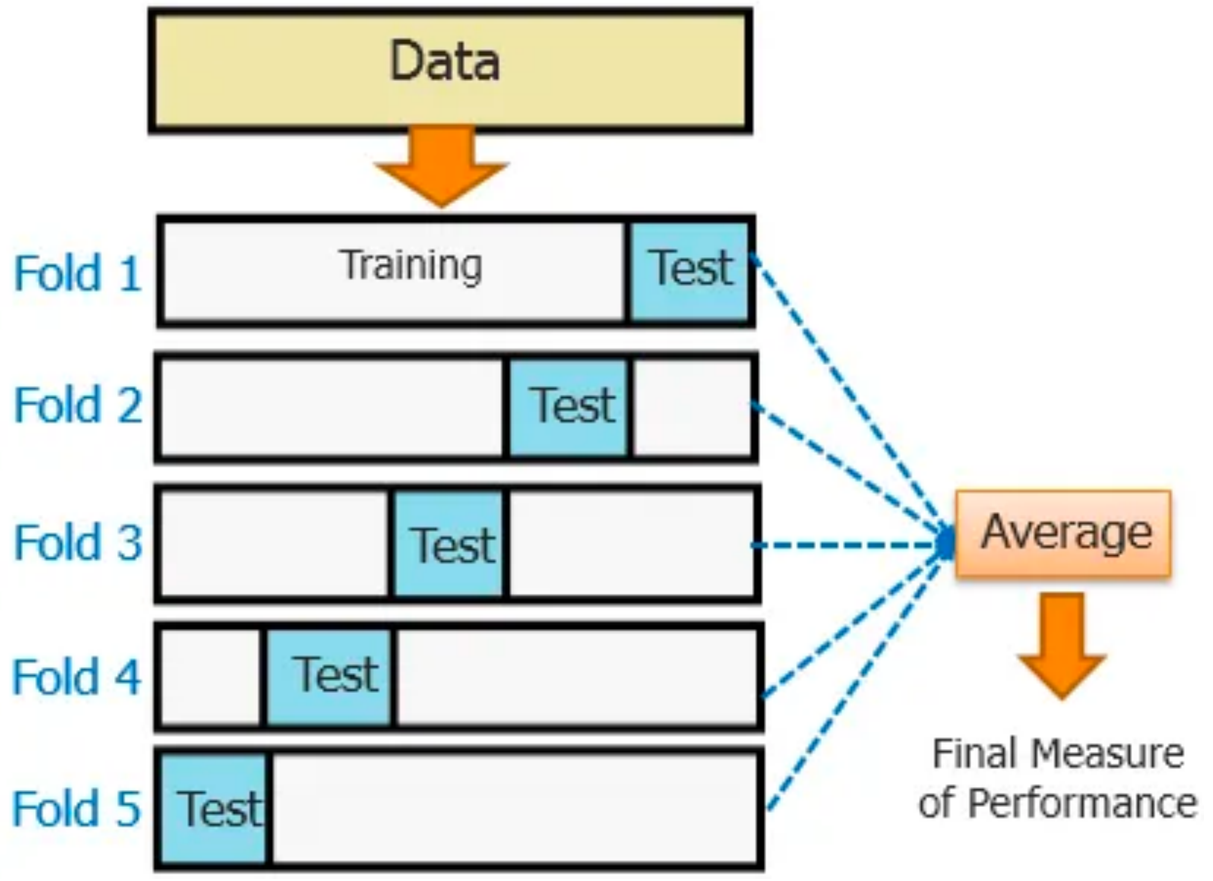
$$L(h) = \int \left(\hat{f}_n(x) - f(x) \right)^2 dx$$

- Here, $\hat{f}_n(x)$ is the density estimate of the histogram with n samples
- Expanding this, minimizing $L(h)$ becomes equivalent to minimizing:

$$J(h) = \underbrace{\int \left(\hat{f}_n(x) \right)^2 dx}_{\textbf{Can compute directly}} - 2 \underbrace{\int \hat{f}_n(x) f(x) dx}_{\textbf{Need to estimate}}$$

cross validation

- **Cross validation** techniques assess how well a model will generalize to new (unseen) data
  - Build model on part of the dataset, use the remainder as a "test"
  - Repeat over multiple combinations, and (typically) average
- Leave-one-out (**LOO**) cross validation
  - Build model on all but one datapoint
  - In the histogram case, for each h, construct n different histograms, and average heights
  - If you carry out the math, you will get this cross-validation estimator:

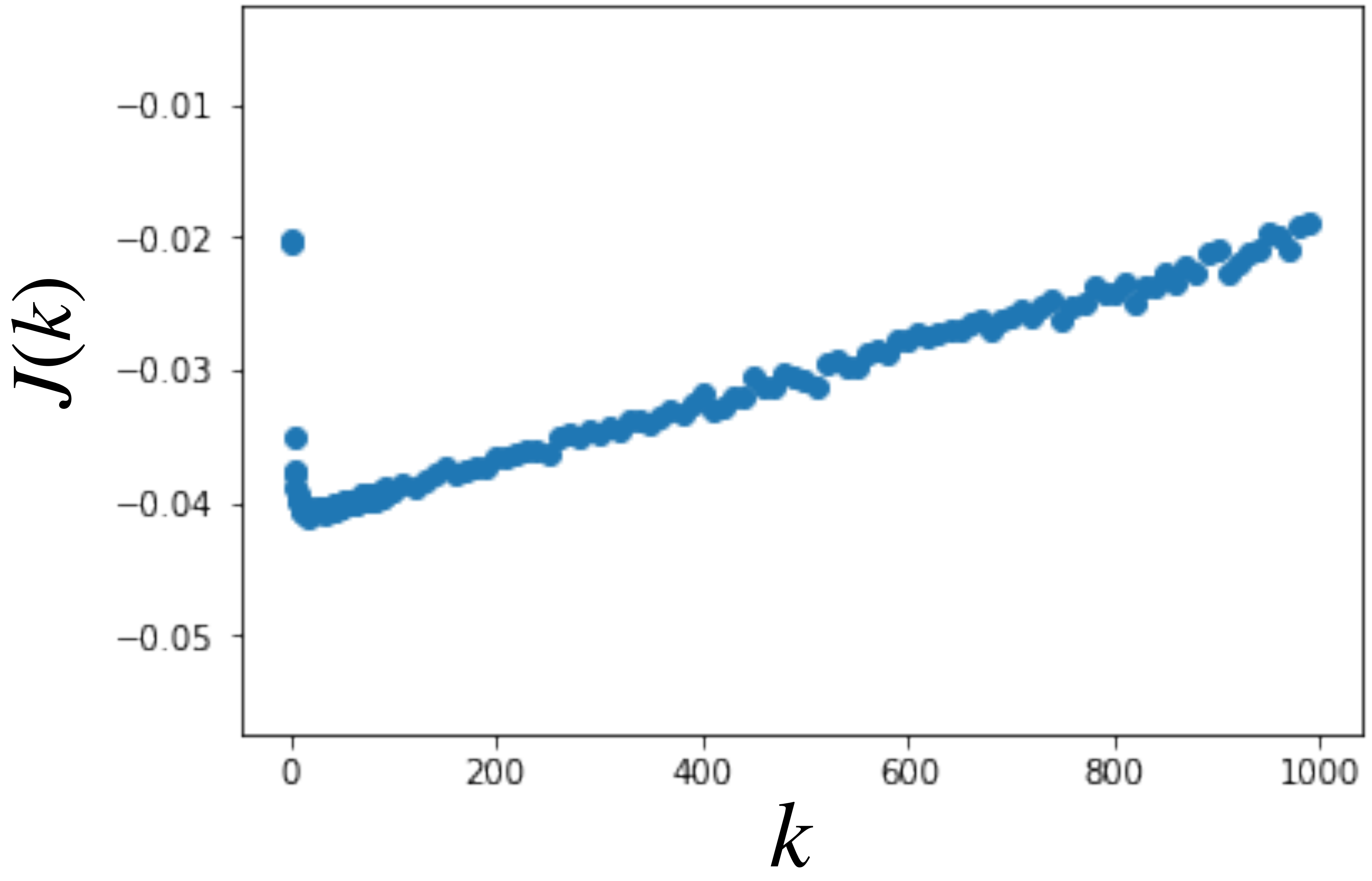


$$J(h) = \frac{2}{(n-1)h} - \frac{n+1}{(n-1)h}(\hat{p}_1^2 + \hat{p}_1^2 + \cdots + \hat{p}_k^2)$$

cross validation procedure

1. Choose a number of #bins m to test in each iteration
2. Choose an initial range size r
3. Initialize *kset* $= \{0, r/m, \ldots, r\}$
4. For each number k in *kset*:
    - Compute $J(k)$
    - Keep track of $k^\star$ with lowest $J(k)$
5. If $J(k^\star)$ is significantly different than the previous
    - Set $r = r/m$, *kset* $= \{\text{int}(h^\star - 0.5r/m), \ldots, \text{int}(h^\star + 0.5r/m)\}$
    - Go to Step 4

Testing all numbers of bins



−0.01
−0.02
−0.03
−0.04
−0.05
$J(k)$
0
200
400
600
800
1000
k