Living the sample life

SAMPLING IN PYTHON

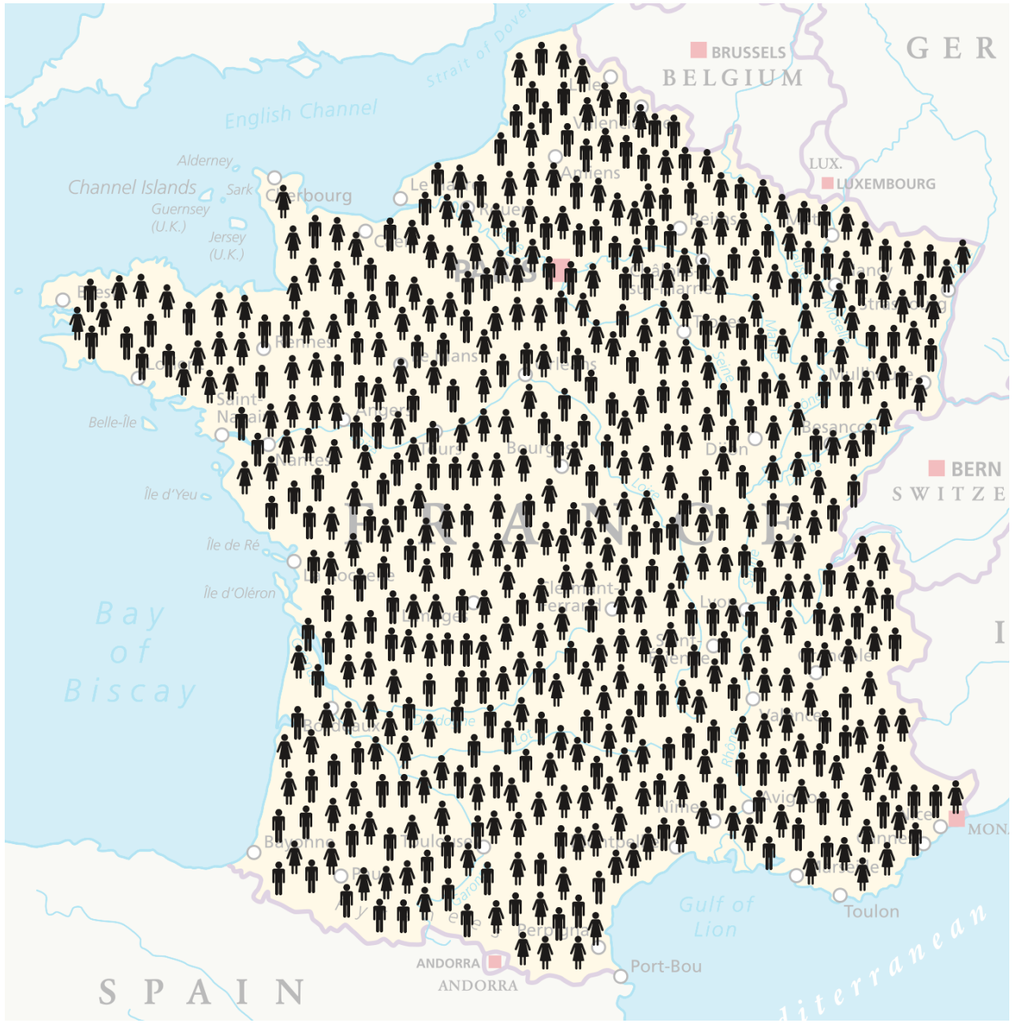
James Chapman
Content Developer, DataCamp

datacamp

Estimating the population of France

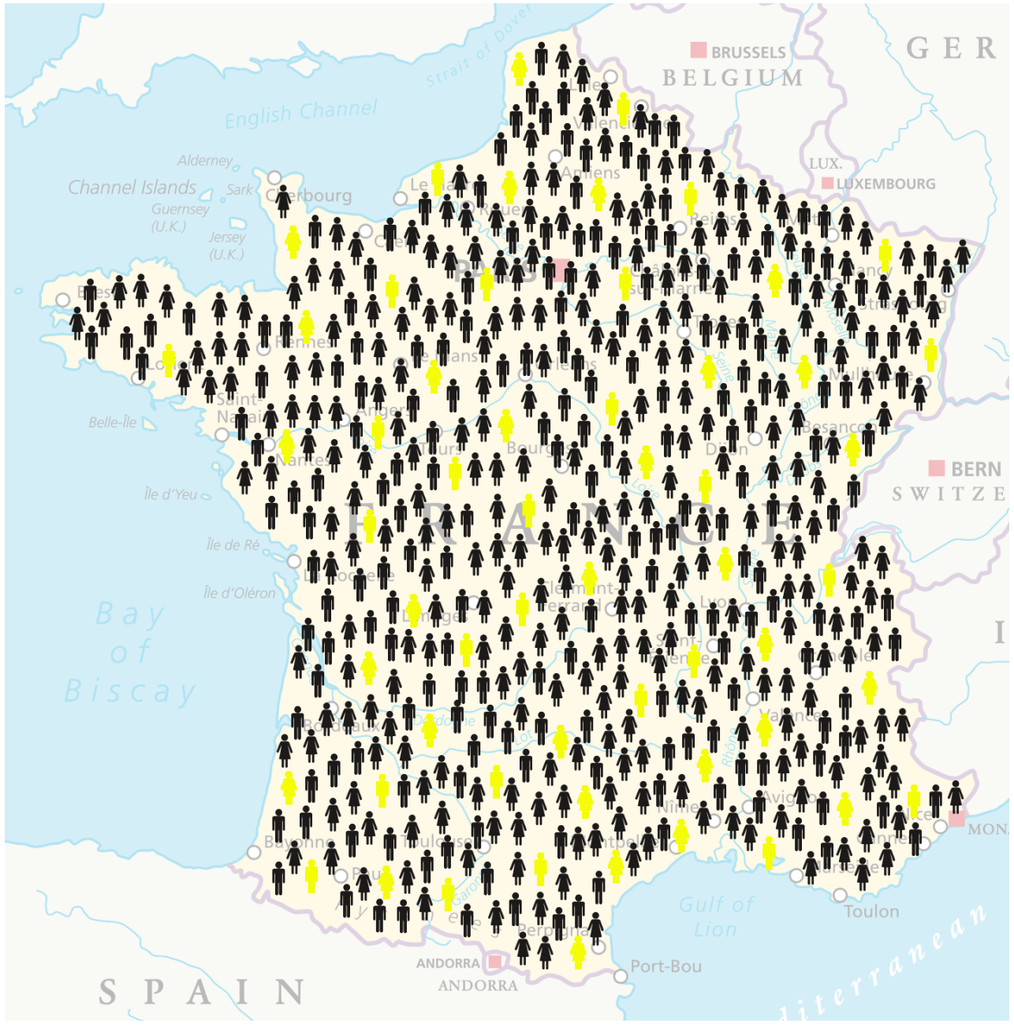
A census asks every household how many people live there.

There are lots of people in France



Censuses are really expensive!

Sampling households



Cheaper to ask a small number of households and use statistics to estimate the population

Working with a subset of the whole population is called *sampling*

Population vs. sample

The *population* is the complete dataset

- Doesn't have to refer to people
- Typically, don't know what the whole population is

The *sample* is the subset of data you calculate on

Coffee rating dataset

total_cup_points	variety	country_of_origin	aroma	flavor	aftertaste	body	balance
90.58	NA	Ethiopia	8.67	8.83	8.67	8.50	8.42
89.92	Other	Ethiopia	8.75	8.67	8.50	8.42	8.42
...	...	...	...	...	...	...	...
73.75	NA	Vietnam	6.75	6.67	6.5	6.92	6.83

- Each row represents 1 coffee
- 1338 rows
- We'll treat this as the population

Points vs. flavor: population

```
pts_vs_flavor_pop = coffee_ratings[["total_cup_points", "flavor"]]
```

```
      total_cup_points  flavor
0                90.58    8.83
1                89.92    8.67
2                89.75    8.50
3                89.00    8.58
4                88.83    8.50
...                ...     ...
1333             78.75    7.58
1334             78.08    7.67
1335             77.17    7.33
1336             75.08    6.83
1337             73.75    6.67

[1338 rows x 2 columns]
```

Points vs. flavor: 10 row sample

```
pts_vs_flavor_samp = pts_vs_flavor_pop.sample(n=10)
```

```
      total_cup_points  flavor
1088             80.33    7.17
1157             79.67    7.42
1267             76.17    7.33
506              83.00    7.67
659              82.50    7.42
817              81.92    7.50
1050             80.67    7.42
685              82.42    7.50
1027             80.92    7.25
62               85.58    8.17

[10 rows x 2 columns]
```

Python sampling for Series

Use `.sample()` for `pandas` DataFrames and Series

```
cup_points_samp = coffee_ratings['total_cup_points'].sample(n=10)
```

```
1088    80.33
1157    79.67
1267    76.17
...     ...
685     82.42
1027    80.92
62      85.58
Name: total_cup_points, dtype: float64
```

Population parameters & point estimates

A *population parameter* is a calculation made on the population dataset

```
import numpy as np
np.mean(pts_vs_flavor_pop['total_cup_points'])
```

```
82.15120328849028
```

A *point estimate* or *sample statistic* is a calculation made on the sample dataset

```
np.mean(cup_points_samp)
```

```
81.31800000000001
```

Point estimates with pandas

```
pts_vs_flavor_pop['flavor'].mean()
```

```
7.526046337817639
```

```
pts_vs_flavor_samp['flavor'].mean()
```

```
7.485000000000001
```

Let's practice!

SAMPLING IN PYTHON

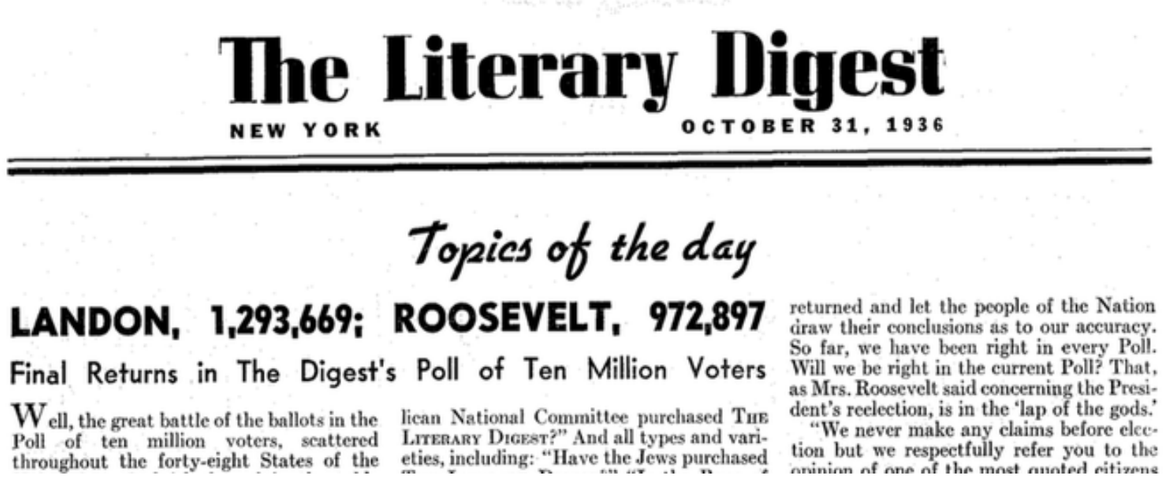
A little too convenient

SAMPLING IN PYTHON

James Chapman
Content Developer, DataCamp

datacamp

The Literary Digest election prediction



The Literary Digest

NEW YORK OCTOBER 31, 1936

Topics of the day

LANDON, 1,293,669; ROOSEVELT, 972,897

Final Returns in The Digest's Poll of Ten Million Voters

Well, the great battle of the ballots in the Poll of ten million voters, scattered throughout the forty-eight States of the

lican National Committee purchased THE LITERARY DIGEST?" And all types and varieties, including: "Have the Jews purchased

returned and let the people of the Nation draw their conclusions as to our accuracy. So far, we have been right in every Poll. Will we be right in the current Poll? That, as Mrs. Roosevelt said concerning the President's reelection, is in the 'lap of the gods.'

"We never make any claims before election but we respectfully refer you to the

- Prediction: Landon gets 57%; Roosevelt gets 43%
- Actual results: Landon got 38%; Roosevelt got 62%
- Sample not representative of population, causing *sample bias*
- Collecting data by the easiest method is called *convenience sampling*

Finding the mean age of French people

- Survey 10 people at Disneyland Paris
- Mean age of 24.6 years
- Will this be a good estimate for all of France?

[1] Image by Sean MacEntee

How accurate was the survey?

Year	Average French Age
1975	31.6
1985	33.6
1995	36.2
2005	38.9
2015	41.2

- 24.6 years is a poor estimate
- People who visit Disneyland aren't representative of the whole population

Convenience sampling coffee ratings

```
coffee_ratings["total_cup_points"].mean()
```

```
82.15120328849028
```

```
coffee_ratings_first10 = coffee_ratings.head(10)
```

```
coffee_ratings_first10["total_cup_points"].mean()
```

```
89.1
```

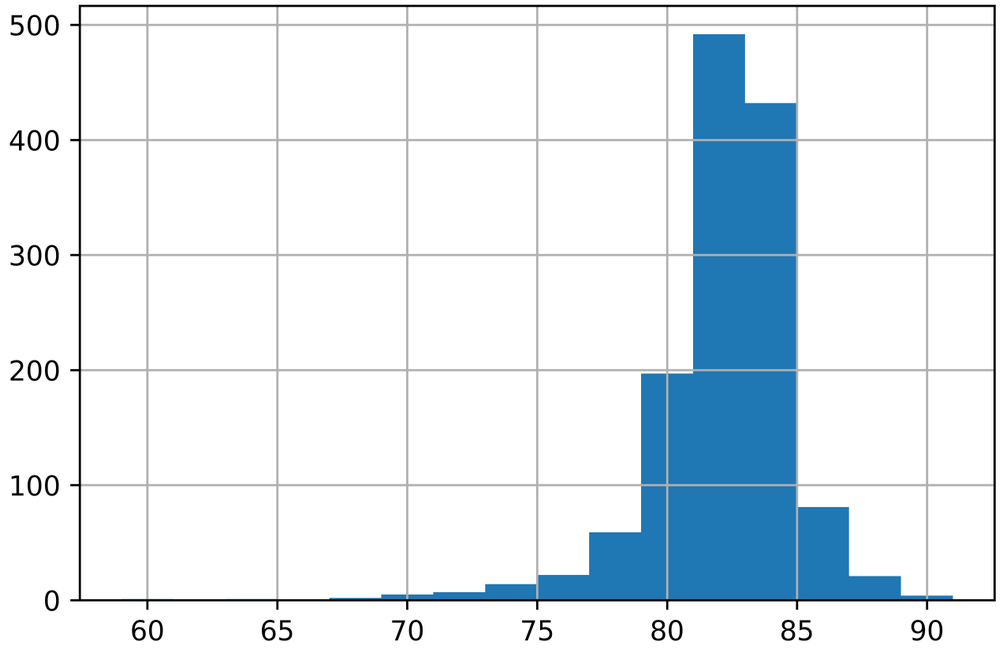
Visualizing selection bias

```
import matplotlib.pyplot as plt
import numpy as np
coffee_ratings["total_cup_points"].hist(bins=np.arange(59, 93, 2))
plt.show()
```

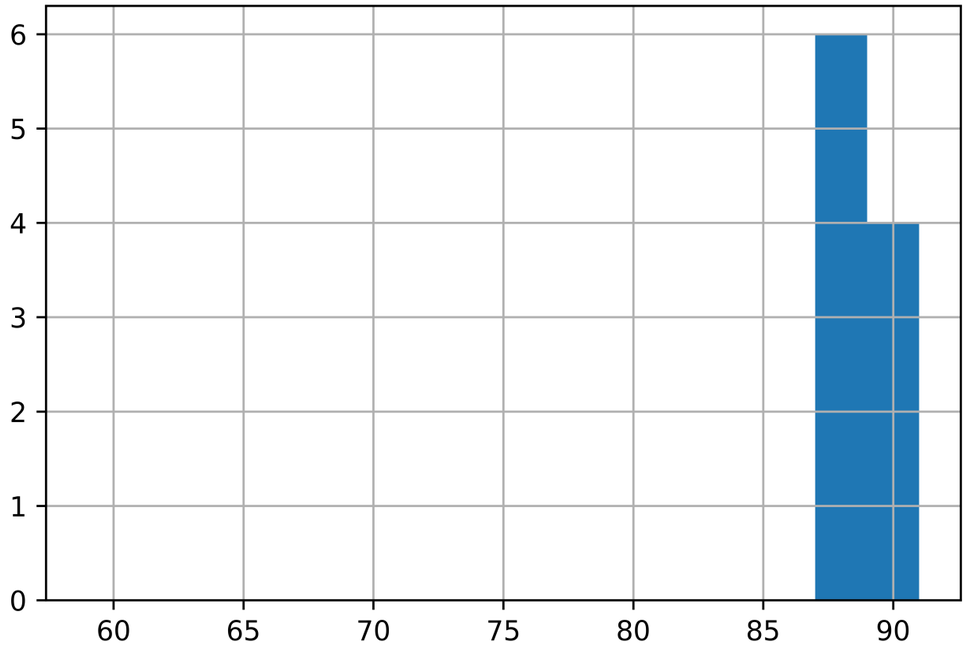
```
coffee_ratings_first10["total_cup_points"].hist(bins=np.arange(59, 93, 2))
plt.show()
```

Distribution of a population and of a convenience sample

Population:



Convenience sample:

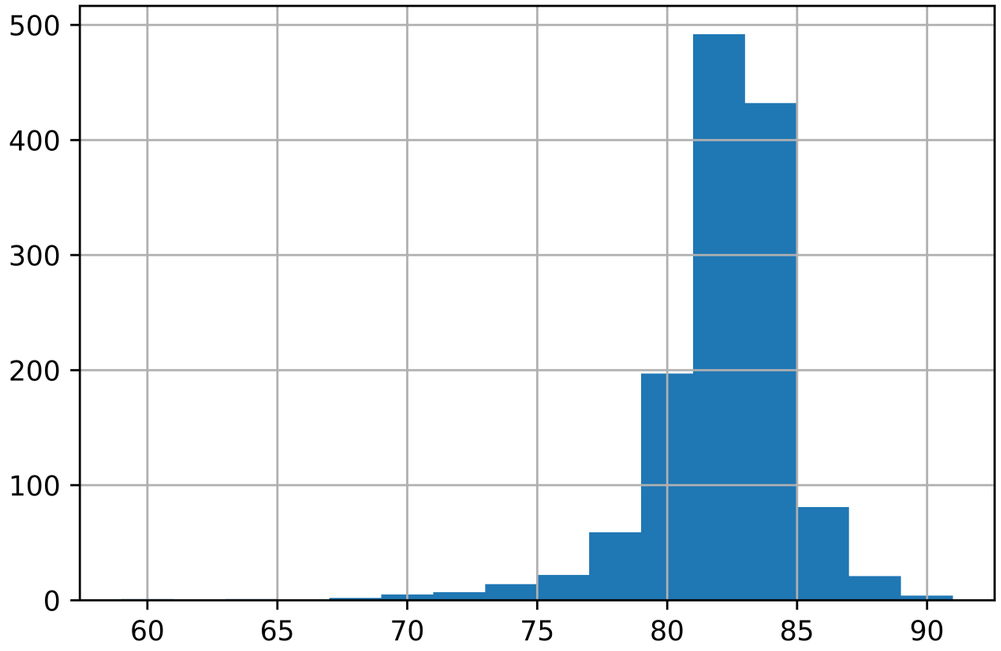


Visualizing selection bias for a random sample

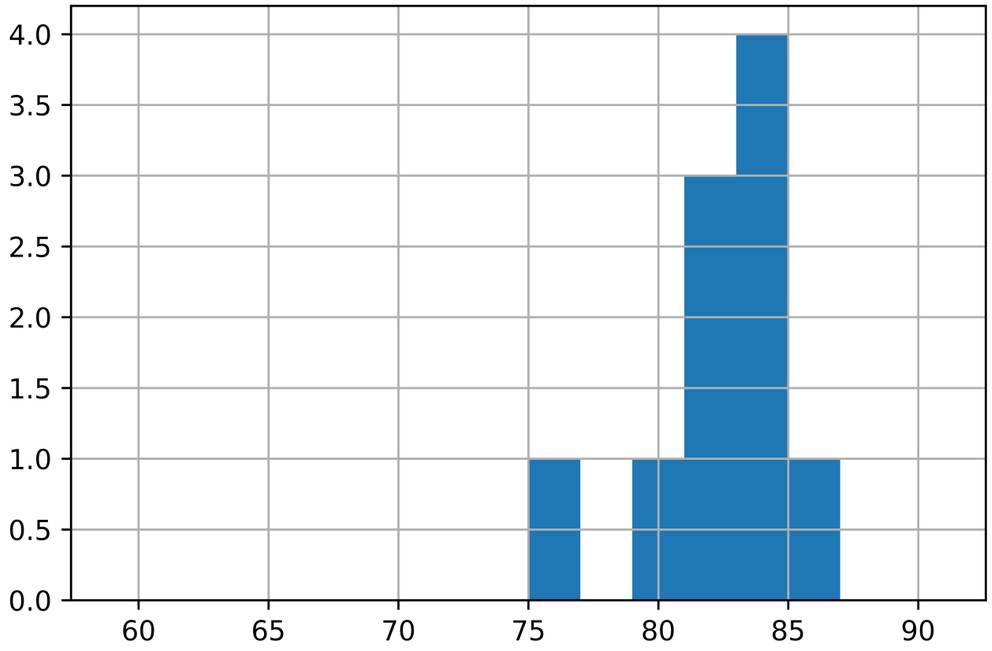
```
coffee_sample = coffee_ratings.sample(n=10)
coffee_sample["total_cup_points"].hist(bins=np.arange(59, 93, 2))
plt.show()
```

Distribution of a population and of a simple random sample

Population:



Random Sample:



Let's practice!

SAMPLING IN PYTHON

How does Sue do sampling?

SAMPLING IN PYTHON

James Chapman
Content Developer, DataCamp

datacamp

What does random mean?

> *{adjective}* made, done, happening, or chosen without method or conscious decision.

[1] Oxford Languages

True random numbers

- Generated from physical processes, like flipping coins
- Hotbits uses radioactive decay
- RANDOM.ORG uses atmospheric noise
- True randomness is expensive

[1] https://www.fourmilab.ch/hotbits [2] https://www.random.org

Pseudo-random number generation

- Pseudo-random number generation is **cheap** and **fast**
- Next "random" number calculated from previous "random" number
- The first "random" number calculated from a *seed*
- The same seed value yields the same random numbers

Pseudo-random number generation example

```
seed = 1
calc_next_random(seed)
```

```
3
```

```
calc_next_random(3)
```

```
2
```

```
calc_next_random(2)
```

```
6
```

Random number generating functions

- Prepend with `numpy.random` , such as `numpy.random.beta()`

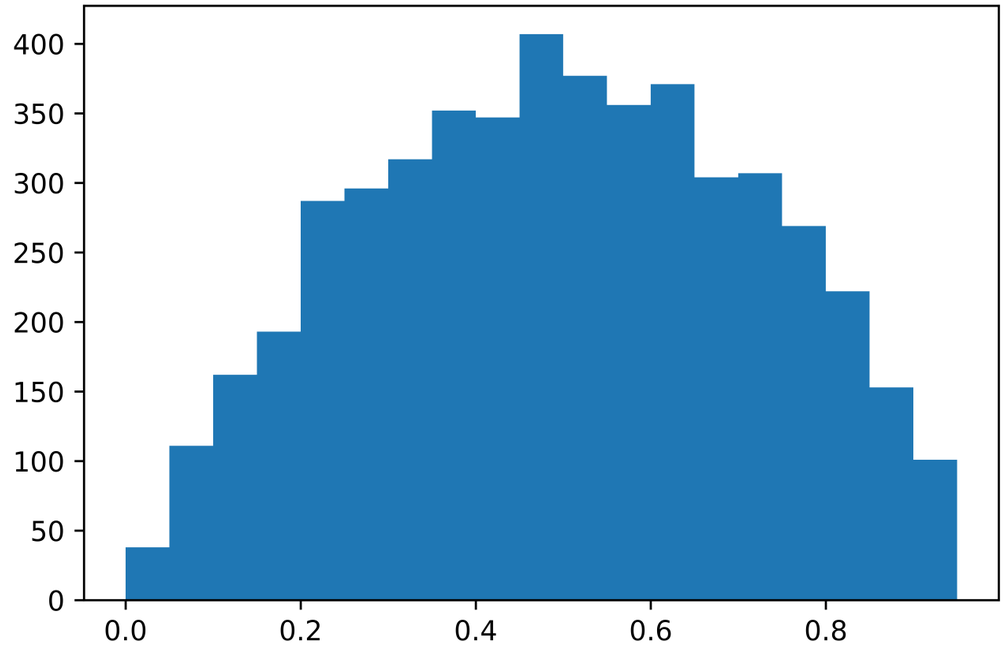
function	distribution	function	distribution
.beta	Beta	.hypergeometric	Hypergeometric
.binomial	Binomial	.lognormal	Lognormal
.chisquare	Chi-squared	.negative_binomial	Negative binomial
.exponential	Exponential	.normal	Normal
.f	F	.poisson	Poisson
.gamma	Gamma	.standard_t	t
.geometric	Geometric	.uniform	Uniform

Visualizing random numbers

```
randoms = np.random.beta(a=2, b=2, size=5000)
randoms
```

```
array([0.6208281 , 0.73216171, 0.44298403, ...,
       0.13411873, 0.52198411, 0.72355098])
```

```
plt.hist(randoms, bins=np.arange(0, 1, 0.05))
plt.show()
```



Random numbers seeds

```
np.random.seed(20000229)
```

```
np.random.normal(loc=2, scale=1.5, size=2)
```

```
array([-0.59030264, 1.87821258])
```

```
np.random.normal(loc=2, scale=1.5, size=2)
```

```
array([2.52619561, 4.9684949 ])
```


```
np.random.seed(20000229)
```

```
np.random.normal(loc=2, scale=1.5, size=2)
```

```
array([-0.59030264, 1.87821258])
```

```
np.random.normal(loc=2, scale=1.5, size=2)
```

```
array([2.52619561, 4.9684949 ])
```

Using a different seed

```
np.random.seed(20000229)
```

```
np.random.normal(loc=2, scale=1.5, size=2)
```

```
array([-0.59030264, 1.87821258])
```

```
np.random.normal(loc=2, scale=1.5, size=2)
```

```
array([2.52619561, 4.9684949 ])
```

```
np.random.seed(20041004)
```

```
np.random.normal(loc=2, scale=1.5, size=2)
```

```
array([1.09364337, 4.55285159])
```

```
np.random.normal(loc=2, scale=1.5, size=2)
```

```
array([2.67038916, 2.36677492])
```

Let's practice!

SAMPLING IN PYTHON