

9.2) Sampling and Empirical Distributions

Vitor Kamada

December 2019

Tables, Graphics, and Figures from

**Computational and Inferential Thinking:
The Foundations of Data Science**

Adhikari & DeNero (2019): Ch 10 Sampling and
Empirical Distributions

<https://www.inferentialthinking.com/>

Top Movies Dataset

```
import numpy as np
from datascience import *
path_data = 'https://github.com/data-8/textbook/raw/gh-pages/data/'
top1 = Table.read_table(path_data + 'top_movies.csv')
top2 = top1.with_column('Row Index', np.arange(top1.num_rows))
top = top2.move_to_start('Row Index')
top.set_format(make_array(3, 4), NumberFormatter)
```

Row Index	Title	Studio	Gross
0	Star Wars: The Force Awakens	Buena Vista (Disney)	906,723,418
1	Avatar	Fox	760,507,625
2	Titanic	Paramount	658,672,302

Deterministic Samples

```
top.take(make_array(3, 18, 100))
```

Row Index	Title	Studio	Gross
3	Jurassic World	Universal	652,270,625
18	Spider-Man	Sony	403,706,375
100	Gone with the Wind	MGM	198,676,459

```
top.where('Title', are.containing('Harry Potter'))
```

Row Index	Title	Studio	Gross
22	Harry Potter and the Deathly Hallows Part 2	Warner Bros.	381,011,219
43	Harry Potter and the Sorcerer's Stone	Warner Bros.	317,575,550

Systematic Sample

Random start among rows 0 through 9

```
start = np.random.choice(np.arange(10))  
top.take(np.arange(start, top.num_rows, 10))
```

Row Index	Title	Studio
2	Titanic	Paramount
12	The Hunger Games: Catching Fire	Lionsgate
22	Harry Potter and the Deathly Hallows Part 2	Warner Bros.

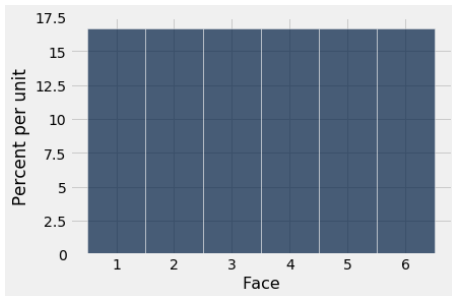
Probability Distribution

```
die = Table().with_column('Face', np.arange(1, 7, 1))
```

```
%matplotlib inline
```

```
import matplotlib.pyplot as plots  
plots.style.use('fivethirtyeight')  
die_bins = np.arange(0.5, 6.6, 1)  
die.hist(bins = die_bins)
```

$$\left(\frac{1}{6} \cong 0.17\right)$$

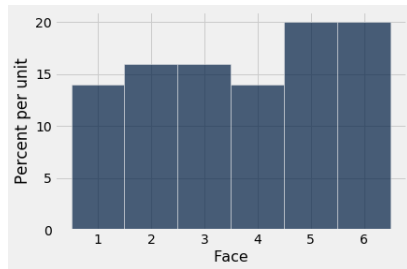
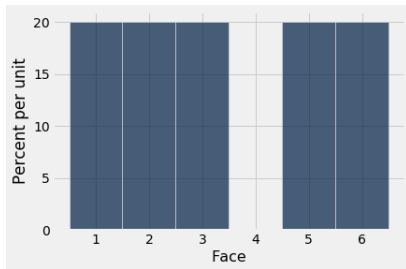


Empirical Distributions

```
def empirical_hist_die(n):  
    die.sample(n).hist(bins = die_bins)
```

```
empirical_hist_die(10)
```

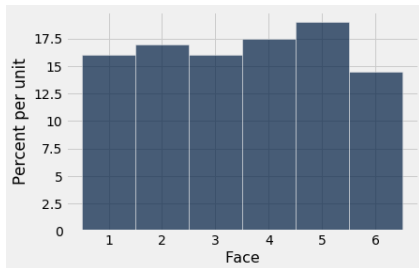
```
empirical_hist_die(100)
```



The Law of Averages

In the long run, the empirical probability gets closer and closer to the theoretical probability of the event

```
empirical_hist_die(1000)
```



$$\left(\frac{1}{6} \cong 0.17\right)$$

Independently and under identical conditions: every repetition is performed in the same way regardless of the results of all the other repetitions

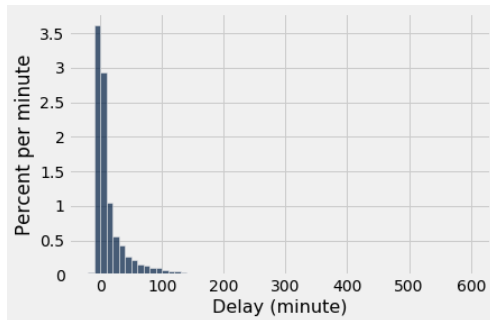
United Airlines domestic flights departing from San Francisco in Summer of 2015

```
united = Table.read_table(path_data + 'united_summer2015.csv')
```

Date	Flight Number	Destination	Delay
6/1/15	73	HNL	257
6/1/15	217	EWR	28
6/1/15	237	STL	-3

Normal Distribution

```
delay_bins = np.append(np.arange(-20, 301, 10), 600)  
united.hist('Delay', bins = delay_bins, unit = 'minute')
```

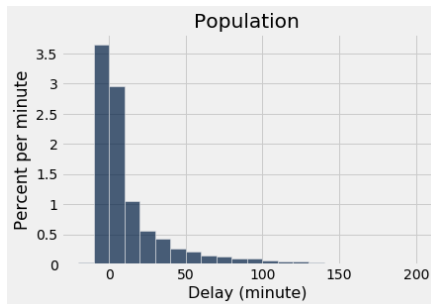


```
united.where('Delay', are.above(200)).num_rows/united.num_rows
```

0.008390596745027125

Ignore the 0.8% of flights

```
delay_bins = np.arange(-20, 201, 10)
united.hist('Delay', bins = delay_bins, unit = 'minute')
plots.title('Population');
```



```
united.where('Delay', are.between(0,
10)).num_rows/united.num_rows
```

0.2935985533453888

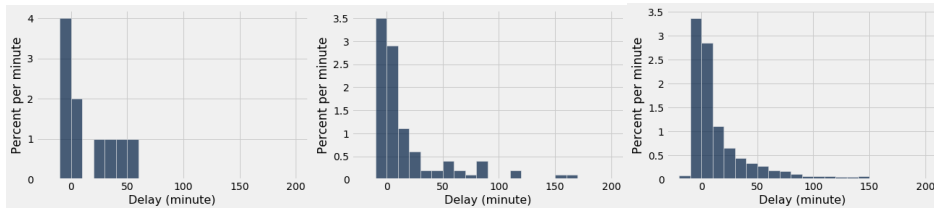
Random Sample converges to the Population

```
def empirical_hist_delay(n):  
    united.sample(n).hist('Delay',  
        bins = delay_bins, unit = 'minute')
```

```
empirical_hist_delay(10)
```

```
empirical_hist_delay(100)
```

```
empirical_hist_delay(1000)
```



Parameter vs Statistic

```
np.median(united.column('Delay'))    2.0
```

```
united.where('Delay',  
             are.below_or_equal_to(2)).num_rows / united.num_rows  
  
0.5018444846292948
```

```
united.where('Delay', are.equal_to(2)).num_rows  
  
480
```

```
np.median(sample_1000.column('Delay'))    1.0
```

```
np.median(united.sample(1000).column('Delay'))  
  
2.5
```

Simulating a Statistic 5,000

```
def random_sample_median():  
    return np.median(united.sample(1000).column('Delay'))  
  
medians = make_array()  
  
for i in np.arange(5000):  
    medians = np.append(medians, random_sample_median())  
  
simulated_medians = Table().with_column('Sample Median', medians)  
  
simulated_medians.hist(bins=np.arange(0.5, 5, 1))
```

