8.1) Conditional Statements, Iteration, and Simulation

Vitor Kamada

December 2019

Reference

Tables, Graphics, and Figures from

Computational and Inferential Thinking: The Foundations of Data Science

Adhikari & DeNero (2019): Ch 9 Randomness

https://www.inferentialthinking.com/

Conditional Statements

```
from datascience import *
import numpy as np
```

```
def bet_on_one_roll():
    """Returns my net gain on one bet"""
    # roll a die once and record the number of spots
    x = np.random.choice(np.arange(1, 7))
    if x <= 2:
        return -1
    elif x <= 4:
        return 0
    elif x <= 6:
        return 1
```

Loop or Iteration: 5 times

```
bet_on_one_roll()
```

```
-1
```

```
for i in np.arange(5):
    print(bet_on_one_roll())
```

```
1
1
1
0
-1
```

Collection Array

```
outcomes = make_array()
for i in np.arange(5):
    outcome_of_bet = bet_on_one_roll()
    outcomes = np.append(outcomes, outcome_of_bet)
outcomes
```

```
array([-1.,  0.,  1., -1.,  0.])
```

```
np.count_nonzero(outcomes)
```

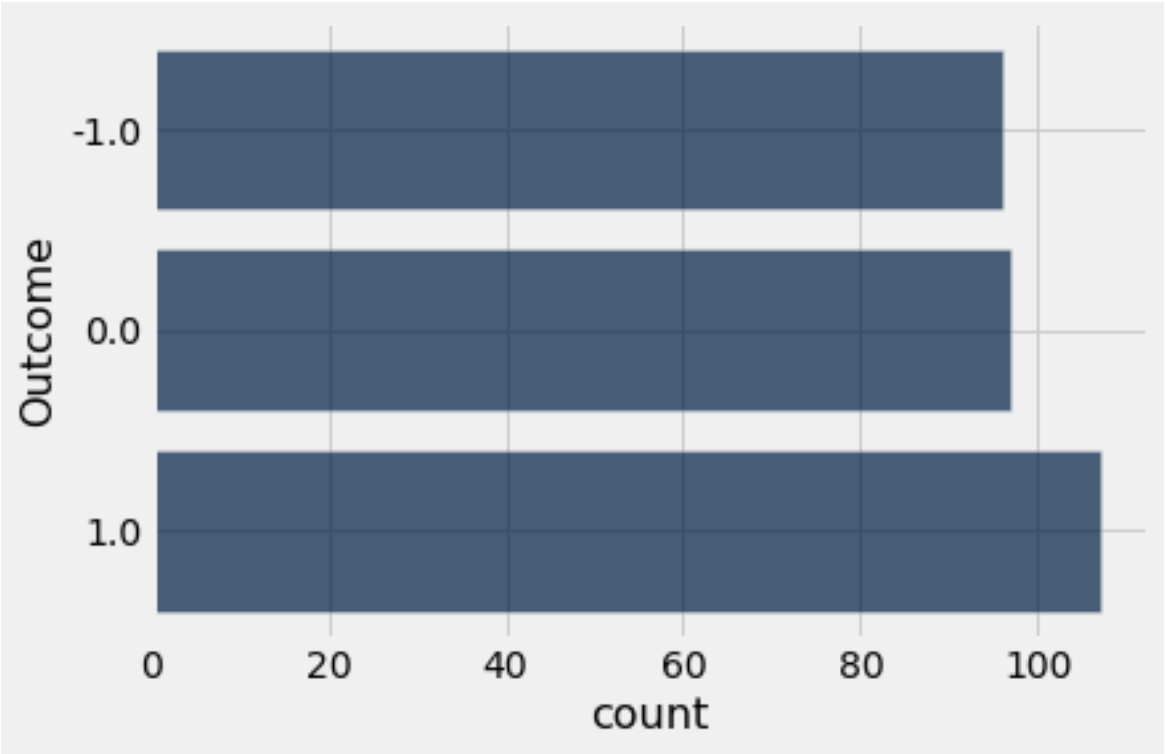
```
3
```

```
outcomes = make_array()
for i in np.arange(300):
    outcome_of_bet = bet_on_one_roll()
    outcomes = np.append(outcomes, outcome_of_bet)
len(outcomes)
```

```
300
```

Betting on 300 Rolls

```
%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
outcome_table = Table().with_column('Outcome', outcomes)
outcome_table.group('Outcome').barh(0)
```



Simulate 100 tosses

```
coin = make_array('Heads', 'Tails')
ten_tosses = np.random.choice(coin, 10)
```

```
array(['Tails', 'Heads', 'Heads', 'Heads', 'Tails', 'Tails', 'Tails'
       'Tails', 'Heads', 'Tails'], dtype='<U5')
```

```
np.count_nonzero(ten_tosses == 'Heads')
```

4

```
outcomes = np.random.choice(coin, 100)
num_heads = np.count_nonzero(outcomes == 'Heads')
```

47

Simulating 10,000 repetitions

```
heads = make_array()

num_repetitions = 10000

for i in np.arange(num_repetitions):
    outcomes = np.random.choice(coin, 100)
    heads = np.append(heads, np.count_nonzero(outcomes == 'Heads'))

heads
```

```
array([50., 45., 56., ..., 54., 57., 48.])
```

```
len(heads)
```

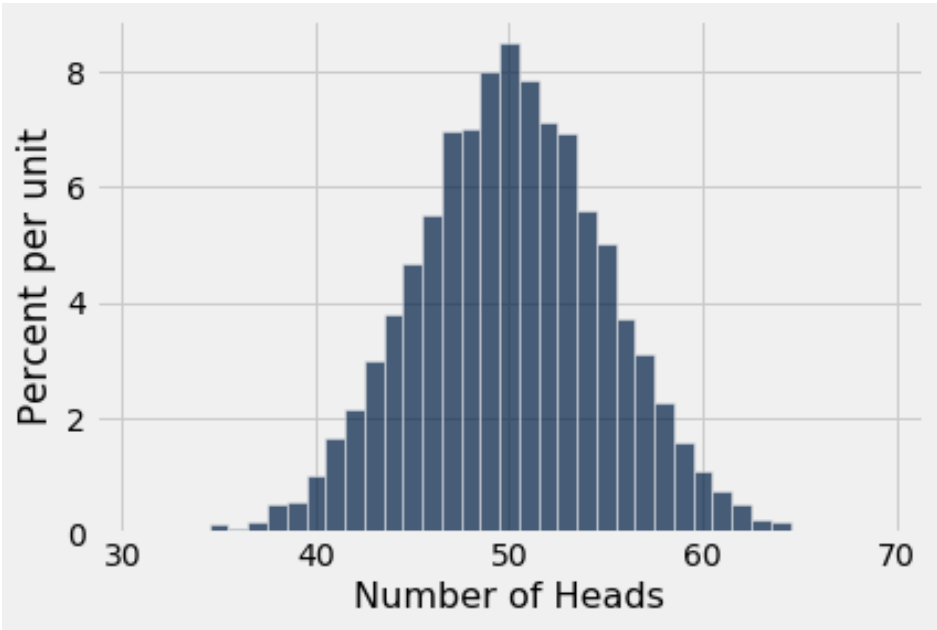
```
10000
```

Normal Distribution

```
simulation_results = Table().with_column(
    'Repetition', np.arange(1, num_repetitions + 1),
    'Number of Heads', heads
)
```

```
simulation_results.hist('Number of Heads',
          bins = np.arange(30.5, 69.6, 1))
```

Repetition	Number of Heads
1	50
2	45
3	56
4	45



Two Rolls of a Die

```
die = np.arange(1, 7)
sum(np.random.choice(die, 2))
```

6

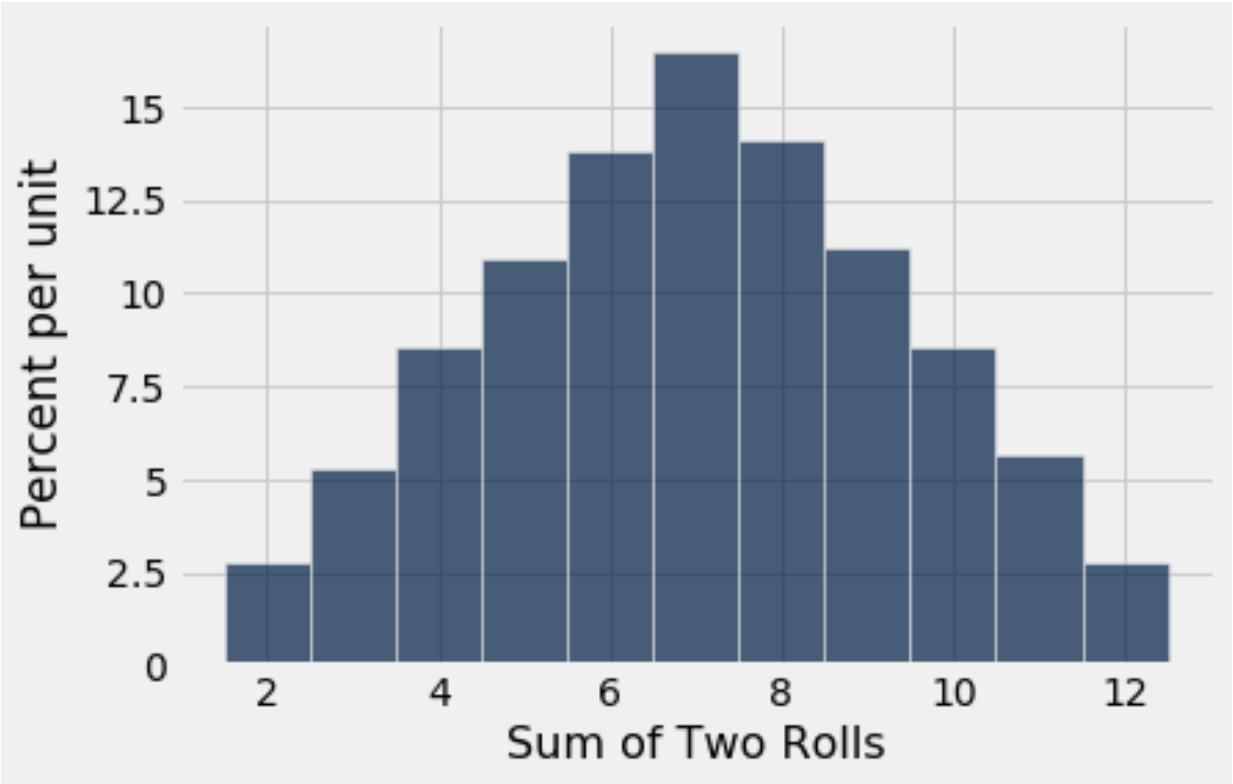
```
moves = make_array()

num_repetitions = 10000

for i in np.arange(num_repetitions):
    one_move = sum(np.random.choice(die, 2))
    moves = np.append(moves, one_move)
```

Simulation 10,000 times

```
results = Table().with_column(
    'Repetition', np.arange(1, num_repetitions + 1),
    'Sum of Two Rolls', moves
)
```



Chance that six spot comes up at least once in rolls of a die

$$P(\neg 6) = P(1) + P(2) + P(3) + P(4) + P(5) = \frac{5}{6}$$

$$P(at\_least\_one\_6\_in\_two\_rolls)$$

$$= 1 - P(both\_rolls\_are\_not\_6) = 1 - (\frac{5}{6})^2$$

$$P(at\_least\_one\_17\_in\_two\_rolls) = 1 - (\frac{5}{6})^{17}$$

The chance that a 6 appears at least once:

```
rolls = np.arange(1, 51, 1)
results = Table().with_columns(
    'Rolls', rolls,
    'Chance of at least one 6', 1 - (5/6)**rolls
)
```

Rolls	Chance of at least one 6
1	0.166667
2	0.305556
3	0.421296
4	0.517747