Lab Assignment 3

March 31, 2022

```
[1]: from datascience import *
     import numpy as np

     %matplotlib inline
     import matplotlib.pyplot as plots
     plots.style.use('fivethirtyeight')
```

1 Question 1

```
[2]: flights = Table.read_table('united_summer2015.csv')
     flights
```

```
[2]: Date     | Flight Number | Destination | Delay
     6/1/2015 | 73            | HNL         | 257
     6/1/2015 | 217           | EWR         | 28
     6/1/2015 | 237           | STL         | -3
     6/1/2015 | 250           | SAN         | 0
     6/1/2015 | 267           | PHL         | 64
     6/1/2015 | 273           | SEA         | -6
     6/1/2015 | 278           | SEA         | -8
     6/1/2015 | 292           | EWR         | 12
     6/1/2015 | 300           | HNL         | 20
     6/1/2015 | 317           | IND         | -10
     … (13815 rows omitted)
```

```
[3]: delay= flights.select('Delay')
```

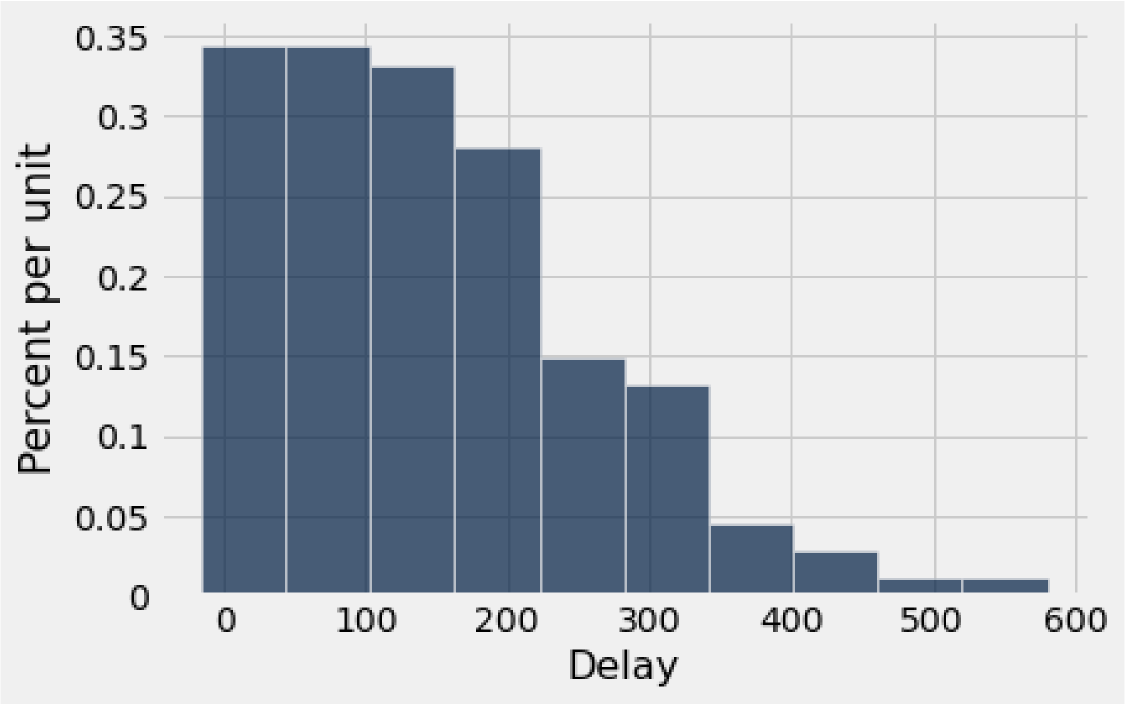
```
[4]: delay_distribution = delay.group('Delay')
     delay_distribution
```

```
[4]: Delay | count
     -16   | 2
     -15   | 1
     -14   | 1
     -13   | 8
     -12   | 16
     -11   | 27
```

```
-10   | 44
-9    | 123
-8    | 207
-7    | 313
… (283 rows omitted)
```

[5]:

```
delay_distribution.sort('count', descending=True).hist('Delay')
```



[6]:

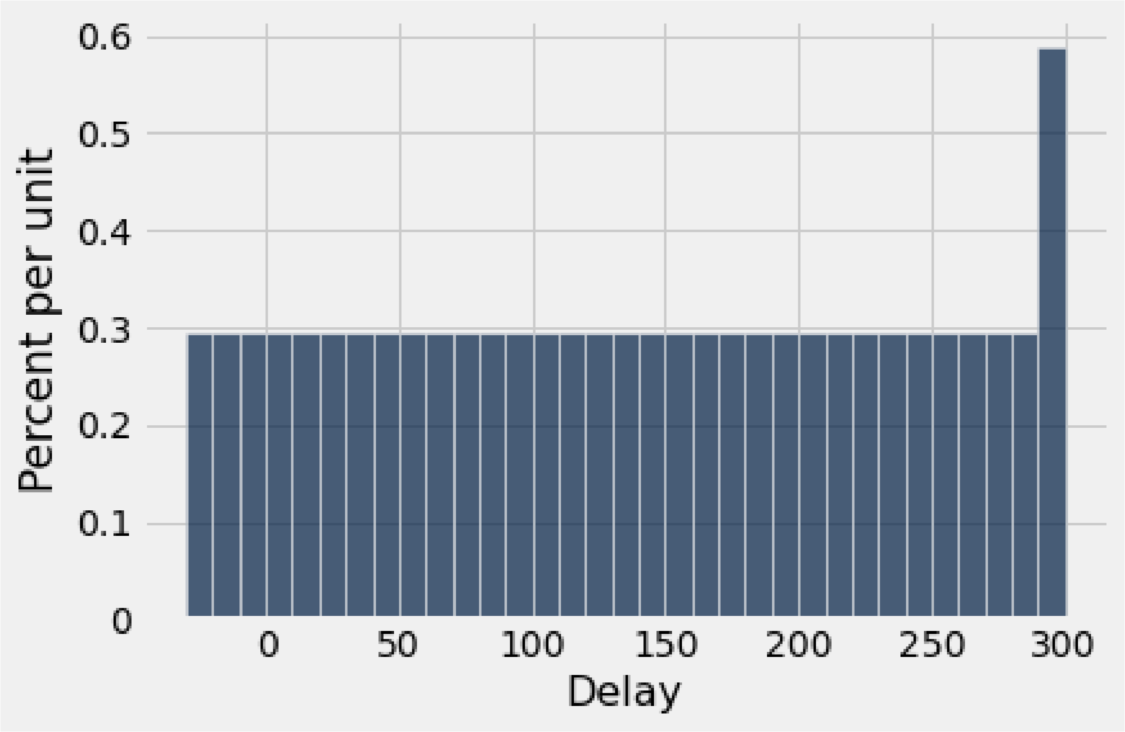
```
die = Table().with_column('Delay', np.arange(-30, 301, 10))
die
```

[6]:

```
Delay
-30
-20
-10
0
10
20
30
40
50
60
… (24 rows omitted)
```

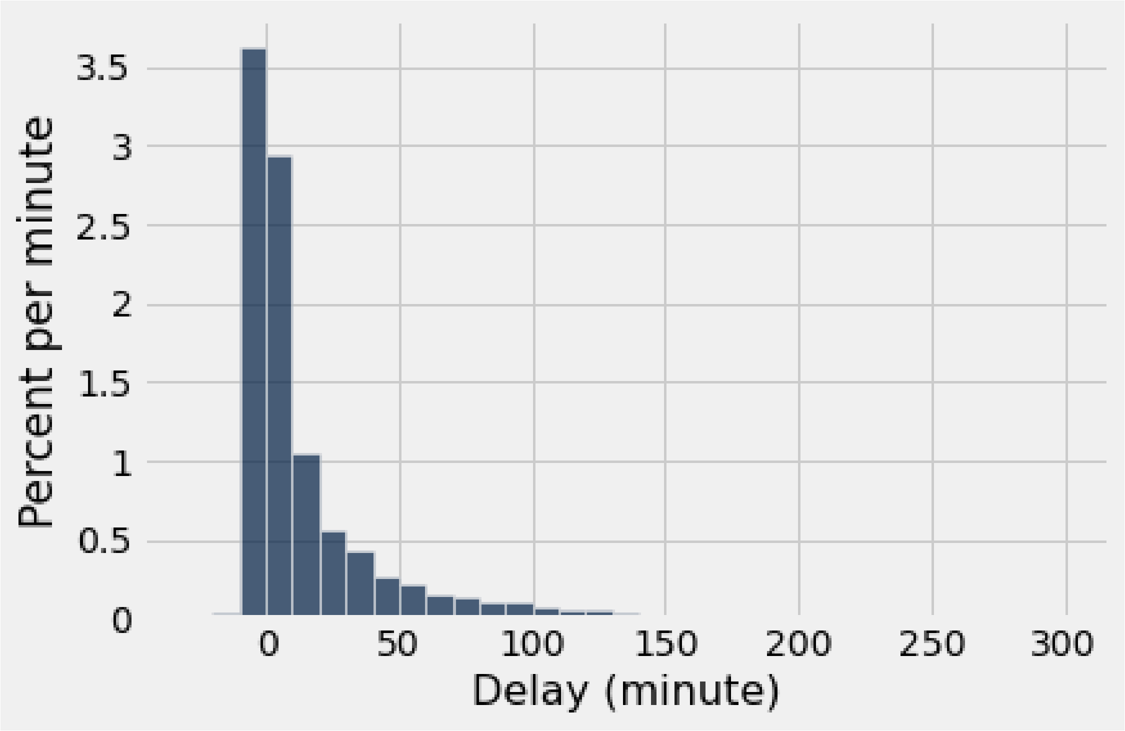
[7]:

```
die_bins = np.arange(-30, 301, 10)
die.hist(bins = die_bins)
```



[8]:

```
delay_bins = np.arange(-30, 301, 10)
flights.hist('Delay', bins = delay_bins, unit = 'minute')
```



Most flights arrived on time and very few flights arrived late.

[9]:
```
flights.where('Delay', are.between(10, 20)).num_rows/flights.num_rows
```

[9]:
```
0.10452079566003616
```

[10]:
```
def random_sample_median():
    return np.median(flights.sample(1000).column('Delay'))
```

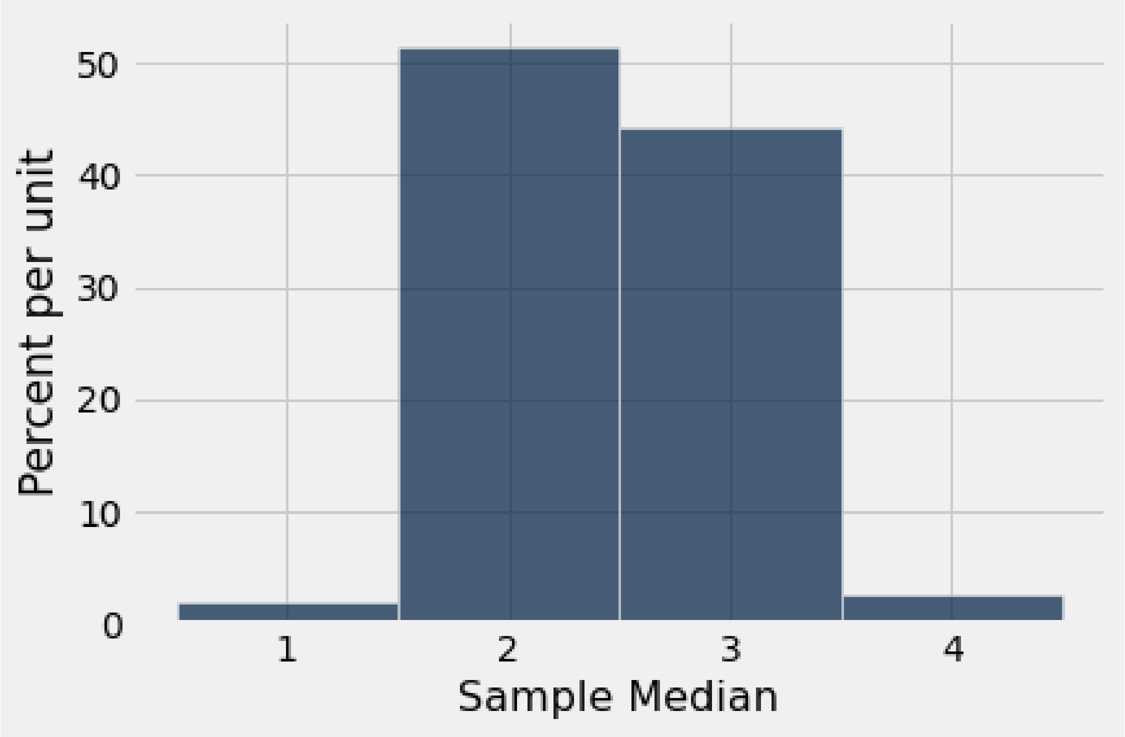
[11]:
```
medians = make_array()

for i in np.arange(5000):
    medians = np.append(medians, random_sample_median())
```

[12]:
```
simulated_medians = Table().with_column('Sample Median', medians)
simulated_medians
```

[12]:
```
Sample Median
4
3
2
3
3
2
```

```
2
2
2
2
… (4990 rows omitted)
```

[13]:
```
simulated_medians.hist(bins=np.arange(0.5, 5, 1))
```



The value 2 is the most probable

2 Question 2

[107]:
```
card = Table.read_table('deck.csv')
card
```

[107]:
```
Rank | Suit
2    |
2    |
2    |
2    |
3    |
3    |
3    |
3    |
4    |
4    |
… (42 rows omitted)
```

[119]:

```
rank_and_suit = card.select('Rank', 'Suit')
```

[120]:

```
suit_distribution = rank_and_suit.group('Suit')
suit_distribution
```

[120]:

```
Suit | count
     | 13
     | 13
     | 13
     | 13
```

[122]:

```
rank_distribution = rank_and_suit.group('Rank').show(13)
rank_distribution
```

```
<IPython.core.display.HTML object>
```

[126]:

```
# Simple random sample of 5000 cards
five_cards = card.sample(5000, with_replacement=True)
five_cards
```

[126]:

```
Rank | Suit
3    |
8    |
A    |
10   |
6    |
9    |
K    |
6    |
2    |
6    |
… (4990 rows omitted)
```

The optimal way is with_replacement equal true

[163]:

```
possible_point_values = np.arange(1, 14)
tosses = 4
total_score = 2
for i in np.arange(tosses):
    total_score = total_score + np.random.choice(possible_point_values)
total_score
```

[163]:

```
42
```

[164]:

```
card = np.arange(1, 13)
(np.random.choice(card))
```

[164]:

```
7
```

[165]:

```
def one_simulated_move():
    return (np.random.choice(card))
```

[166]:

```
num_repetitions = 10000

moves = make_array()
for i in np.arange(num_repetitions):
    new_move = one_simulated_move()
    moves = np.append(moves, new_move)
```

[167]:

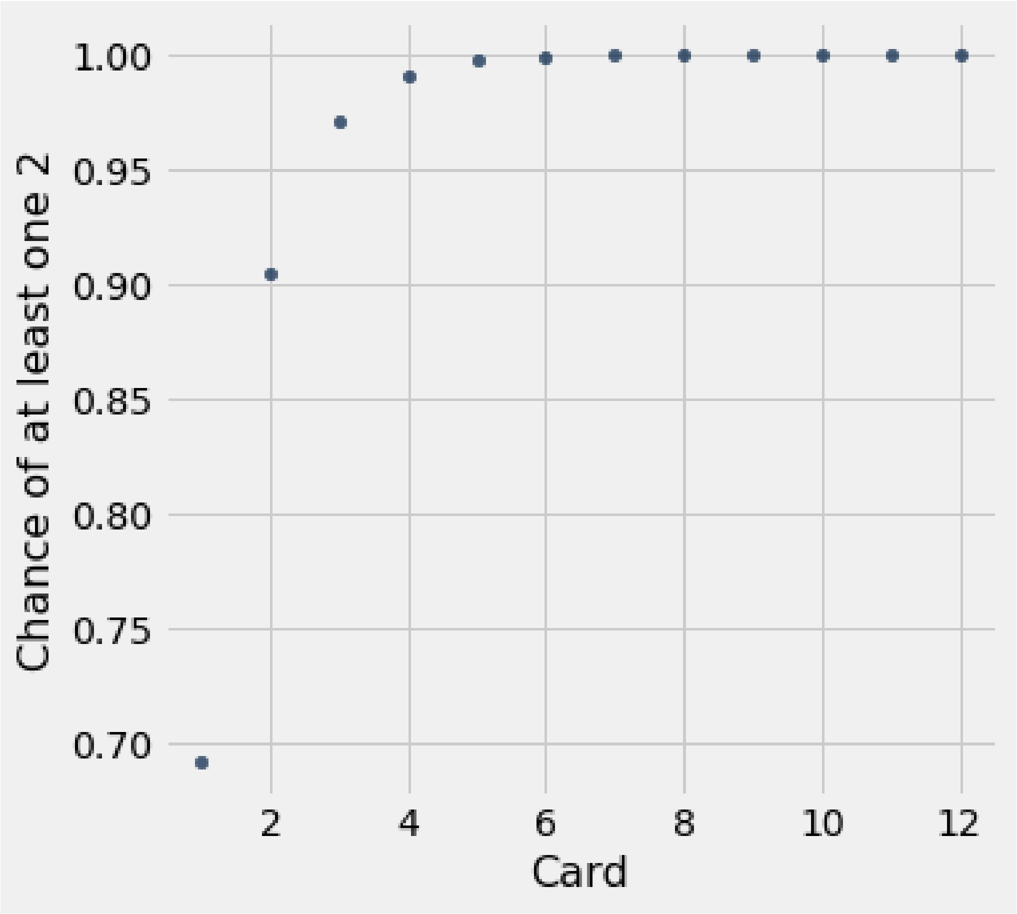
```
card = np.arange(1, 13, 1)
results = Table().with_columns(
    'Card', card,
    'Chance of at least one 2', 1 - (4/13)**card
)
results
```

[167]:

```
Card | Chance of at least one 2
1    | 0.692308
2    | 0.905325
3    | 0.970869
4    | 0.991037
5    | 0.997242
6    | 0.999151
7    | 0.999739
8    | 0.99992
9    | 0.999975
10   | 0.999992
… (2 rows omitted)
```

[168]:

```
results.scatter('Card')
```


Chance of at least one 2
1.00
0.95
0.90
0.85
0.80
0.75
0.70
2
4
6
8
10
12
Card

[ ]:

[ ]: