

Poker hand simulation - royal straight flush probability

Global parameters

```
In [1]: sample_size = 30
simulation_size = 700000
poker_hand_size = 5
confidence = 0.95
```

Create class for deck cards

```
In [2]: from enum import Enum

class Suit(Enum):
    __order__ = 'CLUBS DIAMONDS HEARTS SPADES'
    CLUBS = 1
    DIAMONDS = 2
    HEARTS = 3
    SPADES = 4

class Card:
    def __init__(self, value, suit):
        self.value = value
        self.suit = suit
```

Create deck

```
In [3]: deck = []
for v in range(13):
    for s in Suit:
        deck.append(Card(v, s))
```

Declare simulation function

```

In [4]: import random

def runSimulation():

    royal_straight_flush = 0

    for n in range(simulation_size):

        #shuffle deck
        random.shuffle(deck)

        #counts
        suited_count = 1
        straight_count = 1
        royal_count = 1

        #check if hand is suited
        suit = deck[0].suit
        for i in range(poker_hand_size):
            if(deck[i].suit != suit):
                suited_count = 0
                break

        #check if hand is straight
        hand = []
        for i in range(poker_hand_size):
            hand.append(deck[i].value)
        hand.sort()
        for i in range(poker_hand_size - 1):
            if(hand[i] != hand[i + 1] - 1):
                straight_count = 0

        #check if hand is royal
        if(hand[0] != 0):
            royal_count = 0

        #count
        if suited_count == 1 and straight_count == 1 and royal_count == 1:
            royal_straight_flush += 1

    probability = royal_straight_flush / simulation_size

    return probability

```

Run simulations

```

In [5]: data = []
        for i in range(sample_size):
            probability = runSimulation()
            data.append(probability)

```

Print statistics

```
In [6]: import pandas as pd

df = pd.DataFrame(data)
df.describe()
```

Out[6]:

	0
count	30.000000
mean	0.000002
std	0.000002
min	0.000000
25%	0.000000
50%	0.000001
75%	0.000003
max	0.000006

```
In [7]: import numpy as np
import scipy.stats

a = 1.0 * np.array(data)
m, se = np.mean(a), scipy.stats.sem(a)
h = se * scipy.stats.t.ppf((1 + confidence) / 2., sample_size - 1)
print("confidence interval is: [" + str(m - h) + ", " + str(m + h) + '

confidence interval is: [1.1469745451219004e-06, 2.281596883449528e-
06] at 95%
```

Print global parameters

```
In [8]: print("confidence: " + str(confidence))
print("sample_size: " + str(sample_size))
print("simulation_size: " + str(simulation_size))
print("poker_hand_size: " + str(poker_hand_size))

confidence: 0.95
sample_size: 30
simulation_size: 700000
poker_hand_size: 5
```