AI Investigation: Using "Rock Paper Scissors" to experiment using Confidence Intervals to determine inconclusive victories

**with double mean CI instead of single mean CI*

AI Investigation Part 3+

Goal: further test using Confidence Intervals to determine inconclusive victories: 2+ options and non-binary game results

Also amended previous problem of using single mean to construct CI

Creator: `Lang (Ron) Chen 2022`

In [1]:
```
import random
import numpy as np
import statistics as s
import scipy.stats
```

Define Game

Rock Paper Scissors

```
Win chance 1/3; Lose chance 1/3; Draw chance 1/3
```

Set a random variable X for the result of the game to be 1 (if win), 0 (if draw), -1 (if lose)

Thus, `E(X) = 0` ; `V(X) = 2/3`

In [2]:
```
CHOICE = ['R', 'P', 'S']
```

In [3]:
```
def winloss(player1, player2):
    if player1 == 'R':
        if player2 == 'S':
            return 1
        elif player2 == 'P':
            return -1

    elif player1 == 'P':
        if player2 == 'R':
            return 1
        elif player2 == 'S':
            return -1

    else:
        if player2 == 'P':
            return 1
        elif player2 == 'R':
            return -1

    return 0
```

Simulation

In [4]:

```
RUNS = 1000000

data = {'R': list(), 'P': list(), 'S': list()}

sample = list()
victory = list()
for i in range(RUNS):
    obs1 = random.sample(CHOICE, 1)[0]
    obs2 = random.sample(CHOICE, 1)[0]

    data[obs1].append(winloss(obs1, obs2))
```

Algorithm for final choice

First: Manipulate data so that it is in a dictionary and the dictionary value is [mean, stdev, length]

In [5]:

```
stats = dict()
for choice in CHOICE:
    tmp = list()
    tmp.append(s.mean(data[choice]))
    tmp.append(s.stdev(data[choice]))
    tmp.append(len(data[choice]))

    stats[choice] = tmp

stats
```

Out[5]:

```
{'R': [0.0006029336772954975, 0.8167883765719353, 333370],
 'P': [0.0008683685488329607, 0.8159248986436104, 332808],
 'S': [-0.0006021172960439995, 0.8164663959420605, 333822]}
```

In [6]:

```
def final_choice(stats):

    stats.sort(key = lambda x:x[1][0], reverse = True)

    # Then tests whether the second, third and so forth values contain 0 within their joi
    inconclusive_list = [stats[0][0]]
    inconclusive = False
    for i in range(1, len(stats)):
        if in_range(construct_CI(stats[0], stats[i])):
            inconclusive_list.append(stats[i][0])
            inconclusive = True
        else: # Because all values are sorted, if the current choice's joint Confidence In
            break

    if inconclusive: # If inconclusive, return statement with the list of 'drawed' choices
        return f'Inconclusive: the following came to a draw {inconclusive_list}'

    return stats[0][0] # Else, return the dominant strategy
```

In [7]:

```
def construct_CI(stat1, stat2):
    """ Uses Welch's approximation to construct a joint CI of two means, unknown populatio

    xbar1 = stat1[1][0]
    xbar2 = stat2[1][0]
    s1 = stat1[1][1]
    s2 = stat2[1][1]
    n = stat1[1][2]
    m = stat2[1][2]
```

```python
    r = (s1**2 /n + s2**2 /m)**2 /(s1**4 /(n**2 * (n-1)) + s2**4 /(m**2 * (m-1)))

    q = scipy.stats.t.ppf(0.95, df = r)

    poolsd = np.sqrt(s1**2 /n + s2**2 /m)

    CI = (xbar1 - xbar2 - q * poolsd, xbar1 - xbar2 + q * poolsd)

    print(f'{stat1[0]} {stat2[0]}: {CI}')
    print('\n')

    return CI
```

In [8]:

```python
def in_range(CI):
    """ Helper function to check whether 0 is within the Confidence Interval """

    if CI[0] <= 0 and CI[1] >= 0:
        return True
    return False
```

In [9]:

```python
stats
```

Out[9]:

```
{'R': [0.0006029336772954975, 0.8167883765719353, 333370],
 'P': [0.0008683685488329607, 0.8159248986436104, 332808],
 'S': [-0.0006021172960439995, 0.8164663959420605, 333822]}
```

In [10]:

```python
final_choice(list(stats.items()))
```

```
P R: (-0.0030249194161819553, 0.0035557891592568813)


P S: (-0.0018181062651184583, 0.004759077954872379)
```

Out[10]:

```
"Inconclusive: the following came to a draw ['P', 'R', 'S']"
```

A few words on experimental results:

Overall when using the 2-mean CI the results were better. However there were still times when the 95% CI did not give the correct answer of all three being draws. Thus conducting a proper experiment for what % CI to use is vital.

This time did not test Normal distribution because most of the time we work assuming we don't know the population variance.