# Assignment 2 - Monty Hall - Ben Polasek

January 28, 2019

```
In [1]: import numpy as np
        import pandas as pd
        import random
        import timeit
        import matplotlib.pyplot as plt
```

## 0.1 Step 1: Building piece by piece

```
In [2]: #place a car behind one door represented as '1' and goats behind two represented as '2
        doors = [0,0,1]
        np.random.shuffle(doors)
        doors

Out[2]: [1, 0, 0]
```

```
In [3]: #choose one door
        choice = np.asscalar(np.random.choice(3, 1))
        choice

Out[3]: 0
```

```
In [4]: #open one of the doors with a goat (0) behind it
        open_door = [0,1,2]
        del open_door[choice]
        if doors[choice] == 1:
            #del open_door[choice]
            del open_door[random.randint(0,1)]
        else:
            for i in range (0,2):
                if doors[i] == 1:
                    del open_door[i]
                    #del open_door[random.randint(0,1)]
        open_door = open_door[0]
        open_door

Out[4]: 2
```

```
In [5]: #Strategy 1: Don't change doors
        win_rate = 0
        if doors[choice] == 1:
            win_rate = (win_rate + 1)
            print('Win')
        else:
            print('Lose')
```

```
Win
```

```
In [6]: #Strategy 2: Change doors
        x = 3 - choice - open_door
        choice = x
        if doors[choice] == 1:
            win_rate = (win_rate + 1)
            print('Win')
        else:
            print('Lose')
```

```
Lose
```

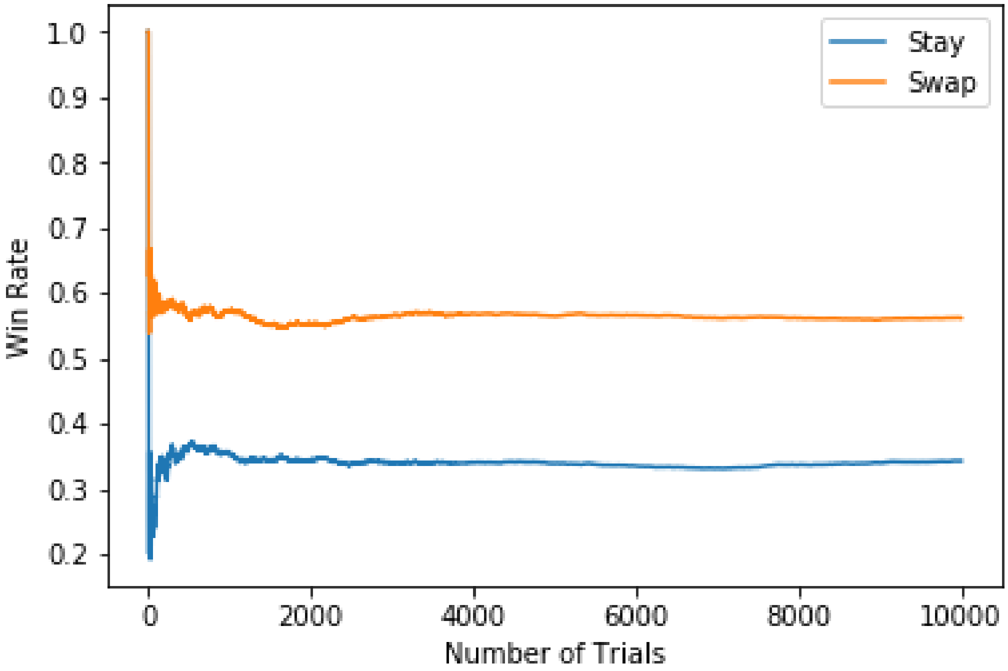
## 0.2 Step 2: Glue pieces together

```
In [7]: #The stay strategy
        n = 10000
        doors = [0,0,1]
        stay_rate = []
        def stay():
            stay_rate.clear()
            win_rate = 0
            for j in range (1,n):
                #shuffle doors
                #doors = [0,0,1]
                np.random.shuffle(doors)

                #pick a random door
                choice = np.asscalar(np.random.choice(3, 1))

            #Monty opens a door with a goat behind it
                open_door = [0,1,2]
                del open_door[choice]
                if doors[choice] == 1:
                    #del open_door[choice]
                    del open_door[random.randint(0,1)]
                else:
                    for i in range (0,2):
                        if doors[i] == 1:
                            del open_door[i]
                            #del open_door[random.randint(0,1)]
                open_door = open_door[0]

                #We stay with our decision
                if doors[choice] == 1:
                    win_rate = (win_rate + 1)

                stay_rate.append(win_rate/j)
            print("The win rate for the stay strategy after", n ,"trails is: ", win_rate/j)
```

```
In [8]: #The switch strategy
        n = 10000
        doors = [0,0,1]
        switch_rate = []
        def switch():
            switch_rate.clear()
            win_rate = 0
            for j in range (1,n):
                #shuffle doors
                #doors = [0,0,1]
                np.random.shuffle(doors)

                #pick a random door
                choice = np.asscalar(np.random.choice(3, 1))

            #Monty opens a door with a goat behind it
                open_door = [0,1,2]
                del open_door[choice]
                if doors[choice] == 1:
                    #del open_door[choice]
                    del open_door[random.randint(0,1)]
                else:
                    for i in range (0,2):
                        if doors[i] == 1:
                            del open_door[i]
                            #del open_door[random.randint(0,1)]
                open_door = open_door[0]

                #We switch doors
                x = 3 - choice - open_door
                choice = x
                if doors[choice] == 1:
                    win_rate = (win_rate + 1)


                switch_rate.append(win_rate/j)
            print("The win rate for the switch strategy after",n ,"trails is: ", win_rate/j)
```

```
In [11]: stay()
         switch()
         plt.plot(stay_rate, label='Stay')
         plt.plot(switch_rate, label ='Swap')
         plt.xlabel('Number of Trials')
         plt.ylabel('Win Rate')
         plt.legend()
         plt.show()
```

```
The win rate for the stay strategy after 10000 trails is:  0.343034303430343
The win rate for the switch strategy after 10000 trails is:  0.5615561556155616
```



As we can see, the stay strategy has a win rate of ~35% (approximately 1/3) and the switch strategy has a winrate of about ~56% (approximately 2/3) so we conclude that always switching is the better strategy