分析彩票游戏的期望值与风险

1. 引言

本报告旨在通过计算机模拟分析一个简单彩票游戏的期望收益和风险。在此游戏中，玩家选择六个数字，如果这些数字与随机生成的六个彩票号码匹配，则可获得奖金。奖金的多少取决于匹配的号码数量，范围从$5到$100,000不等。

2. 方法论

我们使用Python语言进行编程，利用其强大的随机数生成和数据处理能力，以及matplotlib库进行数据可视化。

首先，通过random.sample函数生成一组随机的彩票号码。

1. **def** generate\_lottery\_numbers():
2. """
3. 生成彩票号码。
4. 返回一个包含6个从1到21的随机不重复数字的集合。
5. """
6. **return** set(random.sample(range(1, 22), 6))

随后，我们定义了一个函数来根据匹配的号码数量计算奖金。

1. **def** calculate\_prize(matches):
2. """
3. 根据匹配的数字数量计算奖金。
4. :param matches: 匹配的数字数量
5. :return: 对应的奖金金额
6. """
7. prize\_dict = {0: 0, 1: 5, 2: 5, 3: 10, 4: 100, 5: 100, 6: 100000}
8. **return** prize\_dict.get(matches, 0)

最后，通过模拟多次游戏来估计平均成本和奖金。

1. **def** simulate\_lottery\_plays(num\_plays):
2. """
3. 模拟彩票游戏的一系列玩法。
4. :param num\_plays: 游戏的次数
5. :return: 总花费和总奖金
6. """
7. total\_cost = num\_plays \* 10  # 每次游戏的花费是$10
8. total\_prize = 0
10. **for** \_ **in** range(num\_plays):
11. lottery\_numbers = generate\_lottery\_numbers()
12. player\_numbers = generate\_lottery\_numbers()
13. matches = len(lottery\_numbers.intersection(player\_numbers))
14. total\_prize += calculate\_prize(matches)
16. **return** total\_cost, total\_prize
18. **def** average\_simulation(num\_plays, num\_experiments):
19. """
20. 对一个特定的游戏次数进行多次实验以获取平均结果。
21. :param num\_plays: 游戏次数
22. :param num\_experiments: 实验次数
23. :return: 平均总花费和平均总奖金
24. """
25. total\_costs = []
26. total\_prizes = []
28. **for** \_ **in** range(num\_experiments):
29. cost, prize = simulate\_lottery\_plays(num\_plays)
30. total\_costs.append(cost)
31. total\_prizes.append(prize)
33. avg\_cost = sum(total\_costs) / num\_experiments
34. avg\_prize = sum(total\_prizes) / num\_experiments
35. **return** avg\_cost, avg\_prize

3. 实验设计

实验被设计为对不同次数的游戏（100次、1000次、100000次）进行多次（10次）模拟，以获得稳定和可靠的平均结果。每次游戏的成本设定为$10。

1. # 准备数据
2. num\_plays\_list = [100, 1000, 100000]
3. num\_experiments = 10
4. avg\_costs = []
5. avg\_prizes = []
7. # 对每个游戏次数执行多次实验并收集数据
8. **for** num\_plays **in** num\_plays\_list:
9. avg\_cost, avg\_prize = average\_simulation(num\_plays, num\_experiments)
10. avg\_costs.append(avg\_cost)
11. avg\_prizes.append(avg\_prize)
12. **print**(
13. f"Playing {num\_plays} times: Average Cost = ${avg\_cost}, Average Prize = ${avg\_prize}, Expected Maximum Loss = ${avg\_cost - avg\_prize}")

4. 结果

模拟结果显示，

玩100次：

平均成本为$1000.0，平均奖金为$801.0，预期的最大损失为$199.0。

这意味着如果只玩100次，玩家通常会损失约20%的投资金额。

玩1000次：

平均成本为$10000.0，平均奖金为$8144.5，预期的最大损失为$1855.5。

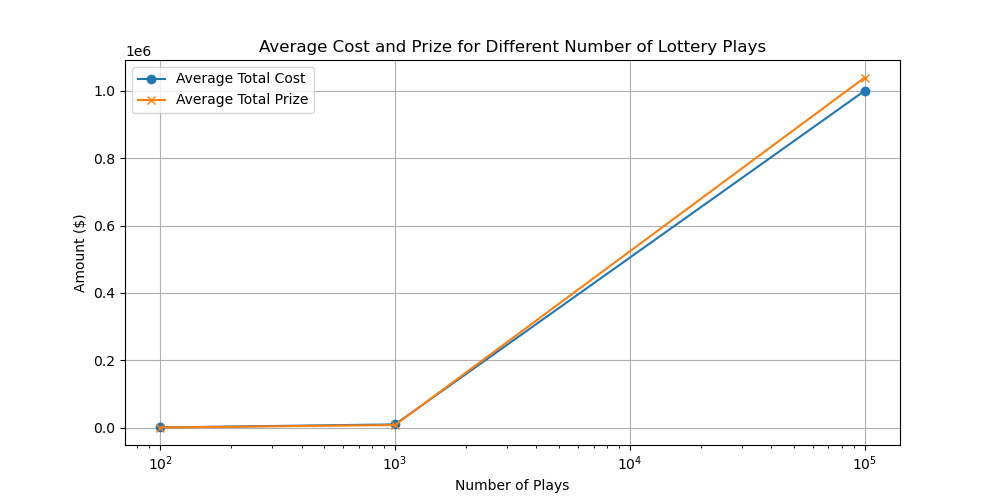
在这个级别上，虽然损失的比例有所减少（大约18.5%的损失），但总损失金额有所增加。

玩100000次：

平均成本为$1000000.0，平均奖金为$1039206.0，预期的最大损失实际上变成了盈利，为$-39206.0。

这表明，在大量的游戏次数下，玩家平均会获得约3.9%的盈利。然而，这个结果可能受到随机性的显著影响，特别是如果出现了一次或多次大奖。

1. Playing 100 times: Average Cost = $1000.0, Average Prize = $801.0, Expected Maximum Loss = $199.0
2. Playing 1000 times: Average Cost = $10000.0, Average Prize = $8144.5, Expected Maximum Loss = $1855.5
3. Playing 100000 times: Average Cost = $1000000.0, Average Prize = $1039206.0, Expected Maximum Loss = $-39206.0



5. 结论

此模拟研究表明，短期内，彩票是一个预期负收益的游戏。玩家应该预期会损失一部分投入的金钱。在玩100次或1000次时，损失的金额与总花费的比例较高。

长期来看，由于大奖的概率分布，可能会出现平均收益超过成本的情况。但这种情况的出现很可能是因为极少数的大奖偏离了统计学上的平均期望值。这样的结果不应被看作是长期可持续的，因为它可能是由大奖的随机性所驱动。

风险与回报：更高的游戏次数可能导致结果的极端化——即可能会有较大的损失或偶尔的大幅盈利。玩家在考虑投资彩票时，应考虑到这种高风险性。

6. 附件

Python代码

1. **import** random
2. **import** matplotlib.pyplot as plt
4. **def** generate\_lottery\_numbers():
5. """
6. 生成彩票号码。
7. 返回一个包含6个从1到21的随机不重复数字的集合。
8. """
9. **return** set(random.sample(range(1, 22), 6))
11. **def** calculate\_prize(matches):
12. """
13. 根据匹配的数字数量计算奖金。
14. :param matches: 匹配的数字数量
15. :return: 对应的奖金金额
16. """
17. prize\_dict = {0: 0, 1: 5, 2: 5, 3: 10, 4: 100, 5: 100, 6: 100000}
18. **return** prize\_dict.get(matches, 0)
20. **def** simulate\_lottery\_plays(num\_plays):
21. """
22. 模拟彩票游戏的一系列玩法。
23. :param num\_plays: 游戏的次数
24. :return: 总花费和总奖金
25. """
26. total\_cost = num\_plays \* 10  # 每次游戏的花费是$10
27. total\_prize = 0
29. **for** \_ **in** range(num\_plays):
30. lottery\_numbers = generate\_lottery\_numbers()
31. player\_numbers = generate\_lottery\_numbers()
32. matches = len(lottery\_numbers.intersection(player\_numbers))
33. total\_prize += calculate\_prize(matches)
35. **return** total\_cost, total\_prize
37. **def** average\_simulation(num\_plays, num\_experiments):
38. """
39. 对一个特定的游戏次数进行多次实验以获取平均结果。
40. :param num\_plays: 游戏次数
41. :param num\_experiments: 实验次数
42. :return: 平均总花费和平均总奖金
43. """
44. total\_costs = []
45. total\_prizes = []
47. **for** \_ **in** range(num\_experiments):
48. cost, prize = simulate\_lottery\_plays(num\_plays)
49. total\_costs.append(cost)
50. total\_prizes.append(prize)
52. avg\_cost = sum(total\_costs) / num\_experiments
53. avg\_prize = sum(total\_prizes) / num\_experiments
54. **return** avg\_cost, avg\_prize
56. # 准备数据
57. num\_plays\_list = [100, 1000, 100000]
58. num\_experiments = 10
59. avg\_costs = []
60. avg\_prizes = []
62. # 对每个游戏次数执行多次实验并收集数据
63. **for** num\_plays **in** num\_plays\_list:
64. avg\_cost, avg\_prize = average\_simulation(num\_plays, num\_experiments)
65. avg\_costs.append(avg\_cost)
66. avg\_prizes.append(avg\_prize)
67. **print**(
68. f"Playing {num\_plays} times: Average Cost = ${avg\_cost}, Average Prize = ${avg\_prize}, Expected Maximum Loss = ${avg\_cost - avg\_prize}")
70. # 绘制图表展示平均成本和奖金
71. plt.figure(figsize=(10, 5))
72. plt.plot(num\_plays\_list, avg\_costs, label='Average Total Cost', marker='o')
73. plt.plot(num\_plays\_list, avg\_prizes, label='Average Total Prize', marker='x')
74. plt.xlabel('Number of Plays')
75. plt.ylabel('Amount ($)')
76. plt.title('Average Cost and Prize for Different Number of Lottery Plays')
77. plt.xscale('log')  # 使用对数尺度来更好地展示不同游戏次数下的数据
78. plt.legend()
79. plt.grid(True)
80. plt.show()