UPEC
Connaissance - Action

UNIVERSITÉ
PARIS-EST CRÉTEIL
VAL DE MARNE

FINDING THE WINNING STRATEGY IN A CARD GAME

Case Study 1

Data Science BookCamp

Dinding Sally CAMARA
Anaïs COLY
Béatrice DEFENSE
Katucia HEMAT
Cadija INJAI
Theresa KINTENDA KIVUVU

INTRODUCTION OF THE CASE

- We assumed that we have a card game with 52 cards.

26 RED cards and 26 BLACK cards

- All the cards are face down and we will flip over the cards one by one.
- If the last card flipped over is red we win a $, if not we lose a $.
- We can stop the game at any time.

What is the **BEST** approach to win the **GAME**?

Choose to stop randomly?

Planned a card strategy?

USE CASE PRESENTATION AND PROBLEM DEFINITION

Organization of the Chapter:

- Introduction of the Case Study.
- Calculate probabilities by using the tool Python.
- Plotting the probabilities by using the tool Matplotlib.
- Run randomly simulation via the tool NumPy.
- Solution given to the Case Study.

How do we use these tools?

What is the probability to flip a red card and so win a dollar?

How to run the different strategies announced in the case?

DATA ANALYTICS APPROACH DEFINITION AND EXPLANATION

1. **Python introduction**

➔ **The basics of probability theory**

➢ Mathematical foundation of statistics.

➢ Central objects of the theory of probability:

- Random variables
- Stochastic processes
- Event

➔ **How to calculate probabilities : from single observation and over range**

➢ Simulation with python of a random experiment

➢ Two functions:

- The first one to simulate the roll of the die;
- The second one to build the list of results

➢ Three functions:

- A max_val function which returns the maximum value of the list;
- A min_val function that returns the minimum value;
- A mean_val function that returns the average value.

DATA ANALYTICS APPROACH DEFINITION AND EXPLANATION

2. Matplotlib introduction

- Matplotlib is a data visualization library for Python
- Can be used to create line graphs, bar graphs, histograms, scatter plots, box plots, pie charts, and even 3D plots.
- Open - source library

- **Probability distribution**
- Probability distribution allows….
- Matplotlib can be used to plot probability distributions…
- The main function of Matplotlib is not to provide that for a probability distribution

- **Plotting and comparing multiple probability distribution**
- To plot and compare multiple probability distributions with Matplotlib:
- Generate data set for each distribution + statical functions from Numpy library
- Create plots for each distribution using Matplotlib plotting functions + add a legend for each distribution
- It is important to normalize the data for a fair and just comparison.

DATA ANALYTICS APPROACH DEFINITION AND EXPLANATION

3.NumPy Introduction

- ➔ **NumPy** is a python library that is used to work with multidimensional arrays
- ➔ Provides tools for creating, manipulating and analyzing data table + performing
- ➔ advanced mathematical operations.

- ➢ **Visualization of simulated data** : generate synthetic data + visualize their distribution
- ➢ Data is generated by : mathematics models or numerical simulations
- ➢ Can be used to : test assumptions, develop strategies, explore hypothetical scenarios
- ➢ Can take many forms: histograms, point clouds, distribution curves

- **Unknown probabilities from hidden observations** : a method used to estimate the probabilities of rare or extreme events from simulate data.
- Simulated observations follow a known oz assume probability distribution
- This method is used in many fields: finance, insurance, risk management…
- It can be applied to situations where real data is rare, expensive or difficult to collect, or when events of interest are very rare or unpredictable.

SOLUTION DEVELOPMENT AND ILLUSTRATION

- **CHAPTER GOAL** ⟶ Discover the process, that best can predict to flip a red card.

Achievement of the goal with 3 main steps 

- How can we know that the card we will flip will be red?
- Take every strategy and make multiple simulations to calculate the probability of success within a high confidence interval.
- Keep the strategy that offers the highest probability of success.

The first Question: **How do we know when we should say "Halt" to stop the game?**

- STOP! when the Total number of red cards > Total number of black cards remaining.
- End ⟶ The total number of red cards divided per the remaining total cards is higher than 0.5.

What is our probability to flip over a red card?

- 1.000 random shuffles

SOLUTION DEVELOPMENT AND ILLUSTRATION

- Then we calculate the frequency of possible win.
- **How?**

We add up the red cards present in the game (26*1.000 = 26.000) and divide them by the total number of cards in all the shuffles (52.000).

Interpretation of the result: 

- Frequency > 0.5 → More wins than losses → Strategy works → We won money.
- Frequency < 0.5 → More losses than wins → Strategy is not working → We will lose money.

Dollars won = Frequency of wins * Sample size (1.000 shuffles)

Simulation with the plotting method:

- Series of sample size **1 to 10.000 shuffles**

Observation → The bigger the sample size is, the closer the frequency of win is from 0.5

SOLUTION DEVELOPMENT AND ILLUSTRATION

➢ **Plotting simulated frequencies of Wins**

- We will analyze the probability of winning.
- When the strategy also fluctuates above and below 50% during sampling, how can we be sure that the probability of a win is greater than 0.5?
- For this, the use of the analysis of confidence intervals is essential.
- we calculate the confidence interval.

➢ Thus, we need to reduce the 95% confidence interval by increasing the sample size at the cost of running time..

SOLUTION DEVELOPMENT AND ILLUSTRATION

➔ If Unfortunately, the new confidence interval still does not make it possible to discern whether the true probability is above 0.5.

➢ Maybe increasing **min_red_fraction** from 0.5 to 0.75 will bring some improvement.

SOLUTION DEVELOPMENT AND ILLUSTRATION

2. the various strategies put in place → optimal winning strategy

discover why strategies fail :

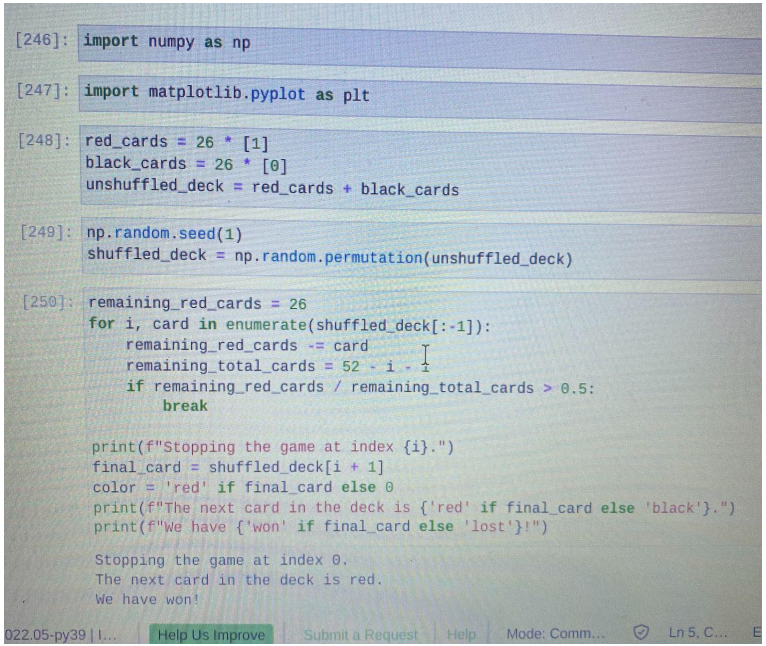
- what strategy to put in place?
- when the strategy stops, there is either a gain or a loss
- In case of loss: any advantage is destroyed
- the chance to win remains always at 50%
- probabilities can be counterintuitive

SOLUTION DEVELOPMENT AND ILLUSTRATION

Setting Up the game

1st step : Import NumPy and Matplotlib, and define them

2nd step : Create the deck with 26 red cards and 26 black cards



```
[246]: import numpy as np

[247]: import matplotlib.pyplot as plt

[248]: red_cards = 26 * [1]
       black_cards = 26 * [0]
       unshuffled_deck = red_cards + black_cards

[249]: np.random.seed(1)
       shuffled_deck = np.random.permutation(unshuffled_deck)

[250]: remaining_red_cards = 26
       for i, card in enumerate(shuffled_deck[:-1]):
           remaining_red_cards -= card
           remaining_total_cards = 52 - i - 1
           if remaining_red_cards / remaining_total_cards > 0.5:
               break

       print(f"Stopping the game at index {i}.")
       final_card = shuffled_deck[i + 1]
       color = 'red' if final_card else 0
       print(f"The next card in the deck is {'red' if final_card else 'black'}.")
       print(f"We have {'won' if final_card else 'lost'}!")

       Stopping the game at index 0.
       The next card in the deck is red.
       We have won!

022.05-py39 | I...   Help Us Improve   Submit a Request   Help   Mode: Comm...   Ln 5, C...
```

SOLUTION DEVELOPMENT AND ILLUSTRATION

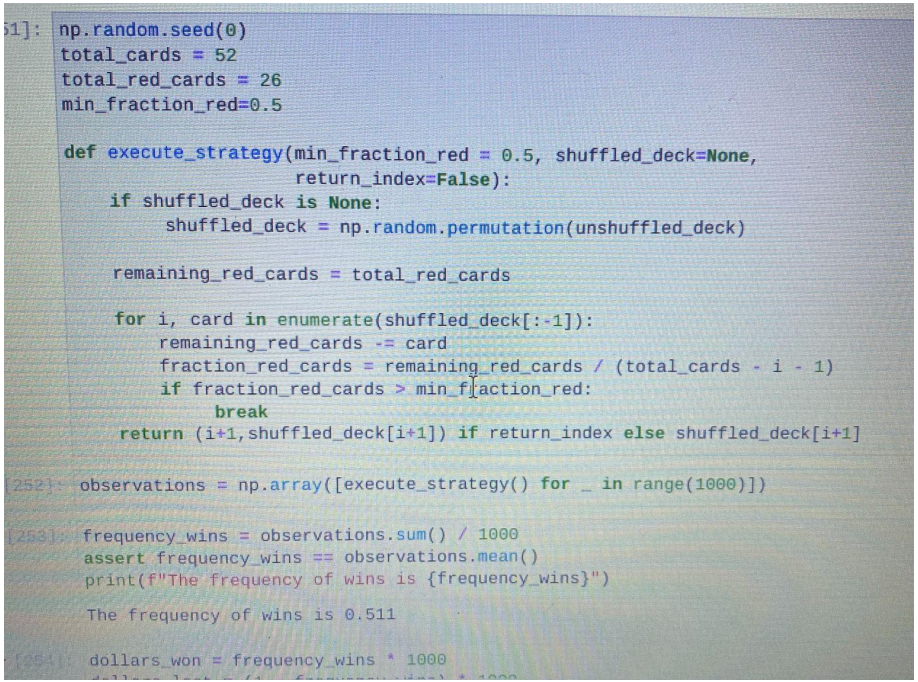
Coding the card game strategy

1st step : the strategy is the following

⇒terminate the game when the number of red cards remaining in the deck is greater than the number of black cards remaining in the deck.

2nd step : execute this strategy

3rd step: generalize this strategy



```
51]: np.random.seed(0)
     total_cards = 52
     total_red_cards = 26
     min_fraction_red=0.5

     def execute_strategy(min_fraction_red = 0.5, shuffled_deck=None,
                          return_index=False):
         if shuffled_deck is None:
             shuffled_deck = np.random.permutation(unshuffled_deck)

         remaining_red_cards = total_red_cards

         for i, card in enumerate(shuffled_deck[:-1]):
             remaining_red_cards -= card
             fraction_red_cards = remaining_red_cards / (total_cards - i - 1)
             if fraction_red_cards > min_fraction_red:
                 break
         return (i+1,shuffled_deck[i+1]) if return_index else shuffled_deck[i+1]

252]: observations = np.array([execute_strategy() for _ in range(1000)])

253]: frequency_wins = observations.sum() / 1000
      assert frequency_wins == observations.mean()
      print(f"The frequency of wins is {frequency_wins}")

      The frequency of wins is 0.511

254]: dollars_won = frequency_wins * 1000
```

SOLUTION DEVELOPMENT AND ILLUSTRATION

Running the strategy over 1,000 random shuffles

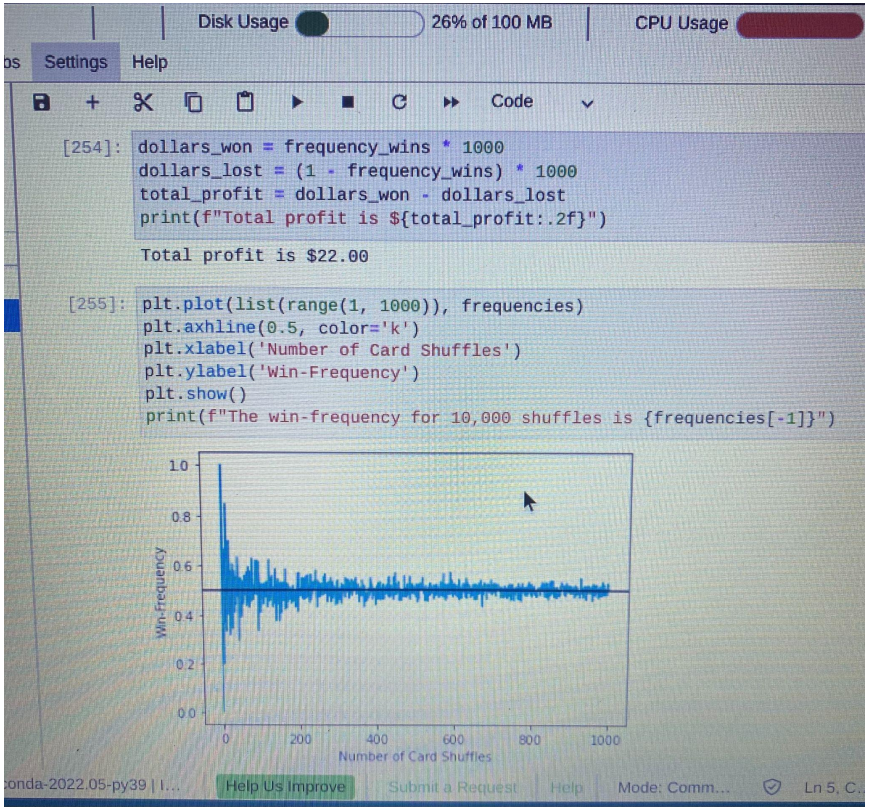
1st step: apply our basic strategy to a series of shuffles

2st step: run the strategy and compute the frequency of wins and total profit

Result : 51,1% win of the total games / 511 wins and 489 losses

Total profit = $22.00

Figure : We plot the strategy's win-frequency convergence over a series of sample sizes ranging from 1 through 10,000

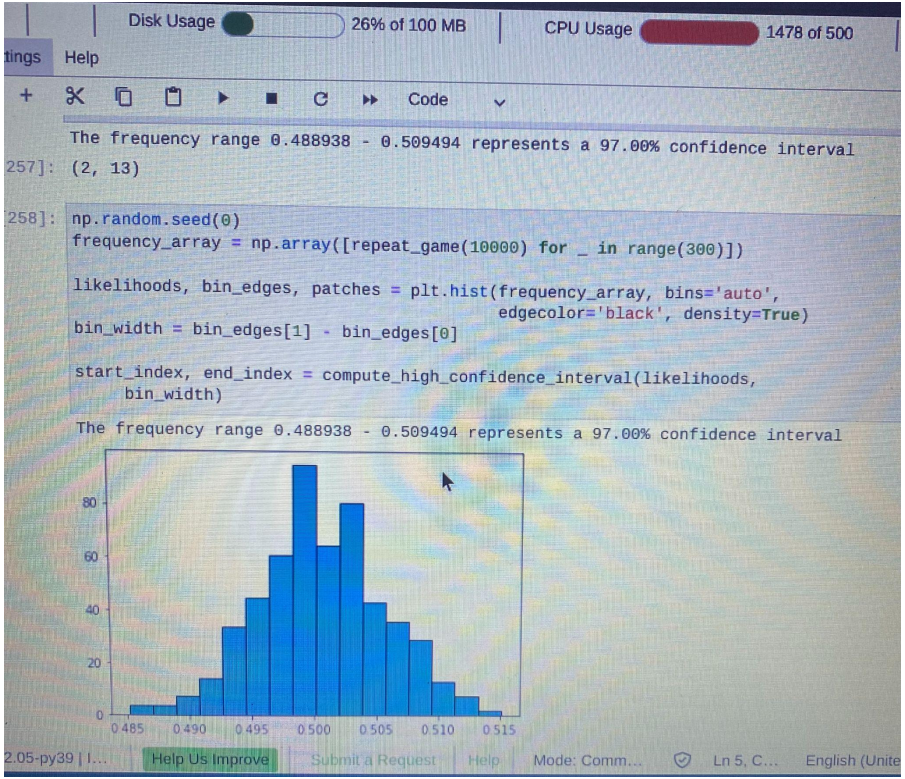


SOLUTION DEVELOPMENT AND ILLUSTRATION

Simulated the frequency of wins

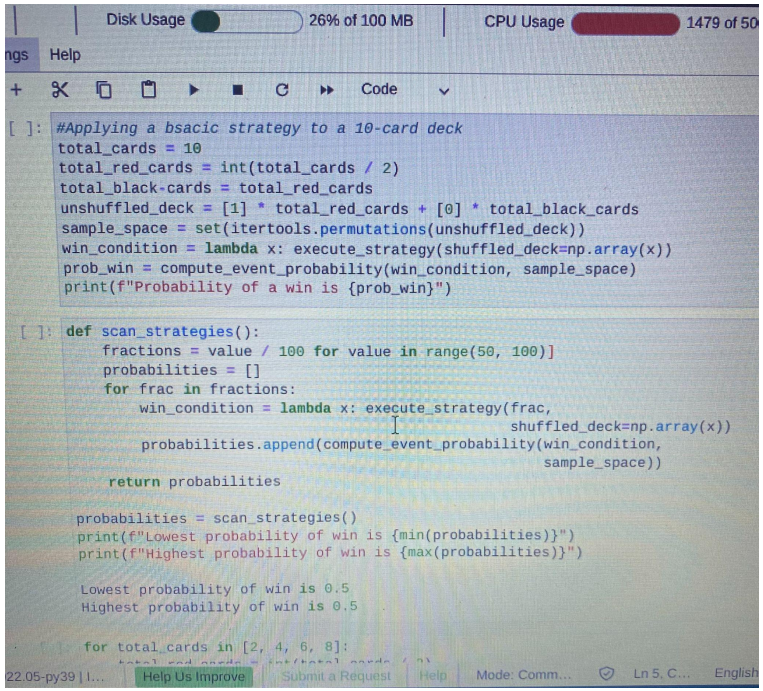
Figure : histogram of the binned frequency plotted against their associated relative likelihoods

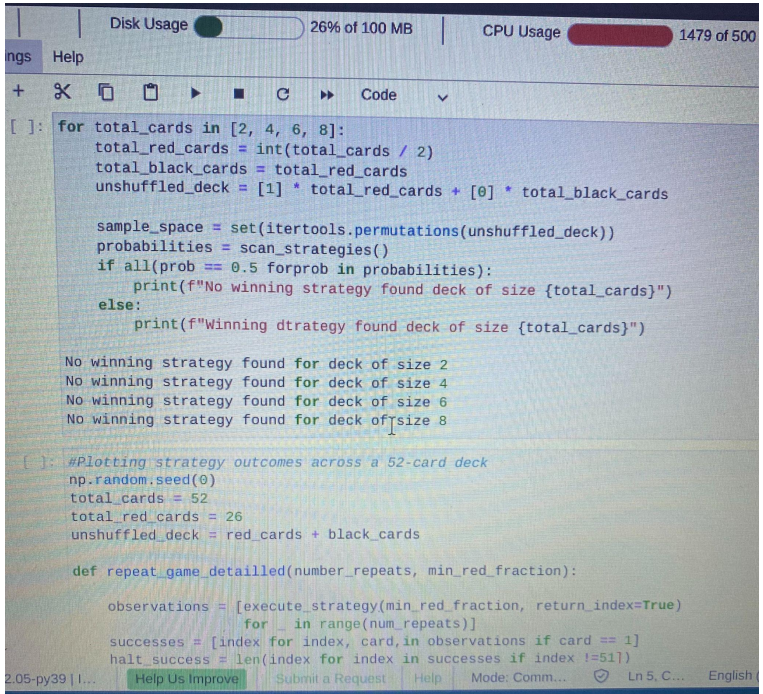
Than, we compute the confidence interval for 3 million and 150 million shuffles



SOLUTION DEVELOPMENT AND ILLUSTRATION

Applying a basic and a multiples strategies to a 10-card deck, and a multiple strategies to multiple decks





SOLUTION DEVELOPMENT AND ILLUSTRATION

Figure : The_min_fraction

4 Strategy :

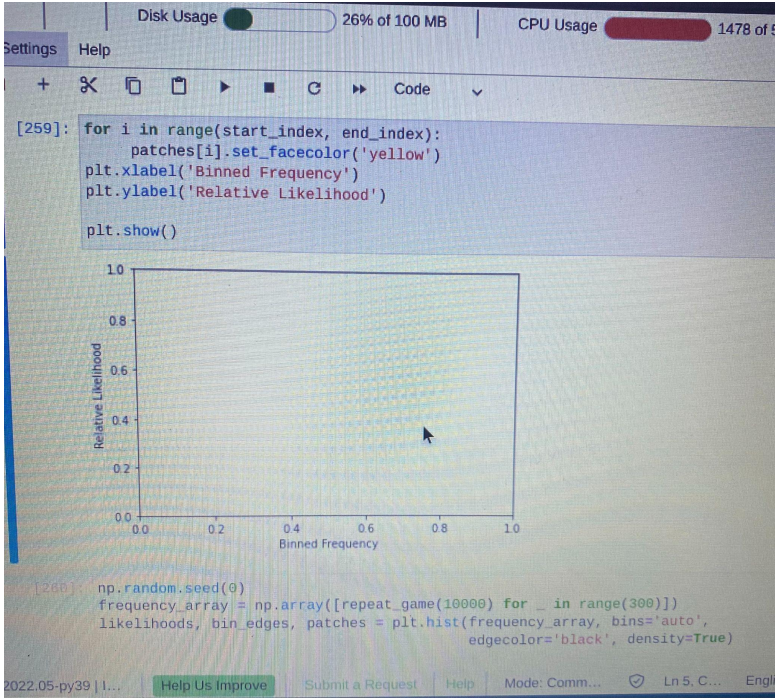
A : Strategy halts. We win.

B. Strategy halts. We lose.

C. No halt. We win.

D.No halt we lose.

Can't display the graph because of a mistake in the min_red_fraction parameter plot against the sampled frequencies for all four possible scenarios.



Final : defining the optimal winning strategy

EVALUATION AND FEEDBACKS

- This case study was very interesting to discover : its highlights the practical applications of data science in games and strategic decision-making
- You need to be focused and read the instructions carefully to perform the coding, but it’s still doable for students
- Studying this case has been very interesting. It is the first time for many of us to have a significant contact with big data and the coding world.
- Probabilities can help us making choice in real life : Reducing the possibilities of uncertainty.
- On the whole, the solutions given in this chapter were understandable.
- The demonstration with the simulation was a part very technical, it was a challenge .
- Even if we have some errors on the simulation we have tried to do our best to find the solution.
- The interpretation of the result is the most important things.
- We had to search and learn very fast how works the programming language.

CONCLUSION

It was very interesting to learn more about Matplotlib and Numpy. This case study gave us quite a bit of work, though at the beginning we had some trouble understanding what the professor expected from us. But once we understood the task, we were able to manage and learn a bit more about big data.

it also shows that Big Data, seems today the miraculous solution for an efficient management of data masses.
It encompasses new working methods, new skills, an effect of decompartmentalization in organizations that will have to rely on new computer architectures.
Its power comes down to: Big Data is a true promise of value driven by the exploration and exploitation of use cases

• Analyze data from different sources from a single application;
. Visualize data with sharp, powerful graphics;
. Naturally explore associations between data;
. Access your data from mobile devices for analytics;
. Develop real-time and secure collaborative decision making