Simulare Monte Carlo: Impactul inflației asupra economiilor

Trifan Bogdan-Cristian, Mihăilă Denisa, Țîncu Alexandru

Introducere

Acest proiect oferă o simulare care estimează valoarea reală a economiilor în timp, luând în considerare inflația din România. Am ales această temă pentru că este important să știm cum inflația ne afectează economiile avute. În acest proiect vom vedea cum 100 de lei economisiți în anul 2024 își vor pierde puterea de cumpărare peste 5 ani.

Caracteristici

- Simulează impactul inflației asupra economiilor pe un interval de 5 ani.
- Utilizează date istorice ale inflației României din perioada anilor 2000 - 2023.
- Valorile inflației sunt reale, puse la dispoziție de către Institutul Național de Statistică.
- Include grafice pentru vizualizarea rezultatelor și datelor (grafice de distribuție și tendințe istorice ale inflației).
- Oferă un cod ușor de ajustat pentru scenarii personalizate.
- Pentru calcule a fost folosită librăria NumPy.
- Pentru grafice a fost folosită librăria Matplotlib.

Inflația în România

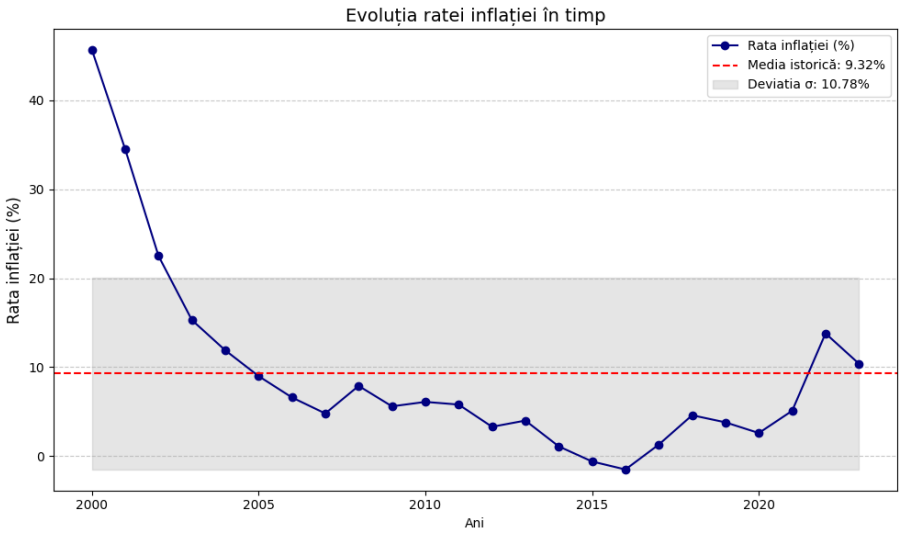

Figure 1: Evolutia inflației în România începând din anul 2000

Formularea matematică

Pentru calculul puterii de cumpărare după cei 5 ani au fost folosite conceptele învățate la cursul de Probabilități și Statistică:

- Simulări Monte Carlo
- Inegalitatea Chernoff-Hoeffding
- Teorema Limită Centrală

1. Valoarea reală a economiilor

Valoarea reală a economiilor după t ani este calculată astfel:

$$P_t = P_0 \cdot \prod_{n=1}^{t} \frac{1}{1 + i_n}$$

Unde:

- P_0: Suma inițială economisită.
- i_n: Rata inflației pentru anul n.
- t: Numărul de ani.

2. Simularea ratei inflației

Ratele inflației sunt generate cu ajutorul librăriei NumPy care dispune de un generator de numere aleatorii având distribuție normală (Gaussiană).

$$\text{simulare\_inflatie} \sim \mathcal{N}(\mu, \sigma)$$

Unde:

- μ: Media ratelor inflației începând cu anul 2000.
- σ: Deviația standard a ratelor inflației.

3. Convergența estimărilor

Conform Teoremei Limită Centrală, ratele de inflație generate vor avea distribuție Gaussiană, așa cum reiese din graficul intitulat **Distribuția simulărilor inflației** (forma de clopot).

4. Folosirea inegalităților stochastice

Inițial, proiectul a calculat puterea de cumpărare a 100 de lei peste 5 ani folosind 100.000 de simulări. Rezultatul a fost că, după cei 5 ani, 100 de lei mai valorează defapt aproximativ 67 de lei!

Folosind inegalitatea Chernoff-Hoeffding, am calculat că este posibil să obținem aceeași valoare cu o marjă de eroare de $\pm 2\%$ și cu un nivel de încredere de 95% după cel puțin 4.611 simulări.

Formula folosită pentru a obține numărul minim n de simulări necesare este:

$$n \geq \frac{1}{2\varepsilon^2} \ln\left(\frac{2}{1-\alpha}\right)$$

Unde:

- ε: 0.02 (Marja de eroare $\pm 2\%$).
- α: 95% (Nivelul de încredere).
- n: Numărul de simulări.

Efectul inegalității Chernoff-Hoeffding în calculul puterii de cumpărare a fost să reducă costul computațional al simulărilor de la 100.000 la 4.611.

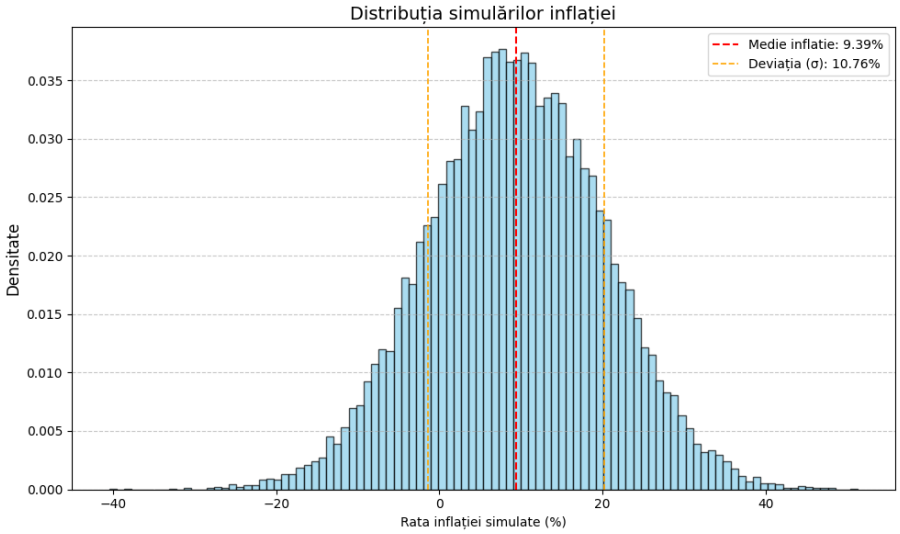

Figure 2: Distribuția datelor folosind numărul de simulări dat de Chernoff-Hoeffding (4611).

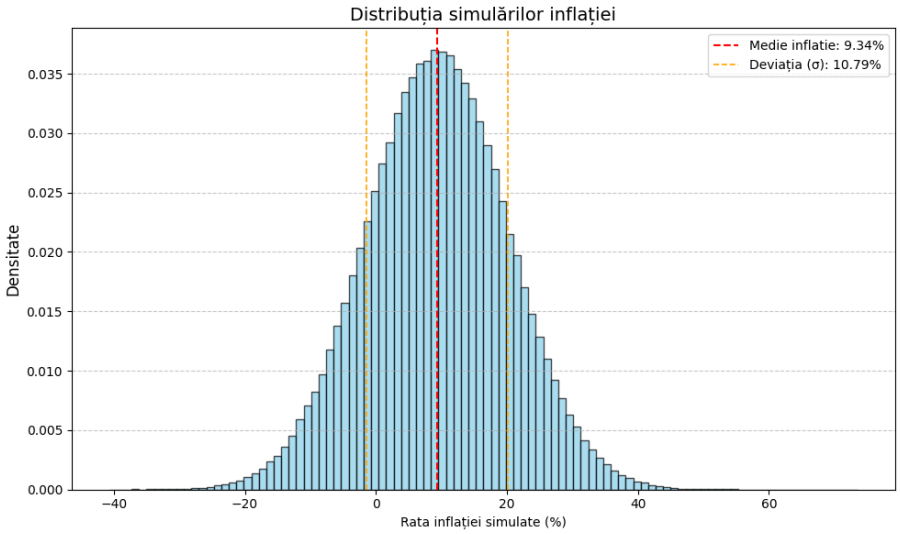

Figure 3: Distribuția datelor după 100.000 de simulări.

Rezultate

Rezultatul proiectului ne arată că 100 de lei economisiți în anul 2024 își vor pierde puterea de cumpărare. Acest fenomen se poate întâmpla cu orice sumă economisită.

Structura codului

Codul are o structură foarte ușoară ce permite modificarea datelor pentru diferite scenarii. Pentru ușurință, am ales să creăm clasa **Inflation** care are 2 funcții:

- **__init__**: constructorul clasei
  - **rata_inflatie**: un array cu rata inflației pusă la dispoziție de INS.
  - **suma_economisita**: suma pentru care vrem să îi calculăm valoarea peste 5 ani
  - **numar_simulari**: numărul de simulări dorite; are valoare implicită de 100.000
- **compute**: funcția care calculează noua valoare a sumei economisite
  - **nivel_de_incredere**: parametru folosit pentru a estima un număr de simulări mai mic folosind inegalitatea Chernoff-Hoeffding; are valoarea implicită 0.95
  - **marja_eroare**: marja de eroare dorită; are valoare implicită 0.02
- **plot**: funcția care generează grafice
  - **Evoluția ratei inflației în timp**: acest grafic arată rata inflației din România începând din anul 2000 până în anul 2023.
  - **Distribuția simulărilor inflației**: graficul arată cum sunt distribuite ratele inflației din simulări; se poate observa forma de clopot.

Pentru a modifica numărul de ani pentru care se face simularea trebuie modificată variabila **PERIOADA_EXPERIMENT** care se află la începutul codului.

Implementare

```
import numpy as np
import matplotlib.pyplot as plt

PERIOADA_EXPERIMENT = 5 # 5 ani

class Inflation:
```

```
def __init__(self, ani, rata_inflatie, suma_economisita, numar_simulari = 100_000):
    self.ani = ani
    self.rata_inflatie = rata_inflatie
    self.numar_simulari = numar_simulari
    self.suma_economisita = suma_economisita

    self.medie_inflatie = np.mean(self.rata_inflatie)
    self.deviatia_standard = np.std(self.rata_inflatie)
    print(f'Media inflatiei in Romania (2000 -> 2023): {round(self.medie_inflatie, 2)}%
    print(f'Deviatia standard a mediei (2000 -> 2023): {round(self.deviatia_standard, 2)

    # generez o matrice cu perioada_experiment coloane si numar_simulari linii
    self.simulare_inflatie = np.random.normal(self.medie_inflatie, self.deviatia_standar

def compute(self, nivel_de_incredere = 0.95, marja_eroare = 0.02):
    # ‘suma_viitoare‘ = array de dimensiune ‘numar_simulari‘ cu valoarea ‘suma_economisi
    suma_viitoare = np.full(self.numar_simulari, self.suma_economisita, dtype = np.float

    for i in range(PERIOADA_EXPERIMENT): # se aplica inflatia fiecarui an
        suma_viitoare = suma_viitoare / (1 + self.simulare_inflatie[:, i] / 100)

    medie_viitor = np.mean(suma_viitoare) # calculez media inflatiei peste cei 5 ani.
    deviatia_standard_viitor = np.std(suma_viitoare) # calculez deviatia standard a infl

    # chernoff-hoeffding
    # vezi cursul 11 "Q4: Pot imbunatati estimarile din Q2 si Q3?"
    numar_necesar_simulari = 2 * (marja_eroare ** 2)
    numar_necesar_simulari = np.log(2 / (1 - nivel_de_incredere)) / numar_necesar_simula

    print(f'Peste {PERIOADA_EXPERIMENT} ani, {self.suma_economisita} lei vor insemna {ro
    print(f'Pentru o eroare de ±{round(marja_eroare * 100, 2)}% cu nivelul de încredere

def plot(self):
    # Grafic 1: Evoluția ratei inflației în timp
    lower_bound = self.medie_inflatie - self.deviatia_standard
    upper_bound = self.medie_inflatie + self.deviatia_standard

    plt.figure(figsize=(10, 6))
    plt.plot(self.ani, self.rata_inflatie, marker='o', linestyle='-', color='navy', labe
    plt.axhline(self.medie_inflatie, color='red', linestyle='dashed', linewidth=1.5,
                label=f'Media istorică: {round(self.medie_inflatie, 2)}%')
    plt.fill_between(self.ani, lower_bound, upper_bound, color='gray', alpha=0.2,
                     label=f'Deviatia : {round(self.deviatia_standard, 2)}%')
    plt.title('Evoluția ratei inflației în timp', fontsize=14)
    plt.xlabel('Ani', fontsize=10)
    plt.ylabel('Rata inflației (%)', fontsize=12)
```

```
        plt.legend()
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show()

        # Grafic 2: Distribuția simulărilor inflației
        medie_simulari = np.mean(self.simulare_inflatie)
        std_simulari = np.std(self.simulare_inflatie)
        lower_bound = medie_simulari - std_simulari
        upper_bound = medie_simulari + std_simulari

        plt.figure(figsize=(10, 6))
        plt.hist(self.simulare_inflatie.flatten(), bins=100, color='skyblue', edgecolor='bla
                 density=True)
        plt.axvline(medie_simulari, color='red', linestyle='dashed', linewidth=1.5,
                    label=f'Medie inflatie: {medie_simulari:.2f}%')
        plt.axvline(lower_bound, color='orange', linestyle='dashed', linewidth=1.2, label=f'
        plt.axvline(upper_bound, color='orange', linestyle='dashed', linewidth=1.2)
        plt.title('Distribuția simulărilor inflației', fontsize=14)
        plt.xlabel('Rata inflației simulate (%)', fontsize=10)
        plt.ylabel('Densitate', fontsize=12)
        plt.legend()
        plt.grid(axis='y', linestyle='--', alpha=0.7)
        plt.tight_layout()
        plt.show()


ani = [2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2
rata_inflatie = [45.7, 34.5, 22.5, 15.3, 11.9, 9.0, 6.6, 4.8, 7.9, 5.6, 6.1, 5.8, 3.3, 4.0,

inflatie = Inflation(ani, rata_inflatie, 100)
inflatie.compute()
inflatie.plot()
```

Instalare

Urmați pașii următori:

```
git clone https://github.com/Bogdanctx/inflation-monte-carlo.git
cd inflation-monte-carlo
pip install numpy matplotlib
python main.py
```

Limitări

- Presupune o distribuție normală a ratelor inflației, ceea ce simplifică comportamentul din lumea reală.
- Ignoră factori economici externi, precum recesiuni sau schimbări de politici.
- Nu ia în considerare ratele dobânzilor sau randamentele investițiilor asupra economiilor.

Referințe

- Cursurile și laboratoarele susținute de profesorul Mihai Bucătaru la Universitatea din București.
- Documentația Numpy: Random Normal
- Probability and Statistics for Computer Scientists, pagina 90.
- Institutul Național de Statistică