Nome: Joao Otavio Rodrigues de Castro Manieri
Matrícula: 12021BSI263

Relatório: Análise Econômica do Endereço `1JHH1pmHujcVa1aXjRrA13BJ13iCfgfBqj`

---

1. Introdução

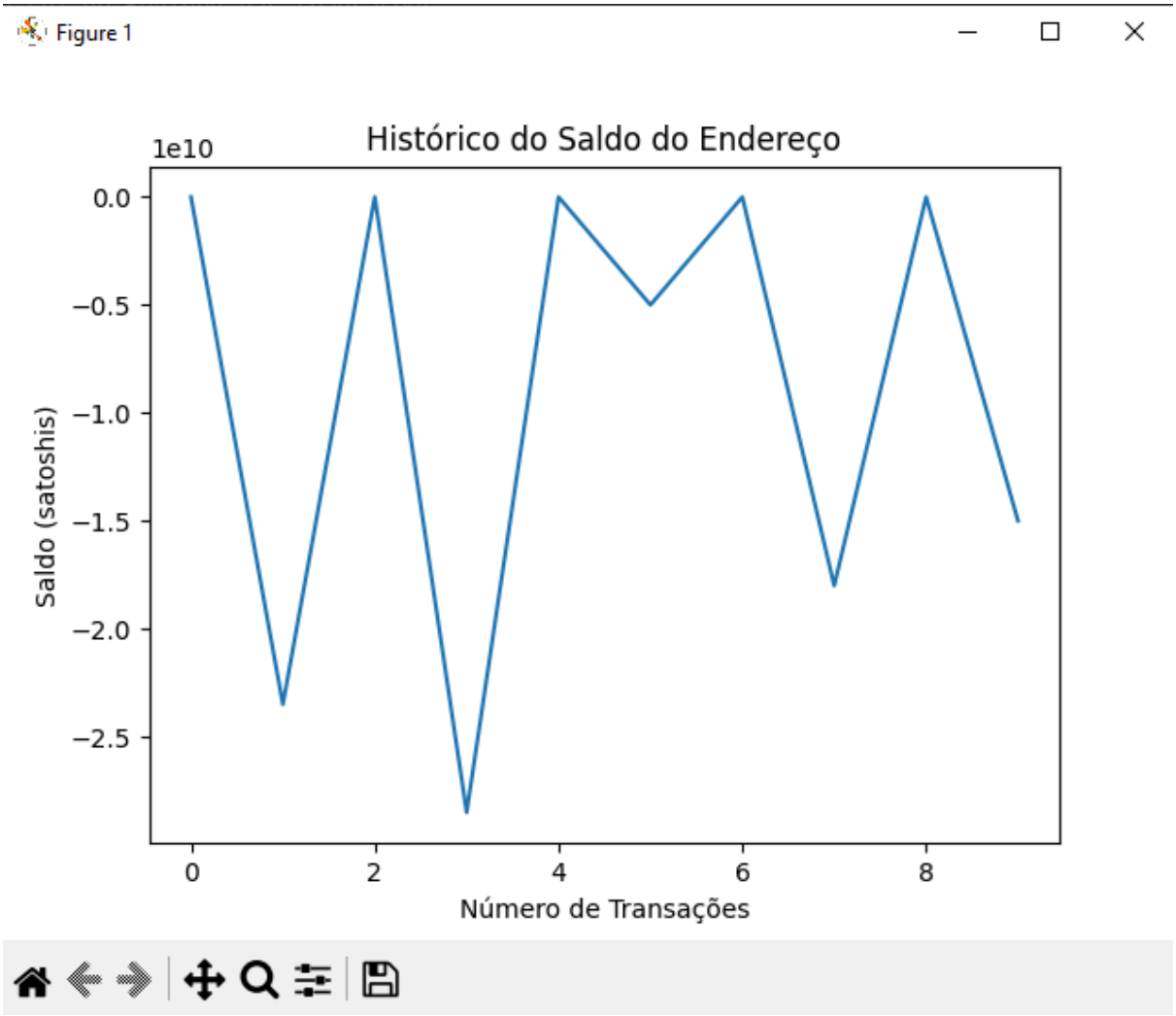
Este relatório apresenta uma análise econômica do endereço Bitcoin `1JHH1pmHujcVa1aXjRrA13BJ13iCfgfBqj`, concentrando-se em três aspectos principais:

- **Histórico do Saldo**: Evolução do saldo do endereço ao longo do tempo.
- **Índice de Gini das Transações**: Medida de desigualdade entre os valores transacionados.
- **Distribuição de Benford**: Verificação da aderência dos valores das transações à Lei de Benford, que define a frequência dos dígitos nas transações financeiras.

---

2. Histórico do Saldo

Foi realizada uma análise do saldo histórico, registrando todas as transações (entradas e saídas) associadas ao endereço. O gráfico abaixo mostra a evolução do saldo ao longo das transações.



O saldo do endereço variou conforme as transações realizadas, demonstrando o fluxo econômico envolvendo o endereço.

---

3. Índice de Gini das Transações

O índice de Gini foi calculado para medir a desigualdade na distribuição dos valores transacionados. Valores próximos de 1 indicam maior desigualdade, enquanto valores próximos de 0 indicam uma distribuição mais igualitária.

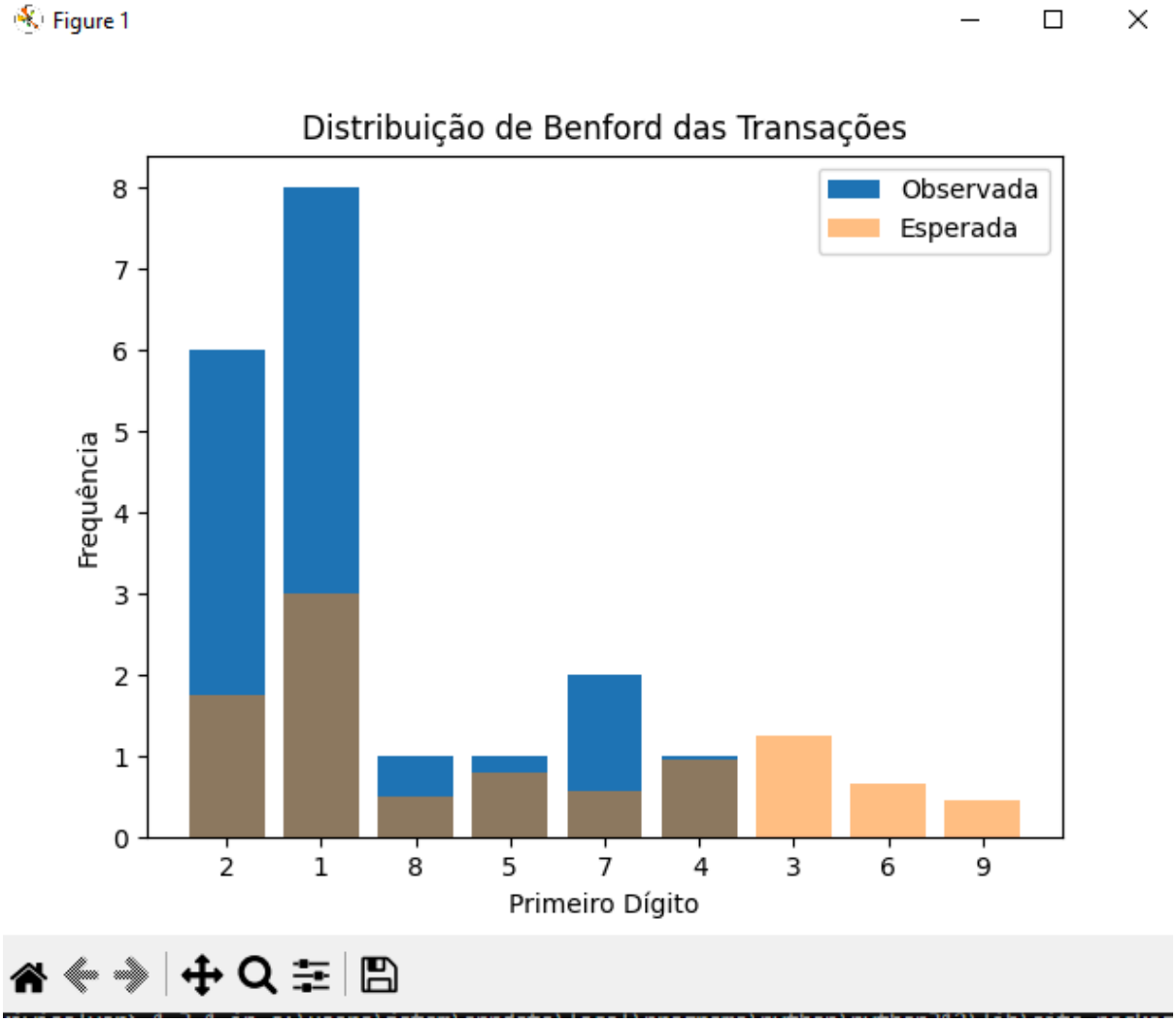
- **Índice de Gini calculado**: `0.614743373024673`
    - `gini index = (2 * np.sum((i + 1) * sorted_values[i] for i in range(n)) / (n * np.sum(sorted_values))) - (n + 1) / n`

O índice sugere que a maioria dos valores transacionados está concentrada em um pequeno número de transações, refletindo desigualdade na movimentação econômica do endereço.

---

4. Análise de Benford

A Lei de Benford foi aplicada às transações, analisando a frequência do primeiro dígito dos valores. A distribuição observada foi comparada com a distribuição esperada pela Lei de Benford, conforme mostrado no gráfico abaixo.



A análise demonstra que os valores transacionados seguem, em certa medida, o padrão esperado pela Lei de Benford, indicando um comportamento financeiro típico para este tipo de endereço.

---

5. Conclusão

A análise do endereço `1JHH1pmHujcVa1aXjRrA13BJ13iCfgfBqj` revelou as seguintes observações:

- O histórico do saldo mostra uma variação significativa, indicando uma movimentação econômica ativa.
- O índice de Gini indicou desigualdade nos valores transacionados, sugerindo concentração de grandes quantias em poucas transações.
- A análise de Benford mostrou que os valores transacionados seguem parcialmente a distribuição esperada, o que é comum em transações financeiras genuínas.

Essas análises oferecem uma visão geral da movimentação econômica do endereço e ajudam a identificar padrões de transação que podem ser úteis em investigações ou estudos futuros.

---

6. Referências

- **API utilizada**: BlockCypher (https://www.blockcypher.com)
    - A https://www.blockchain.com/explorer/api/blockchain_api estava dando
    - muito erro de timeout
- **Ferramentas**: Python (bibliotecas `requests`, `numpy`, `matplotlib`)

---

7. Código

```
import requests
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
import math

# API base para BlockCypher
blockcypher_base_url = 'https://api.blockcypher.com/v1/btc/main/'

# Função para obter as transações de um endereço específico
def get_address_transactions(address):
    url = blockcypher_base_url + f'addrs/{address}/full'
    response = requests.get(url)

    if response.status_code == 200:
        try:
            return response.json()
        except requests.exceptions.JSONDecodeError:
            print("Erro: Não foi possível interpretar a resposta como
JSON.")
            return None
    else:
```

```
        print(f"Erro: Recebido status code {response.status_code}")
        return None

# Função para calcular o saldo histórico do endereço
def calculate_balance_history(transactions, address):
    balance_history = []
    balance = 0
    for tx in transactions:
        # Filtrar apenas as transações de saída e entrada do endereço
em questão
        for output in tx['outputs']:
            if 'addresses' in output and address in
output['addresses']:
                balance += output['value']
        for input_tx in tx['inputs']:
            if 'addresses' in input_tx and address in
input_tx['addresses']:
                balance -= input_tx['output_value']
        balance_history.append(balance)
    return balance_history

# Função para calcular o índice de Gini
def calculate_gini(values):
    sorted_values = sorted(values)
    n = len(values)
    cumulative_values = np.cumsum(sorted_values)
    gini_index = (2 * np.sum((i + 1) * sorted_values[i] for i in
range(n)) / (n * np.sum(sorted_values))) - (n + 1) / n
    return gini_index

# Função para aplicar a Lei de Benford nas transações
def apply_benford(transactions):
    first_digits = [str(output['value'])[0] for tx in transactions for
output in tx['outputs'] if output['value'] > 0]
    return Counter(first_digits)

# Função para calcular a distribuição esperada de Benford
def benford_distribution():
    return {str(digit): math.log10(1 + 1 / digit) for digit in range(1,
10)}

# Endereço a ser analisado
address = "1JHH1pmHujcVa1aXjRrA13BJ13iCfgfBqj"
```

```
# Obter as transações do endereço
address_data = get_address_transactions(address)

if address_data and 'txs' in address_data:
    transactions = address_data['txs']

    # 1. Análise do saldo histórico
    balance_history = calculate_balance_history(transactions, address)

    # Plotar o saldo histórico
    plt.plot(balance_history)
    plt.title('Histórico do Saldo do Endereço')
    plt.xlabel('Número de Transações')
    plt.ylabel('Saldo (satoshis)')
    plt.show()

    # 2. Calcular o índice de Gini das transações
    transaction_values = [sum(output['value'] for output in
tx['outputs']) for tx in transactions]
    gini = calculate_gini(transaction_values)
    print(f'Índice de Gini das transações: {gini}')

    # 3. Aplicar a Lei de Benford nas transações
    benford_actual = apply_benford(transactions)
    benford_expected = benford_distribution()

    # Comparar a distribuição de Benford observada com a esperada
    plt.bar(benford_actual.keys(), [benford_actual[digit] for digit in
benford_actual.keys()], label='Observada')
    plt.bar(benford_expected.keys(), [benford_expected[digit] *
len(transactions) for digit in benford_expected.keys()], alpha=0.5,
label='Esperada')
    plt.title('Distribuição de Benford das Transações')
    plt.xlabel('Primeiro Dígito')
    plt.ylabel('Frequência')
    plt.legend()
    plt.show()
else:
    print("Nenhum dado de transação disponível.")
```