Uma breve incursão pelo Caos - Guilherme e Victor

September 5, 2021

1 Física Estatística Computacional

1.1 Uma breve incursão pelo Caos

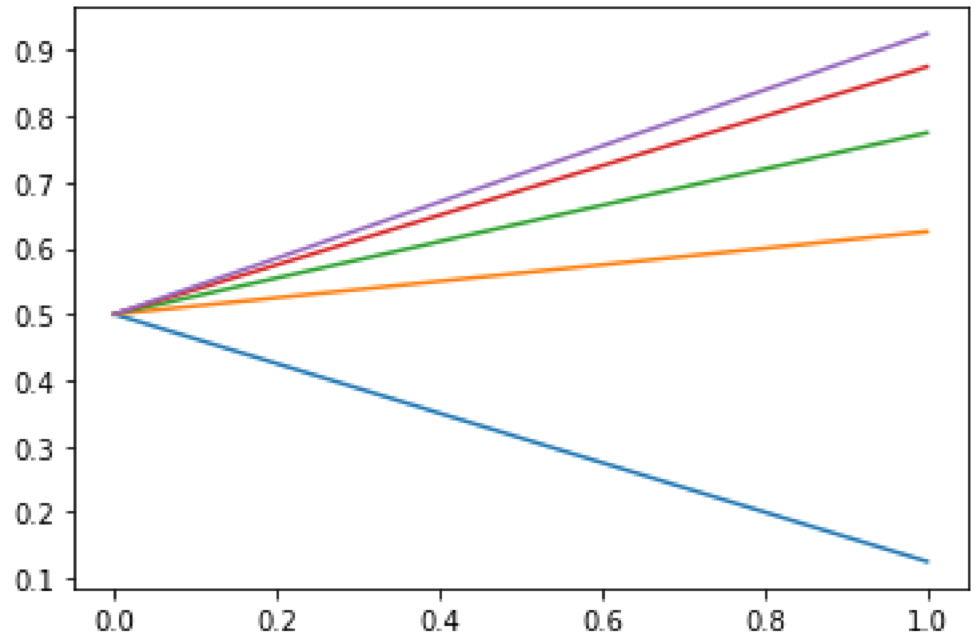
Guilherme de Abreu Lima Buitrago Miranda - 2018054788 Victor Hugo Silva Moura - 2018054958

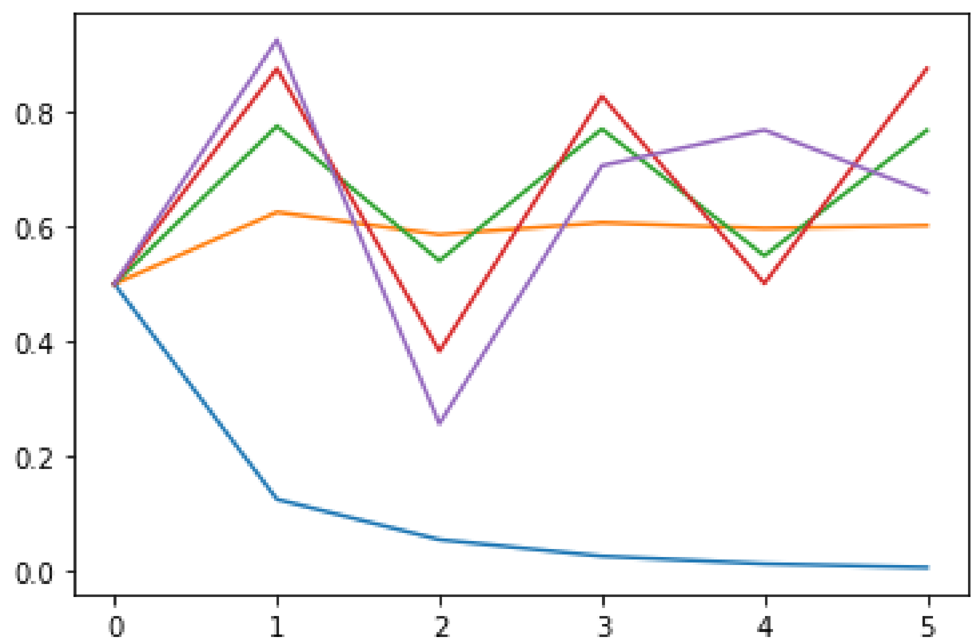
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

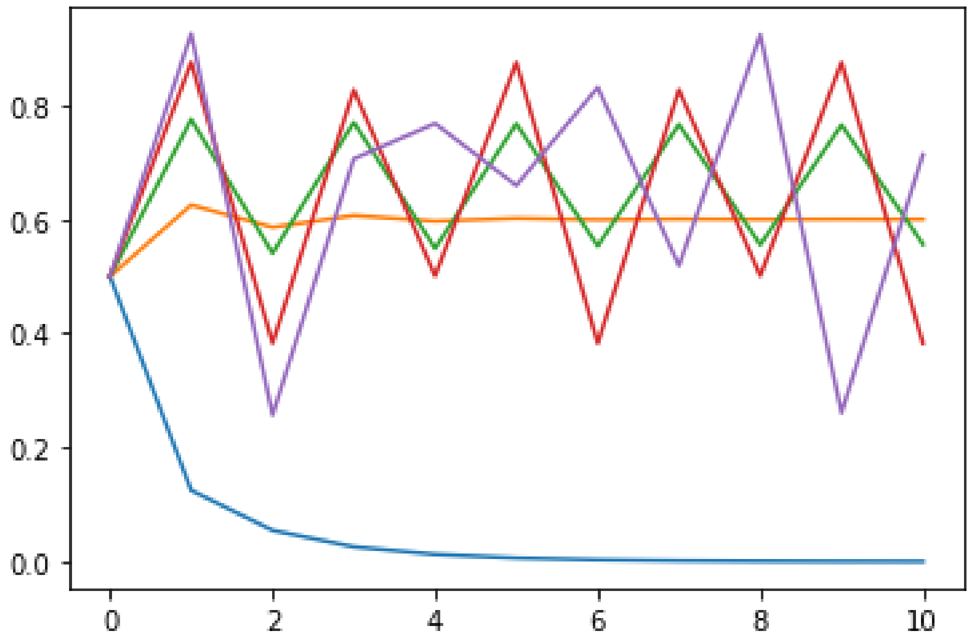
1.2 Testes

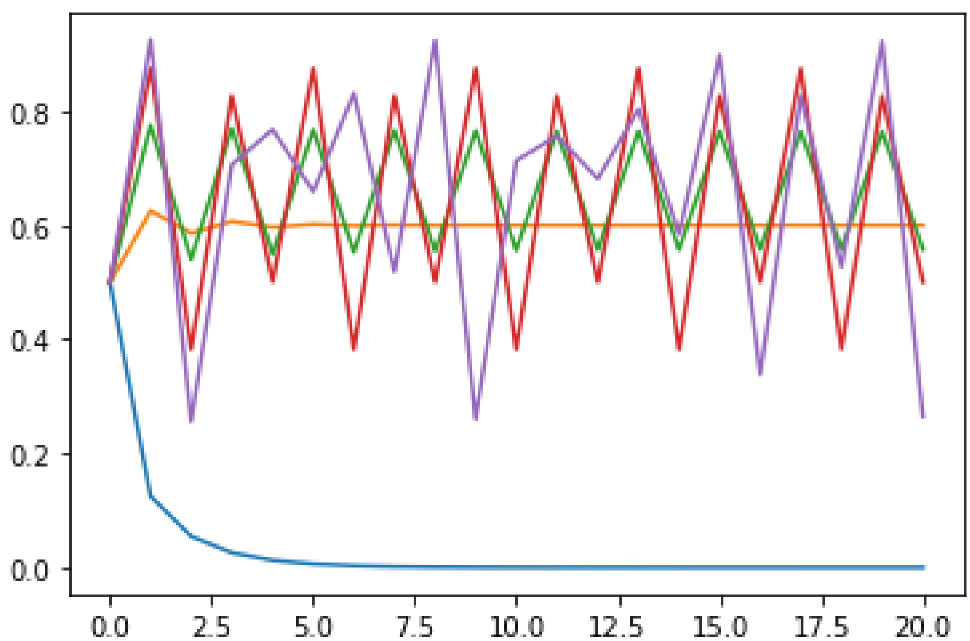
Testando diferentes valores de N

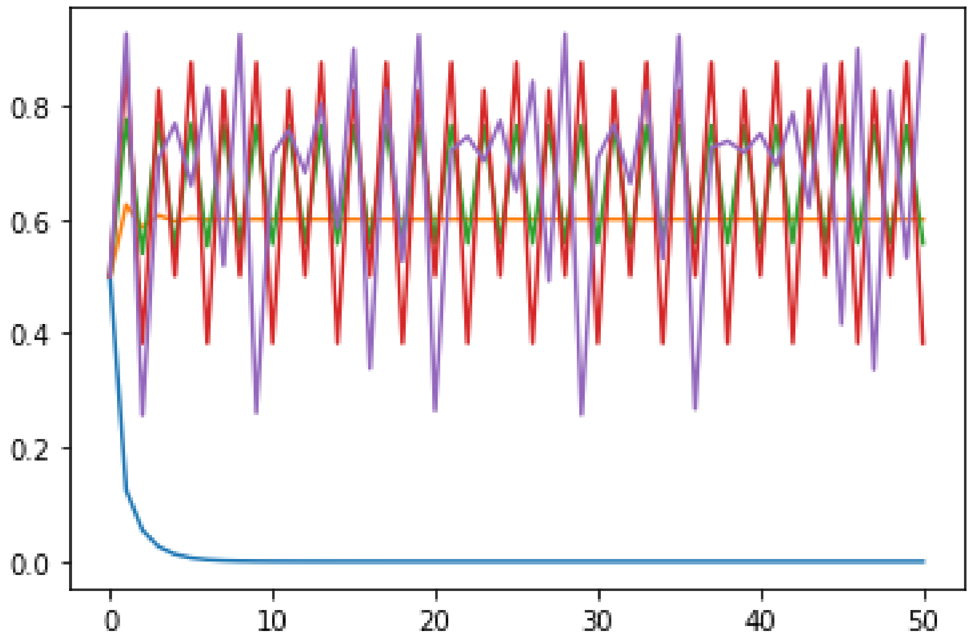
[2]:
```
rs = np.array([.5, 2.5, 3.1, 3.5, 3.7])
x0 = .5
ns = [1, 5, 10, 20, 50]
for n_range in ns:
    for r in rs:
        x = [x0]
        for n in range(n_range):
            x.append(r * x[n] * (1 - x[n]))
        plt.plot(np.arange(n_range + 1), x)
    plt.show()
```


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.2
0.4
0.6
0.8
1.0


0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5


0.0
0.2
0.4
0.6
0.8
0
2
4
6
8
10

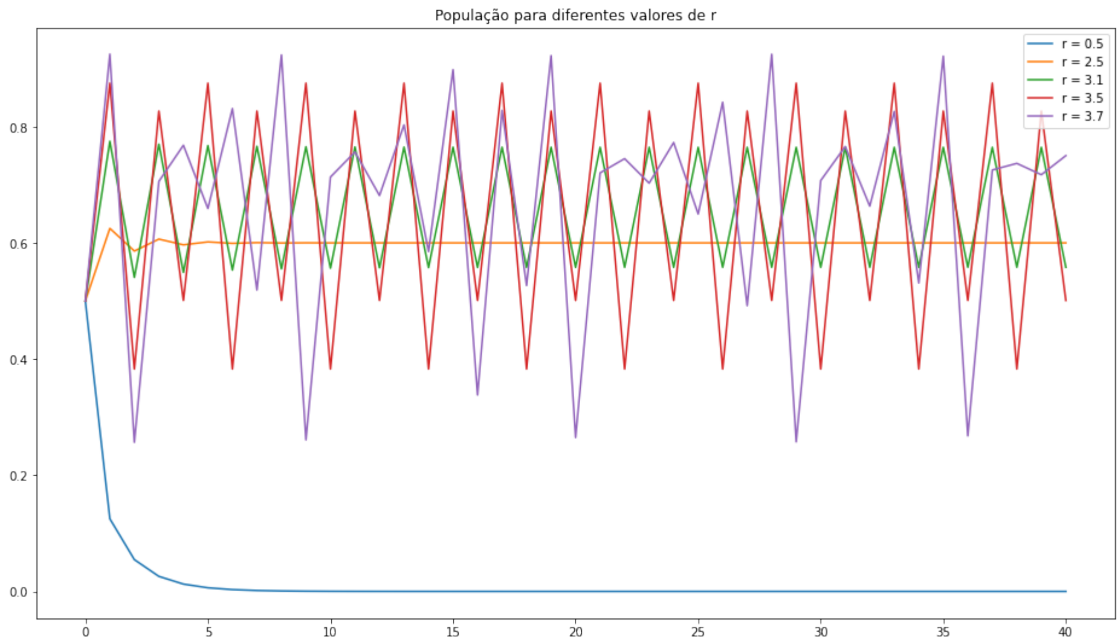

0.0
0.2
0.4
0.6
0.8
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



1.3 Passo 1

Vê-se que o gráfico abaixo rapidamente converge para 0 (população extinta) para o valor de r = 0.5. Para os outros valores de r, observa-se, em cada um deles, um padrão distinto. Por exemplo: para r = 2.5, observa-se um padrão linear. Para r = 3.1, observa-se um padrão cíclico de tamanho 2. Para valores maiores, observa-se um padrão bem mais espaçado. Para r = 3.7, não há qualquer padrão observado.

[3]:
```
rs = np.array([.5, 2.5, 3.1, 3.5, 3.7])
x0 = .5
n = 40
plt.figure(figsize=(16, 9))
for r in rs:
    x = [x0]
    for n_ in range(n):
        x.append(r * x[n_] * (1 - x[n_]))
    plt.plot(np.arange(n + 1), x, label="r = " + str(r))
plt.title("População para diferentes valores de r")
plt.legend()
plt.show()
```



1.4 Passo 2

Observa-se, nos gráficos a seguir, que para r = 0.5 e 2.5, a mudança no valor de x0 em nada altera a convergência do gráfico. Para valores de r maiores, as alterações são mais significativas, sobretudo para r = 3.7, em que são gerados gráficos completamente diferentes uns dos outros.

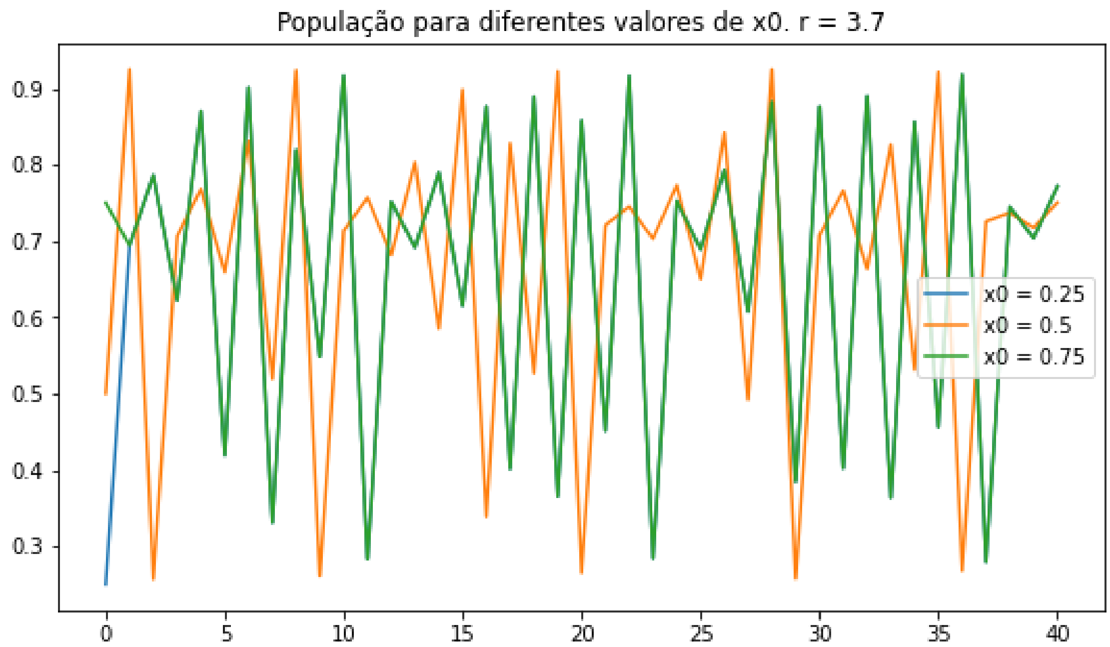
[4]:
```
rs = np.array([.5, 2.5, 3.1, 3.5, 3.7])
x0s = [.25, .5, .75]
n = 40
for r in rs:
    plt.figure(figsize=(9, 5))
    for x0 in x0s:
        x = [x0]
        for n_ in range(n):
            x.append(r * x[n_] * (1 - x[n_]))
        plt.plot(np.arange(n + 1), x, label="x0 = " + str(x0))
    plt.title("População para diferentes valores de x0. r = " + str(r))
    plt.legend()
    plt.show()
```


População para diferentes valores de x0. r = 0.5
x0 = 0.25
x0 = 0.5
x0 = 0.75
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
5
10
15
20
25
30
35
40


População para diferentes valores de x0. r = 2.5
x0 = 0.25
x0 = 0.5
x0 = 0.75
0.3
0.4
0.5
0.6
0.7
0
5
10
15
20
25
30
35
40


População para diferentes valores de x0. r = 3.1
x0 = 0.25
x0 = 0.5
x0 = 0.75


População para diferentes valores de x0. r = 3.5
x0 = 0.25
x0 = 0.5
x0 = 0.75

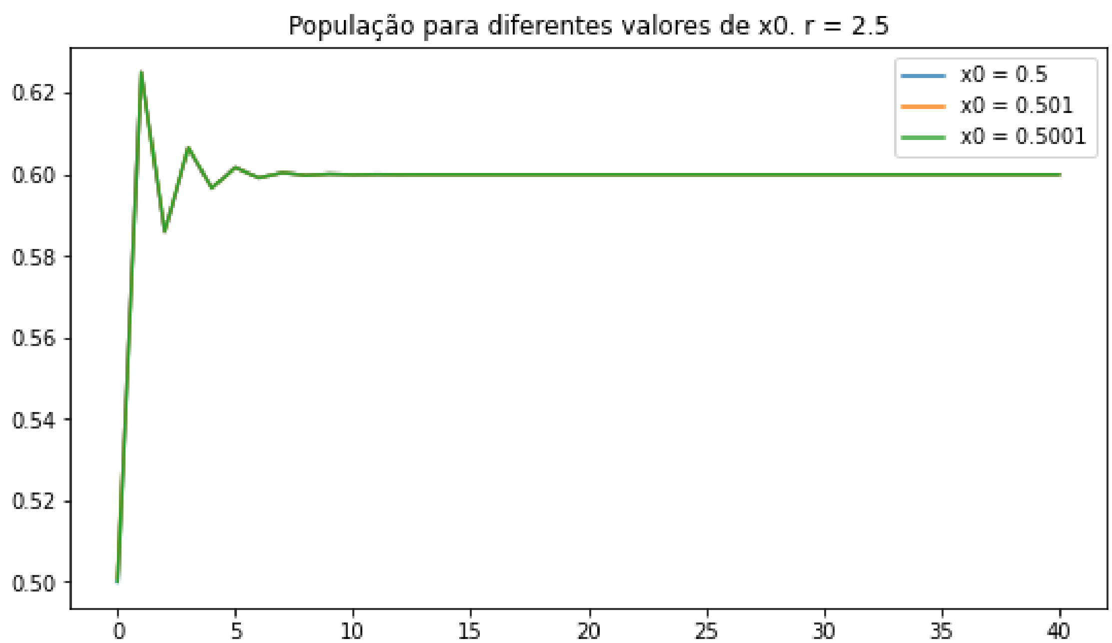


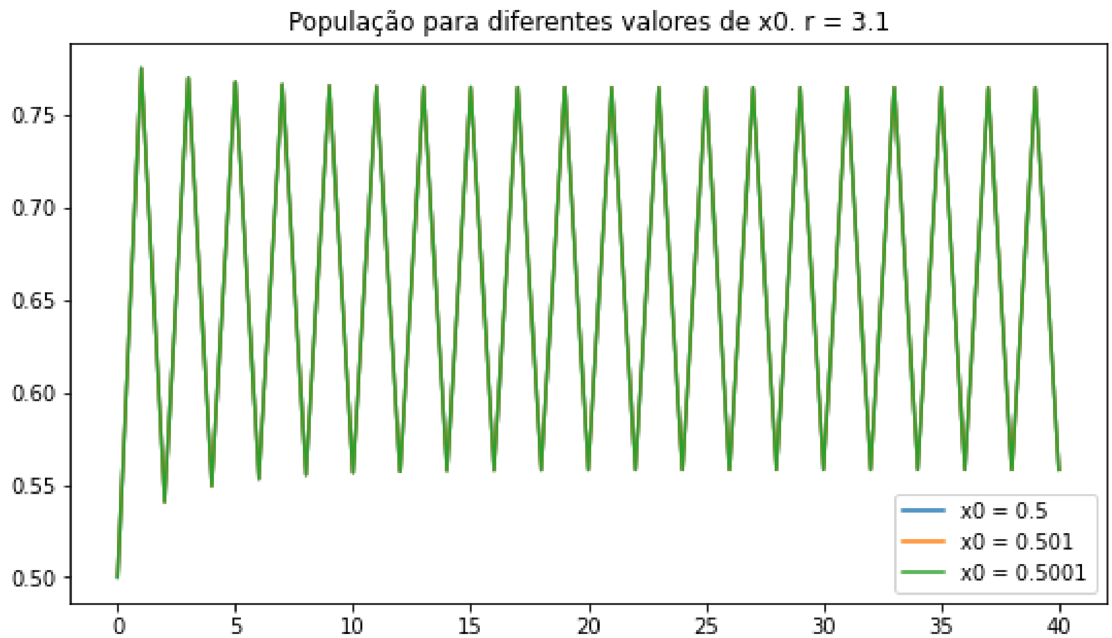
1.5 Passo 3

Para alterações tão pequenas, observa-se que o único valor em que os gráficos efetivamente tem mudanças significativas é quando r = 3.7, em que já observa-se o caos.

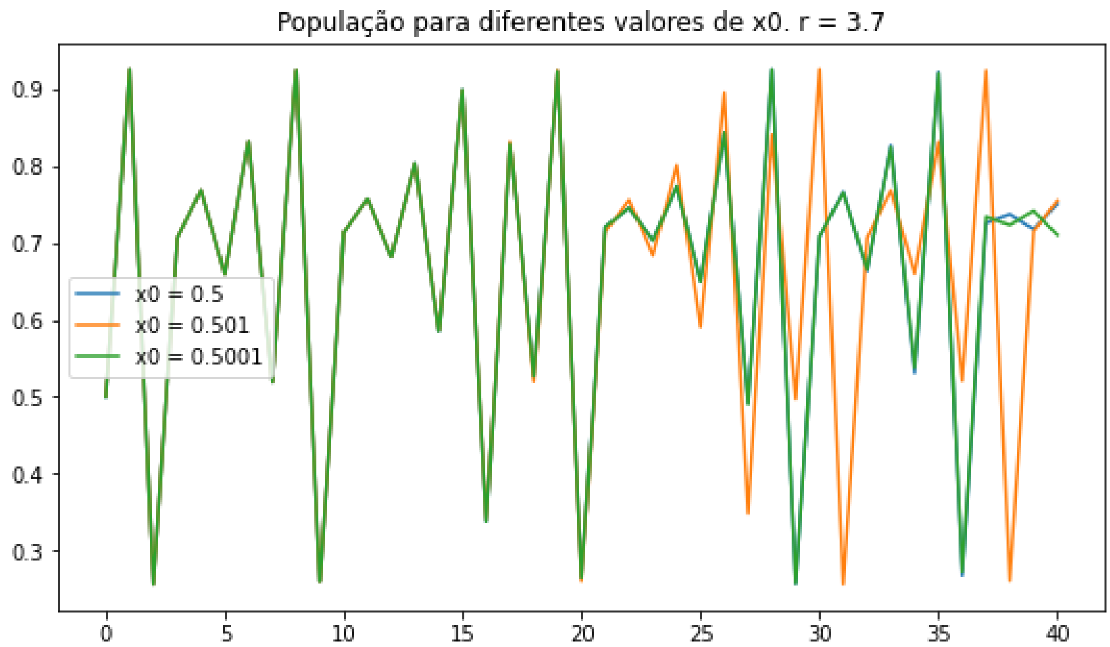
[5]:
```
rs = np.array([.5, 2.5, 3.1, 3.5, 3.7])
x0s = [.5, .501, .5001]
n = 40
for r in rs:
    plt.figure(figsize=(9, 5))
    for x0 in x0s:
        x = [x0]
        for n_ in range(n):
            x.append(r * x[n_] * (1 - x[n_]))
        plt.plot(np.arange(n + 1), x, label="x0 = " + str(x0))
    plt.title("População para diferentes valores de x0. r = " + str(r))
    plt.legend()
    plt.show()
```


População para diferentes valores de x0. r = 0.5
x0 = 0.5
x0 = 0.501
x0 = 0.5001
0.5
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25
30
35
40


População para diferentes valores de x0. r = 2.5
x0 = 0.5
x0 = 0.501
x0 = 0.5001
0.62
0.60
0.58
0.56
0.54
0.52
0.50
0
5
10
15
20
25
30
35
40

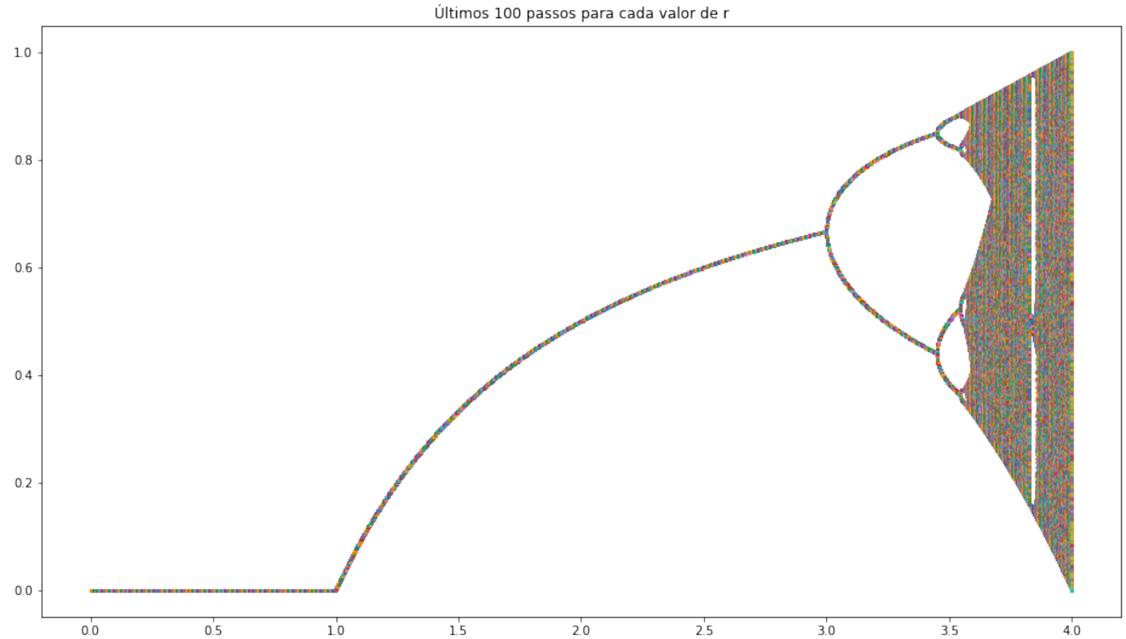

População para diferentes valores de x0. r = 3.1
0.75
0.70
0.65
0.60
0.55
0.50
0
5
10
15
20
25
30
35
40
x0 = 0.5
x0 = 0.501
x0 = 0.5001


População para diferentes valores de x0. r = 3.5
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30
35
40
x0 = 0.5
x0 = 0.501
x0 = 0.5001



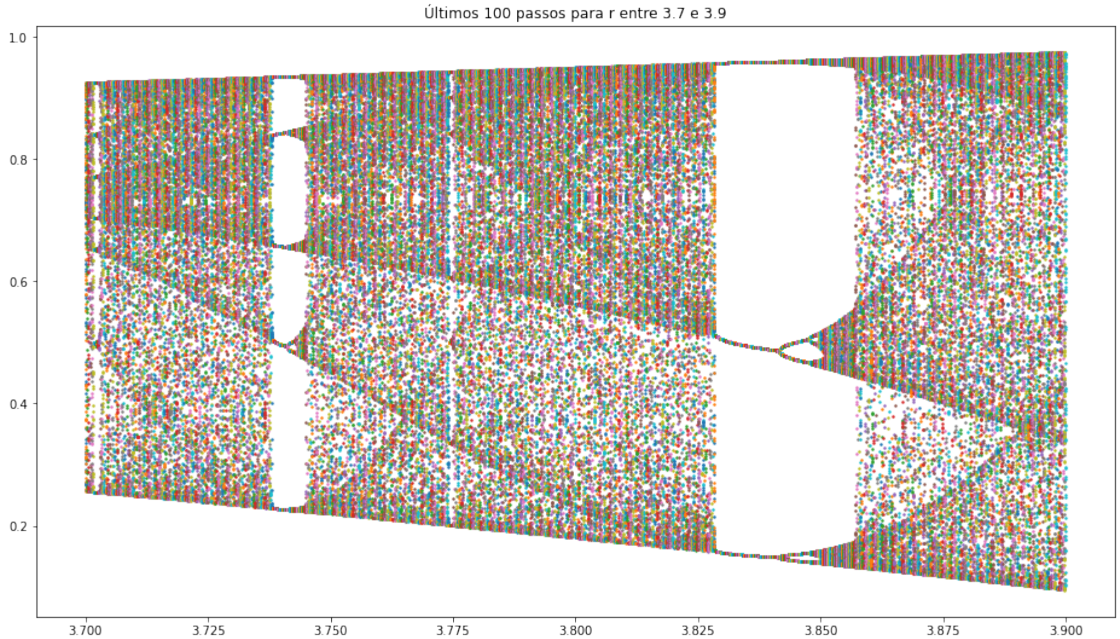
1.6 Passo 4

[6]:
```
rs = np.linspace(10e-5, 4, 30000)
x0 = .5
n = 1000
plt.figure(figsize=(16, 9))
for r in rs:
    x = [x0]
    for n_ in range(n):
        x.append(r * x[n_] * (1 - x[n_]))
    plt.scatter([r]*100, x[-100:], s=2)
plt.title("Últimos 100 passos para cada valor de r")
plt.show()
```

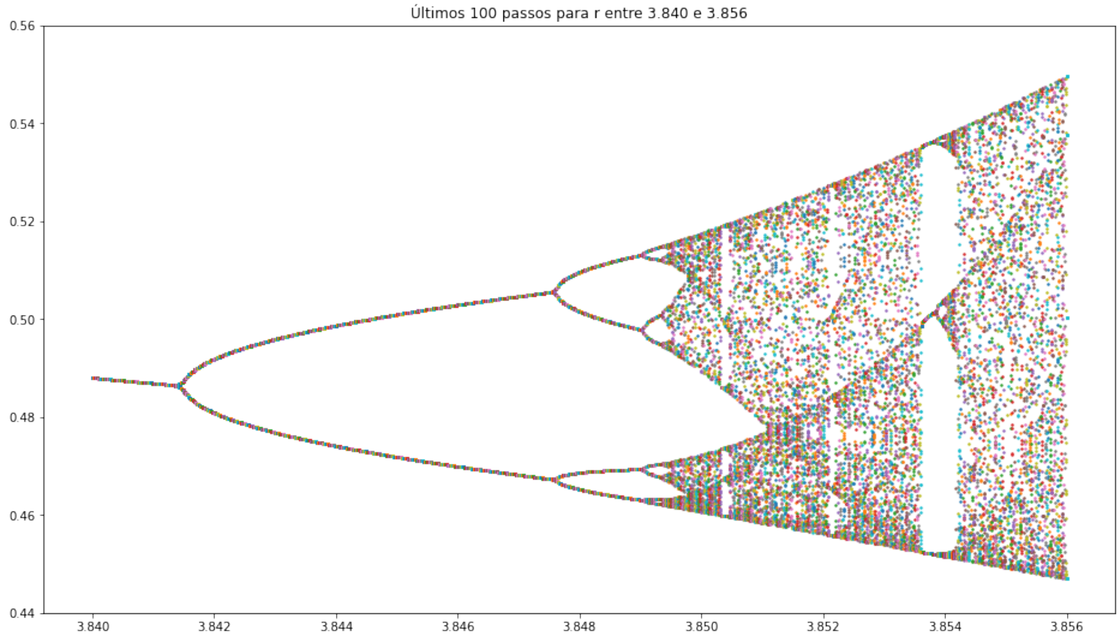


1.7 Passo 5

[7]:
```
rs = np.linspace(3.7, 3.9, 1000)
x0 = .5
n = 1000
plt.figure(figsize=(16, 9))
for r in rs:
    x = [x0]
    for n_ in range(n):
        x.append(r * x[n_] * (1 - x[n_]))
    plt.scatter([r]*100, x[-100:], s=2)
plt.title("Últimos 100 passos para r entre 3.7 e 3.9")
plt.show()
```



[8]:
```
rs = np.linspace(3.840, 3.856, 1000)
x0 = .5
n = 1000
plt.figure(figsize=(16, 9))
plt.ylim(0.44, 0.56)
for r in rs:
    x = [x0]
    for n_ in range(n):
        x.append(r * x[n_] * (1 - x[n_]))
    plt.scatter([r]*100, x[-100:], s=2)
plt.title("Últimos 100 passos para r entre 3.840 e 3.856")
plt.show()
```



1.8 Passo 6

1.8.1 Comentários

Por meio dos experimentos, conseguimos verificar que a partir do valor 3, ocorre a bifurcação inicial dos valores, e, após outro determinado valor (aproximadamente 3.5), ocorre outra bifurcação, obtendo-se 4 "caminhos". Em seguida, cada uma das 4 bifurcações gera outras bifurcações e, rapidamente, o gráfico fica sem um padrão bem definido. Em outras palavras, tem-se um gráfico caótico.

Em relação aos dois últimos gráficos, nota-se que, em meio ao caos, é possível se observar alguns padrões bem definidos, caracterizados por bifurcações em determinados valores e espaços em branco em outros.

```
[ ]:
```