Imports

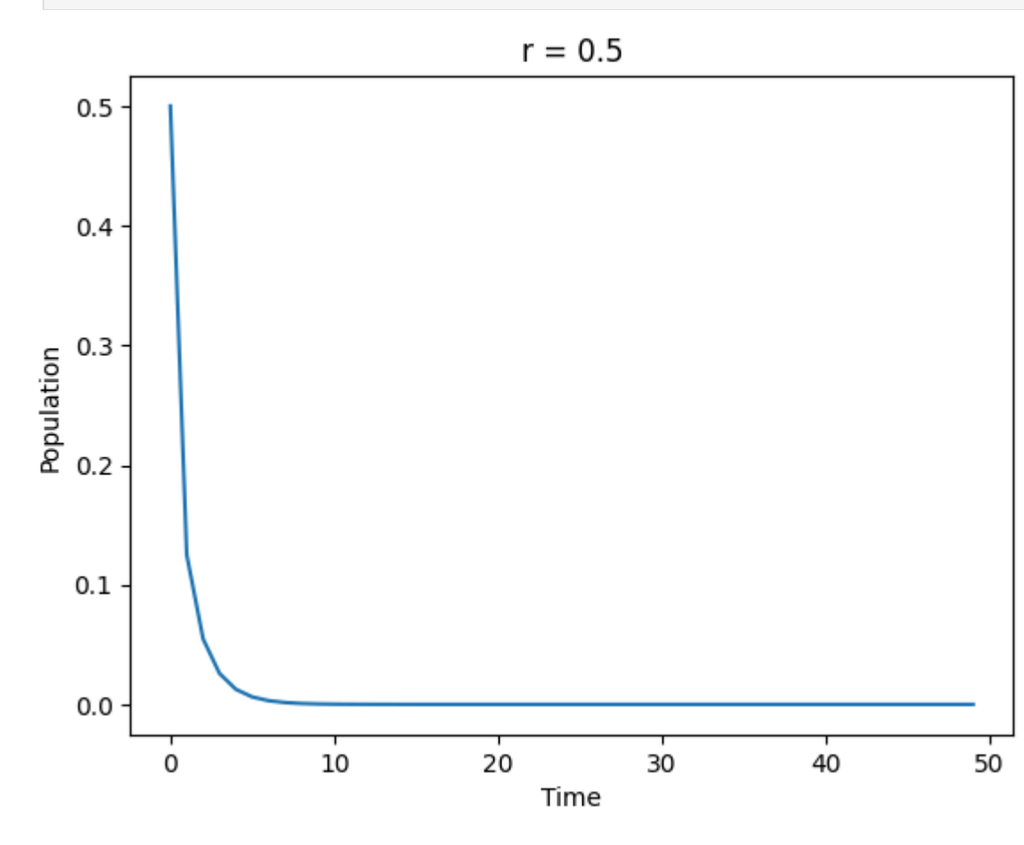
```
import matplotlib.pyplot as plt
import numpy as np
```

Tarefa 1

1. Fa¸ca gr´aficos de yn como fun¸c˜ao de n para r = 0, 5. Use y0 = 0, 5 e n entre 1 e 50.

```
def func(r, yn):
  return r * yn * (1 - yn)

def plot(r, yo, t, title):
  y = []
  y.append(yo)

  for i in range(1, t):
    y.append(func(r, y[i - 1]))

  plt.plot(range(0, t), y)

  plt.title(title)
  plt.xlabel('Time')
  plt.ylabel('Population')

  plt.show()
```

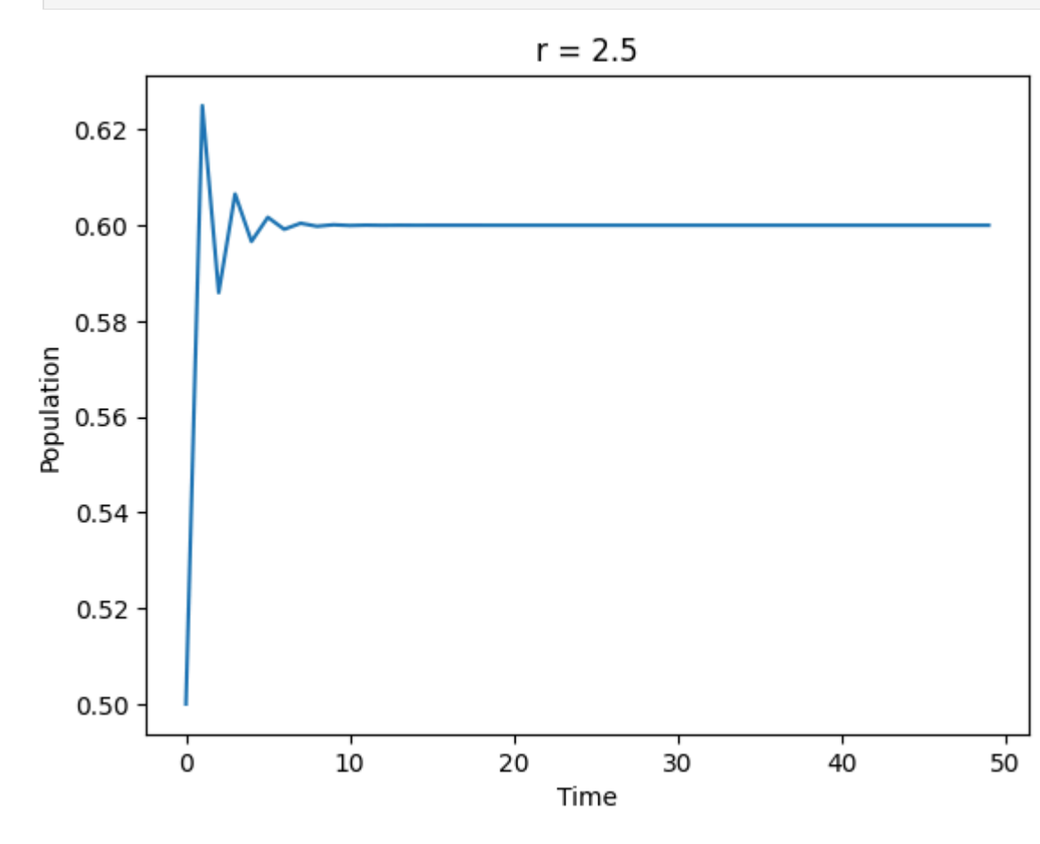
```
plot(r=0.5, yo=0.5, t=50, title='r = 0.5')
```

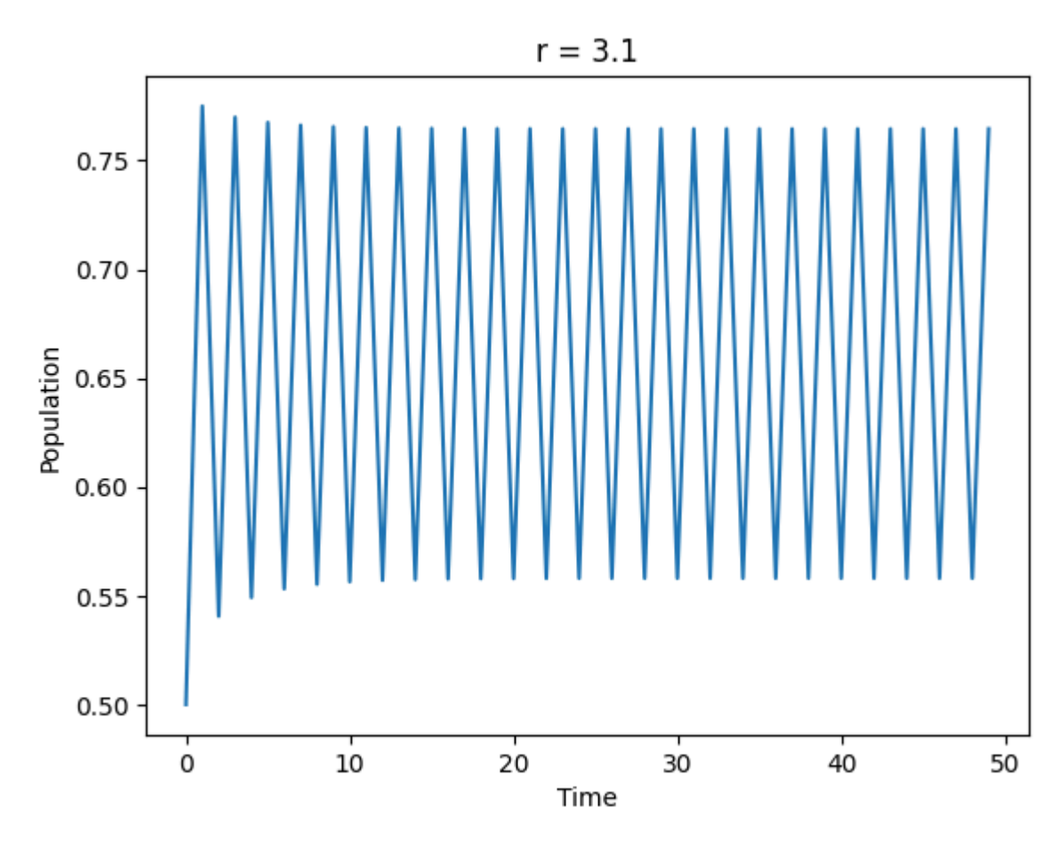


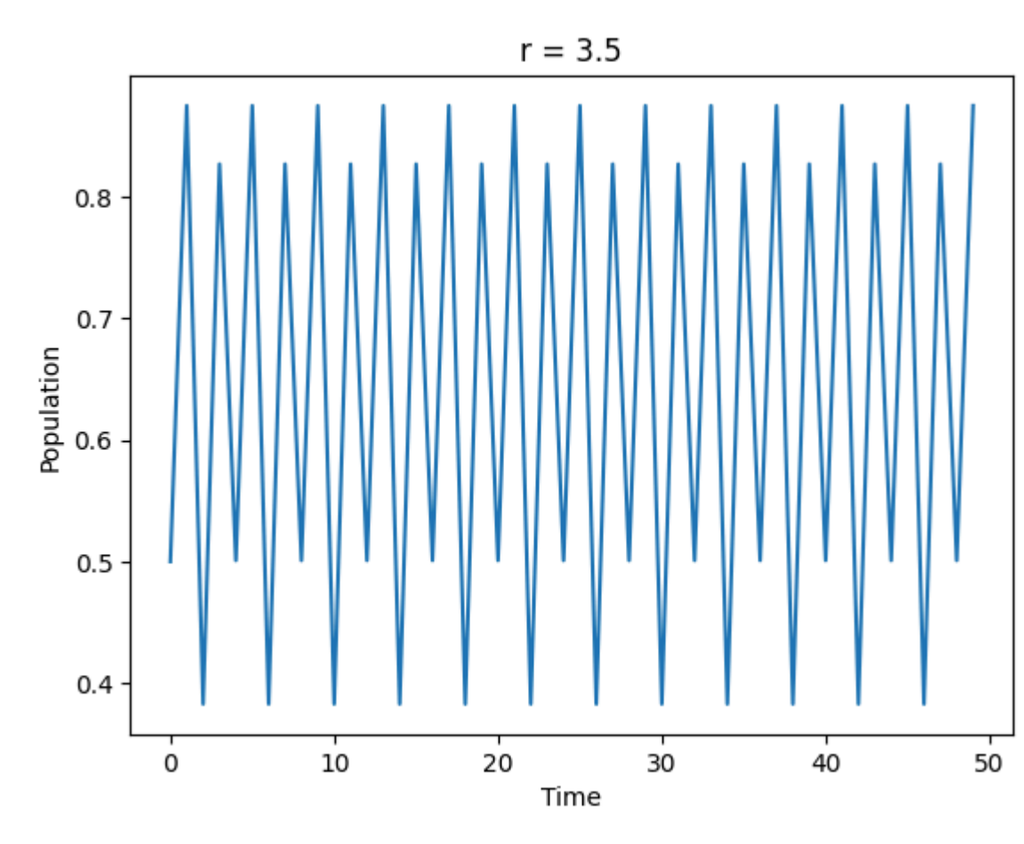
Tarefa 2

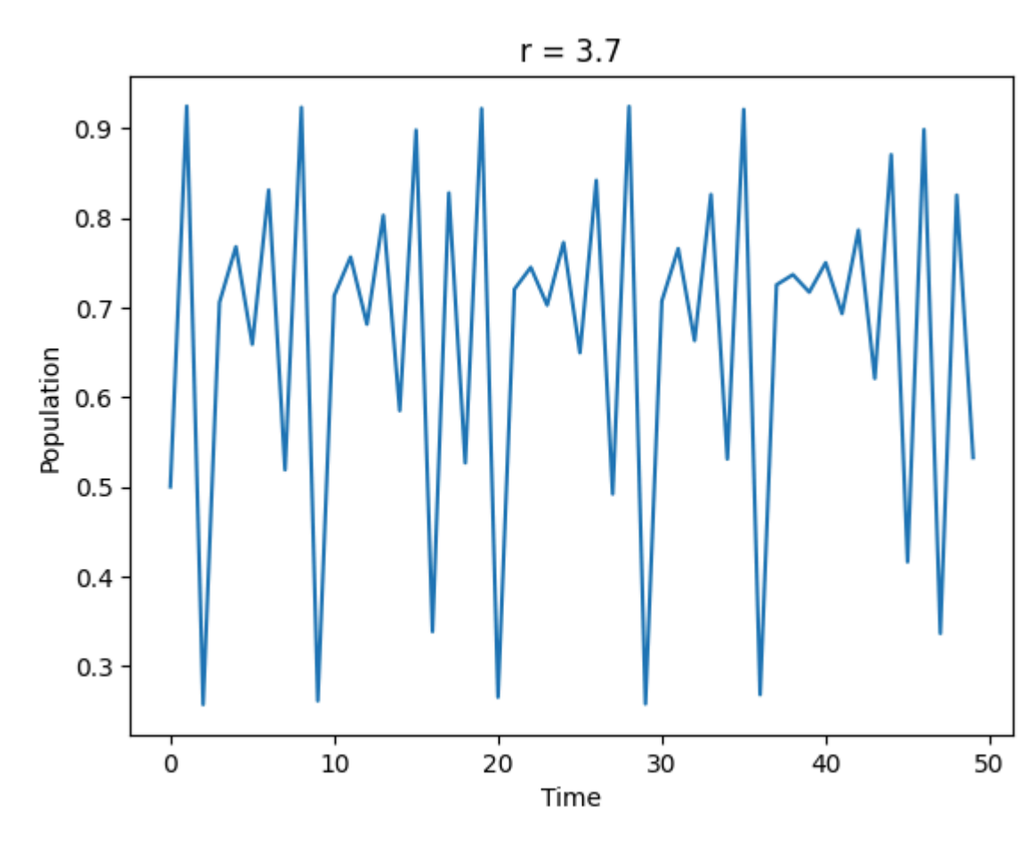
2. Repita o procedimento acima para os seguintes valores de r: 2.5; 3.1; 3.5 e 3.7. Comente sobre cada resultado. Em particular comente sobre a exist^encia de padr˜oes.

```
plot(r=2.5, yo=0.5, t=50, title='r = 2.5')
plot(r=3.1, yo=0.5, t=50, title='r = 3.1')
plot(r=3.5, yo=0.5, t=50, title='r = 3.5')
plot(r=3.7, yo=0.5, t=50, title='r = 3.7')
```









Comentário:

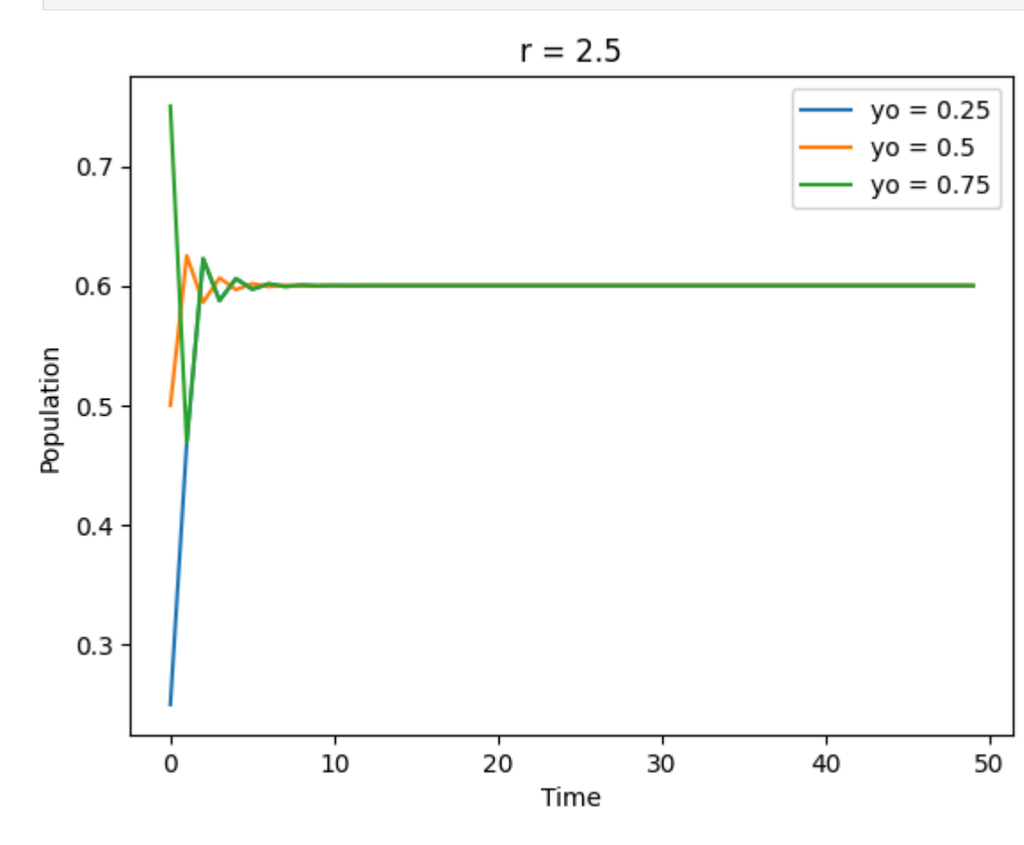
Veja que há divergência na existência de padrões que se repetem nos gráficos. Veja que no gráfico r = 3.5, há uma padrão muito bem definido. Já em outros o padrão não é perfeito mas possui similaridades entre diferentes intervalos do eixo x, como no gráfico r = 3.7. Já em outros não podemos perceber padrão algum como o r = 2.5 que no início é instável mas depois converge.

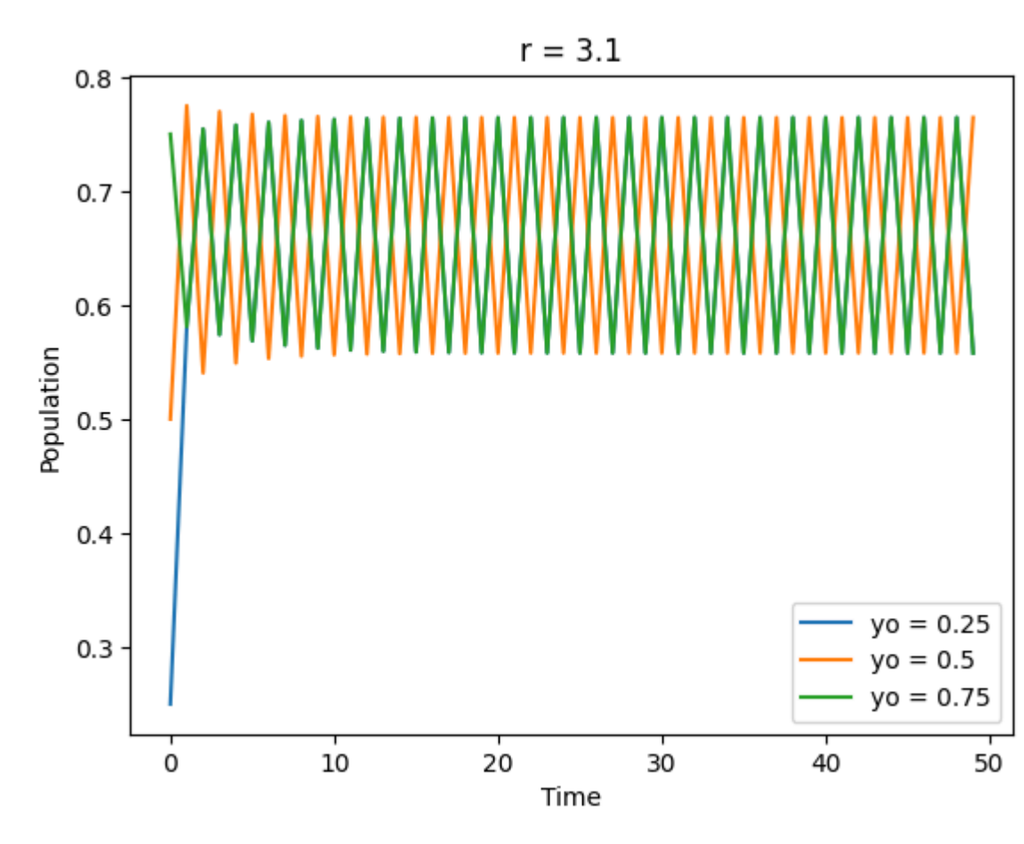
Tarefa 3

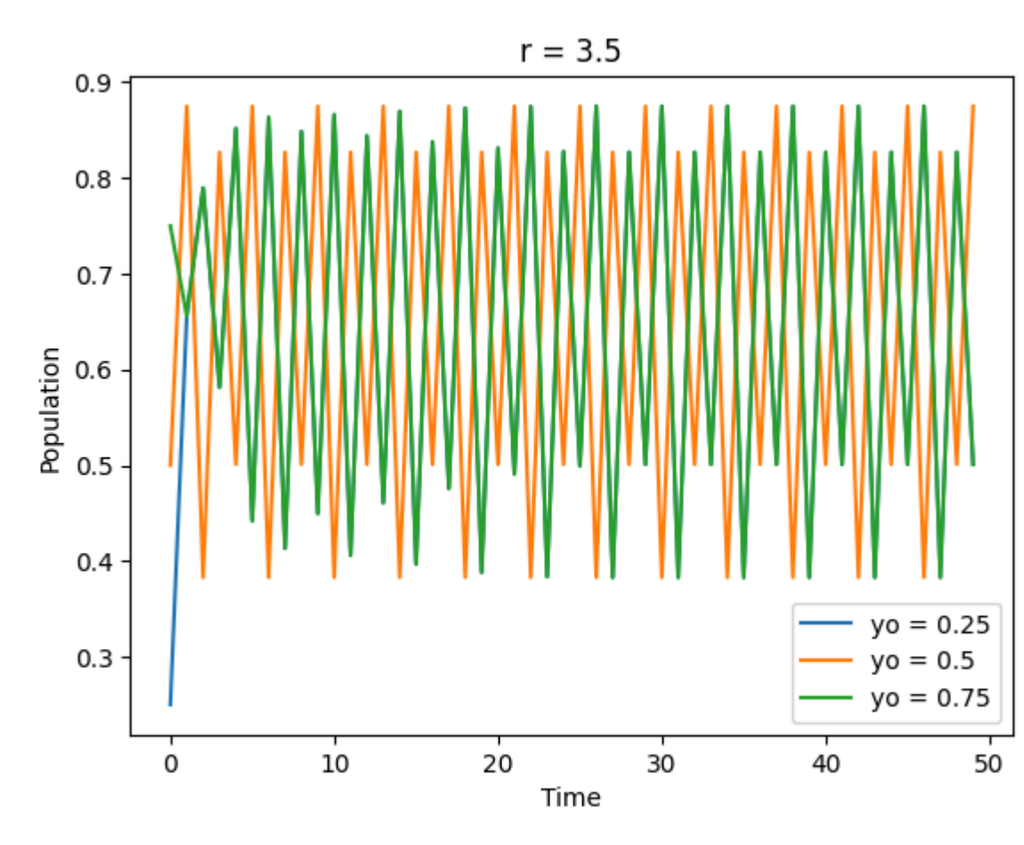
3. Refa¸ca os gr´aficos anteriores considerando tr^es condi¸c˜oes iniciais consideravelmente diferentes, ou seja, trace, em um mesmo gr´afico, as curvas para y0 = 0.25, y0 = 0.5 e y0 = 0.75. Comente

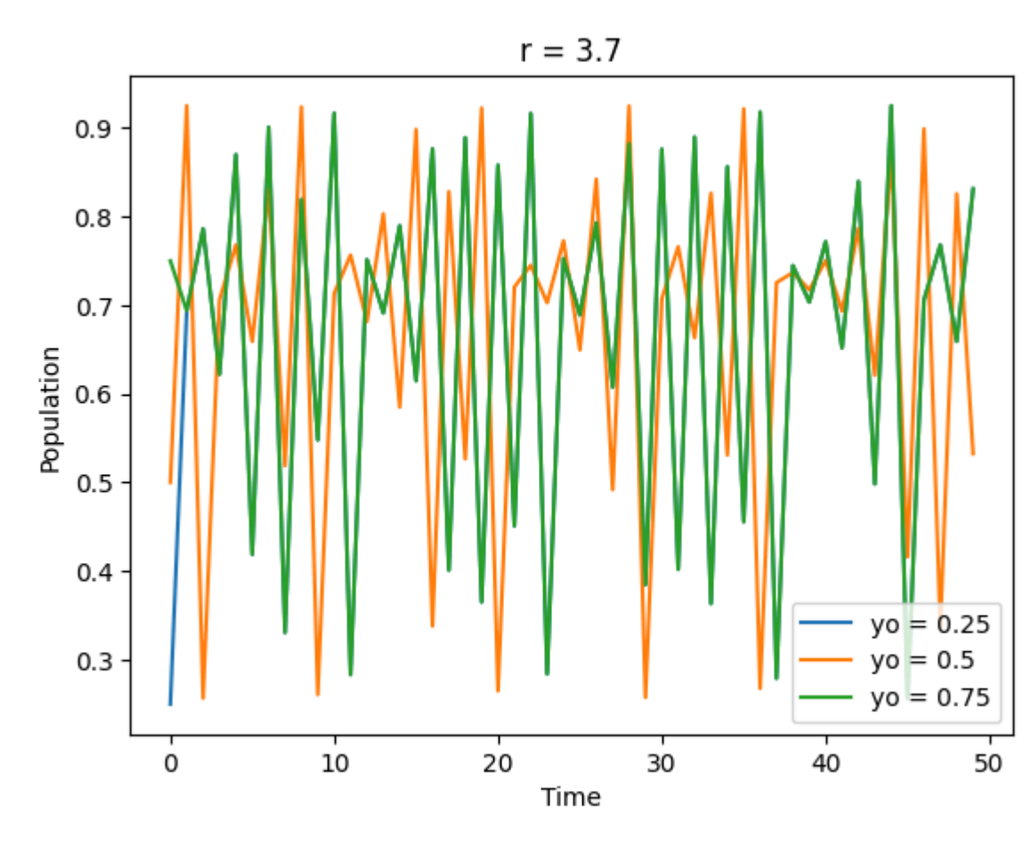
```
def plot_same_graph(r, yo, t, title):
    for yo_i in yo:
        y = []
        y.append(yo_i)

        for i in range(1, t):
            y.append(func(r, y[i - 1]))

        plt.plot(range(0, t), y, label=f'yo = {yo_i}')

    plt.title(title)
    plt.xlabel('Time')
    plt.ylabel('Population')

    plt.legend()
    plt.show()
```

```
plot_same_graph(r=2.5, yo=[0.25,0.5,0.75], t=50, title='r = 2.5')
plot_same_graph(r=3.1, yo=[0.25,0.5,0.75], t=50, title='r = 3.1')
plot_same_graph(r=3.5, yo=[0.25,0.5,0.75], t=50, title='r = 3.5')
plot_same_graph(r=3.7, yo=[0.25,0.5,0.75], t=50, title='r = 3.7')
```









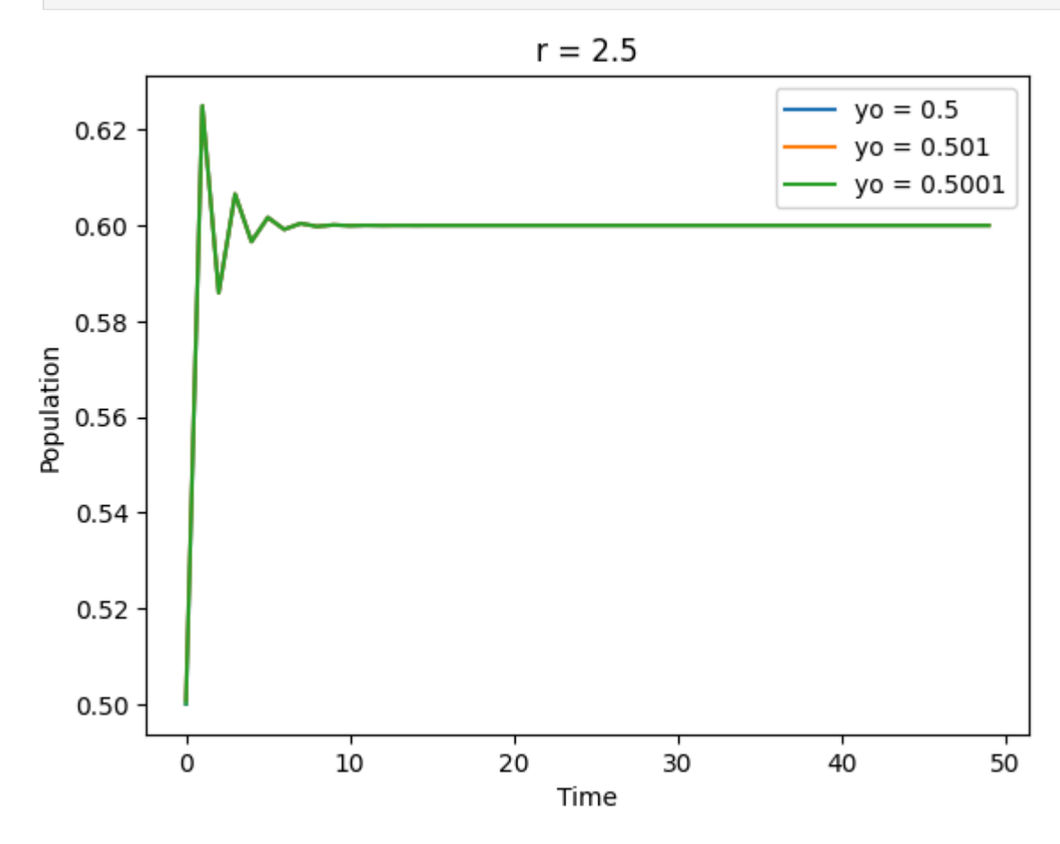
Comentário:

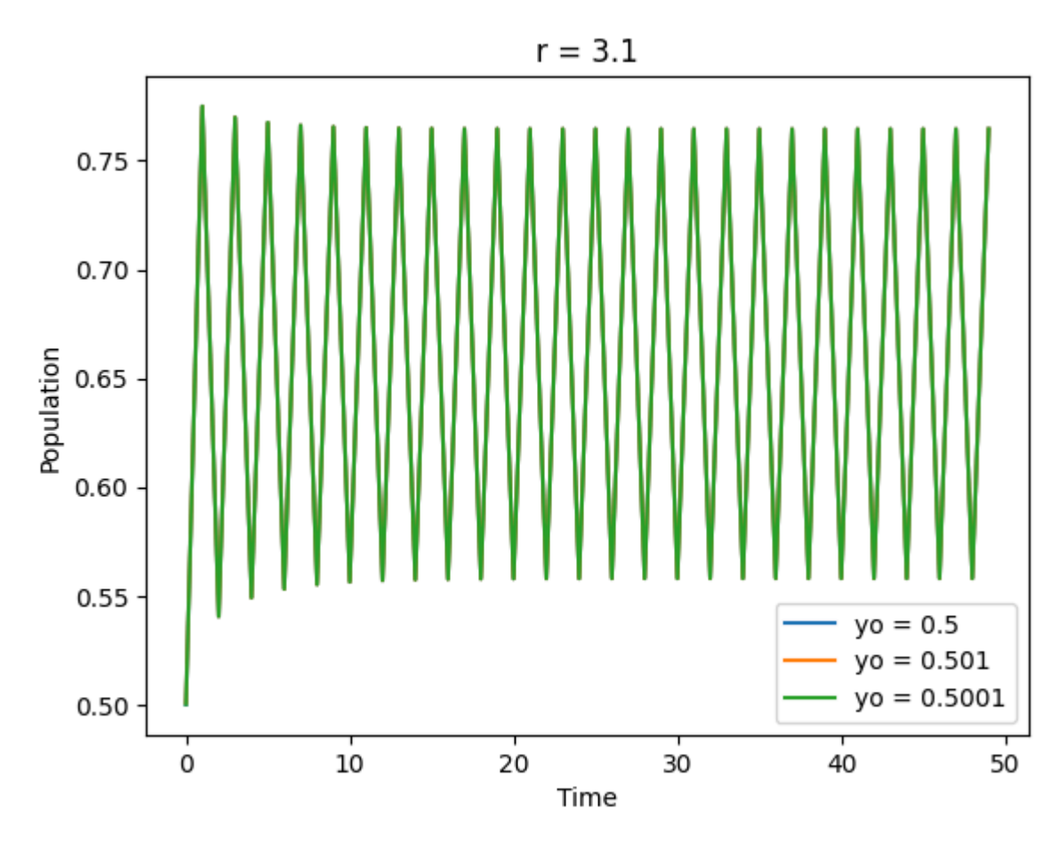
Veja que á medida que o r aumenta, mais caos o sistema tende a ter

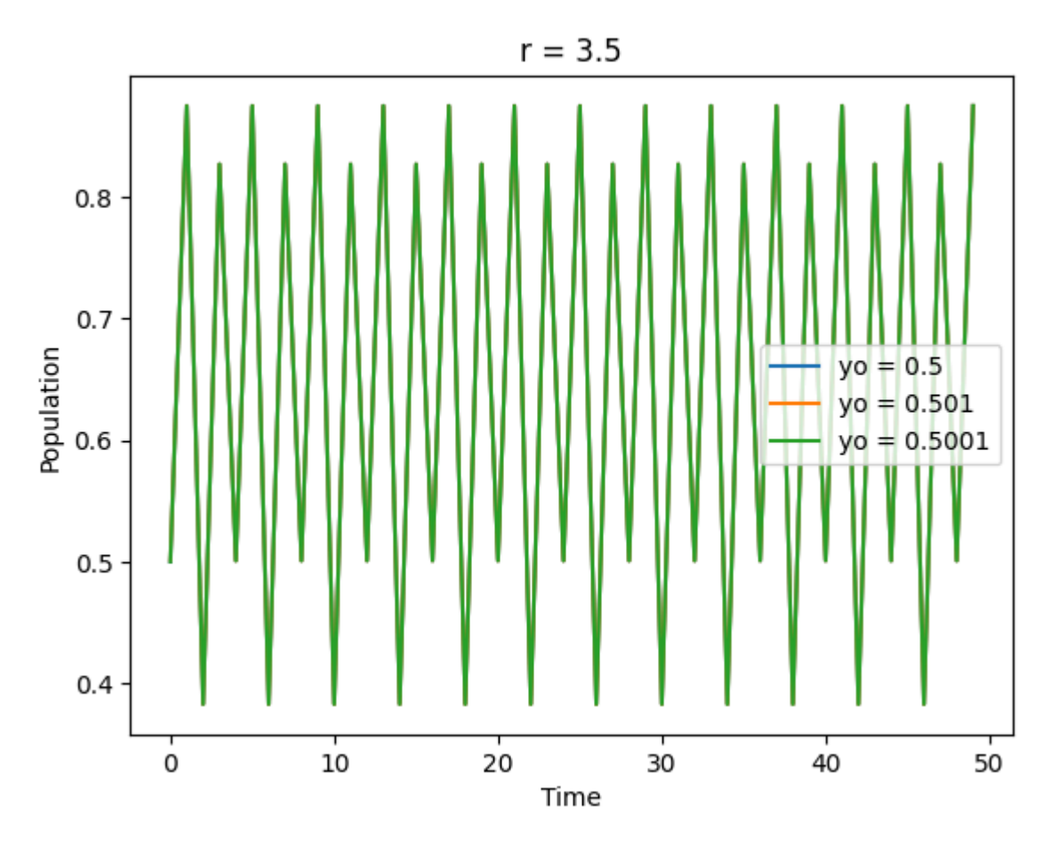
Tarefa 4

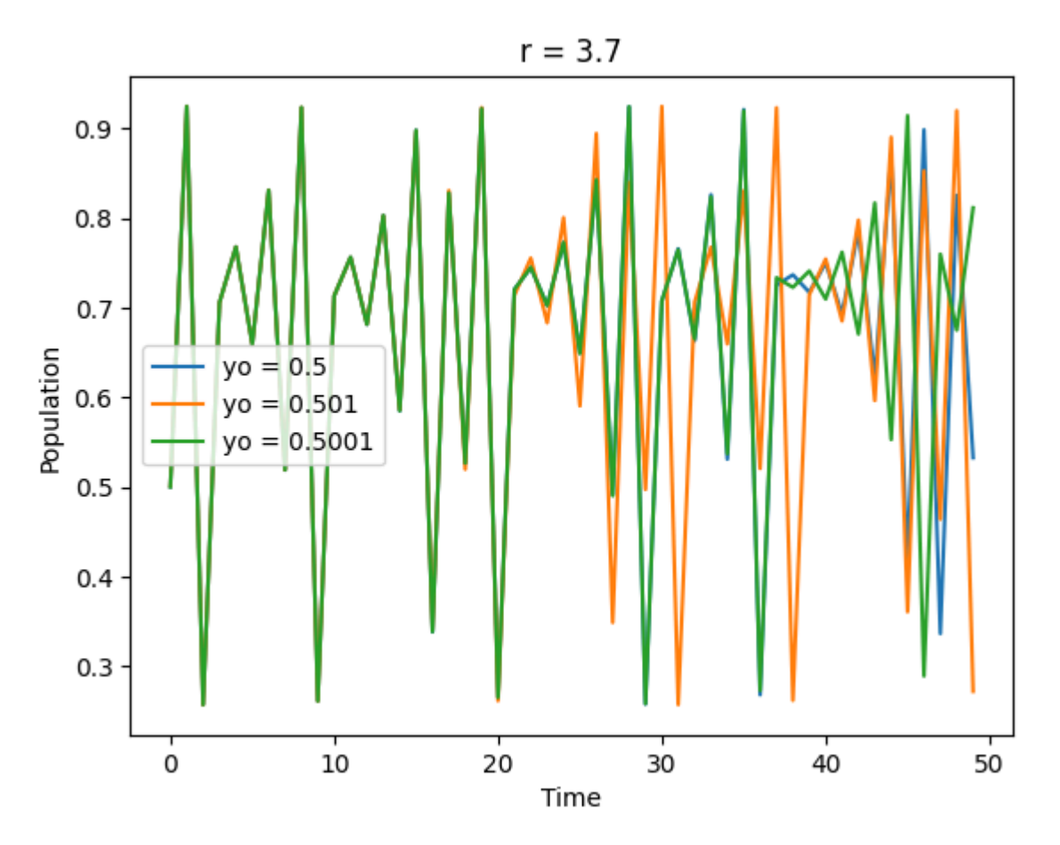
4. Refa¸ca o item anterior considerando tr^es condi¸c˜oes iniciais ligeiramente diferentes, ou seja, trace as curvas para y0 = 0.5, y0 = 0.501 e y0 = 0.5001. Comente

```
plot_same_graph(r=2.5, yo=[0.5,0.501,0.5001], t=50, title='r = 2.5')
plot_same_graph(r=3.1, yo=[0.5,0.501,0.5001], t=50, title='r = 3.1')
plot_same_graph(r=3.5, yo=[0.5,0.501,0.5001], t=50, title='r = 3.5')
plot_same_graph(r=3.7, yo=[0.5,0.501,0.5001], t=50, title='r = 3.7')
```









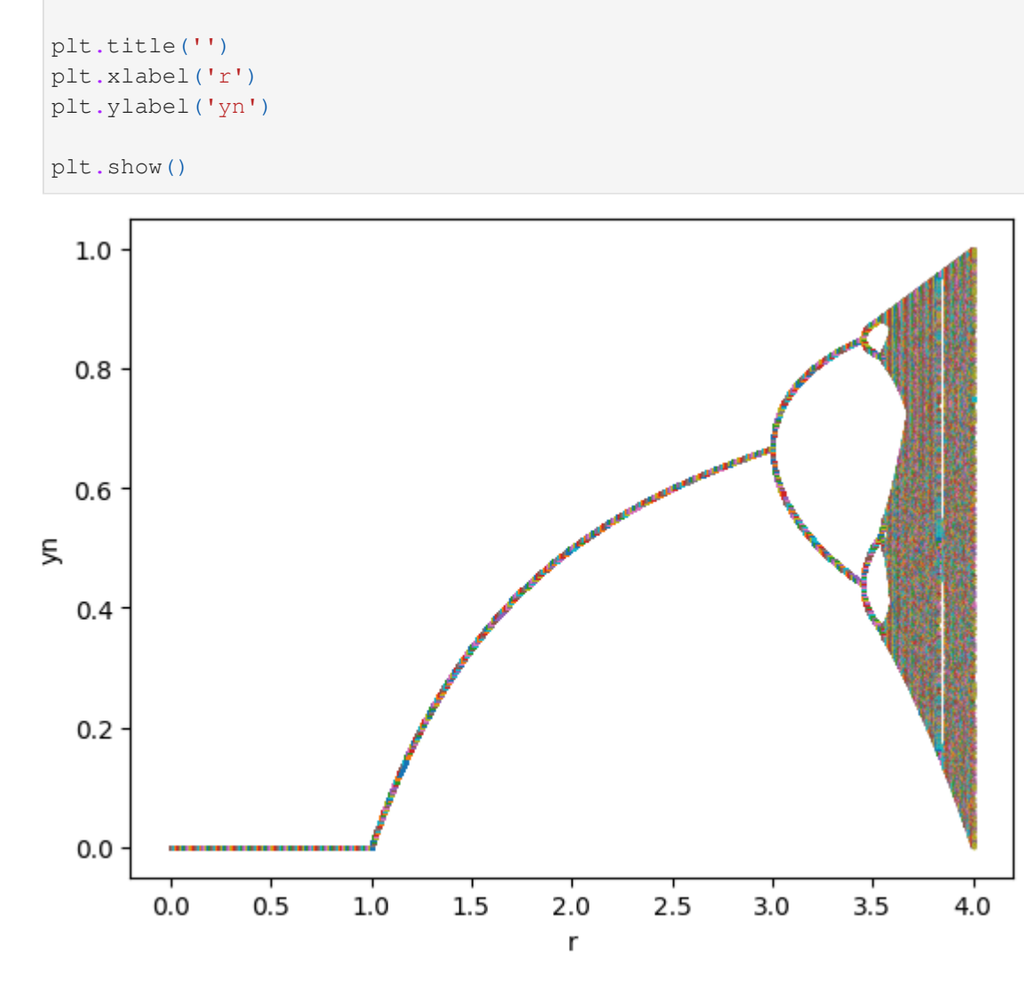
Comentário:

No início, os sistemas parecem se comportar da mesma maneira por terem condições iniciais muito próximas. Contudo, a medida que o tempo passa, as pequenas diferençãs inciais são amplificadas e o estados dos sistemas são completamente diferentes. Quanto maior o valor de r, mais rápido ocorre essa divergência entre os estados dos sistemas

Tarefa 5

5. Considere 10000 valores de r igualmente espa¸cados entre 10−5 e 4. Para cada um desses valores de r, vamos iterar a equa¸c˜ao log´ıstica 1000 vezes, i.e., nmax = 1000. Em um gr´afico de yn como fun¸c˜ao de r, vamos plotar para cada valor de r os 100 ´ultimos passos da evolu¸c˜ao. Teremos 100 pontos no eixo y do gr´afico para cada valor de r, que estar´a no eixo x. A figura gerada ´e conhecida como diagrama de bifurca¸c˜ao. Essa figura traz in´umeras informa¸c˜oes importantes. Tire um tempo para pensar nela e analisar o que encontrou.

```
def last_100(r, yo, t=1000):
    y = []
    y.append(yo)

    for i in range(1, t):
        y.append(func(r, y[i - 1]))

    return y[(t-100):]

r_values = np.linspace(10e-5, 4, 10000)

yo = 0.25

for r in r_values:
    yn = last_100(r=r, yo=yo)
    plt.plot([r for _ in range(100)], yn, marker='o', markersize=0.5, markerfacecolor='black', linestyle='None')

plt.title('')
plt.xlabel('r')
plt.ylabel('yn')

plt.show()
```



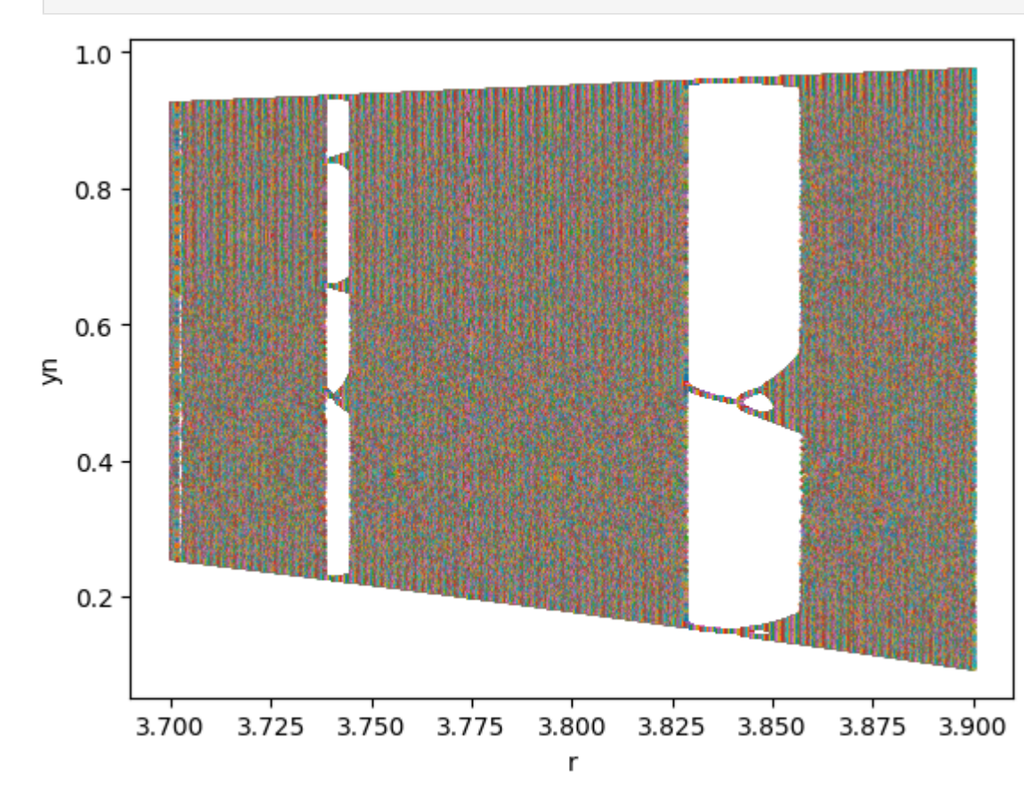
Comentário:

Para valores baixos de r, o sistema tende a um ponto fixo estável. Porém à medida que o valor de r aumenta, o ponto fixo se torna instável e o sistema bifurca. Em valores cada vez maiores de r, o sistema se torna caótico, fato que se torna claro com r > 3.7 (aproximadamente)

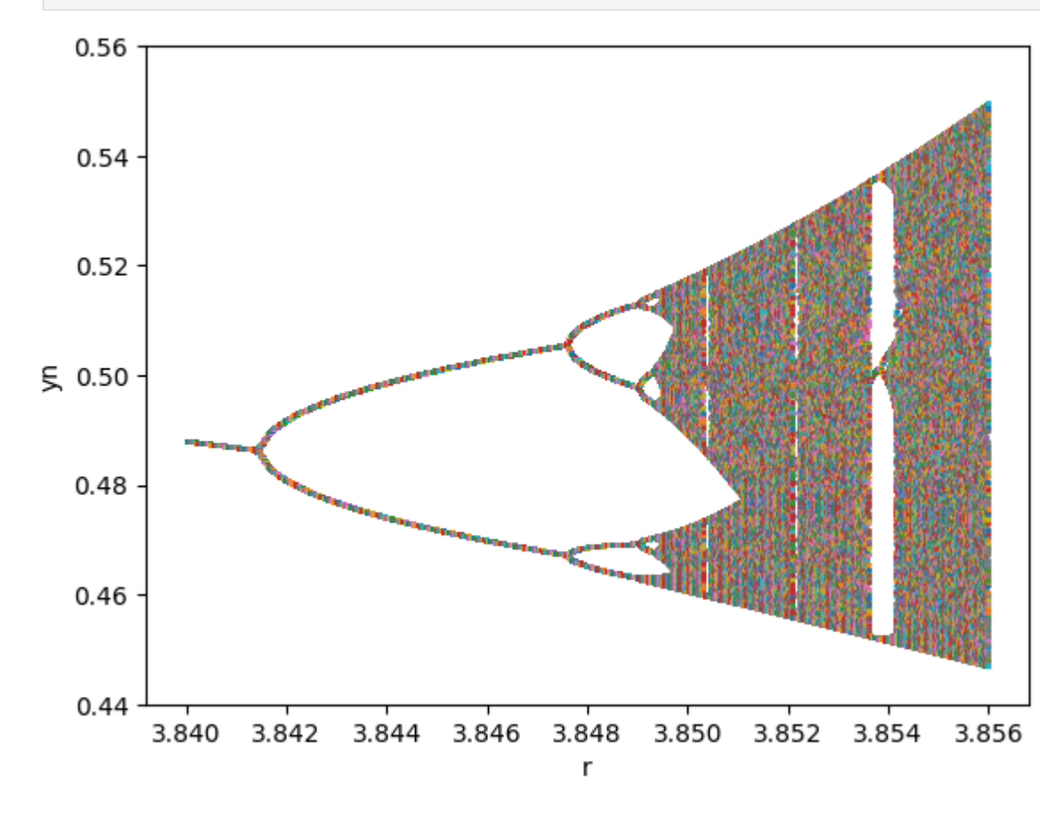
Tarefa 6

6. Para explorar melhor uma das interessantes propriedades desse diagrama, refa¸ca o procedimento anterior para valores de r entre 3.7 e 3.9 e para valores de r entre 3.840 e 3.856. No ´ultimo caso, para melhorar a visualiza¸c˜ao, restrinja os valores do eixo y para o intervalo entre 0,44 e 0,56. Comente seus resultados.

```
r_values = np.linspace(3.7, 3.9, 10000)

yo = 0.25

for r in r_values:
    yn = last_100(r=r, yo=yo)
    plt.plot([r for _ in range(100)], yn, marker='o', markersize=0.5, markerfacecolor='black', linestyle='None')

plt.title('')
plt.xlabel('r')
plt.ylabel('yn')

plt.show()
```




```
r_values = np.linspace(3.84, 3.856, 10000)

yo = 0.25

for r in r_values:
    yn = last_100(r=r, yo=yo)
    plt.plot([r for _ in range(100)], yn, marker='o', markersize=1, linestyle='None')

plt.title('')
plt.xlabel('r')
plt.ylabel('yn')
plt.ylim(0.44,0.56)

plt.show()
```



Comentário:

No gráfico com r no intervalo de [1.7, 3.9], como há um salto maior entre os valores de r, o sistema é caótico na sua maior parte tendo algumas janelas de estabilidade. Já no gráfico com r no intervalo de [3.840, 3.856], há saltos menores nos valores de r e consequentemente pertubamos uma estabilidade maior no início. Contudo, podemos observar o gráfico com r no intervalo de [3.840, 3.856] como uma senóide "zoomed in" do gráfico com r no intervalo de [1.7, 3.9], como se fosse um fractal.