Nome: Arthur Pontes Nader

Matrícula: 2019022294

Primeira Parte

In [16]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
def gerarResultados(r, y0, n = 50):

    resultados = []
    y_aux = y0

    for i in range(0,n):
        resultados.append(y_aux)
        y_aux = r*y_aux*(1-y_aux)

    return resultados
```

In [3]:
```
def gerarGrafico(resultados, titulo):

    plt.figure(figsize=(10, 6))
    plt.xlabel("Tempo", fontsize=14)
    plt.ylabel("População", fontsize=14)
    plt.title(titulo, fontsize=18)
    plt.plot(resultados)
    plt.show()
```

Gerando os resultados variando r

In [4]:
```
valores_r = [0.5, 2.5, 3.1, 3.5, 3.7]
```

In [5]:
```
for r in valores_r:
    resultados = gerarResultados(r, 0.5)
    gerarGrafico(resultados, "Grafico para r = "+ str(r) +" e y0 = 0.5")
```

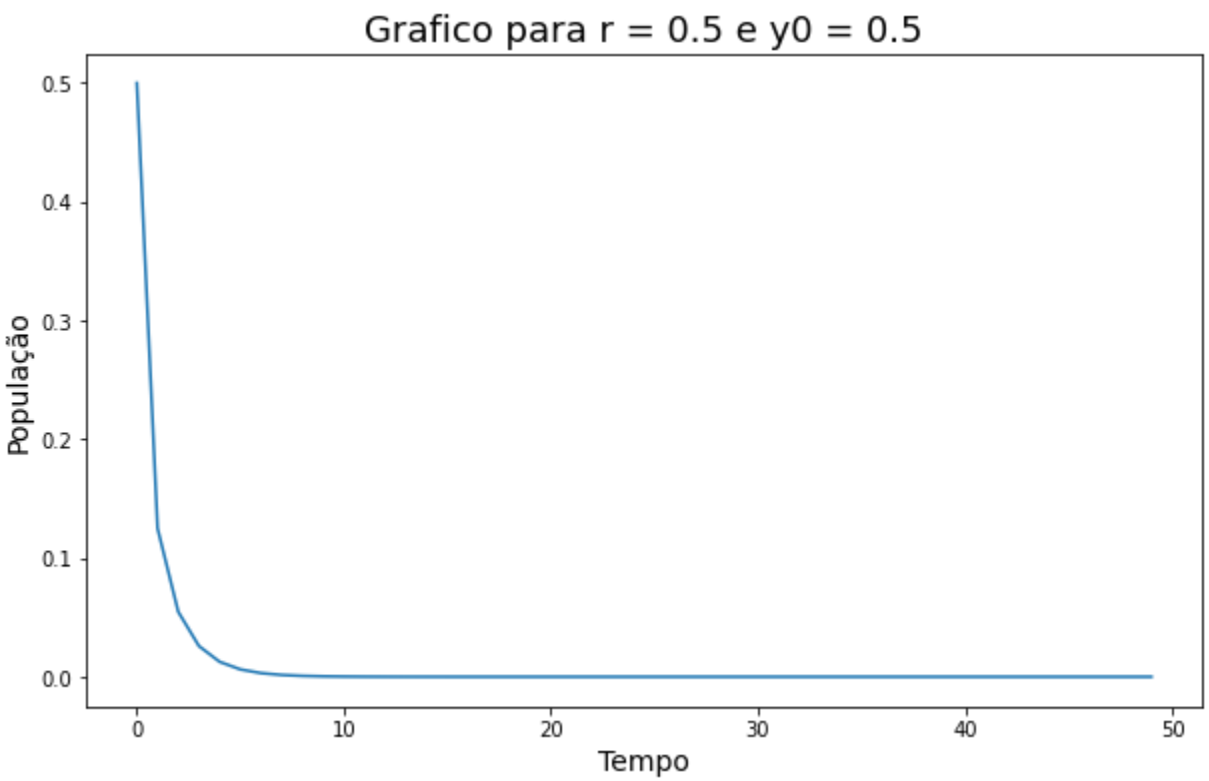


Grafico para r = 0.5 e y0 = 0.5
População
Tempo
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50

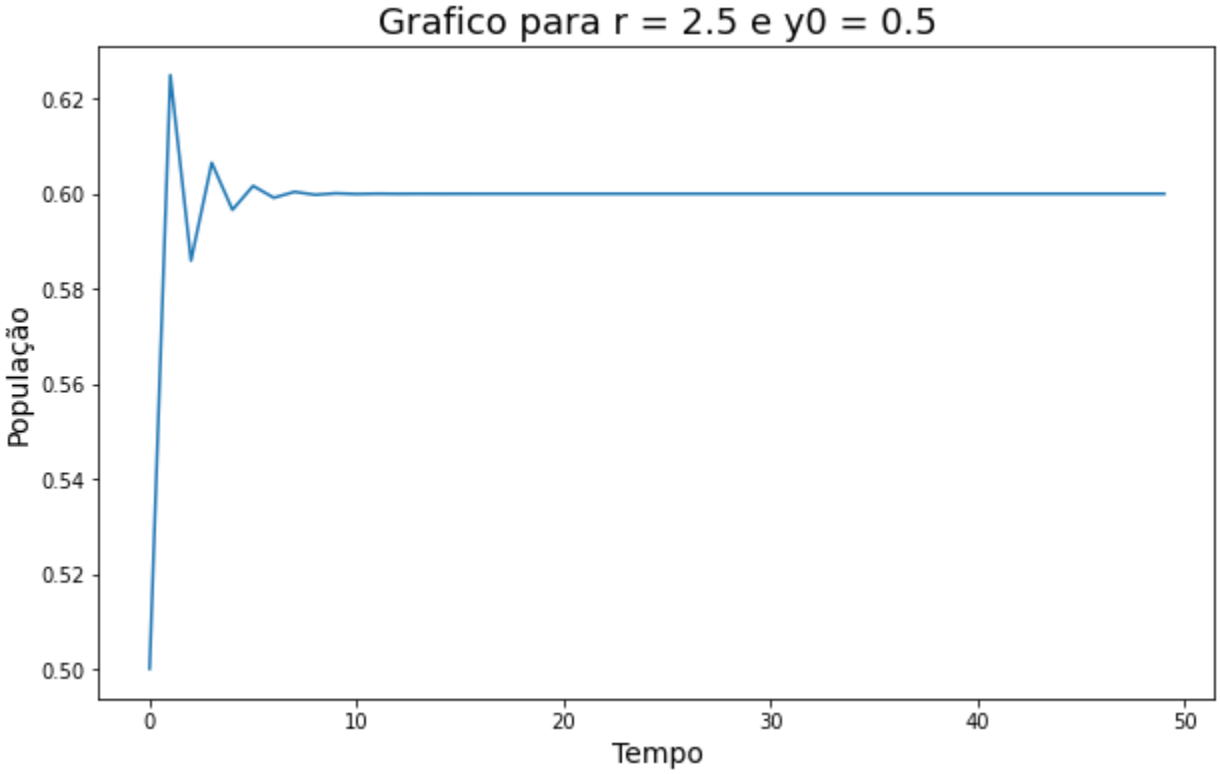


Grafico para r = 2.5 e y0 = 0.5
População
Tempo
0.62
0.60
0.58
0.56
0.54
0.52
0.50
0
10
20
30
40
50

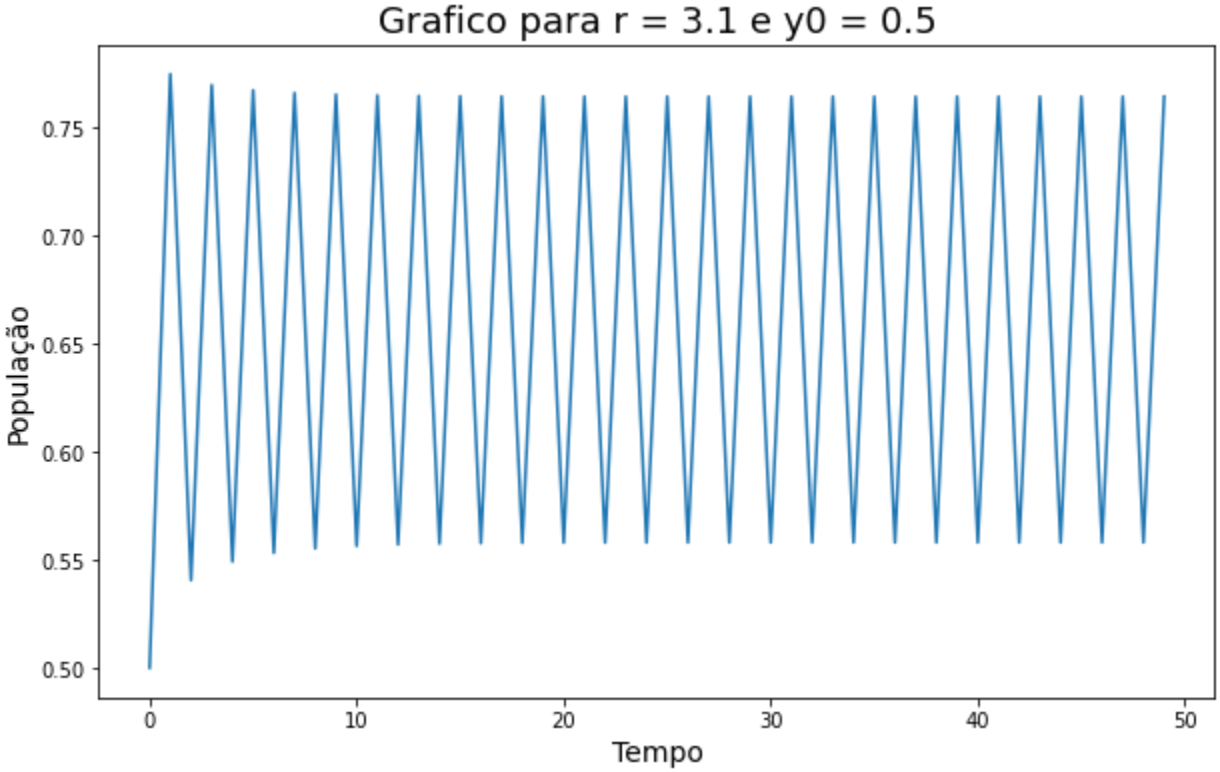


Grafico para r = 3.1 e y0 = 0.5
População
Tempo
0.75
0.70
0.65
0.60
0.55
0.50
0
10
20
30
40
50

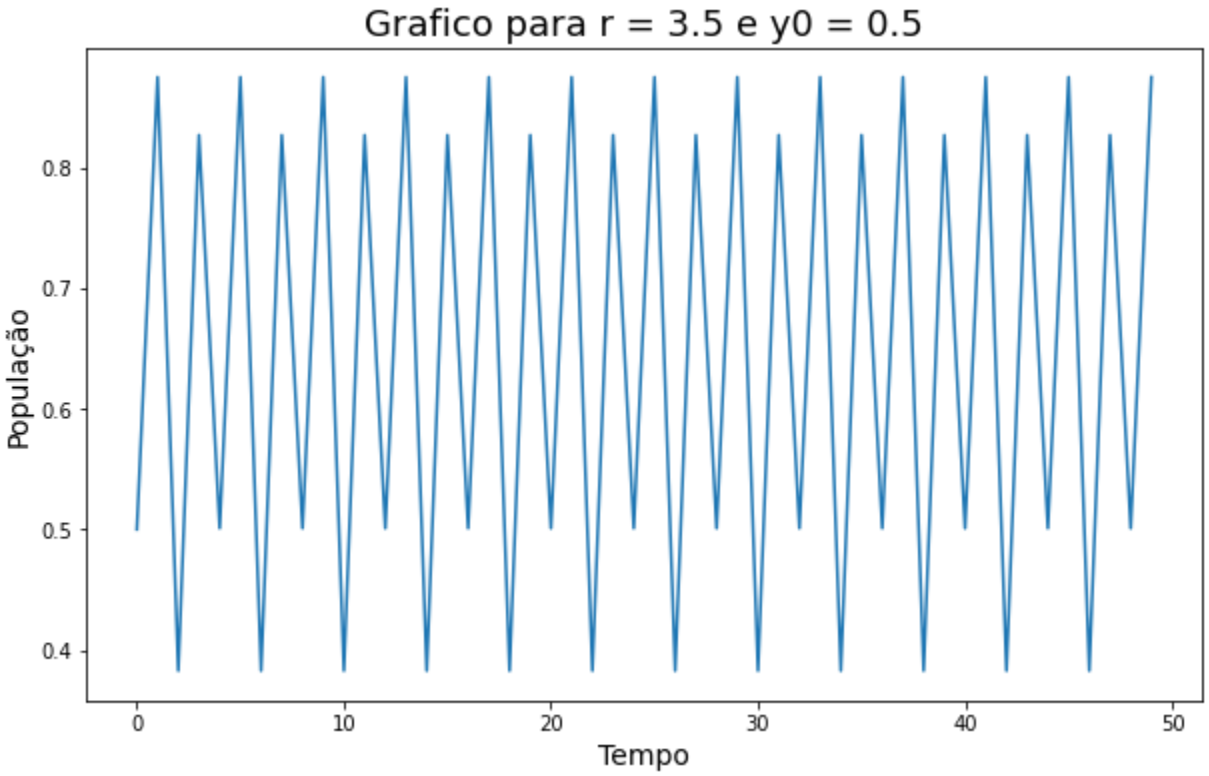
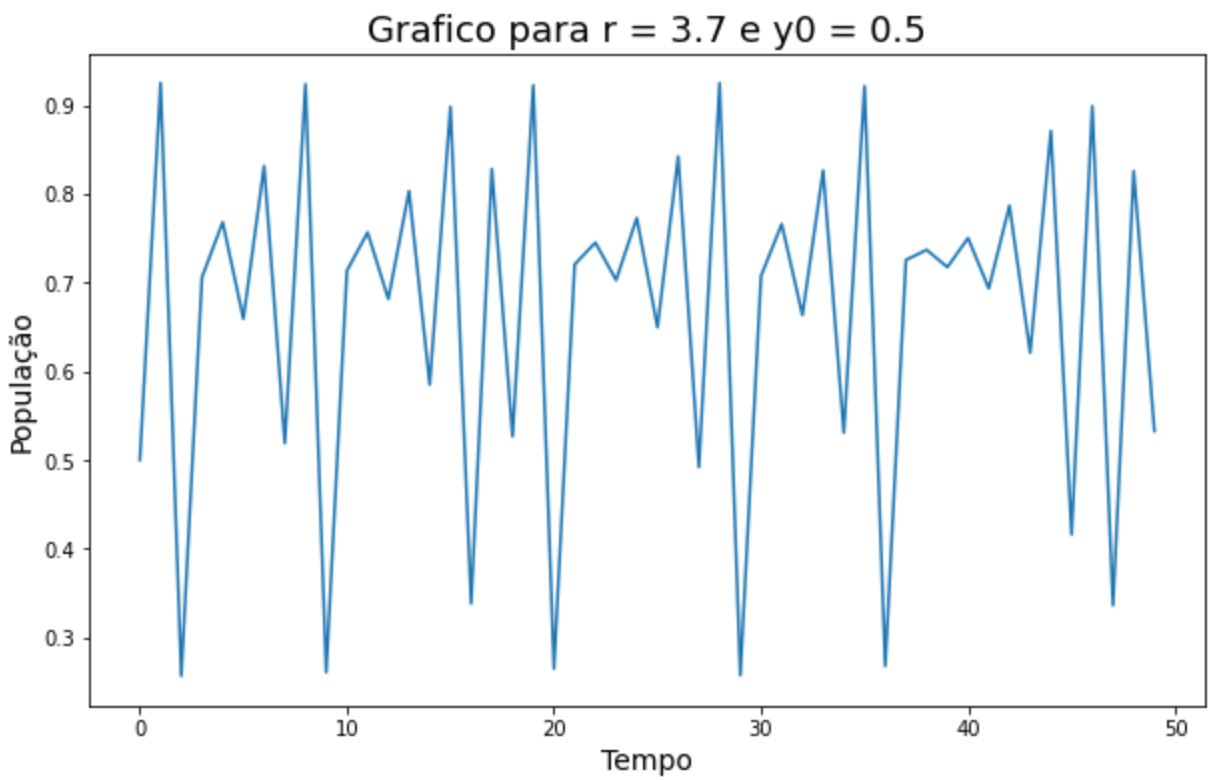


Grafico para r = 3.5 e y0 = 0.5
População
Tempo
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50



Percebe-se que no primeiro caso a população tende a 0 com o passar do tempo. Já no segundo, ela tende a um valor constante diferente de 0. Já nos casos restantes a população tem a tendência de oscilar de acordo com um determinado padrão de repetição.

Gerando resultados variando consideravelmente as condições iniciais

In [6]:
```
valores_r = [0.5, 2.5, 3.1, 3.5, 3.7]
valores_y0 = [0.25, 0.5, 0.75]
```

In [7]:
```
def gerarGraficoUnico(resultados, titulo):

    plt.figure(figsize=(12, 8))
    plt.xlabel("Tempo", fontsize=14)
    plt.ylabel("População", fontsize=14)
    plt.title(titulo, fontsize=18)

    colors=['orange', 'red', 'green']
    plt.gca().set_prop_cycle(color=colors)

    for i in range(len(resultados)):
        plt.plot(resultados[i], label="y0 = "+str(valores_y0[i]))

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

In [8]:
```
for r in valores_r:
    resultados = []
    for y in valores_y0:
        resul = gerarResultados(r, y)
        resultados.append(resul)

    gerarGraficoUnico(resultados, "Grafico para r = "+ str(r))
```

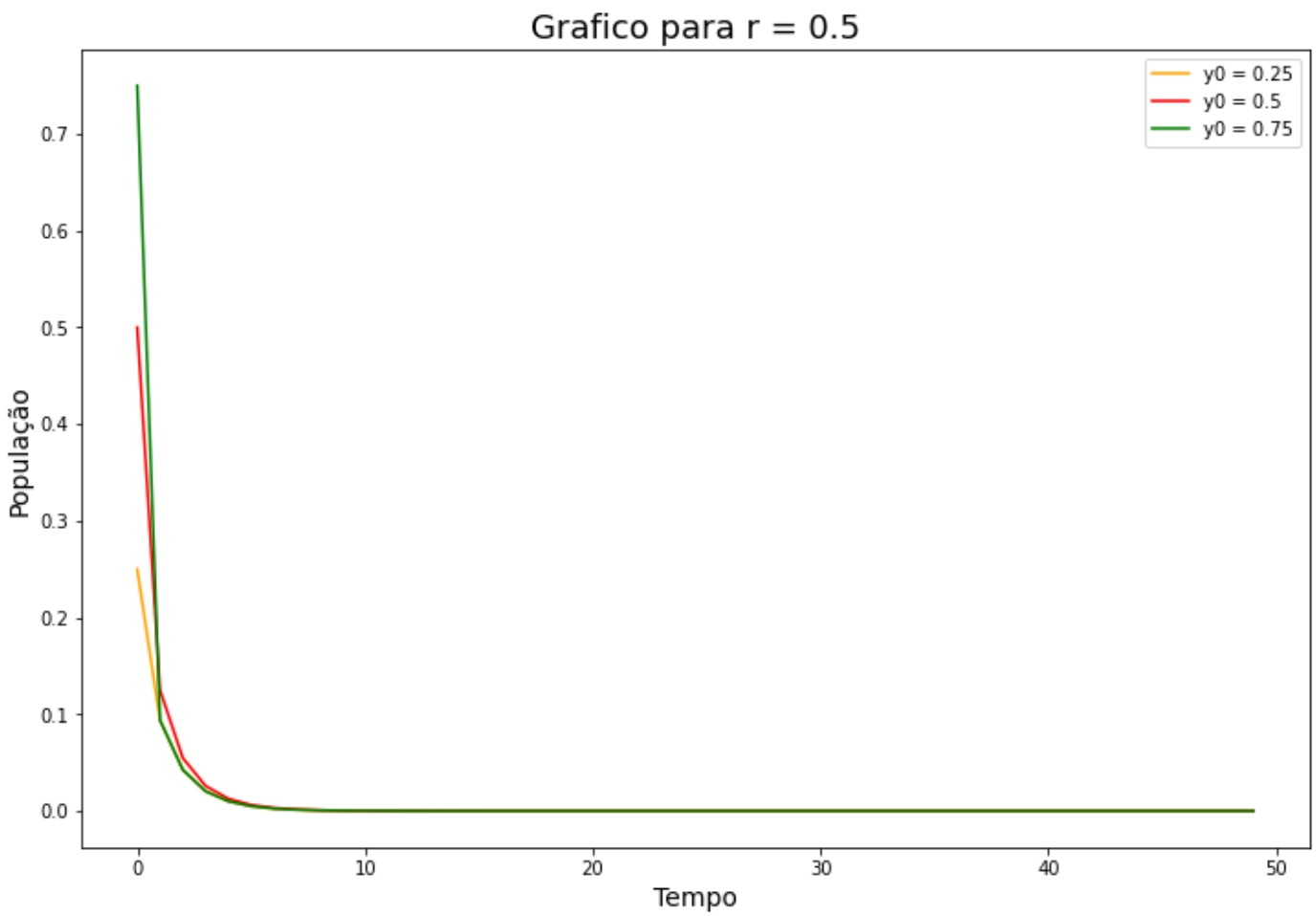


Grafico para r = 0.5
y0 = 0.25
y0 = 0.5
y0 = 0.75
População
Tempo
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
10
20
30
40
50

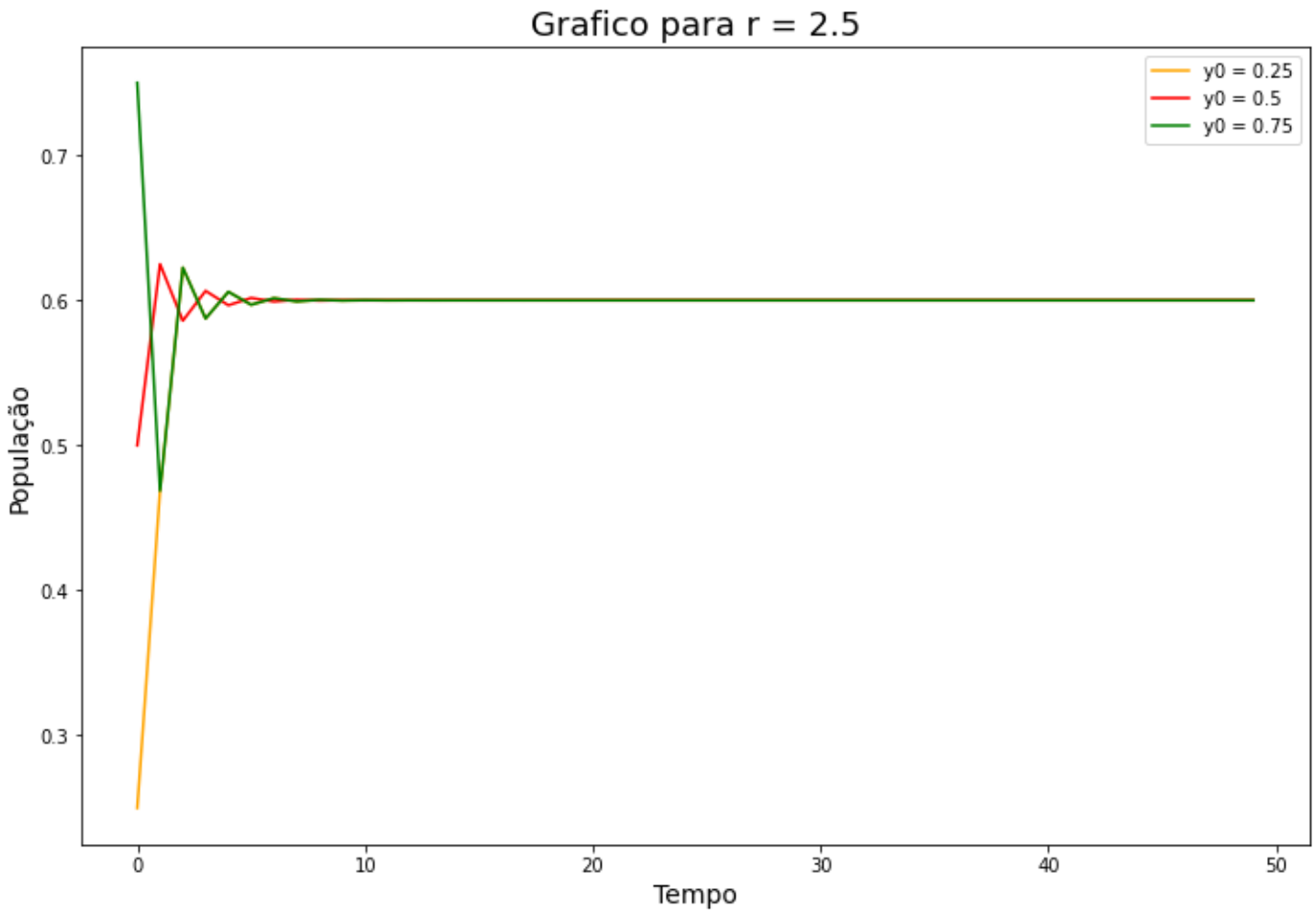


Grafico para r = 2.5
y0 = 0.25
y0 = 0.5
y0 = 0.75
População
Tempo
0.3
0.4
0.5
0.6
0.7
0
10
20
30
40
50

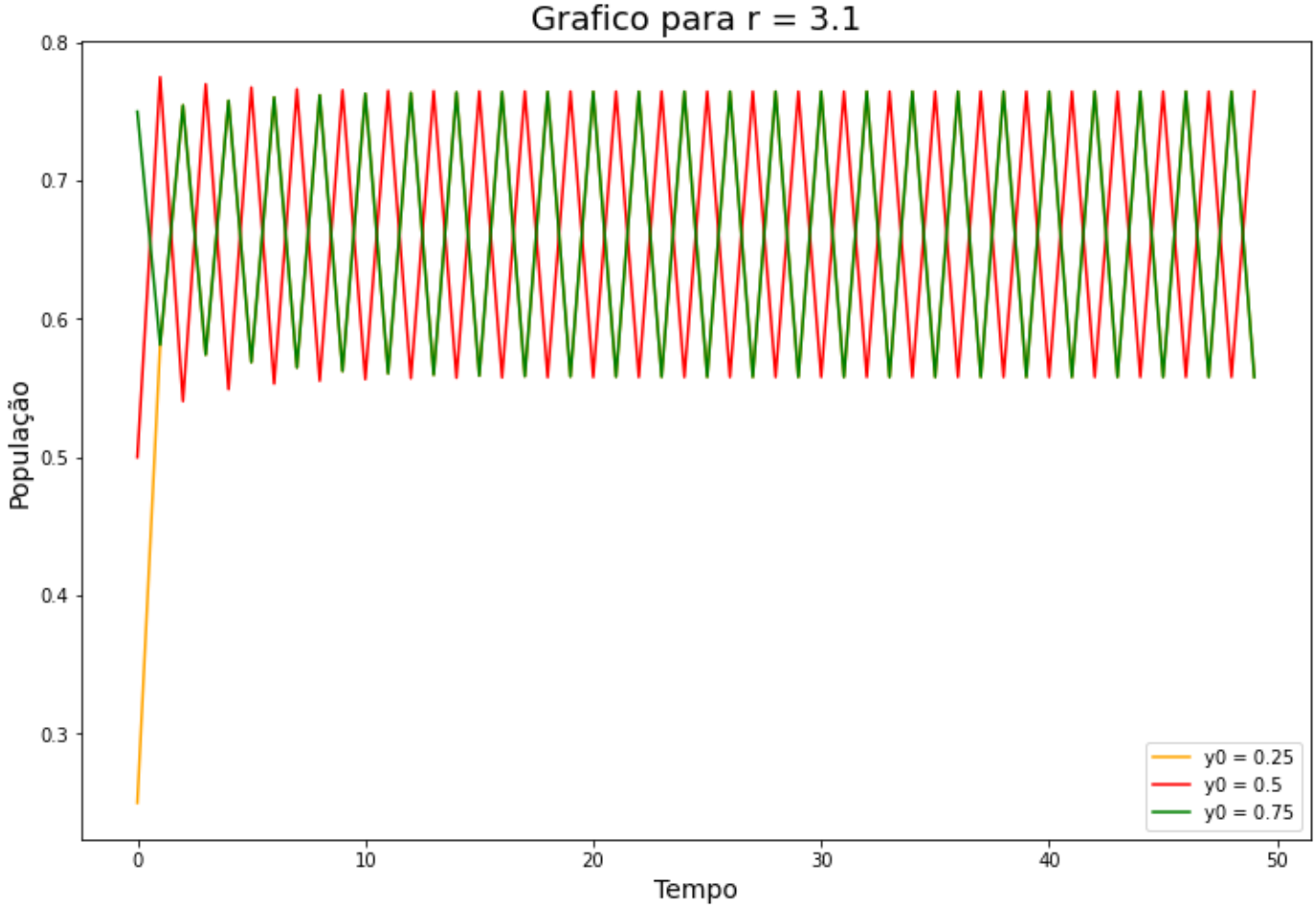


Grafico para r = 3.1
População
Tempo
y0 = 0.25
y0 = 0.5
y0 = 0.75

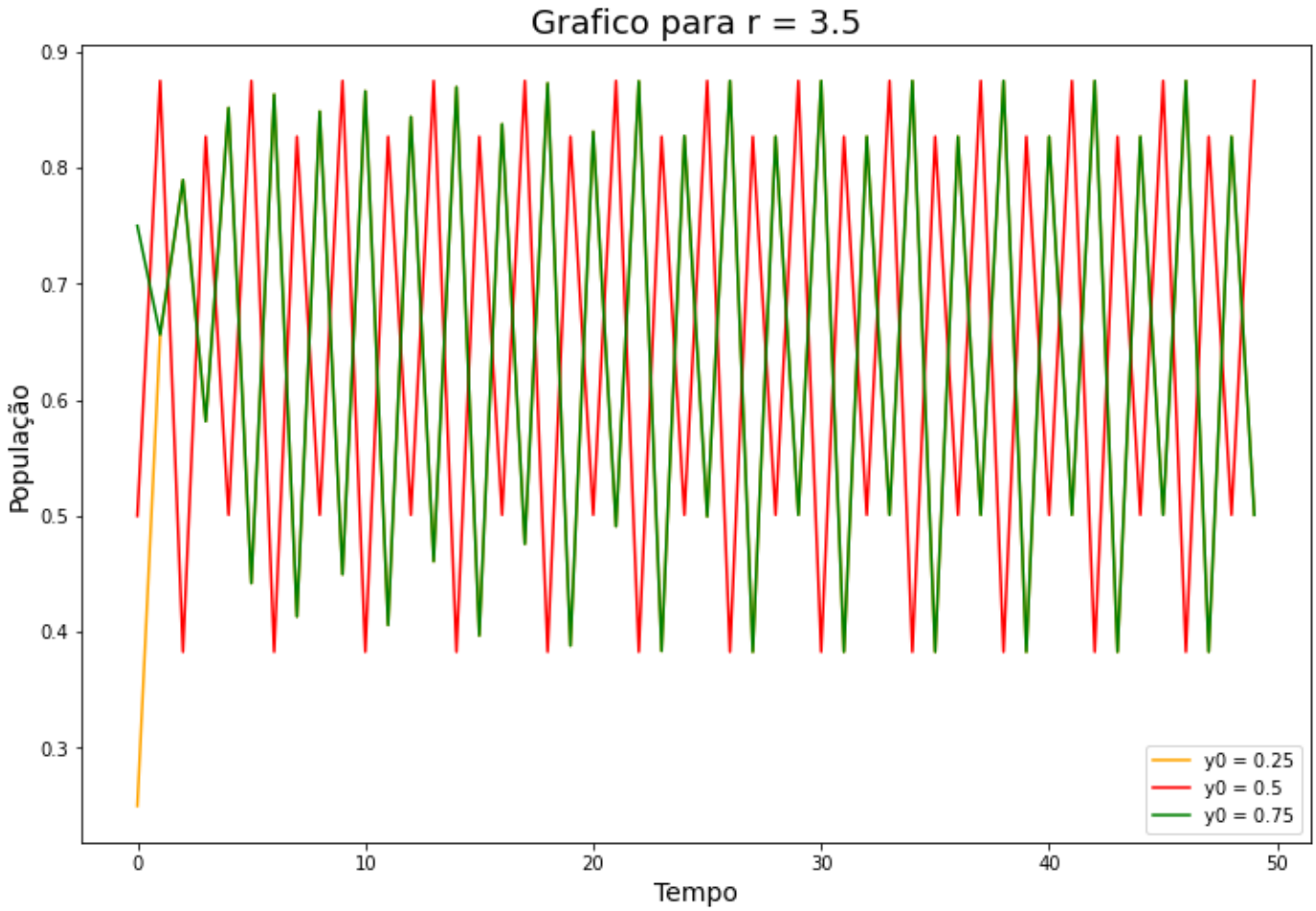
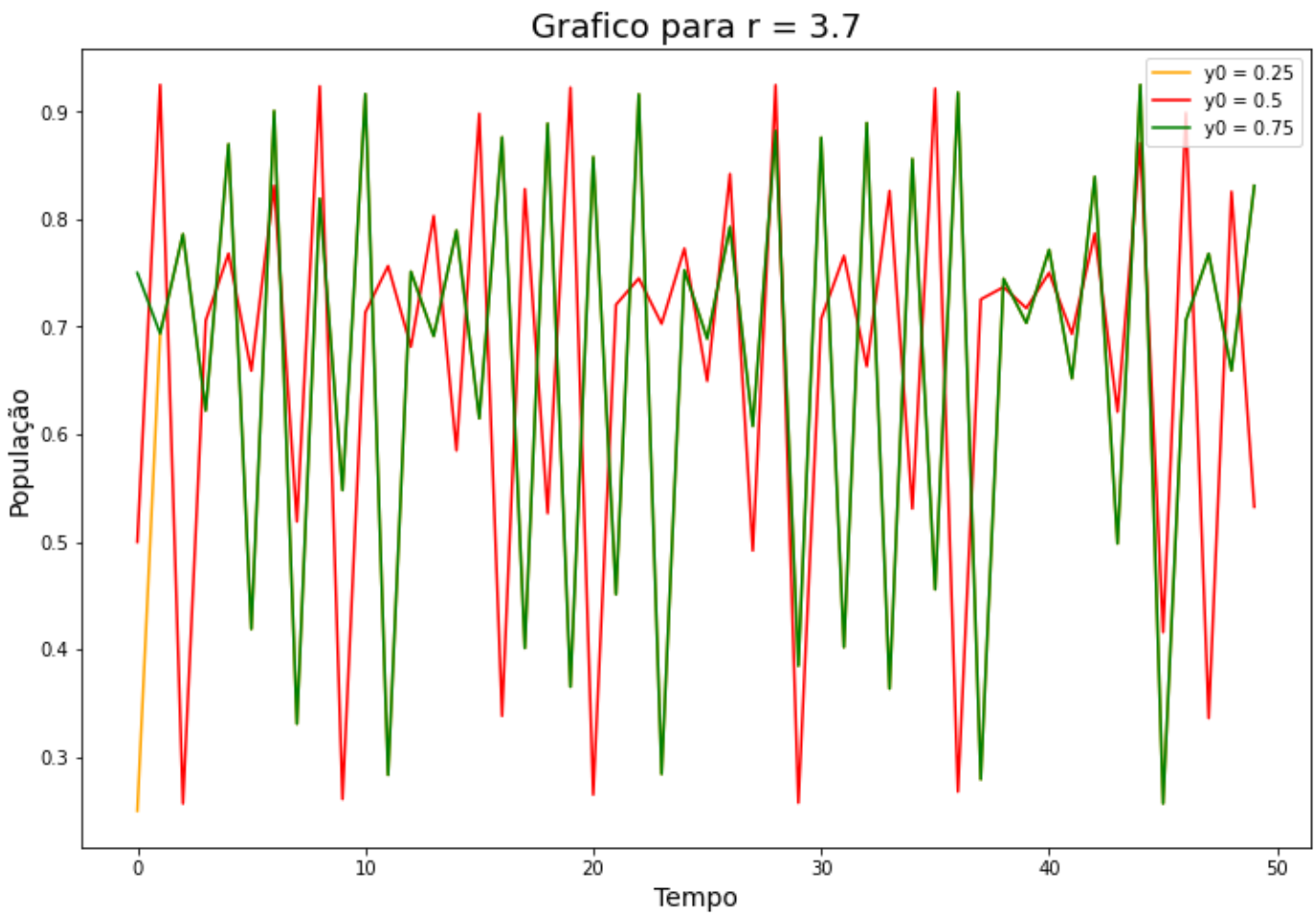


Grafico para r = 3.5
População
Tempo
y0 = 0.25
y0 = 0.5
y0 = 0.75



Com as condições iniciais consideravelmente diferentes, percebe-se que as populações de y0 igual a 0.25 e 0.75 produzem os mesmos resultados após a primeira iteração. Esses valores, comparados as gerados por y0 = 0.5, permitem concluir que as condições iniciais mudam significativamente os valores da população ao longo do tempo.

Gerando resultados variando ligeiramente as condições iniciais

In [9]:
```
valores_r = [0.5, 2.5, 3.1, 3.5, 3.7]
valores_y0 = [0.5, 0.501, 0.5001]
```

In [10]:
```
for r in valores_r:
    resultados = []
    for y in valores_y0:
        resul = gerarResultados(r, y)
        resultados.append(resul)

    gerarGraficoUnico(resultados, "Grafico para r = "+ str(r))
```

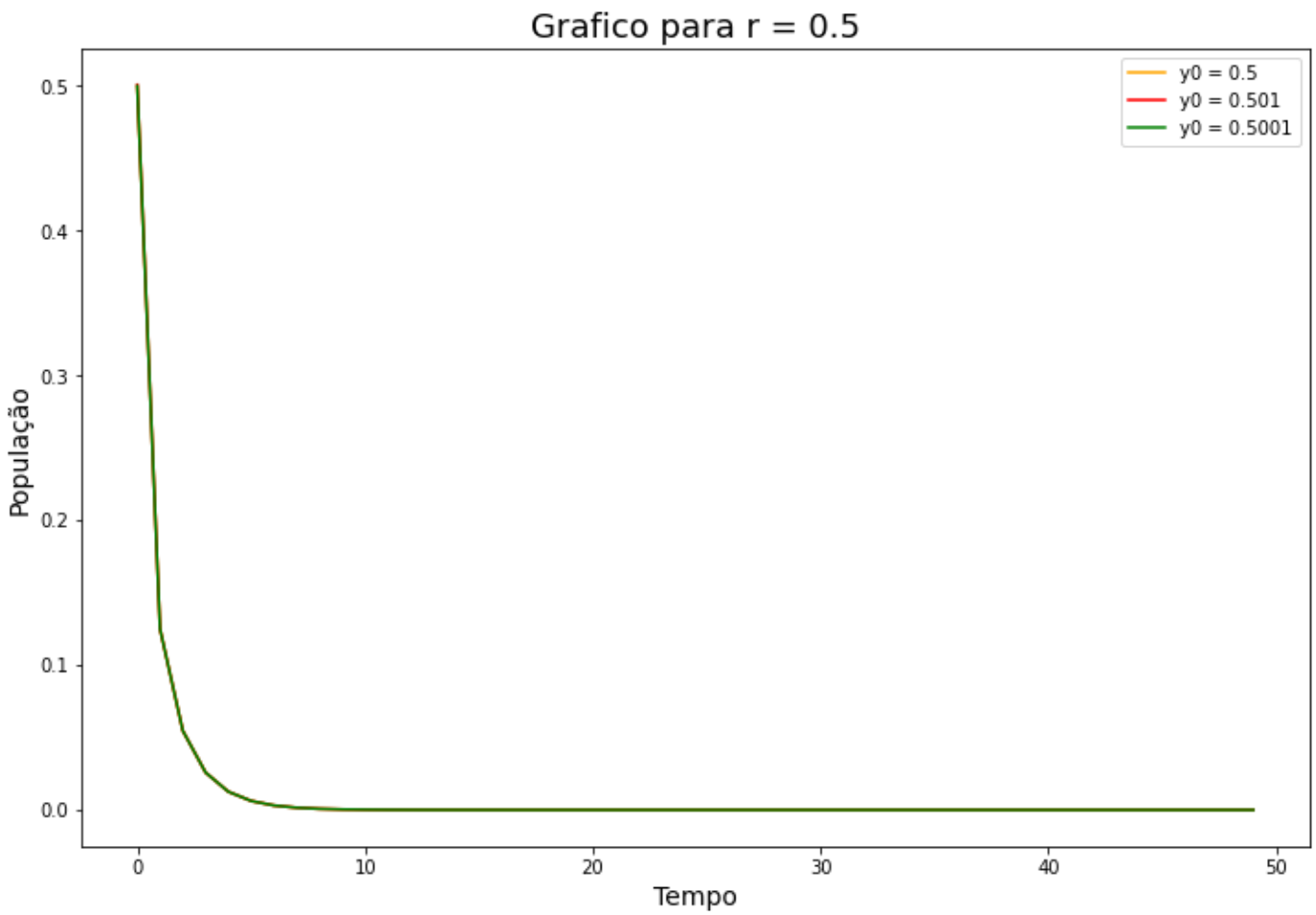


Grafico para r = 0.5
y0 = 0.5
y0 = 0.501
y0 = 0.5001
População
Tempo
0.0
0.1
0.2
0.3
0.4
0.5
0
10
20
30
40
50

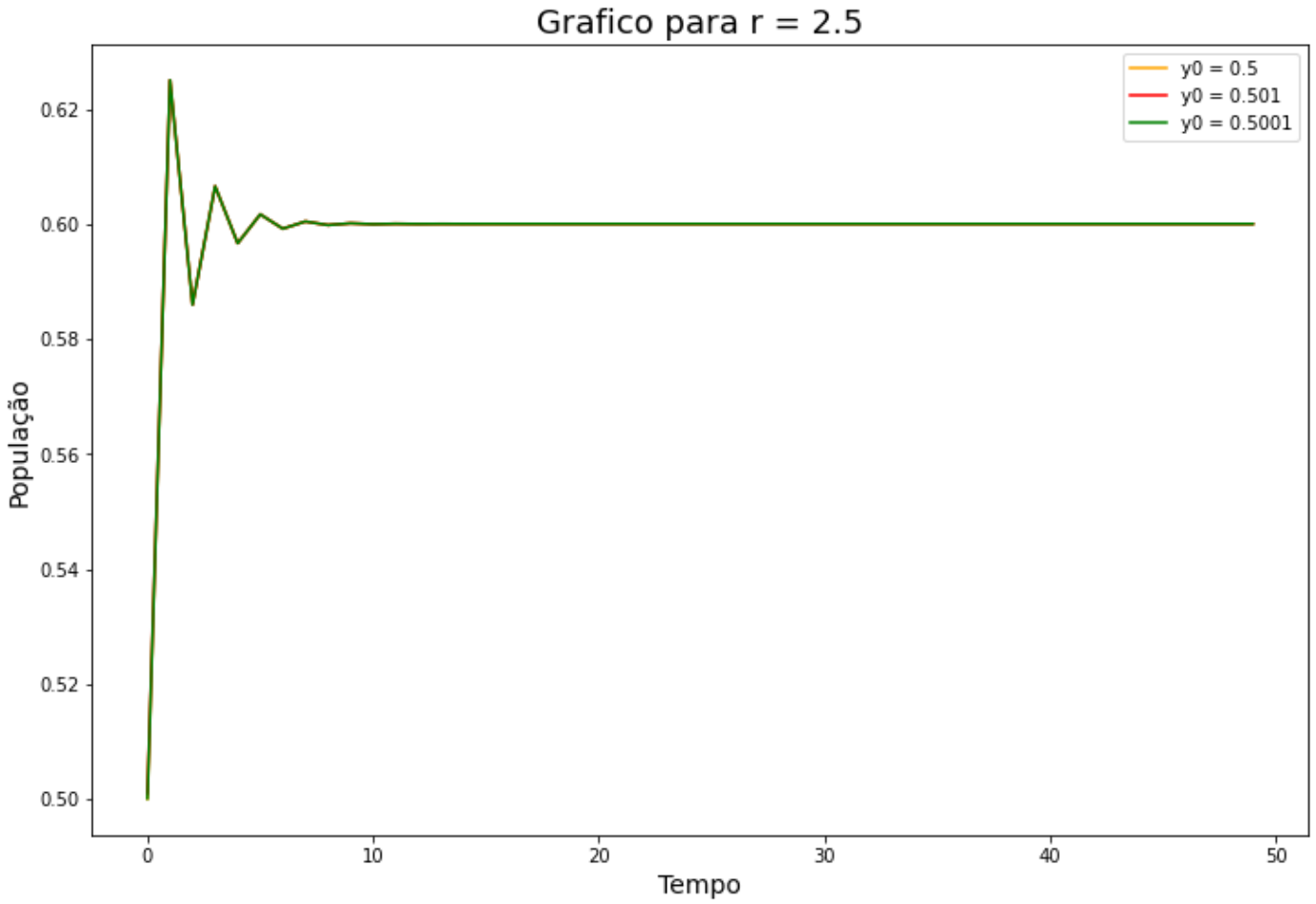


Grafico para r = 2.5
y0 = 0.5
y0 = 0.501
y0 = 0.5001
População
Tempo
0.50
0.52
0.54
0.56
0.58
0.60
0.62
0
10
20
30
40
50

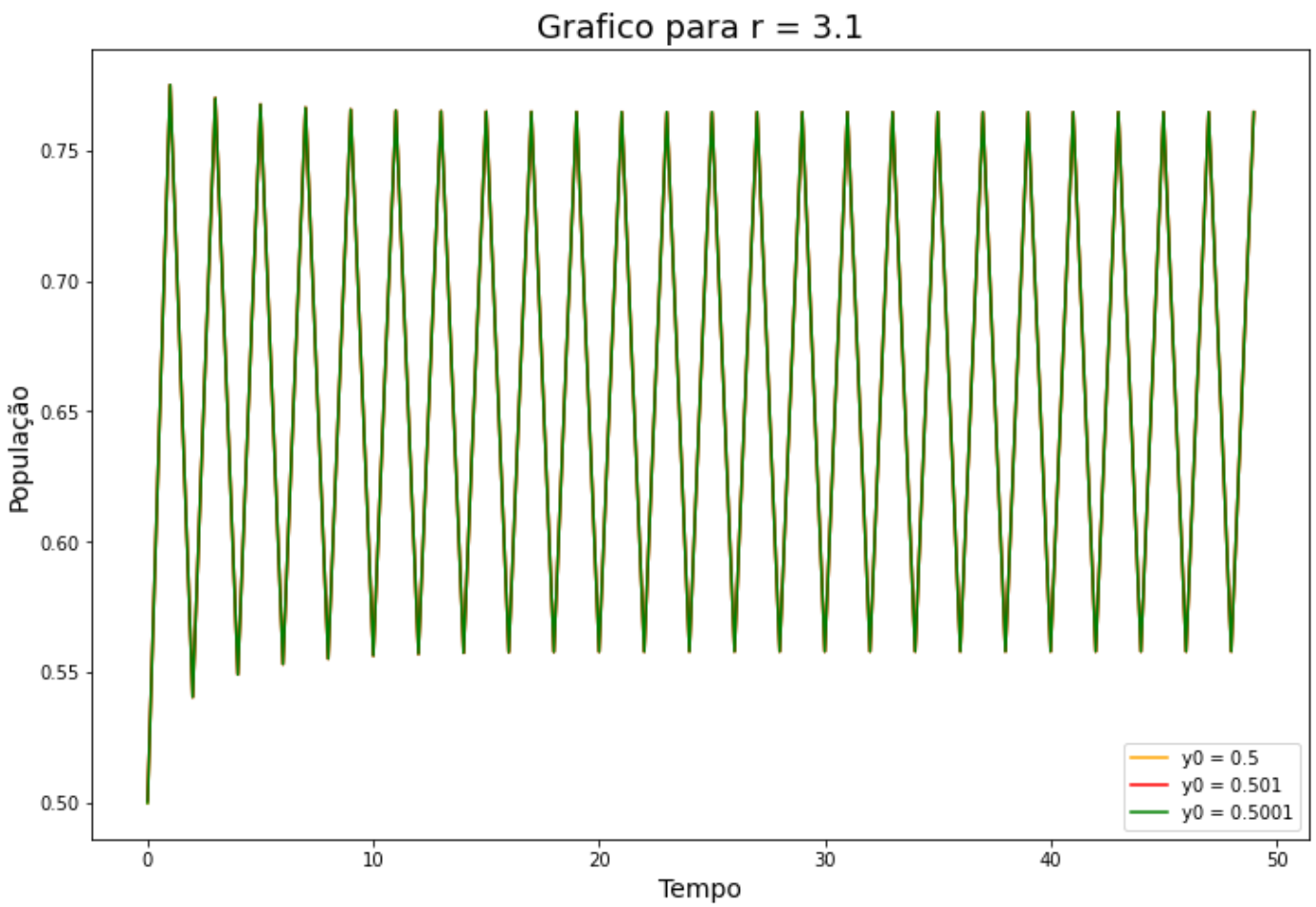


Grafico para r = 3.1
População
Tempo
y0 = 0.5
y0 = 0.501
y0 = 0.5001

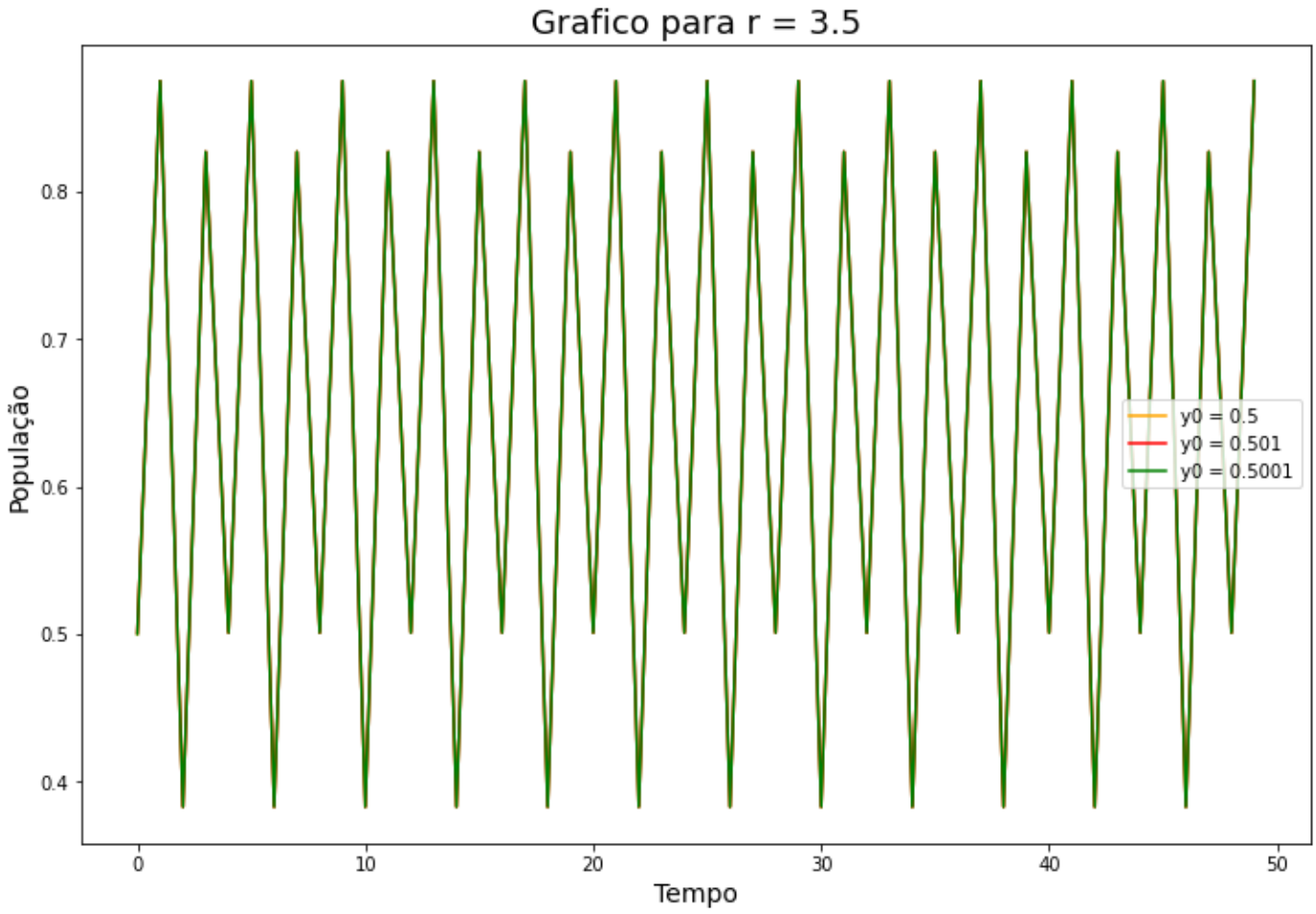
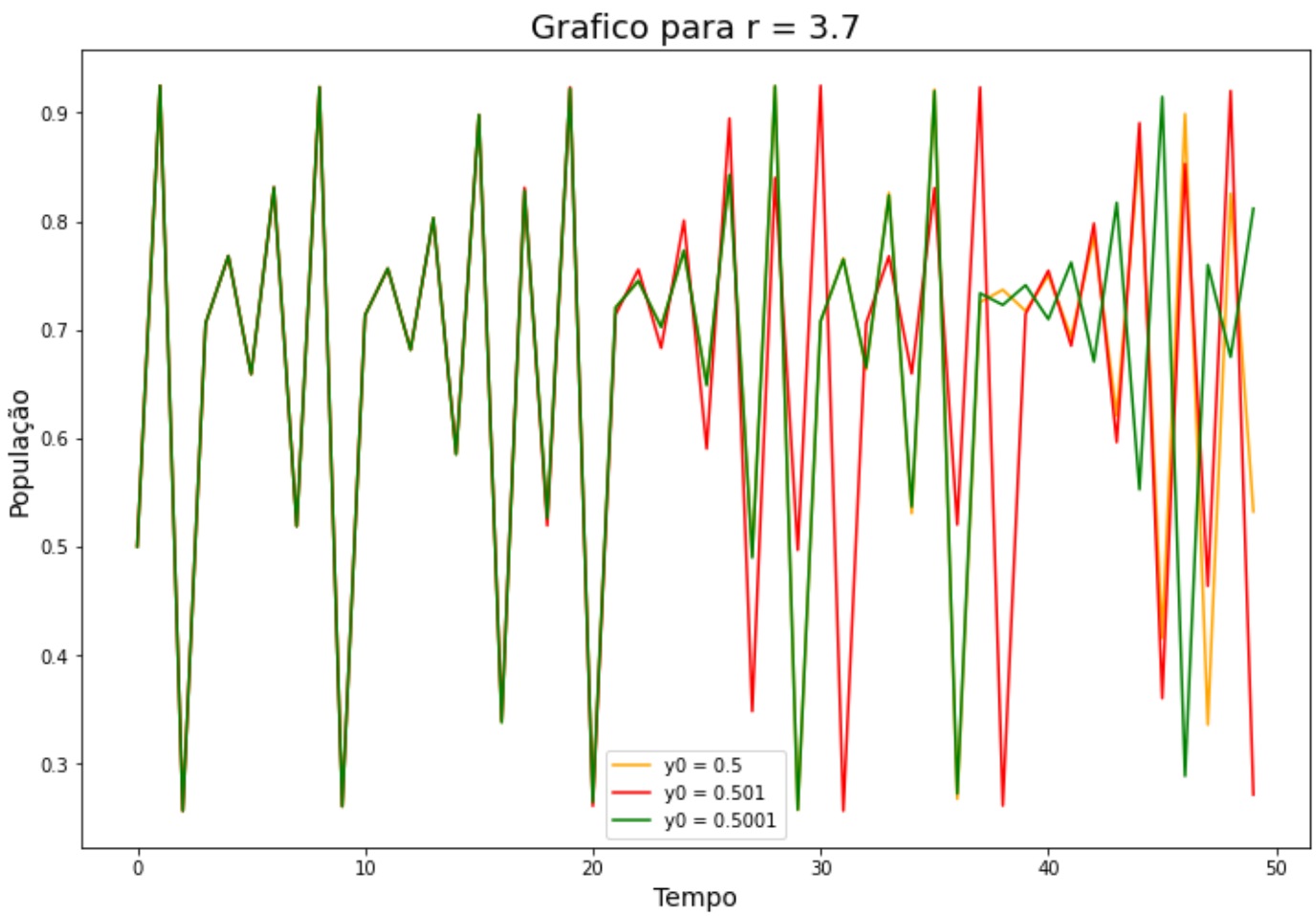


Grafico para r = 3.5
População
Tempo
y0 = 0.5
y0 = 0.501
y0 = 0.5001



Nesse caso em que as condições iniciais são bastante próximas, nota-se que a diferença nos valores da população só é perceptível para r = 3.7 e a partir da vigésima iteração.

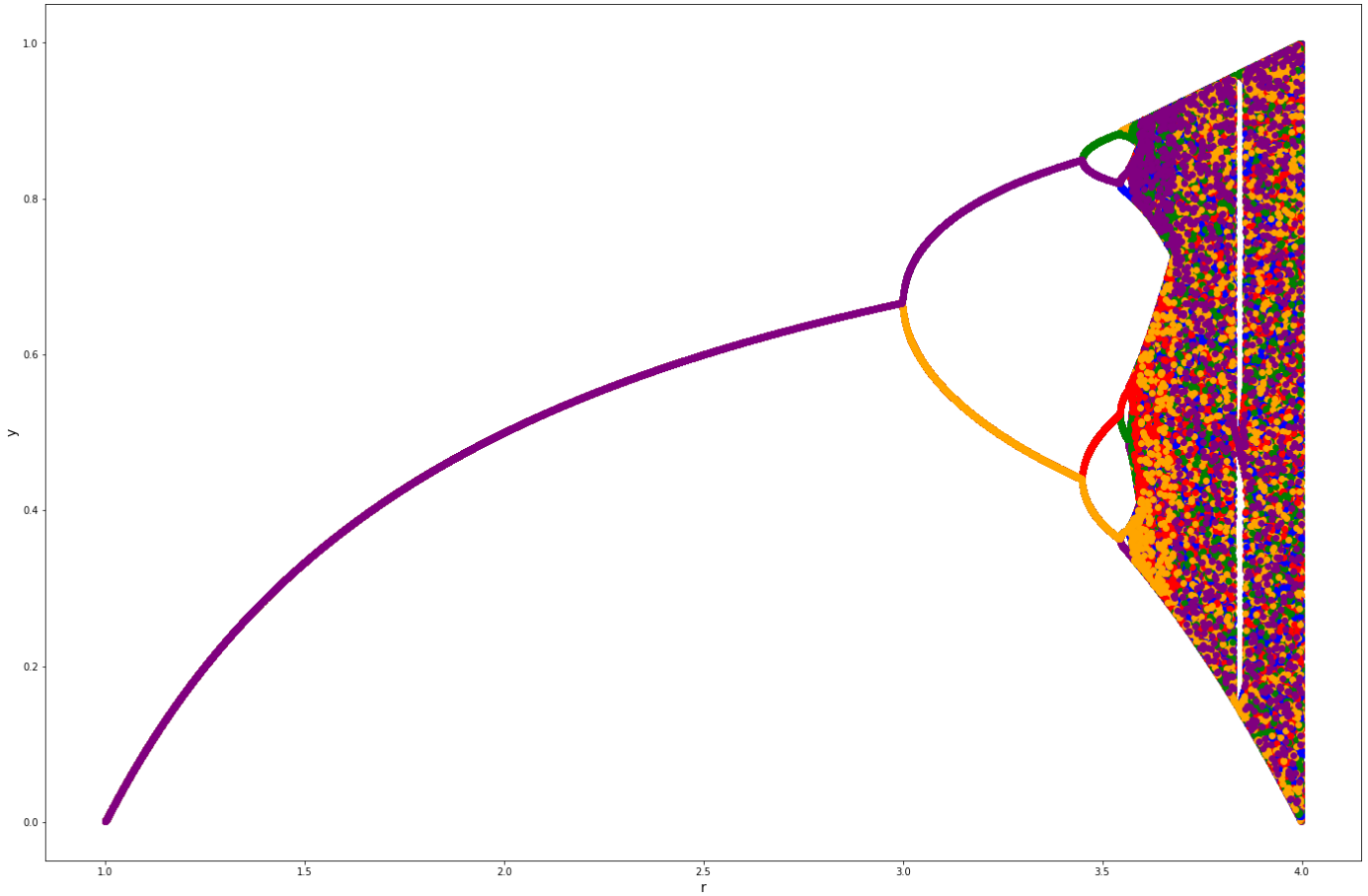
Segunda Parte

In [11]:
```
def gerarGraficoCompleto(valores_r, resultados, titulo):

    plt.figure(figsize=(24, 16))
    plt.xlabel("r", fontsize=14)
    plt.ylabel("y", fontsize=14)
    plt.title(titulo, fontsize=18)

    colors=['blue', 'red', 'green', 'orange', 'purple']

    valores = np.array(resultados).T

    for i in range(len(valores)):
        plt.scatter(valores_r,valores[i], c = colors[i%len(colors)])
    plt.show()
```

Para r entre 1 e 4:

In [12]:
```
r = 1
r_final = 4
passo = (r_final-r)/10000

resultados = []
valores_r = []
for i in range(10000):
    valores_r.append(r)
```

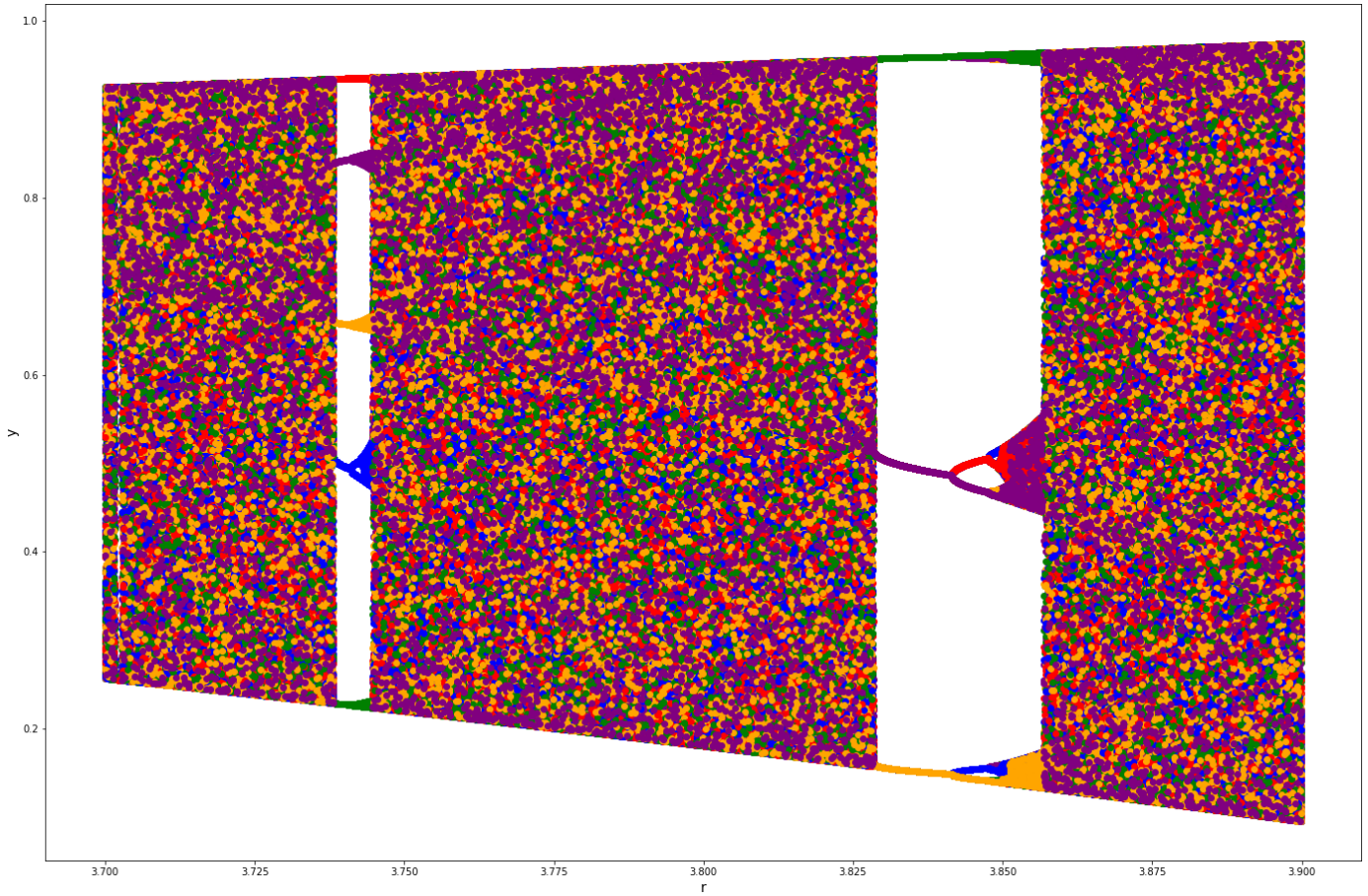
```
    resul = gerarResultados(r, 0.5, 1000)
    resultados.append(resul[900:])
    r = r + passo
gerarGraficoCompleto(valores_r,resultados, "")
```



Nota-se que ocorrem diversas bifurcações à medida que o valor de r aumenta. Essas bifurcações que ocorrem fazem com que o sistema possuam uma grande quantidade de valores quando r fica próximo de 4, o que indica o surgimento de um comportamento caótico.

Para r entre 3.7 e 3.9:

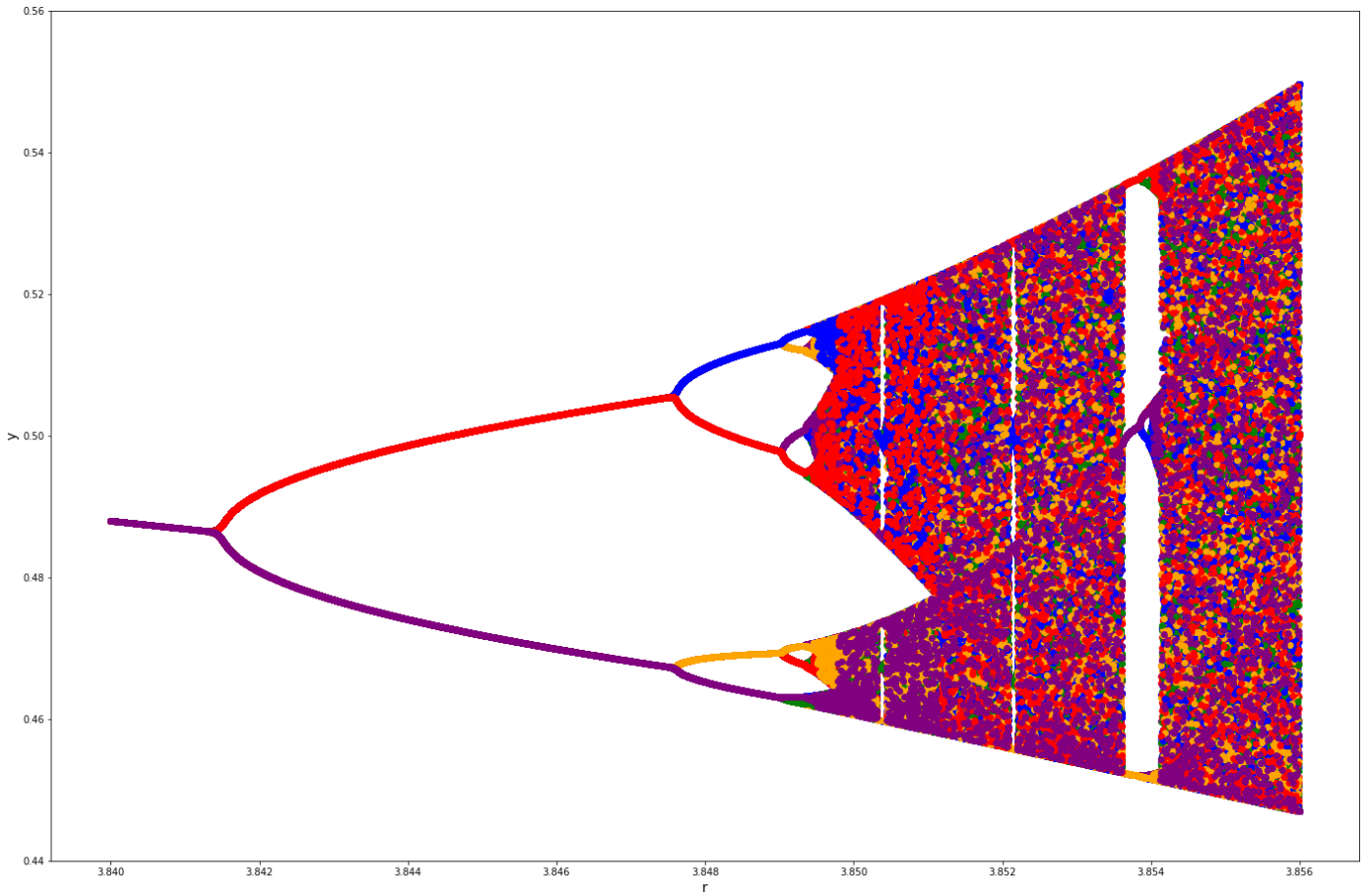
In [13]:
```
r = 3.7
r_final = 3.9
passo = (r_final-r)/10000

resultados = []
valores_r = []
for i in range(10000):
    valores_r.append(r)
    resul = gerarResultados(r, 0.5, 1000)
    resultados.append(resul[900:])
    r = r + passo
gerarGraficoCompleto(valores_r,resultados, "")
```



Para r entre 3.840 e 3.856:

In [17]:
```
def gerarGraficoRestrito(valores_r, resultados, titulo):

    plt.figure(figsize=(24, 16))
    plt.xlabel("r", fontsize=14)
    plt.ylabel("y", fontsize=14)
    plt.ylim(0.44, 0.56)
    plt.title(titulo, fontsize=18)

    colors=['blue', 'red', 'green', 'orange', 'purple']

    valores = np.array(resultados).T

    for i in range(len(valores)):
        plt.scatter(valores_r,valores[i], c = colors[i%len(colors)])
    plt.show()
```

In [18]:
```
r = 3.840
r_final = 3.856
passo = (r_final-r)/10000

resultados = []
valores_r = []
for i in range(10000):
    valores_r.append(r)
    resul = gerarResultados(r, 0.5, 1000)
    resultados.append(resul[900:])
    r = r + passo
gerarGraficoRestrito(valores_r,resultados, "")
```



Essas execuções para valores de r em um intervalo menor permitem que possamos observar que as mesmas bifurcações ocorrem muitas vezes ao longo da execução, o que não era muito perceptível no gráfico de r variando de 1 a 4.

In [ ]: