UNIVERSIDAD DE SONORA

DEPARTAMENTO DE FÍSICA

ACTIVIDAD 9

Alumno:
Luis Alfonso Torres Flores

Profesor
Carlos Lizárraga Celaya

19 de Mayo de 2017

Breve resumen

Al igual que en la actividad anterior que generamos un gif, también podemos crear archivos e videos mp4, como lo será en esta actividad.

Introducción

En este reporte mostraremos el código empleado. Dicho código tiene como función mostrar distintos pasos para la obtención de la animación que deseamos generar como un archivo mp4. El archivo, al igual que la actividad anterior, muestra el efecto mariposa. En el código iremos generando diversas imágenes que mostraremos más adelante en el orden en el que aparecerán al abrir y usar el archivo de Python correspondiente a este trabajo.

Para poder realizar el archivo mp4 se necesita la instalación de pynamical, cosa que necesitaremos si podemos observar los archivos que importamos al inicio de nuestro código. También se pudo encontrar algunos problemas a la hora de generar el archivo deseado debido a la falta de un código, mismo que se encuentra ahora en el trabajo. Si revisamos el código cuidadosamente podremos observar que podemos cambiar algunas características a placer, tal y como es el color de las gráficas y del video mismo, por lo que en cierto grado puede ser modificado a gusto.

Codigo

import pynamical from pynamical import simulate, save_fig, phase_diagram, phase_diagram_3d import pandas as pd, numpy as np, matplotlib.pyplot as plt, IPython.display as IPdisplay %matplotlib inline

title_font = pynamical.get_title_font() label_font = pynamical.get_label_font()

draw a phase diagram for 100 generations for the growth rate parameter 2.9 # it shows points converging on 0.655 because the logistic map has a fixed-point attractor at 0.655 when r=2.9 pops = simulate(num_gens=100, rate_min=2.9, num_rates=1, num_discard=100) phase_diagram(pops, title='Logistic Map Attractor, r=2.9', size=20)

draw a phase diagram for 100 generations for the growth rate parameter 3.5 # it shows 4 points because the logistic map has a period of 4 when r=3.5 pops = simulate(num_gens=100, rate_min=3.5, num_rates=1, num_discard=100) phase_diagram(pops, title='Logistic Map Attractor, r=3.5', size=20)

draw a phase diagram for 100 generations for the growth rate parameter 3.56 # it shows 8 points because the logistic map has a period of 8 when r=3.56 pops = simulate(num_gens=100, rate_min=3.56, num_rates=1, num_discard=100) phase_diagram(pops, title='Logistic Map Attractor, r=3.56', size=20)

draw a phase diagram for 100 generations for the growth rate parameter 3.57 # it shows n points because the logistic map has a period of n when r=3.57 pops =

simulate(num_gens=100, rate_min=3.57, num_rates=1, num_discard=100) phase_diagram(pops, title='Logistic Map Attractor, r=3.57', size=20)

draw a phase diagram for 2,000 generations for the growth rate parameter 3.9 # the plot reveals the strange attractor - the logistic map is chaotic when r=3.9 pops = simulate(num_gens=2000, rate_min=3.9, num_rates=1) phase_diagram(pops, xmin=0.25, xmax=0.75, ymin=0.8, ymax=1.01, size=20, title='Logistic Map Attractor, r=3.9')

draw a phase diagram for 2,000 generations across 50 growth rate steps from 3.6 to 4.0 # each chaotic growth rate has its own parabola pops = simulate(num_gens=2000, rate_min=3.6, rate_max=4.0, num_rates=50) phase_diagram(pops, xmin=0.25, xmax=0.75, ymin=0.8, ymax=1.01, size=7, title='Logistic Map Attractor, r=3.6 to r=4.0', color='viridis')

sometimes it is hard to tell if a time series is chaotic or random # generate two time series of 1,000 steps, one chaotic and one random # generate 30,000 time steps for the chaotic series but only keep the final 1,000 (when system is fully evolved) total_gens = 30000 gens = 1000 np.random.seed(1)

chaos_pops = simulate(num_gens=total_gens, rate_min=3.99, num_rates=1) chaos_pops = chaos_pops.iloc[total_gens-gens:].reset_index().drop(labels='index', axis=1)

random_pops = pd.DataFrame(np.random.random(gens), columns=['value']) time_series = pd.concat([chaos_pops, random_pops], axis=1) time_series.columns = ['chaos', 'random'] time_series.head()

plot the chaotic and random time series to show how they are sometimes tough to differentiate ax = time_series.plot(kind='line', figsize=[10, 6], linewidth=3, alpha=0.6, style=['#003399','#cc0000']) ax.grid(True) ax.set_xlim(40, 90) ax.set_ylim(0, 1) ax.set_title('Time Series, Deterministic Chaos vs Random Data', fontproperties=title_font) ax.set_xlabel('Generation', fontproperties=label_font) ax.set_ylabel('Population', fontproperties=label_font) ax.legend(loc=3)

save_fig('chaos-vs-random-line') plt.show()

plot same data as 2D phase diagram instead pops = pd.concat([chaos_pops, random_pops], axis=1) pops.columns = ['chaos', 'random'] phase_diagram(pops, size=20, color=['#003399','#cc0000'], ymax=1.005, legend=True, filename='logistic-attractor-chaos-random')

plot same data as 3D phase diagram instead phase_diagram_3d(pops, color=['#003399','#cc0000'], filename='logistic-attractor-chaos-random-3d', legend=True, legend_bbox_to_anchor=(0.94, 0.9))

plot same data as 3D phase diagram instead phase_diagram_3d(pops, color=['#003399','#cc0000'], filename='logistic-attractor-chaos-random-3d', legend=True, legend_bbox_to_anchor=(0.94, 0.9))

run logistic model for 4,000 generations across 50 growth rate steps from 3.6 to 4.0 pops = simulate(num_gens=4000, rate_min=3.6, rate_max=4.0, num_rates=50)

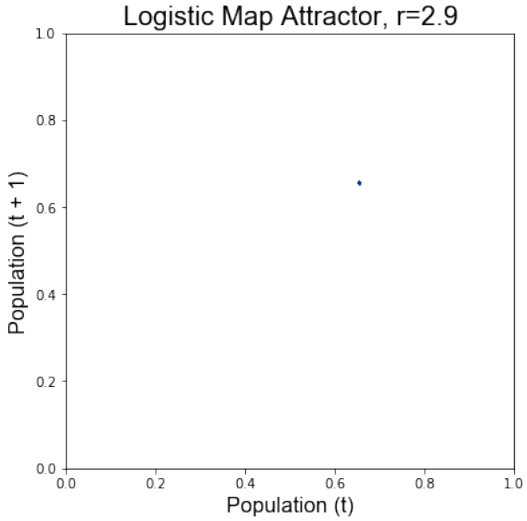
phase diagram: each chaotic growth rate has its own strange attractor curling through state space phase_diagram_3d(pops, title='Mapa logistico de atractor de r=3.6 a r=4.0', alpha=0.1, color='inferno', color_reverse=False, azim=230,

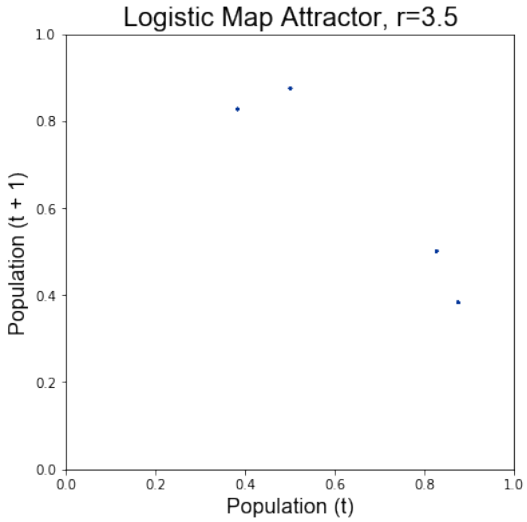
filename='3d-logistic-map-attractor-1', xlabel='Población (t)', ylabel='Población (t+1)', zlabel='Población (t+2)')

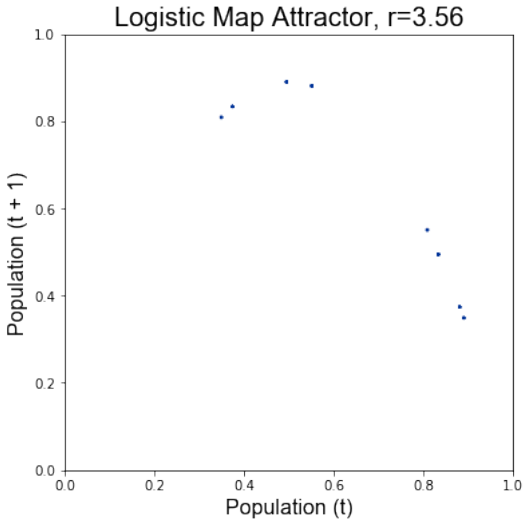
phase diagram: each chaotic growth rate has its own strange attractor curling through state space phase_diagram_3d(pops, title='Mapa logistico de atractor de r=3.6 a r=4.0', alpha=0.1, color='inferno', color_reverse=False, elev=7, azim=320, filename='3d-logistic-map-attractor-2', xlabel='Población (t)', ylabel='Población (t+1)', zlabel='Población (t+2)')

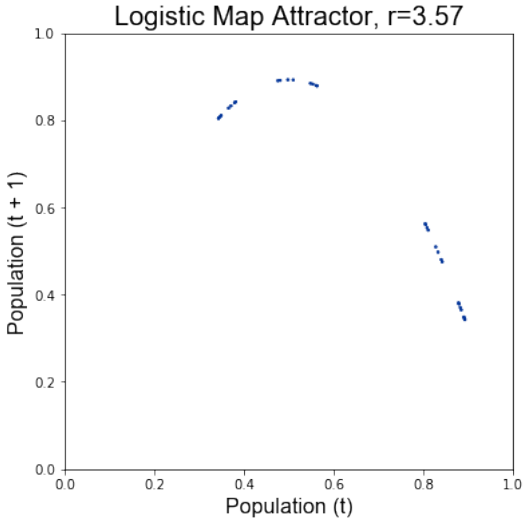
here's an example of the animated phase diagrams that I create in pynamical-demo-3d-animation.ipynb IPdisplay.Image(url='images/phase-animate/05-logistic-3d-phase-diagram-chaotic-regime.gif')

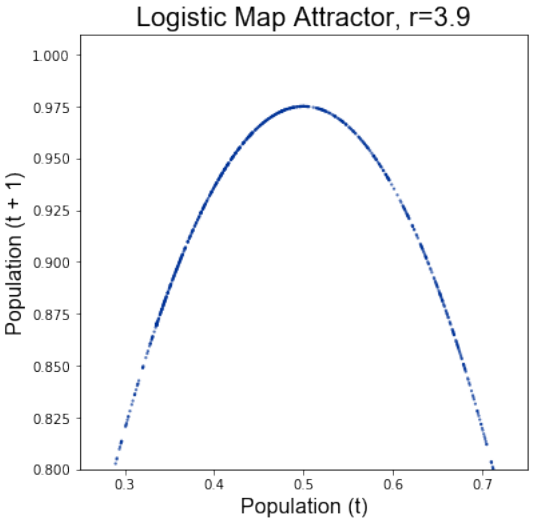
Imagenes obtenidas

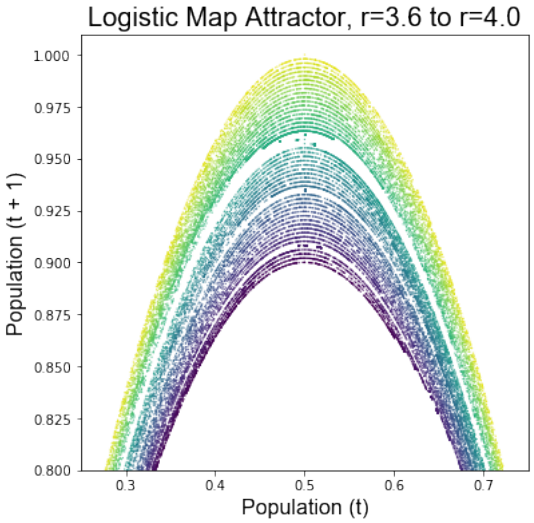


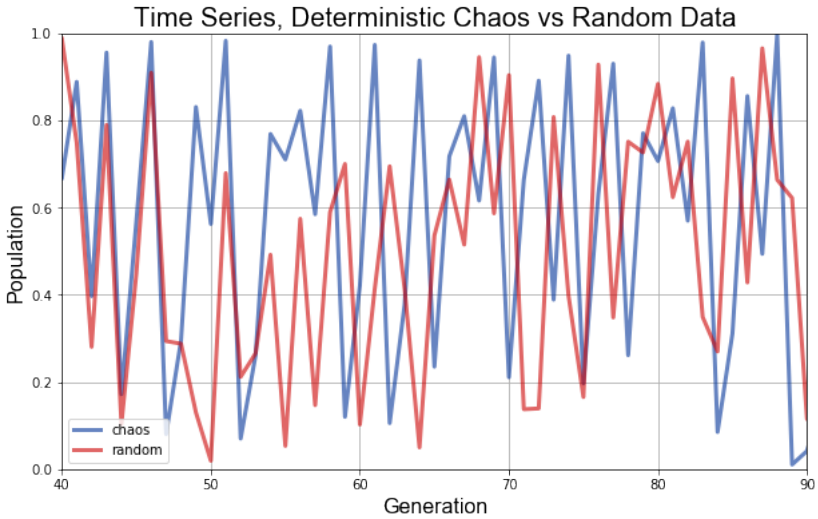


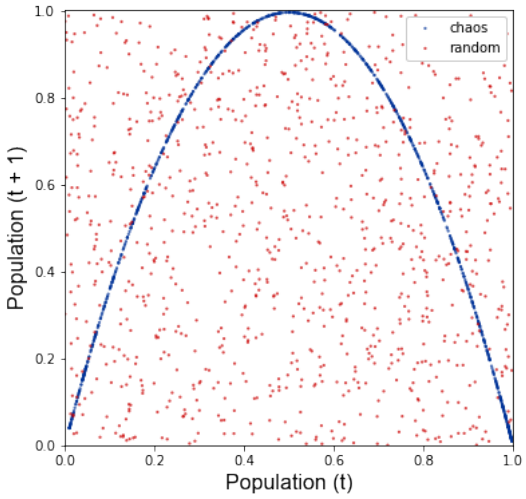

Logistic Map Attractor, r=3.56
Population (t + 1)
Population (t)
0.0
0.2
0.4
0.6
0.8
1.0

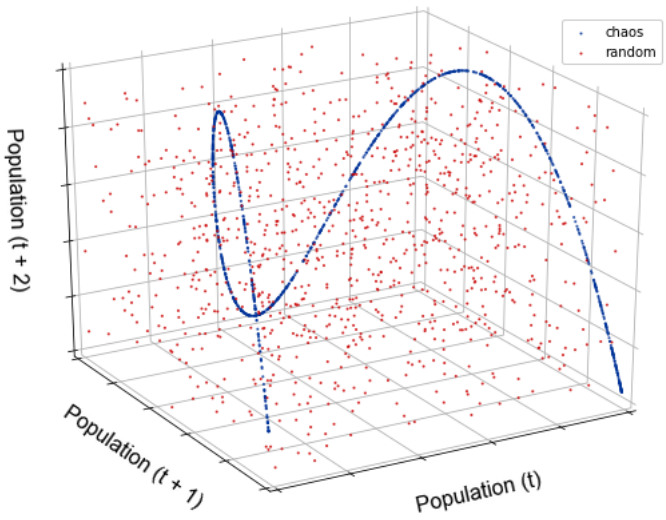

Logistic Map Attractor, r=3.57
Population (t + 1)
Population (t)
0.0
0.2
0.4
0.6
0.8
1.0


Logistic Map Attractor, r=3.9
Population (t + 1)
Population (t)
0.800
0.825
0.850
0.875
0.900
0.925
0.950
0.975
1.000
0.3
0.4
0.5
0.6
0.7

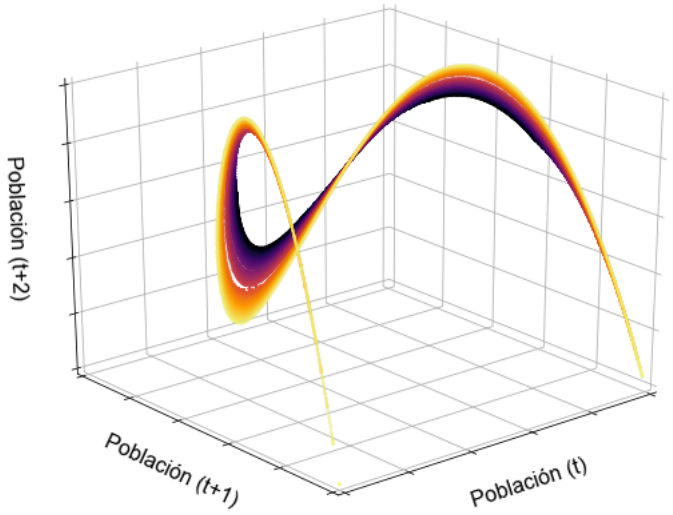

Logistic Map Attractor, r=3.6 to r=4.0
Population (t + 1)
Population (t)
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0.3
0.4
0.5
0.6
0.7


Time Series, Deterministic Chaos vs Random Data
Population
Generation
chaos
random
1.0
0.8
0.6
0.4
0.2
0.0
40
50
60
70
80
90


Population (t + 1)
Population (t)
chaos
random
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



Mapa logistico de atractor de r=3.6 a r=4.0



Mapa logistico de atractor de r=3.6 a r=4.0