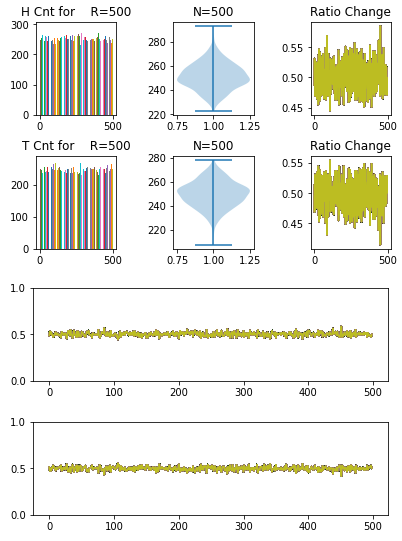
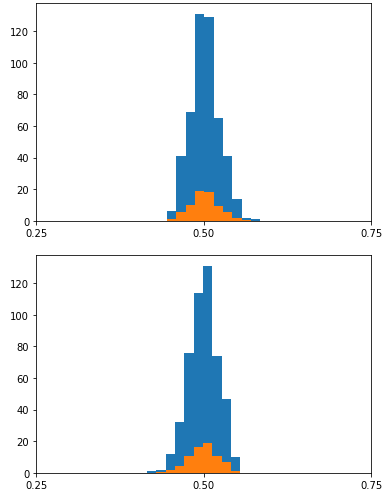
**This code shows visualization of the convergence of expected average for a fair coin flipping via repetitive H/T trials**





import numpy as np

from matplotlib import pyplot as plt

# The statespace

states = ["H"]

# Possible sequences of events

transitionName = [["HH","HT"]]

# Probabilities of transition matrix

transitionMatrix = [[0.5,0.5]]

print(sum(transitionMatrix[0]))

if sum(transitionMatrix[0]) == 1:

print("Move on! No Error in Transition matrix 1.")

else: print("Error Transition matrix")

# Function implementing the Markov model to forecast next state.

def state\_transition(steps):

# Choose the starting state

currentState = "H"

# print("Start state: " + currentState)

# Store the sequence of states taken.

stateList = [currentState]

i = 0

# To calculate the probability of the stateList

prob = 1

while i != steps:

if currentState == "H":

change = np.random.choice(transitionName[0],replace=True,p=transitionMatrix[0])

# print("change:", change)

if change == "HH":

prob = prob \* transitionMatrix[0][0]

stateList.append("H")

# print("HH:", stateList)

pass

elif change == "HT":

prob = prob \* transitionMatrix[0][1]

currentState = "H"

stateList.append("T")

# print("HT:", stateList)

pass

i += 1

# print("Possible states: " + str(stateList))

# print("End state after "+ str(days) + " days: " + currentState)

# print("Probability of the possible sequence of states: " + str(prob))

# print(stateList)

cnt\_H = stateList.count('H')

cnt\_T = stateList.count('T')

ratio\_H = cnt\_H /(cnt\_H + cnt\_T)

ratio\_T = cnt\_T /(cnt\_H + cnt\_T)

# print(HCnt, CCnt, ratio\_H, ratio\_C)

return stateList, cnt\_H, cnt\_T, ratio\_H, ratio\_T

# Function that iterates func\_name M times to build an appended list

def build\_list(Max, func\_name, func\_input):

HCnt\_List=[]

TCnt\_List=[]

ratioH\_List=[]

ratioT\_List=[]

for out in range(1,Max):

# print(out)

Collected=[]

Collected, HCnt, TCnt, ratioH, ratioT = func\_name(func\_input)

# print(Collected, ratioH, ratioT)

HCnt\_List.append(HCnt)

TCnt\_List.append(TCnt)

ratioH\_List.append(ratioH)

ratioT\_List.append(ratioT)

return HCnt\_List, TCnt\_List, ratioH\_List, ratioT\_List

# Run the function with the data

R=500 # Number of runs

N=500 # Each patter size

HCnt\_List, TCnt\_List, ratioH, ratioT = build\_list(R, state\_transition, N)

# Plot the outcome

fig, axs = plt.subplots(2, 3)

fig.tight\_layout(pad=2.0)

axs[0, 0].set\_title('H Cnt for R=%d' %(R))

axs[0, 1].set\_title('N=%d' %(N))

axs[0, 2].set\_title('Ratio Change')

axs[1, 0].set\_title('T Cnt for R=%d' %(R))

axs[1, 1].set\_title('N=%d' %(N))

axs[1, 2].set\_title('Ratio Change')

for k in range(0,R-1):

axs[0, 0].bar(k,HCnt\_List[k])

axs[0, 1].violinplot(HCnt\_List)

for k in range(0,R-1):

axs[0, 2].plot(ratioH)

for k in range(0,R-1):

axs[1, 0].bar(k,TCnt\_List[k])

axs[1, 1].violinplot(TCnt\_List)

for k in range(0,R-1):

axs[1, 2].plot(ratioT)

fig, axs = plt.subplots(2, 1)

fig.tight\_layout(pad=2.0)

for k in range(0,R-1):

axs[0].plot(ratioH)

axs[0].set\_yticks([0.0, 0.5, 1.0])

for k in range(0,R-1):

axs[1].plot(ratioT)

axs[1].set\_yticks([0.0, 0.5, 1.0])

plt.show()

plt.figure(0)

plt.hist(ratioH)

plt.hist(ratioH, density=True)

plt.xticks([0.25, 0.5, 0.75])

plt.figure(1)

plt.hist(ratioT)

plt.hist(ratioT, density=True)

plt.xticks([0.25, 0.5, 0.75])