Santiago Quiroz Upegui cc: 1036678350. Grupo 1

SIMULACIÓN DE SISTEMAS - 3007331

import random

import math

import matplotlib.pyplot as plt

import numpy as np

from scipy.stats import t

from scipy.stats import kstest

PARCIAL 1

**definimos funciones para generar el tiempo para completar la duatlo que se distribuye a + c**

para a el valor maximo de su fdp es en x = 48 f(48) = M  
osea M = f(48)

el dominio de la distribucion es 48,60

el valor maximo de la fdp en una triangualar es 2/(b-a) en este caso M = 2/(60-48) = 0.16666666666

**segun el metodo de aceptacion - rechazo:**

g(x) = (f(x)/M) = (f(x)/26 , con g(x) < 1

**si genero 2 aleatorios (r1 , r2) y se cumple lo siguiente:**

sea x = a + (b - a)r1 = 2 + (3 - 1) r1

si r2 <= g(x)

**x es una variable aleatoria valida para usar en nuestra simulacion**

**asi que solo hay que repetir la generacion de aleatorios con hasta que se cumpla la condicion**

def fdpTriangular (a,c,b,x):

    if a < x <=c:

        fdp = ((x-a)\*\*2)/((b-a)\*(c-a))

        return fdp

    elif c < x <=b :

        fdp = 1 - ((b-x)\*\*2)/((b-a)\*(b-c))

        return fdp

    else :

        return 0

def genA ():

    a = 48

    b = 60

    c = 48

    M = 2/(60-48)

    while True :

        r1 = random.uniform(0, 1)

        r2 = random.uniform(0, 1)

        x = a + (b - a) \* r1

        g = (fdpTriangular(a,c,b,x)/M)

        if r2 <= g:

            return x

def genC ():

    x = random.uniform(25,55)

    return x

def genTiempoDuatlon ():

    return genA() + genC()

## definimos funciones para calcular estadisticos necesario y para la creacion de un intervalo de confianza

#funcion que calcula los estimadores necesarios para el intervalo de confianza de la media y retorna una tupla (media, desviacion)

def calcularEstimadores(datos):

    media = np.mean(datos)

    desviacionEstandar = np.std(datos)

    return (media, desviacionEstandar)

def calcularIntervaloConfianzaMedia(media, desviacionEstandar, numeroSimulaciones, alfa):

    #calculando el intervalo de confianza

    t\_tabla= t.ppf(1-(alfa/2),(numeroSimulaciones-1))

    print("valor t con grados de libertad α/2 y n-1 ", t\_tabla)

    derecha =media+(t\_tabla)\*(desviacionEstandar/(numeroSimulaciones\*\*1/2))

    izquierda = media-(t\_tabla)\*(desviacionEstandar/(numeroSimulaciones\*\*1/2))

    return (izquierda,derecha)

## simulamos 12 corredores

def simular(n):

    datosSimulacion = []

    for i in range(0,n):

        datosSimulacion.append(genTiempoDuatlon ())

    return datosSimulacion

n = 12

datosSimulacion=simular(n)

tiempoGanador = np.amin(datosSimulacion)

print ("el tiempo del ganador fue ", tiempoGanador)

tiempoPerdedor = np.amax(datosSimulacion)

print("el tiempo del perdedor fue ", tiempoPerdedor)

a) ¿Cuál es el tiempo del ganador? ¿Y cuál es el tiempo del perdedor? (Puntos: 40)

el tiempo del ganador fue 74.67532597662083

b) Calcule un intervalo de confianza para la media del tiempo del duatlón. Considere un alfa de 0,05. (Puntos: 10)

el tiempo del perdedor fue 113.0408306354223

**generamos un intervalo de confianza del 95% para la media**

alfa = 0.05

media, desviacionEstandar = calcularEstimadores(datosSimulacion)

intervalo = calcularIntervaloConfianzaMedia(media, desviacionEstandar, n, alfa)

print("Un intervalo con un 95 por ciento de confianza para la media poblacional del tiempo promedio para completar la duatlon es: ")

print(intervalo)

valor t con grados de libertad α/2 y n-1 2.200985160082949 Un intervalo con un 95 por ciento de confianza para la media poblacional del tiempo promedio para completar la duatlon es: (94.16354611305209, 103.88783537346424)

## AHORA CON LOS DATOS QUE NOS DIERON DE LOS ALEATORIOS

def genTiempoDuatlonQuemados (atletismo,ciclo):

    return [x + y for x, y in zip(atletismo, ciclo)]

atletismo = [0.08,0.59,0.82,0.09,0.78,0.64,0.64,0.80,0.07,0.07,0.14,0.80]

ciclo = [0.22,0.44,0.54,0.82,0.77,0.88,0.65,0.50,0.51,0.46,0.72,0.03]

datosSimulacion = genTiempoDuatlonQuemados(atletismo,ciclo)

tiempoGanador = np.amin(datosSimulacion)

print ("el tiempo del ganador fue ", tiempoGanador)

tiempoPerdedor = np.amax(datosSimulacion)

print("el tiempo del perdedor fue ", tiempoPerdedor)

a) ¿Cuál es el tiempo del ganador? ¿Y cuál es el tiempo del perdedor? (Puntos: 40)

el tiempo del ganador fue 0.3

el tiempo del perdedor fue 1.55

b) Calcule un intervalo de confianza para la media del tiempo del duatlón. Considere un alfa de 0,05. (Puntos: 10)

## generamos un intervalo de confianza del 95% para la media

alfa = 0.05

media, desviacionEstandar = calcularEstimadores(datosSimulacion)

intervalo = calcularIntervaloConfianzaMedia(media, desviacionEstandar, n, alfa)

print("Un intervalo con un 95 por ciento de confianza para la media poblacional del tiempo promedio para completar la duatlon es: ")

print(intervalo)

valor t con grados de libertad α/2 y n-1 2.200985160082949 Un intervalo con un 95 por ciento de confianza para la media poblacional del tiempo promedio para completar la duatlon es: (0.8620967393915606, 1.1479032606084392)

PUNTO 2

print (kstest(aleatorios),'norm')

no me dio tiempo de interpretarlo

def genAleatorios(x, a, c,m, n):

    numeros = []

    random.seed(9)

    numeros.append((a\*x + c)%m)

    for i in range (0, n-1):

        numeros.append(((a\*(numeros[-1])) + c)%m)

    return numeros

a = 17

m = 61

x = 9

c =12

n = 20

aleatorios = genAleatorios(x, a, c, m, n)

print(aleatorios)

[43, 11, 16, 40, 21, 3, 2, 46, 1, 29, 17, 57, 5, 36, 14, 6, 53, 59, 39, 4]

import math

import scipy.stats as st

nuevoarreglo = []

lista = [43, 11, 16, 40, 21, 3, 2, 46, 1, 29, 17, 57, 5, 36, 14, 6, 53, 59, 39, 4]

for i in range (0,len(lista)-1):

if lista[i] < lista[i+1]:

nuevoarreglo.append(1)

else:

nuevoarreglo.append(0)

print(nuevoarreglo)

corridas = 13

miuc0 = ((2\*20)-1)/3

sigma0 = ((16\*20)-29)/90

a = math.sqrt(sigma0)

print(miuc0)

b = corridas - miuc0

z=b/a

print(z)

import scipy.stats as st

print(st.norm.ppf(.95))

SALIDAS

[0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0]

13.0

0.0

1.6448536269514722

Como nuestro estadístico z es menor que el estadístico de la tabla esto significa que si pasa la prueba de las corridas y se puede afirmar que son independientes