Revista de Investigación Multidisciplinaria

CTSCAFE PARA CIUDADANOS.....

http://www.ctscafe.pe

ISSN 2521-8093

Volumen V- Nº 14 Julio 2021

http://www.ctscafe.pe

Lima - Perú

Programa para calcular funciones de distribución de variables aleatorias en formato del lenguaje GPSS

Sr. Michael Eduardo Atoccsa Lopez
Universidad Nacional de Ingeniería
Correo Electrónico: matoccsal@uni.pe

Resumen: El objetivo de este artículo es desarrollar una metodología de cálculo y procesamiento de datos para reducir el tiempo de obtener la función de distribución acumulada de variables continuas que permitan aplicar el Método Montecarlo (1944) en el lenguaje GPSS (lenguaje de programación de propósito general de simulación de eventos discretos). La metodología de cálculo se presenta en el lenguaje de programación Python.

Palabras claves: Programación/ Sturges/ Estadística/ Método Montecarlo/ Simulación.

Abstract: The objective of this paper is to develop a methology and data processing to reduce the time to obtain the cumulative distribution function of continuous variables that allow the application of the MonteCarlo Method (1944) in the GPSS language (general purpose programming language for discrete event simulation). The calculation methodology is presented in the Python programming language.

Keywords: Programming/ Sturges/ Statistics/ Monte Carlo Method/ Simulation.

Résumé : L'objectif de cet article est de développer une méthodologie de calcul et de traitement des données pour réduire le temps d'obtention de la fonction de distribution cumulative des variables continues qui permettent d'appliquer la méthode de Monte Carlo (1944) dans le langage GPSS (langage de programmation général pour la simulation d'événements discrets). La méthodologie de calcul est présentée dans le langage de programmation Python.

Mots-clés: Programmation/ Sturges/ Statistiques/ Méthode Monte Carlo/ Simulation.

1. Introducción

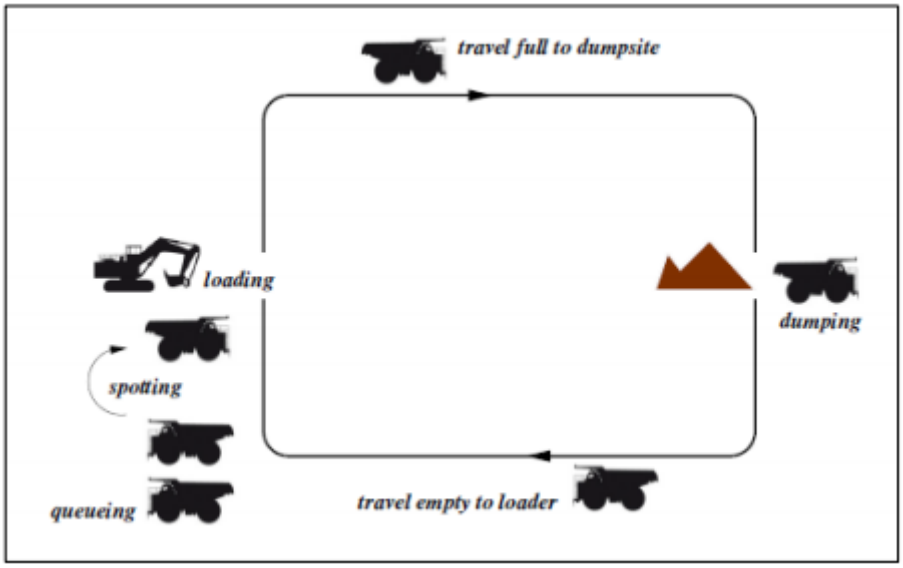
En el presente siglo XXI la programación de computadoras ha logrado imponerse como una herramienta que promueve la eficiencia de los procesos en toda actividad humana. Los ingenieros del siglo XXI cuentan con los lenguajes de programación como medio para interactuar con el computador y aplicar sus conocimientos lógicos y matemáticos para manipular la realidad en beneficio de la sociedad peruana. Con este trabajo se busca mostrar la importancia de un correcto dominio de un lenguaje de programación

moderno como lo es Python y la aplicación en el tratamiento de grandes volúmenes de datos para facilitar el desarrollo de modelos de simulación de sistemas discretos como son los sistemas de transporte de mineral en minería superficial. Se comparte el código desarrollado en este trabajo para beneficio de los ingenieros que trabajan con modelos de simulación y aplican el método de Montecarlo para generar sus variables aleatorias simuladas.

1.1. Planteamiento del problema

Se desea modelar el sistema de transporte de la mina Sur Perú, que opera bajo el método de explotación open pit, mediante el lenguaje GPSS con datos probabilísticos de tiempo, payload, etc. Para ello el programa usa el Método de Montecarlo pidiéndonos la función de distribución acumulada de las variables bajo un formato en específico que permite definir correctamente la función. Podemos observar el ciclo del camión minero básico en la Figura 1

Figura Nº1: Ciclo del camión minero.



Fuente: Burt & Caccetta, (2013)

La variable continua (tiempo, velocidades, tonelaje, etc.) tomados en campo o con algún sistema de gestión y monitoreo de los equipos como es el sistema Dispatch cuentan ya con los tratamientos previos que son:

- La base de datos debe ser analizada para eliminar outliers si los hubiera o filtrar la base de datos de manera que haya un sentido coherente con el modelo.
- La cantidad de registros por variable debe ser una cantidad considerable (del orden de miles).
- El encabezado de la variable debe tener el nombre de la función a utilizar en el lenguaje GPSS.
- La base de datos debe estar en formato .csv delimitado por comas (sin importar la cantidad de registros por variable).

2. Material y métodos

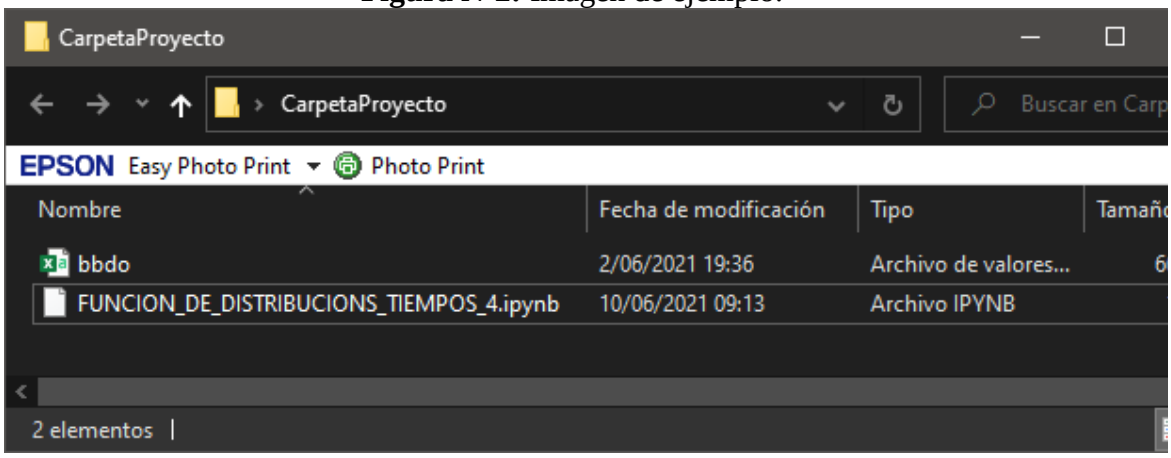
El procedimiento utiliza los archivos .txt y .xlsx, el lenguaje de programación Python. Al tener la(s) variable(s) en un archivo .csv delimitado por comas con encabezados que tengan como nombre(s) la función(es) a utilizar en el lenguaje de programación GPSS. El programa lee todos los registros de la variable y mediante la regla de Sturges calcula el número de intervalos para nuestra tabla de frecuencia y mediante una compensación de rango se obtiene los intervalos de clase identificados. Mediante contadores se captura las cantidades de valores que pertenecen a un intervalo de clase (frecuencia absoluta) y la cantidad acumulada de valores hasta un intervalo de clase (frecuencia absoluta acumulada). Con esta información se completa la tabla de frecuencia con la marca de clase (Xi), frecuencia relativa (hi) y frecuencia relativa acumulada (Hi). Debido al formato de instancia de una función en el lenguaje GPSS tenemos que aumentar un intervalo de clase, con el mismo ancho de clase, previo al primero. Luego capturamos los pares ordenados (Hi, Xi) que con la condición antes mencionada todos los primeros pares ordenados calculados por cada variable debe comenzar con (0, X). Damos una forma de presentación de los pares ordenados que representan la función de distribución acumulada de la variable que es puesta en un archivo .txt.

3. Resultados

3.1. Carpeta del Proyecto

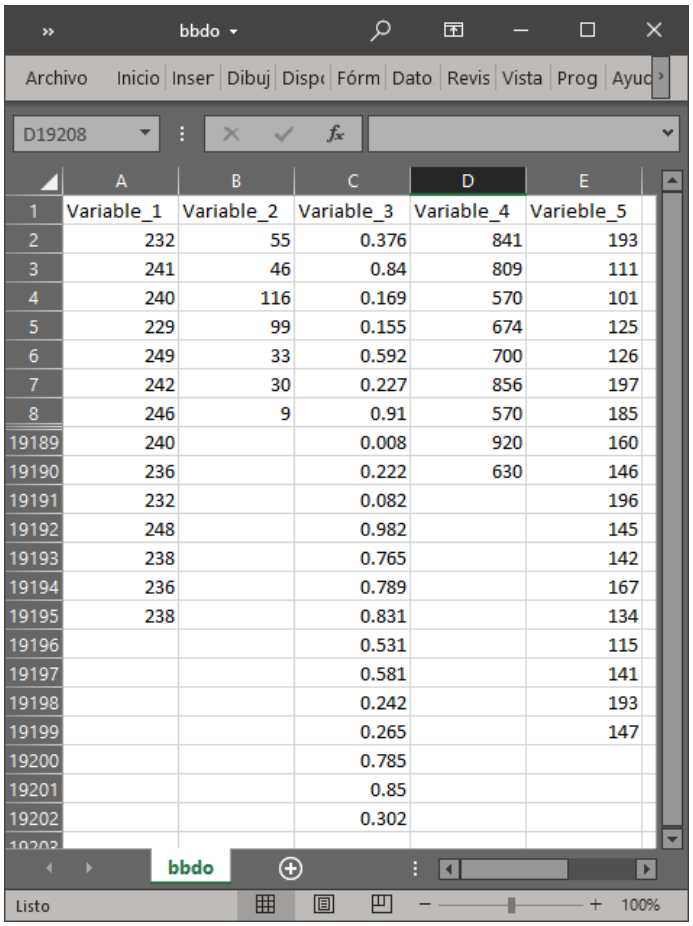
Crea una carpeta en la dirección de su preferencia y agregue la base de datos de las variables continuas. Guarde la ruta tanto de la carpeta creada y de la base de datos que fue agregado.

Figura N°2: Imagen de ejemplo.



Fuente: Elaboración Propia

Figura Nº3: Ejemplo de base de datos. Las variables no necesariamente deben tener la misma cantidad de registros



Fuente: Elaboración Propia

3.2. Ejecutar el algoritmo.

Debemos tener instalado el entorno de programación interactica Jupyter Notebook. Abrir FUNCION_DE_DISTRIBUCION_4.ipynb con Jupyter Notebook brindado con el artículo y a continuación corra el algoritmo. Le pedirá el nombre de los archivos que generar un .xlsx que nos permita visualizar la tabla de frecuencias, polígono de frecuencia y ojiva correspondiente. Una hoja de cálculo por variable y otro .txt que tendremos que copiar y pegar directamente al modelo que estamos haciendo en lenguaje GPSS. Todo este procesamiento lo realiza siguiendo una secuencia de algoritmos representado por el diagrama de flujo mostrado en el Anexo

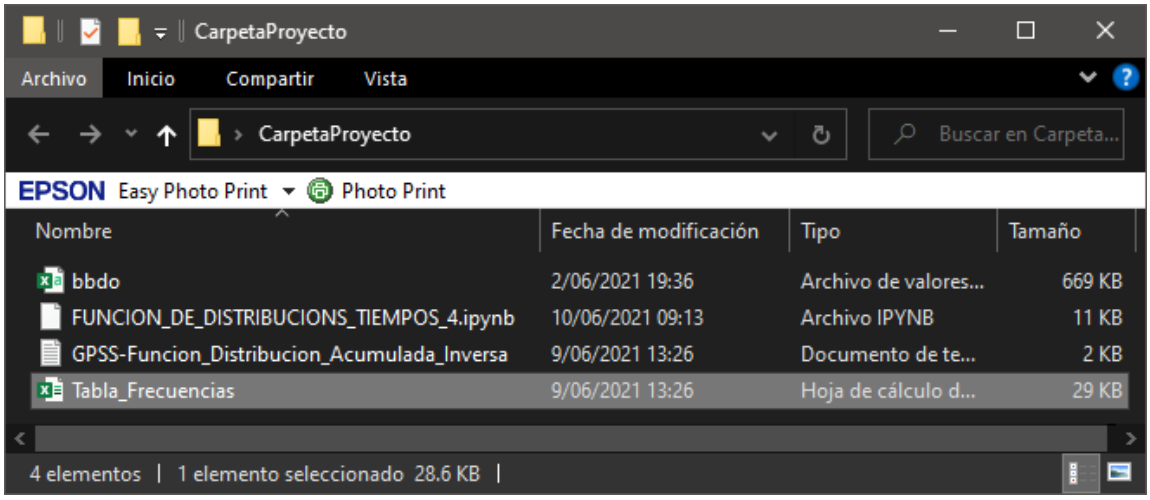
Tendrá que introducir la ruta de la base de datos (recordar que es un archivo .csv). Por ejemplo: C:\Users\MiembroCGP\Desktop\CarpetaProyecto\data.csv. En el explorador de archivo buscar el .csv y hacer un click izquierdo, luego buscar en el MENU INICIO la opción COPIAR RUTA DE ACCESO. De la misma forma nos pedirá la ruta de la carpeta creada para que en esa carpeta cree los outputs (tanto el archivo .xlsx y .xtx). Por ejemplo: C:\Users\MiembroCGP\Desktop\CarpetaProyecto

En el output .xlsx creado los datos estarán con 6 decimales, pero en el output .txt que va al GPSS debe ser especificada un número de decimales correcto de acuerdo con el trabajo a desarrollar es por ello nos pedirá ese parámetro.
Por último, nos pedirá un número entero debido a que todas las funciones usadas en GPSS deben estar enumeradas así que nos pide el entero por el cual comienza a enumerar esas funciones. Por ejemplo, si en la base de datos hay 2 variables y en este campo rellenamos 5, la primera variable tendrá la posición 5 y la siguiente variable tendrá la posición 6.

3.3. Outputs

En la carpeta del proyecto ya podremos observar el archivo .xlsx y .txt generado como vemos en la Figura 4.

Figura Nº4: Carpeta del proyecto ya ejecutada el programa. Se observa los outpus Tabla_Frecuencia.xlsx y GPSS_Funcion_Distribucion_Acumulada_Inversa.txt



Fuente: Elaboración Propia

La base de datos será procesada por el programa propuesto. Los outputs generados a partir de la bbdo.csv (base de datos ejemplo) de la Figura 3 son los siguientes:

- Archivo .txt:

FUNCIÓN DE DISTRIBUCIÓN DE PROBABILIDAD ACUMULADA (INVERSA)********

Variable_1 FUNCTION RN1,C16
0.0,145.833/0.000208,154.167/0.000625,162.5/0.000886,170.833/0.001302,179.167/0.002032,187.5/0.005054,195.833/0.014484,204.167/0.035219,212.5/0.105866,220.833/0.288371,229.167/0.644212,237.5/0.936803,245.833/0.996509,254.167/0.999687,262.5/1.0,270.833

Variable_2 FUNCTION RN2,C16
0.0,3.0/0.067449,4.0/0.13448,12.0/0.201251,20.0/0.271566,28.0/0.337555,36.0/0.40344,44.0/0.470628,52.0/0.533802,60.0/0.598176,68.0/0.661976,76.0/0.724576,84.0/0.792807,92.0/0.861767,100.0/0.925775,108.0/1.0,116.0

Variable_3 FUNCTION RN3,C16
0.0,0.033/0.063695,0.033/0.127285,0.1/0.192855,0.167/0.261341,0.233/0.32816,0.3/0.394771,0.367/0.463465,0.433/0.531222,0.5/0.601583,0.567/0.669184,0.633/0.734024,0.7/0.800635,0.767/0.867715,0.833/0.935055,0.9/1.0,0.967

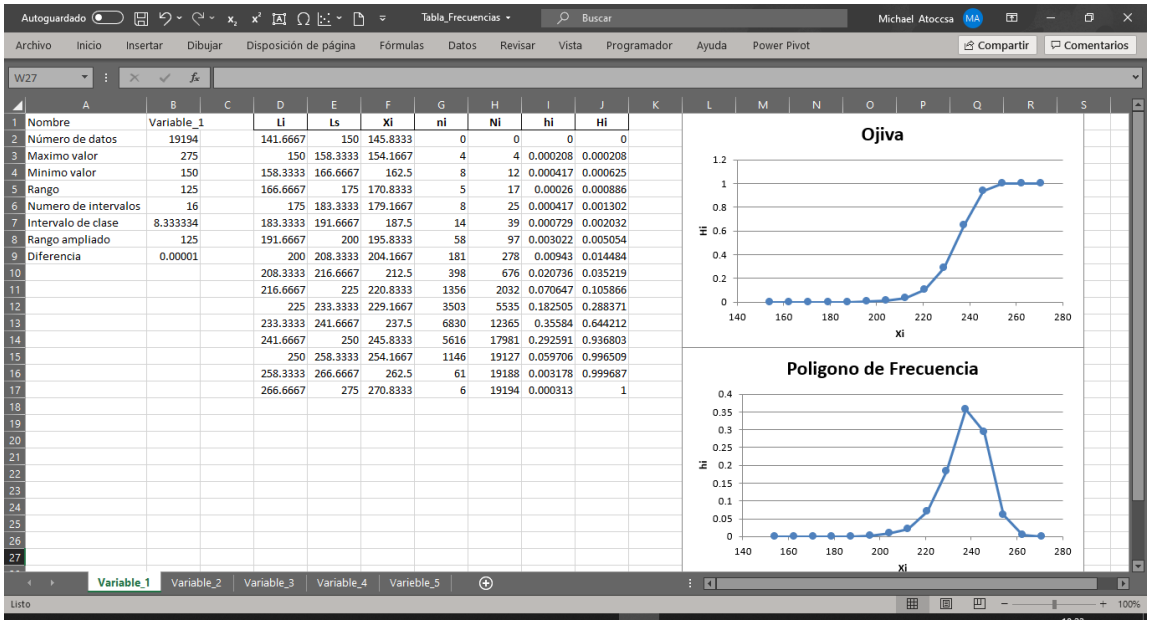
Variable_4 FUNCTION RN4,C16
0.0,483.333/0.07077,516.667/0.138517,550.0/0.202043,583.333/0.271666,616.667/0.338632,650.0/0.403565,683.333/0.469436,716.667/0.534317,750.0/0.597999,783.333/0.665694,816.667/0.730054,850.0/0.799052,883.333/0.865392,916.667/0.930637,950.0/1.0,983.333

Varieble_5 FUNCTION RN5,C16
0.0,96.667/0.068497,103.333/0.137358,110.0/0.195958,116.667/0.268101,123.333/0.337535,130.0/0.398062,136.667/0.466142,143.333/0.531201,150.0/0.604855,156.667/0.663611,163.333/0.733306,170.0/0.800552,176.667/0.862173,183.333/0.932076,190.0/1.0,196.667

Este archivo representa el formato que permite declarar funciones de variables continuas en el lenguaje de programación GPSS.

- Archivo .xlsx:

Figura Nº5: Libro Excel en donde cada hoja de cálculo muestra la tabla de frecuencia, ojiva y polígono de frecuencia correspondiente a cada una de las variables.



	A	B	C	D	E	F	G	H	I	J
1	Nombre	Variable_1		Li	Ls	Xi	ni	Ni	hi	Hi
2	Número de datos	19194		141.6667	150	145.8333	0	0	0	0
3	Maximo valor	275		150	158.3333	154.1667	4	4	0.000208	0.000208
4	Minimo valor	150		158.3333	166.6667	162.5	8	12	0.000417	0.000625
5	Rango	125		166.6667	175	170.8333	5	17	0.00026	0.000886
6	Numero de intervalos	16		175	183.3333	179.1667	8	25	0.000417	0.001302
7	Intervalo de clase	8.333334		183.3333	191.6667	187.5	14	39	0.000729	0.002032
8	Rango ampliado	125		191.6667	200	195.8333	58	97	0.003022	0.005054
9	Diferencia	0.00001		200	208.3333	204.1667	181	278	0.00943	0.014484
10				208.3333	216.6667	212.5	398	676	0.020736	0.035219
11				216.6667	225	220.8333	1356	2032	0.070647	0.105866
12				225	233.3333	229.1667	3503	5535	0.182505	0.288371
13				233.3333	241.6667	237.5	6830	12365	0.35584	0.644212
14				241.6667	250	245.8333	5616	17981	0.292591	0.936803
15				250	258.3333	254.1667	1146	19127	0.059706	0.996509
16				258.3333	266.6667	262.5	61	19188	0.003178	0.999687
17				266.6667	275	270.8333	6	19194	0.000313	1

Fuente: Elaboración Propia

4. Conclusiones

- Se desarrolló un programa en el lenguaje Python que calcule la función de distribución acumulada de una variable(s) en formato del lenguaje GPSS.
- Este programa permite ahorrar tiempo de procesamiento de datos cuando programamos en el lenguaje GPSS y necesitamos definir funciones continuas sea cual sea el número de estas funciones
- Se procesa el 100% de los datos garantizando la representatividad en los cálculos.
- Este programa permite calcular funciones de distribución de variables aleatorias que alimentan a un modelo de simulación del sistema de transporte mineral en minería superficial. En particular se entrega la función en formato GPSS

5. Literatura citada

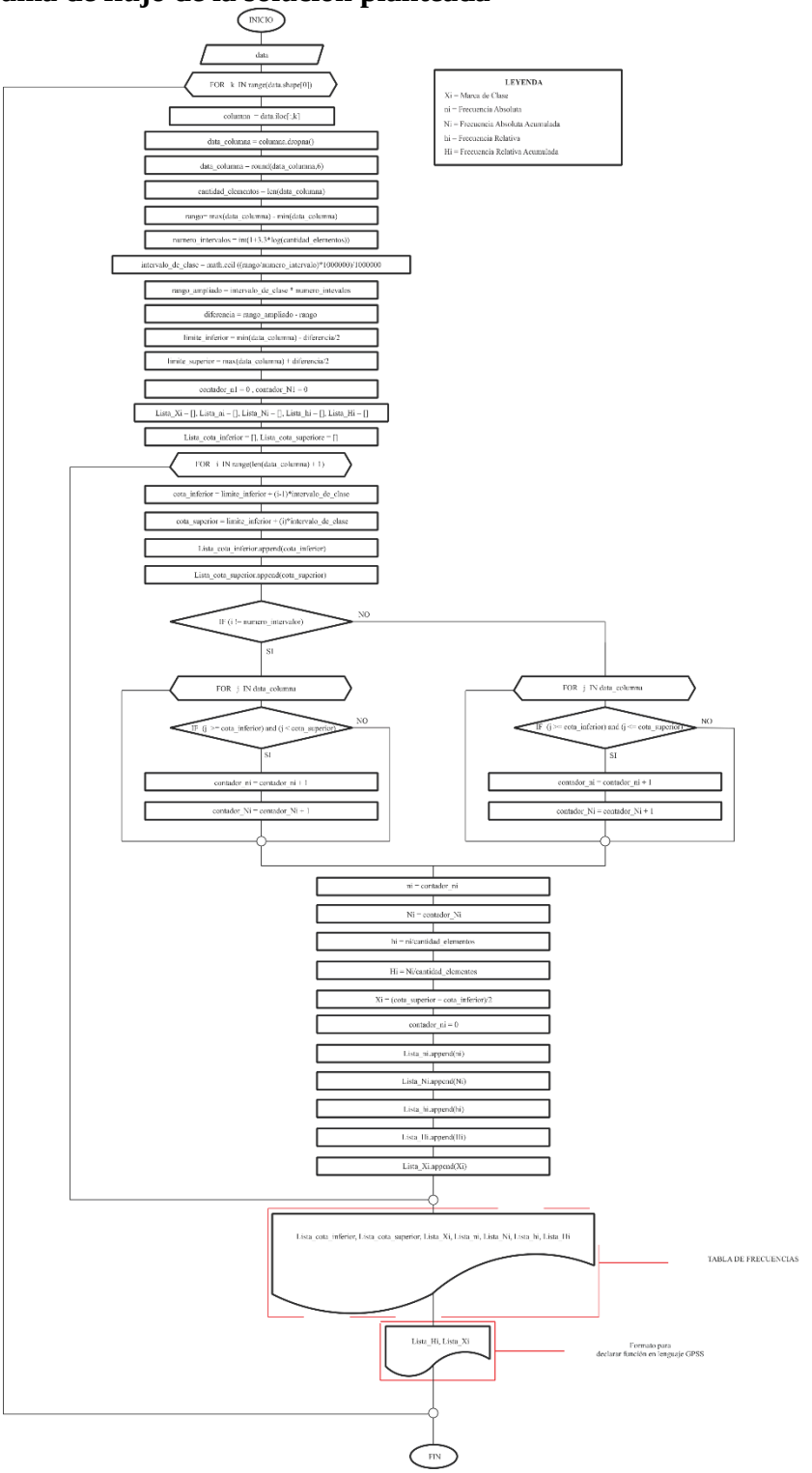
Marín Suárez, Alfredo (1976). Méthodologie de l'estimation et simulation multivariable des grands gisements tridimensionnels / {A. Marin-Suarez} Publication : Fontainebleau : ENSMP, 1978.

Marín Suárez, Alfredo (1986). Modelo geoestadístico de finlones de almaden, FONTAINEBLEAU. ENSMP, 1986

.

6. Anexos

6.1. Diagrama de flujo de la solución planteada



6.2. Código Fuente del programa en Python

Autor: Michael Eduardo Atoccsa Lopez
matoccsal@uni.pe
Centro Geoestadístico Peruano
Escuela de Minas de la Universidad Nacional de Ingeniería

```
!pip install XlsxWriter
import numpy as np
import pandas as pd
import os
import math
import xlsxwriter
Nombre_archivo = input('¿Cúal sera el nombre de los archivos? ')
Nombre_archivo_GPSS = Nombre_archivo
Nombre_archivo_excel = Nombre_archivo + '.xlsx'
Ubicacion_Data = str(input('¿Cual es la ruta del archivo .csv?: '))
Ubicacion_Data = Ubicacion_Data.replace('\\','/')
Ubicacion_output = str(input('Rutee el nombre de la carpeta: '))
Ubicacion_output = Ubicacion_output.replace('\\','/')

Decimales = int(input('¿Cuantos decimales? '))

#Lee la(s) variables(s) en formato .csv
data = pd.read_csv(Ubicacion_Data)

número_columnas = data.shape[1]
#Nombre de las columnas
columnas = data.columns.to_list()

Lista_Nombre_Columna = []
Lista_Numero_Pares = []
Lista_FDA = [] #DFA = Función de Distribución acumulada

#Crear el método la conexión entre el  módulo XlsxWriter y Pandas
#Aca tenemos que poner el nombre del archivo xlsx
#Engine = Se encarga de hacer la conexión
writer = pd.ExcelWriter(Ubicacion_output+'/'+Nombre_archivo_excel,
engine='xlsxwriter')

#Procesamos los dartos
for k in range(data.shape[1]):

    Nombre_Columna = columnas[k]
    #los datos de una columna
    data_columna = round(data[columnas[k]].dropna(),6)
    #¿Cuantos datos tengo?
    cantidad_elementos = len(data_columna)
    # Rango = max - min
    rango = max(data_columna) - min(data_columna)
```

```
# Numero de intervalos de clase. Es un número entero). Tan solo redondeamos al entero más cercano
número_intervalos = int(round(1 + 3.3* np.log10(cantidad_elementos),0))
# Intervalo de clase es el rango entre el número de intervalo. Redondeamos por exceso a la cantidad
# de decimales que tenga nuestros datos
intervalo_de_clase = math.ceil((rango/numero_intervalos)*(10**6))/(10**6)
# Rango amplidado = intervalo de clase * número de inervalos
rango_ampliado = intervalo_de_clase*número_intervalos
diferencia = round((rango_ampliado - rango)*(10**6))/(10**6)
#distribuimos la diferencia
limite_inferior = min(data_columna) - diferencia/2
limite_superior = max(data_columna) + diferencia/2

#Construimos la tabla de frecuencia
Lista_cota_inferior = []
Lista_cota_superior = []
Lista_Xi = []
Lista_ni = []
Lista_Ni = []
Lista_hi = []
Lista_Hi = []
contador_Ni = 0
contador_ni = 0

for i in range(numero_intervalos+1): # 0,1,2,3,4,5,6,7,/8

    cota_inferior = límite_inferior + (i-1)*intervalo_de_clase
    cota_superior = límite_inferior + (i)*intervalo_de_clase
    Lista_cota_inferior.append(cota_inferior)  # Li
    Lista_cota_superior.append(cota_superior) #Ls

    if i != (número_intervalos):

        for j in data_columna:

            if (j >= cota_inferior) and (j < cota_superior):
                contador_Ni = contador_Ni + 1
                contador_ni = contador_ni + 1

    else:

        for j in data_columna:

            if (j >= cota_inferior) and (j <= cota_superior):
                contador_Ni = contador_Ni + 1
                contador_ni = contador_ni + 1
```

```
        ni = contador_ni
        contador_ni = 0
        Ni = contador_Ni
        hi = ni/cantidad_elementos
        Hi = Ni/cantidad_elementos
        Xi = (cota_superior + cota_inferior)/2
        Lista_ni.append(ni)
        Lista_Ni.append(Ni)
        Lista_hi.append(round((hi)*(10**6))/(10**6))
        Lista_Hi.append(round((Hi)*(10**6))/(10**6))
        Lista_Xi.append(Xi)

    #Cramos la tabla de frecuencias
    dic1 = {'Li': Lista_cota_inferior, 'Ls': Lista_cota_superior,'Xi': Lista_Xi, 'ni': Lista_ni,
'Ni': Lista_Ni,'hi':Lista_hi, 'Hi': Lista_Hi}
    Tabla_de_Frecuencia = pd.DataFrame(dic1)

    #Giardamos todos los datos en una lista para poder dar el formato de instancia de
funcion en GPSS
    Lista_Xi = Tabla_de_Frecuencia['Xi'].to_list()
    Lista_Hi = Tabla_de_Frecuencia['Hi'].to_list()
    Lista_Interna_FDA =
[str(round(Lista_Hi[x],6))+','+str(round(Lista_Xi[x],Decimales)) for x in
range(len(Lista_Xi))]
    String_FDA = ''
    for g in range(len(Lista_Interna_FDA)):
        if g < (len(Lista_Interna_FDA)-1):
            String_FDA = String_FDA + Lista_Interna_FDA[g] +'/'
        else:
            String_FDA = String_FDA + Lista_Interna_FDA[g]

    Lista_FDA.append(String_FDA)
    Lista_Nombre_Columna.append(Nombre_Columna)
    Lista_Numero_Pares.append(Tabla_de_Frecuencia.shape[0])

    #Creamos la tabla de informacion
    Info = ['Nombre','Número de datos','Maximo valor','Minimo valor','Rango','Numero
de intervalos','Intervalo de clase','Rango ampliado','Diferencia']
    Valor =
[Nombre_Columna,cantidad_elementos,max(data_columna),min(data_columna),rango,
numero_intervalos+1,intervalo_de_clase,rango_ampliado,diferencia]
    dic = {'Info' : Info, 'Valor':Valor}
    Tabla_Info = pd.DataFrame(dic)
    df = Tabla_Info.copy()
    df1 = Tabla_de_Frecuencia.copy()

    #Convierte el DataFrame a un XlsxWriter Excel object
    #El df utiliza la funcion writer para pasar a un objeto de
    #XlsxWriter Excel object  y lo pone en la hoja mencionada
```

```
    df.to_excel(writer, sheet_name=Nombre_Columna, index = False, header = False)
    df1.to_excel(writer, sheet_name=Nombre_Columna, index = False ,startcol = 3)

    #Ahora para poder escribir en la hoja de excel tenemos lo siguiente
    workbook = writer.book #workbook = xlsxwriter.Workbook('filename.xlsx')
    worksheet = writer.sheets[Nombre_Columna] #worksheet = workbook.add_worksheet()

    #Ajustamos el tamaño de la primera columna
    worksheet.set_column('A:A', 20)

    #Insertamos el grafico de la ojiva
    #creamos el objeto que llama a una grafica
    chart = workbook.add_chart({'type':'scatter',
                                'subtype':'straight_with_markers'})

    #Dandole valores a la grafica y agregamos una serie. asi como esta podemos agregar mas
    max_row = len(df1)
    chart.add_series({
          'name':       [Nombre_Columna, 0, 9],
          'categories': [Nombre_Columna, 1, 5, max_row, 5], #X
          'values':     [Nombre_Columna, 1, 9, max_row, 9], #Y
          'marker':     {'type': 'circle', 'size': 7},
       })

    #Configurar los ejes de la grafica
    chart.set_title({'name': 'Ojiva'})
    chart.set_x_axis({'name':'Xi'})
    chart.set_y_axis({'name':'Hi'})

    #Apagar la legenda porque si no lo ponemos esta por defecto
    chart.set_legend({'position': 'none'})

    #Insertar la gafica en un hoja de calculo
    worksheet.insert_chart('L1', chart)

    #Insertamos el grafico del poligono de frecuencia
    chart1 = workbook.add_chart({'type':'scatter',
                                 'subtype':'straight_with_markers'})
    max_row = len(df1)
    chart1.add_series({
       'name' : [Nombre_Columna, 0, 8],
       'categories' : [Nombre_Columna, 1, 5, max_row, 5],
       'values' : [Nombre_Columna, 1, 8, max_row, 8],
       'marker' : {'type': 'circle', 'size': 7}})
    chart1.set_title({'name':'Poligono de Frecuencia'})
    chart1.set_x_axis({'name':'Xi'})
```

```
    chart1.set_y_axis({'name':'hi'})

    chart1.set_legend({'position':'none'})
    worksheet.insert_chart('L15',chart1)

writer.save()

#En un archivo .txt damos el formato
Numero_de_la_funcion = int(input('¿Con que numero comienza la(s) funcion(es) en GPSS? '))
FileDatos = open(Ubicacion_output+"/GPSS-"+Nombre_archivo_GPSS+".txt","w")
for i in range(len(Lista_Nombre_Columna)):
    FileDatos.write(Lista_Nombre_Columna[i]+"     FUNCTION
          "+'RN'+str(i+Numero_de_la_funcion)+",C"+str(Lista_Numero_Pares[i])+'\n')
    FileDatos.write(Lista_FDA[i]+'\n')

FileDatos.close()
```

REVISTA DE INVESTIGACIÓN MULTIDISCIPLINARIA

CTSCAFE Para ciudadanos…

http://www.ctscafe.pe

Volumen V- N° 13 Marzo 2021

Contáctenos en nuestro correo electrónico
revistactscafe@ctscafe.pe

Página Web:
http://ctscafe.pe

Blog:
https://ctscafeparaciudadanos.blogspot.com/

Facebook
https://www.facebook.com/Revista-CTSCafe-1822923591364746/