Métodos Estadísticos y simulación: Simulación de una distribución binomial

Antonio Sixto López Malo

Universidad de Málaga
Tuesday 23rd May, 2023

UNIVERSIDAD DE MÁLAGA

Contenido

1. Introducción
2. Estructura del código
3. Main
4. Análisis Estadistico
5. Visualización
6. Ejecución
7. Conclusión

Introducción

En esta presentación vamos a ver cómo generar una muestra que sigue una **distribución binomial**, la distribución de frecuencias mediante un **histograma** y la aplicación del **Teorema Central del Límite.** Todo ello mediante el lenguaje de programación Python.

Estructura del código

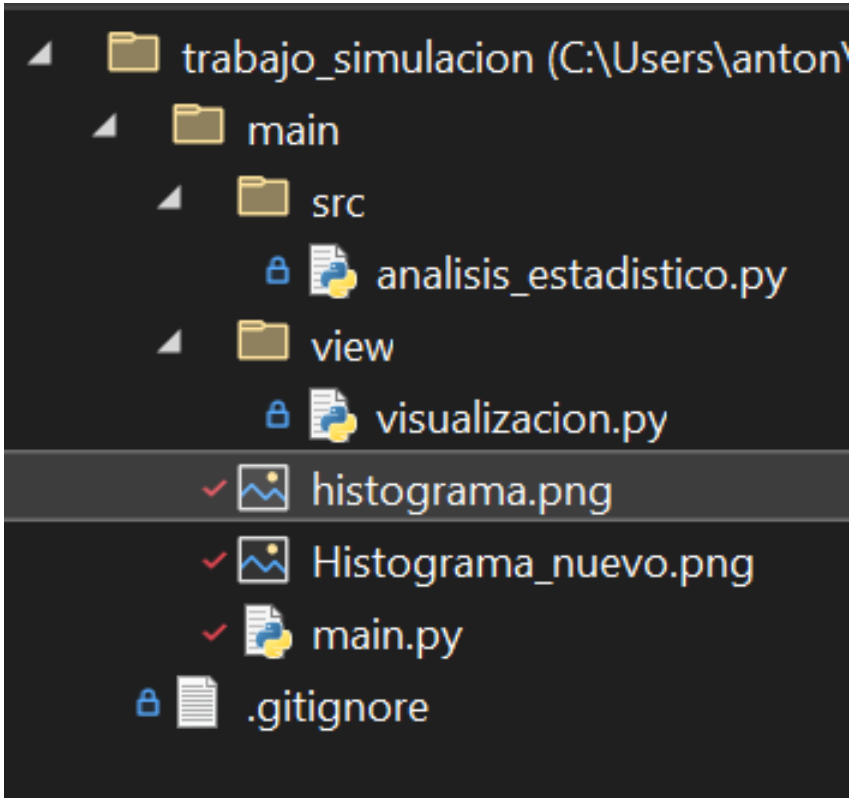


Figure: Estructura del código

- Hay una carpeta main, que es donde se encuentra el main.py.
- Dentro de la carpeta main, hay otras dos carpetas llamadas src y view
- En **src** está todo lo correspondiente a las operaciones en sí (generar la muestra, calcular la media, la varianza, etc.
- En la carpeta **view** se encuentra todo lo referente a la generación de los histogramas.

Main

main.py | analisis_estadistico.py | visualizacion.py

```python
# -*- coding: iso-8859-1 -*-

# main.py

from src.analisis_estadistico import AnalisisEstadistico
from view.visualizacion import crear_histograma_normal, crear_histograma_superpuesto, mostrar_ocurrencias

if __name__ == "__main__":

    #Apartado 1 (src.analisis_estadistico.py)
    tamanio_muestra = 1000
    num_ensayos = 10
    probabilidad_de_exito = 0.5
    muestra_binomial = AnalisisEstadistico.generar_muestra_binomial(tamanio_muestra, num_ensayos, probabilidad_de_exito)
    print(muestra_binomial) #comentar esta linea si va a ejecutar muestras grandes
    #mostrar_ocurrencias(muestra_binomial)

    #Apartado 2 (view.visualizacion.py)
    crear_histograma_normal(muestra_binomial, "histograma")

    #Apartado 3
    lim_inferior_del_intervalo = 3
    lim_superior_del_intervalo = 7
    frec_relativa, prob_real, error = AnalisisEstadistico.estimar_probabilidad_intervalo(
        muestra_binomial, num_ensayos, probabilidad_de_exito, lim_inferior_del_intervalo, lim_superior_del_intervalo)
    print("Frecuencia relativa:", frec_relativa)
    print("Probabilidad real:", prob_real)
    print("Error:", error)

    #Apartado 4
    media = AnalisisEstadistico.calcular_media(muestra_binomial)
    varianza = AnalisisEstadistico.calcular_varianza(muestra_binomial)
    print("Media: ", media)
    print("Varianza: ", varianza)

    #Apartado 5
    muestra_nueva = AnalisisEstadistico.generar_muestra_nueva(muestra_binomial, media, varianza)
    print([round(x, 3) for x in muestra_nueva])
    media_muestra_nueva = AnalisisEstadistico.calcular_media(muestra_nueva)
    varianza_muestra_nueva = AnalisisEstadistico.calcular_varianza(muestra_nueva)
    print("Media_nueva: ", abs(round(media_muestra_nueva, 6)))
    print("Varianza_nueva: ", round(varianza_muestra_nueva, 6))

    #genero valores para la densidad normal
    x, densidad_normal = AnalisisEstadistico.generar_densidad_normal()
    crear_histograma_superpuesto(muestra_nueva, x, densidad_normal)
```

Main

La clase main es la encargada de ejecutar el proyecto. Se divide en varias partes (cada uno de los apartados del trabajo):

- **Apartado 1:** Generamos la muestra binomial.
- **Apartado 2:** Creamos el histograma y lo almacenamos en el archivo "histograma.png".
- **Apartado 3:** Calculamos la probabilidad relativa de un intervalo, la probabilidad real y el error cometido.
- **Apartado 4:** Calculamos la media y la varianza de la muestra obtenida en el apartado 1.
- **Apartado 5:** Generamos una nueva muestra aplicando el Teorema Central del Límite sobre la primera, calculamos su media y su varianza y almacenamos su histograma en "Histograma_nuevo.png".

Análisis Estadístico

main.py | analisis_estadistico.py | visualizacion.py

```python
# analisis_estadistico.py

import random
import numpy as np
from scipy import stats


class AnalisisEstadistico:
    @staticmethod
    def generar_muestra_binomial(tam, n, p):
        muestra = []
        for _ in range(tam):
            exitos = 0
            for _ in range(n):
                resultado_ensayo = random.uniform(0, 1)
                if resultado_ensayo < p:  # probabilidad de éxito
                    exitos += 1
            muestra.append(exitos)
        return muestra

    def estimar_probabilidad_intervalo(muestra, n, p, a, b):
        muestra = np.array(muestra)
        frec_relativa = np.mean((muestra >= a) & (muestra <= b))
        prob_real = stats.binom.cdf(b, n, p) - stats.binom.cdf(a-1, n, p)
        error = abs(frec_relativa - prob_real)

        return frec_relativa, prob_real, error

    def calcular_media(muestra):
        return np.mean(muestra)

    def calcular_varianza(muestra):
        return np.var(muestra)

    def generar_muestra_nueva(muestra_binomial, media, varianza):
        raiz_varianza = np.sqrt(varianza)
        muestra_nueva = [((x-media)/(raiz_varianza)) for x in muestra_binomial]
        return muestra_nueva

    def generar_densidad_normal():
        x = np.linspace(-4, 4, 500)
        densidad_normal = (1 / np.sqrt(2 * np.pi)) * np.exp(-0.5 * x**2)
        return x, densidad_normal

```

Análisis Estadístico

La clase "analisis_estadistico" es la encargada de realizar todos los cálculos de pura estadística. Para ello nos apoyamos en las librerías de "numpy", "random" y "stats" de scipy.

- **Generar_muestra_binomial:** genera una muestra de variables aleatorias binomiales. Recibe el tamaño de la muestra, el número de ensayos y la probabilidad de éxito. Utiliza bucles for anidados para realizar los ensayos y contar los éxitos. Los resultados se almacenan en una lista y se devuelven al final de la función.
- **Estimar_probabilidad_intervalo:** calcula la probabilidad real de que una variable aleatoria binomial caiga dentro de un intervalo especificado. Utiliza la función de distribución acumulativa de la distribución binomial para obtener esta probabilidad real. Luego, compara esta probabilidad con la frecuencia relativa obtenida de una muestra y calcula el error absoluto. La función devuelve la frecuencia relativa, la probabilidad real y el error absoluto como medidas de la precisión de la estimación.

Análisis Estadístico

- **Calcular_media:** calcula la media aritmética de una muestra de datos utilizando "np.mean(muestra)".
- **Calcular_varianza:** calcula la varianza de una muestra de datos utilizando "np.var(muestra)".
- **Generar_muestra_nueva:** calcula la muestra nueva aplicando el Teorema Central del Límite a una muestra dada.
- **Generar_densidad_normal:** genera una densidad de probabilidad normal estándar. Utiliza "np.linspace()" para obtener puntos equidistantes en el eje x y luego calcula la densidad utilizando la fórmula de la campana de Gauss. La función retorna dos arrays: uno con los puntos en el eje x y otro con los valores de la densidad de probabilidad normal correspondientes a esos puntos.

Visualización

main.py | analisis_estadistico.py | visualizacion.py

```python

import matplotlib.pyplot as plt
import numpy as np
import collections


def crear_histograma_normal(muestra_binomial, nombre_archivo):

    fig, ax = plt.subplots()
    bin_edges = np.arange(min(muestra_binomial) - 0.5, max(muestra_binomial) + 1.5, 1)

    # Generar histograma con los límites de los bins ajustados
    ax.hist(muestra_binomial, bins=bin_edges, align='mid', edgecolor='black')
    ax.set_xlabel('Numero de exitos')
    ax.set_ylabel('Frecuencia')
    ax.set_title('Distribucion de frecuencias')
    #plt.xticks(np.arange(min(muestra_binomial), max(muestra_binomial)+1, 1))
    ax.set_xticks(range(0, 11))
    ax.set_gid(True)

    # Crear histograma
    fig.savefig(nombre_archivo + ".png")



def crear_histograma_superpuesto(muestra, x, densidad_normal):
    fig, ax = plt.subplots()
    bin_edges = np.arange(min(muestra) - 0.5, max(muestra) + 1.5, 1)

    # Generar histograma con los límites de los bins ajustados
    n, bins, patches = ax.hist(muestra, bins=bin_edges, edgecolor='black', align='mid')
    escala = np.sum (n * np.diff(bins))
    densidad_normal *= escala
    ax.plot(x, densidad_normal, color='red', label='Distribucion normal')
    ax.set_xlabel('Valores')
    ax.set_ylabel('Frecuencia')
    ax.set_xticks(range(-5, 5))
    ax.set_gid(True)
    ax.legend()

    fig.savefig("Histograma_nuevo.png")


def mostrar_ocurrencias(muestra):
    conteo = collections.Counter(muestra)
    for elemento, frecuencia in sorted(conteo.items()):
        print(f"{elemento} --> {frecuencia}")

```

Visualización

En esta clase se encuentran las funciones encargadas de generar histogramas y salidas del programa.

- **Crear_histograma_normal:** genera un histograma de frecuencias de una muestra dada y lo almacena en un archivo ".png" con el nombre que se le pase como parámetro.
- **Crear_histograma_superpuesto:** genera un histograma de frecuencias de una muestra dada y además se le superpone una grafica de una densidad normal con los parámetros dados.

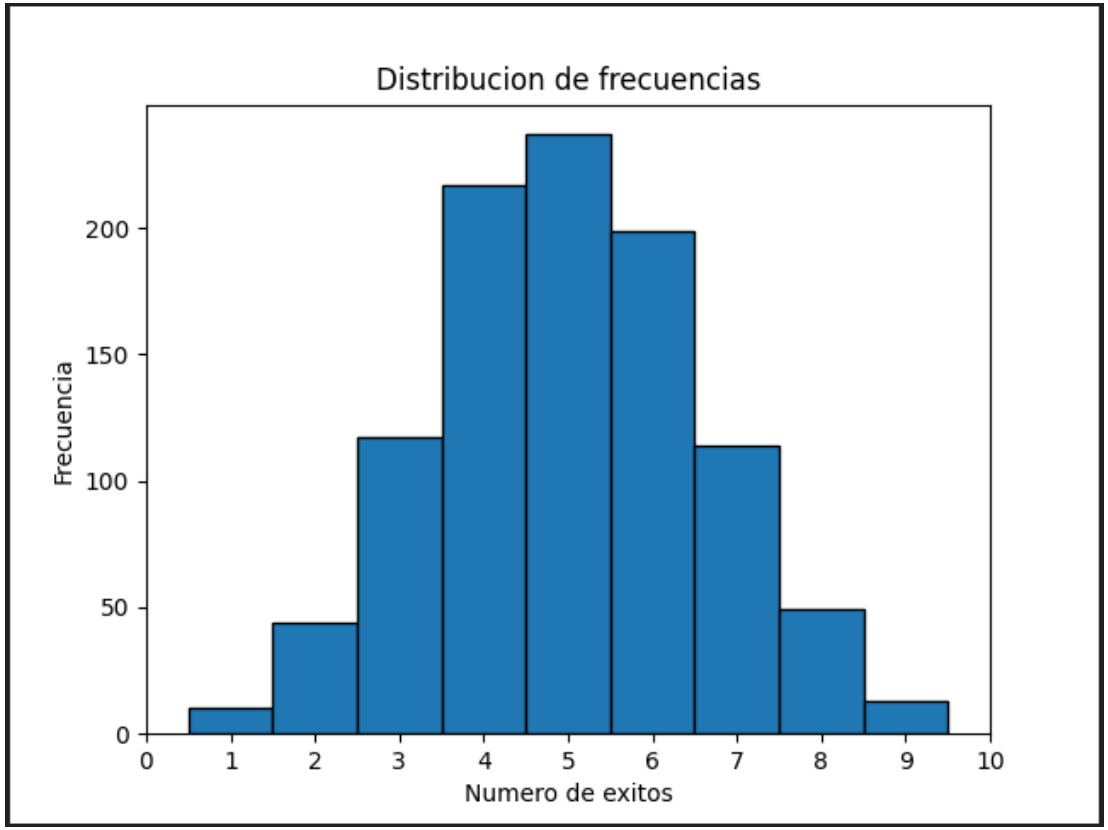
Ejecución

- La salida nos muestra los valores correspondientes a los apartados 3, 4 y 5 del trabajo.

```
C:\Users\anton\Informatica\Tercero\ModelosEstadisticos\trabajo_simulacion\main>python main.py
Frecuencia relativa: 0.884
Probabilidad real: 0.890625
Error: 0.006624999999999992
Media:  5.003
Varianza:  2.5449909999999996
Media_nueva:  0.0
Varianza_nueva:  1.0
```

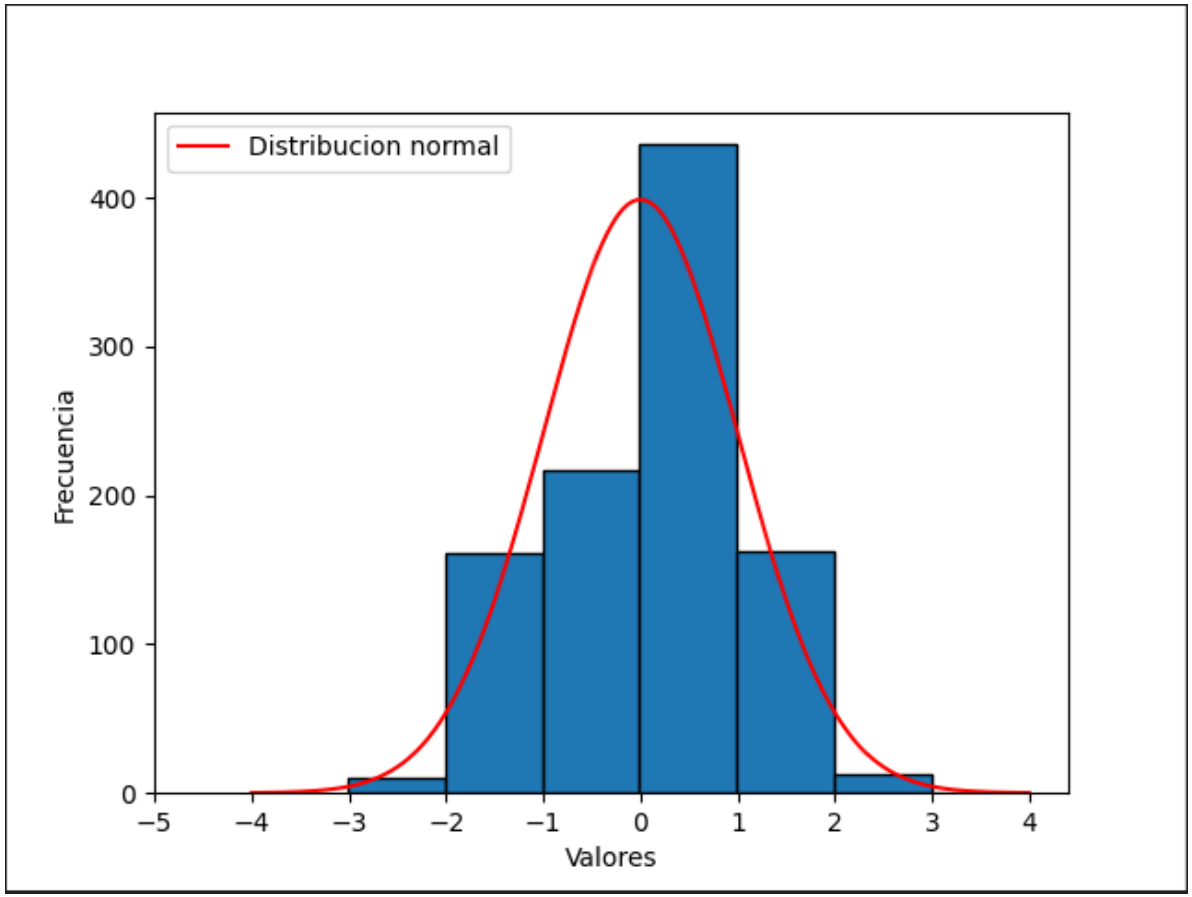
Ejecución

- El siguiente histograma representa la distribución de las frecuencias de la muestra generada en el apartado 1.



Ejecución

- El siguiente histograma representa la distribución de las frecuencias de la muestra generada en el apartado 5 y la superposición de una grafica de densidad normal.



Conclusión

- En este trabajo, hemos explorado la generación de una muestra que sigue una distribución binomial, hemos analizado las frecuencias mediante histogramas y hemos aplicado el Teorema Central del Límite.
- Hemos utilizado el lenguaje de programación Python y las librerías numpy, random y stats de scipy para realizar los cálculos y generar visualizaciones.
- Observamos que a medida que aumentamos el tamaño de la muestra, la distribución de frecuencias se aproxima a una distribución normal, lo que confirma el Teorema Central del Límite.
- Además, hemos calculado la media y la varianza de la muestra y hemos obtenido resultados consistentes con las propiedades de la distribución binomial.