Proyecto parcial 1

**Método Lineal Congruencial**

**Código**

*import* matplotlib.pyplot *as* plt

*# método lineal congruencial (L.C.G.)*

*#Linear congruential method: Generate 10 numbers with the following parameters: X0 = 6, a = 32, c = 3, m = 80.*

def lcg(*seed*):

*while* True:

        seed = ((32 \* seed + 3) % 80)

*yield* seed / 80

rand = lcg(6)

values = [next(rand) *for* \_ *in* range(10)]

print(values)

plt.hist(values, *bins*=10, *edgecolor*='k', *color* = '#0c1881', *histtype* = 'bar')

plt.show()

**Prueba**

Text

Description automatically generated

**Chi-squared test**

**Código**

*import* numpy *as* np

*from* scipy.stats *import* chisquare

*# prueba de aleatoriedad (Prueba de Chi-cuadrada)*

"""

La tabla de frecuencias (con C = 10 y W = 0.1), mostrando los intervalos, frecuencias observadas y esperadas (no es obligatorio mostrar las operaciones, aunque yo las muestre en mi ejemplo)

el valor de χ2

las hipótesis H0 and H1

la conclusión acerca de H0 (es decir, si fue rechazada o no y por qué)

"""

*# Run the chi-squared test with the following numbers: chi\_data.txt*

*# import the observed values*

values = np.loadtxt("chi\_data.txt", *dtype*=float)

*# round the values to 3 decimals*

observed = np.around(values, 3)

*# define the expected values*

expected = np.repeat(3.0, 10)

*# define the intervals*

intervals = np.linspace(0, 1, 11)

*# calculate the observed frequencies*

observedfrequencies, \_ = np.histogram(observed, intervals)

*# we can not calculate the expected*

*# calculate the chi-square statistic and p-value*

chisquared, pvalue = chisquare(observedfrequencies, expected)

*# calculate the chi-square contributions for each interval*

chisquaredcontributions = ((observedfrequencies - expected)\*\*2) / expected

*# print the table*

print("Intervals  \t\t Observed  \t\t  Expected\t(O - E)^2 / E")

*for* i *in* range(len(intervals) - 1):

  print("[{:.3f} - {:.3f})\t{}\t\t{:.3f}\t\t{:.3f}"

    .format(

    intervals[i],

    intervals[i+1],

    observedfrequencies[i],

    expected[i],

    chisquaredcontributions[i]

    )

  )

print()

print("--------------------")

print()

print("χ2 = {: 0.3f}".format(chisquared))

print()

print("H0: Generated numbers are not different from the uniform distribution")

print("H1: Generated numbers are different from the uniform distribution")

*#Conclusion*

*if* chisquared > 16.91:

  print()

  print("Since {:0.3f} > 16.91, H0 is rejected".format(chisquared))

*else*:

  print()

  print("Since {:0.3f} <= 16.91, H0 is not rejected".format(chisquared))

**Prueba**

**Text

Description automatically generated**

**Runs test**

**Código**

*from* scipy.stats *import* norm

*import* numpy *as* np

*# prueba de aleatoriedad (Prueba de Rachas)*

"""

Los signos generados (en caso de que sean muchos, el usuario decidirá si se muestran todos o no)

La cantidad de Rachas calculadas

Los parámetros μ, σ y Zr

Las hipótesis H0 and H1

La conclusión acerca de H0 (es decir, si fue rechazada o no y por qué)

"""

numbers = np.loadtxt("runs\_data.txt", *dtype*=float)

*# Generate the signs*

signs = np.sign(numbers - np.mean(numbers))

signs[signs == 0] = 1

*# Calculate the total number of signs and runs*

total\_signs = len(signs)

total\_runs = len(np.where(np.diff(signs) != 0)[0]) + 1

*# Calculate the mean and standard deviation of the number of runs*

miu = (2 \* total\_signs - 1) / 3

sigma = np.sqrt((16 \* total\_signs - 29) / 90)

*# Calculate the z-score and p-value*

z\_score = (total\_runs - miu) / sigma

p\_value = 2 \* (1 - norm.cdf(abs(z\_score)))

*# Print the results*

print("Generated signs:\n", " ".join(["+" *if* s == 1 *else* "-" *for* s *in* signs]))

print()

print("Total signs:", total\_signs)

print("Total runs:", total\_runs)

print()

print("Statistics")

print("Miu =", miu)

print("Sigma =", sigma)

print("Z-score =", z\_score)

print()

print("H0: Appearance of the numbers is random")

print("H1: Appearance of the numbers is not random")

*if* abs(z\_score) < 1.96:

    print("Since |{}| < |1.96|, H0 is not rejected".format(z\_score))

*else*:

    print("Since |{}| >= |1.96|, H0 is rejected".format(z\_score))

**Prueba**

**Text

Description automatically generated**

Video de evidencia en drive

<https://drive.google.com/drive/folders/1FFy0UHJYJkEQ7ufW2nmqqS4_eqIfDysE?usp=sharing>